Sufficient Conditions for Feasibility and Optimality of Real-Time Optimization Schemes – I. Theoretical Foundations

Gene A. Bunin[a], Grégory François[a], Dominique Bonvin[a,*]

[a]*Laboratoire d'Automatique, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland*

Abstract

The idea of iterative process optimization based on collected output measurements, or "real-time optimization" (RTO), has gained much prominence in recent decades, with many RTO algorithms being proposed, researched, and developed. While the essential goal of these schemes is to drive the process to its true optimal conditions without violating any safety-critical, or "hard", constraints, no generalized, unified approach for *guaranteeing* this behavior exists. In this two-part paper, we propose an implementable set of conditions that can enforce these properties for any RTO algorithm. The first part of the work is dedicated to the theory behind the sufficient conditions for feasibility and optimality (SCFO), together with their basic implementation strategy. RTO algorithms enforcing the SCFO are shown to perform as desired in several numerical examples – allowing for feasible-side convergence to the *plant* optimum where algorithms not enforcing the conditions would fail.

Keywords: real-time optimization, black-box optimization, constraint satisfaction, optimization under uncertainty

1. Real-Time Optimization: Problem Structure, Characteristics, and Challenges

The idea of optimization is present in many experimental settings, as it is often the case that a user/operator is interested in finding the "best" set of decision/design variables so as to minimize (maximize) a certain cost (profit) criterion. Noting that many such problems will also involve constraints that

[☆]Nonstandard abbreviations used in this article: SCFO (sufficient conditions for feasibility and optimality).

[*]Author to whom correspondence should be addressed (tel.: +41 21 6933843, fax: +41 21 6932574).

Email addresses: `gene.bunin@epfl.ch` (Gene A. Bunin), `gregory.francois@epfl.ch` (Grégory François), `dominique.bonvin@epfl.ch` (Dominique Bonvin)

limit the decision/design space of the variables, we may state the resulting problem mathematically as follows:

$$\begin{array}{ll} \underset{\mathbf{u}}{\text{minimize}} & \phi_p(\mathbf{u}) \\ \text{subject to} & \mathbf{G}_p(\mathbf{u}) \preceq \mathbf{0} \\ & \mathbf{u}^L \preceq \mathbf{u} \preceq \mathbf{u}^U \end{array} , \tag{1}$$

with $\mathbf{u} \in \mathbb{R}^{n_u}$ denoting the manipulated decision variables (the "inputs"), $\phi_p : \mathbb{R}^{n_u} \to \mathbb{R}$ the cost function to be minimized, $\mathbf{G}_p$ a set of n_g inequality constraint functions $g_p : \mathbb{R}^{n_u} \to \mathbb{R}$, and $\mathbf{u}^L$ and $\mathbf{u}^U$ a set of lower and upper limits on the inputs (the box constraints). We adopt the subscript p (i.e. the "plant") to signify that the corresponding functions are unknown, black-box, or uncertain, i.e. that one may only evaluate them at various discrete instants by applying various choices of $\mathbf{u}$, with each choice amounting to a single experiment.

Given the generality of (1), it is not surprising that a great number of applications give rise to this problem. Examples include:

- steady-state optimization (Brdys & Tatjewski, 2005; Chen & Joseph, 1987; Cheng & Zafiriou, 2004; Engell, 2007; Fatora & Ayala, 1992; Flemming et al., 2007; Gao & Engell, 2005; Naysmith & Douglas, 1995; Tatjewski, 2008), where a plant operator is interested in finding operating conditions that maximize the profit of a dynamic plant at steady state,

- optimization of a dynamic profile in a batch process (Costello et al., 2011; François et al., 2005; Georgakis, 2009; Kadam et al., 2007; Srinivasan et al., 2003a,b), where the inputs are the "handles" used to parameterize, in some parsimonious fashion, the trajectory being optimized,

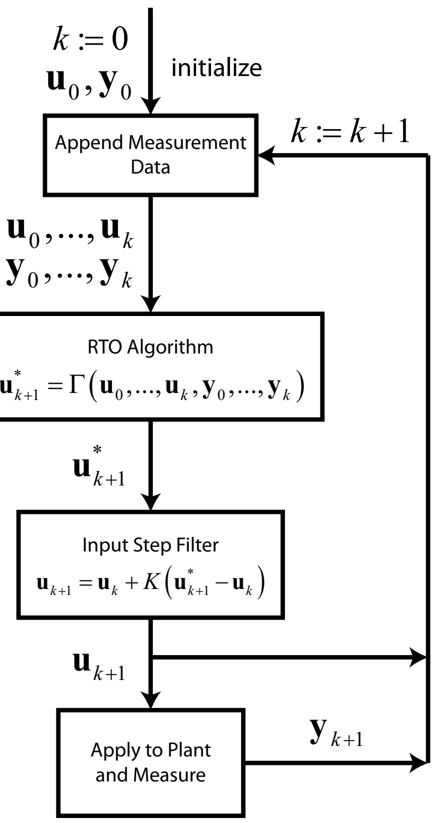
- any application where experiments are carried out to optimize a response for which a model is not available, typically done using design-of-experiments and response-surface methods (see, e.g., Myers et al. (2009) and the examples therein),

- controller tuning/design (Bunin et al., 2012a, 2013a; Hjarmarsson et al., 1998; Karimi et al., 2004; Killingsworth & Krstić, 2006; Magni et al., 2009), where one is interested in finding the controller parameters that yield the best closed-loop performance,

- numerical optimization where function evaluations are "expensive" (Conn et al., 2009), e.g. problems where evaluating a function is time-consuming as it involves simulating a system of differential equations (Vugrin, 2003),

among others.

Because the functions involved are unknown, it is evident that one cannot hope to solve Problem (1) directly and without any experimentation. We may, however, attempt to solve the problem *iteratively* – that is, we may apply certain choices of $\mathbf{u}$, measure the results, and then apply "more intelligent" choices of



$k := 0$
$\mathbf{u}_0, \mathbf{y}_0$
initialize
Append Measurement Data
$k := k+1$
$\mathbf{u}_0, ..., \mathbf{u}_k$
$\mathbf{y}_0, ..., \mathbf{y}_k$
RTO Algorithm
$\mathbf{u}^*_{k+1} = \Gamma(\mathbf{u}_0, ..., \mathbf{u}_k, \mathbf{y}_0, ..., \mathbf{y}_k)$
$\mathbf{u}^*_{k+1}$
Input Step Filter
$\mathbf{u}_{k+1} = \mathbf{u}_k + K(\mathbf{u}^*_{k+1} - \mathbf{u}_k)$
$\mathbf{u}_{k+1}$
Apply to Plant and Measure
$\mathbf{y}_{k+1}$

Figure 1: Generalized feedback structure of RTO schemes.

u based on what we have learned. A general formulation of such an approach may be given as:

$$\begin{array}{l} \mathbf{u}^*_{k+1} = \Gamma(\mathbf{u}_0, ..., \mathbf{u}_k, \mathbf{y}_0, ..., \mathbf{y}_k) \\ \mathbf{u}_{k+1} = \mathbf{u}_k + K(\mathbf{u}^*_{k+1} - \mathbf{u}_k) \end{array}, \tag{2}$$

where $\mathbf{y} \in \mathbb{R}^{n_y}$ denotes the measurements (the "outputs"), k the iteration (experiment) counter, and $\Gamma(\cdot)$ some prescribed adaptation law to yield an "optimal" target $\mathbf{u}^*_{k+1}$ for the next experiment, which is often filtered with a gain of $K \in [0, 1]$ as a safety precaution (Brdys & Tatjewski, 2005). It is precisely this iterative nature of (2) that introduces the "real-time" element. We will, hereafter, refer to $\Gamma(\cdot)$ as a *real-time optimization (RTO) algorithm*, and to any problem having the form of (1) that is solved by (2) as an *RTO problem*. A graphical illustration of the iterative RTO procedure is presented in Figure 1.

Unfortunately, RTO problems rarely have "neat" solutions, and a number of complications arise due to the experimental nature of the problem. The major culprit is, of course, the black-box nature of the problem – one does not know the analytical properties of ϕ_p and $\mathbf{G}_p$, thereby making it impossible to optimize via even the simplest numerical approaches like the gradient descent, let alone via more elegant (e.g. interior-point) methods. A typical strategy in many RTO algorithms (Brdys & Tatjewski, 2005; Bunin et al., 2012b; Gao & Engell, 2005; Marchetti et al., 2010; Rodger, 2010; Tadej & Tatjewski, 2001) is to estimate the gradient from discrete measurements, but this is often made difficult by the presence of *measurement noise*, thereby requiring a number of precautions and/or limitations for such an estimation to be effective. Another pressing difficulty is feasibility, since the inequality constraints $\mathbf{G}_p$ may be safety-critical or "hard", in which case one is strictly forbidden from applying any $\mathbf{u}$ that violates one of the inequalities, as doing so may promote hazardous conditions that either endanger the personnel or irreversibly damage the experimental equipment. Finally, the innately costly nature of performing experiments generally demands that RTO algorithms move quickly and that they move in the right direction – an RTO algorithm that does not lead to immediate improvement may be abandoned after only a few iterations, while an algorithm that promises optimality, but only after 10,000 iterations, may never be implemented in the first place.

In spite of these challenges, a number of RTO algorithms have been proposed over the years and may be summarized (more or less completely) as follows:

- model-based approaches where a parametric model of Problem (1) is available and where the measurements are used for parameter estimation, after which the updated parametric model is optimized numerically to yield $\mathbf{u}^*_{k+1}$ (Chen & Joseph, 1987; Fatora & Ayala, 1992; Jang et al., 1987; Naysmith & Douglas, 1995),

- model-based approaches where bias correction terms are estimated and added to the model, with a numerical optimization of the corrected model used to yield $\mathbf{u}^*_{k+1}$ (Brdys & Tatjewski, 2005; François & Bonvin, 2013; Gao & Engell, 2005; Marchetti, 2009; Roberts, 1978; Xu & Cheng, 2008),

- response-surface methods that are based on optimal experimental design (Georgakis, 2009; Myers et al., 2009), where a (usually quadratic) model is built from the collected measurements and then optimized ($\mathbf{u}^*_{k+1}$ often becoming fixed once this is done),

- direct-search methods that attempt to estimate and use the gradient (Box & Wilson, 1951; Bunin et al., 2012a; François et al., 2005; Garcia & Morari, 1984; Killingsworth & Krstić, 2006), akin to the standard gradient-descent algorithm in numerical optimization ($\mathbf{u}^*_{k+1}$ being a step in the gradient descent direction),

- derivative-free methods that avoid estimating the gradient entirely and optimize purely via zero-order knowledge (Alexandrov et al., 1997; Box & Draper, 1969; Conn et al., 2009; Holmes, 2003), with $\mathbf{u}^*_{k+1}$ defined differently depending on the algorithm used, together with any hybrids/variations of the above.

However, even with this variety of methods, the state-of-the-art in RTO remains largely *ad hoc*, as none of the above approaches are capable of overcoming the stated challenges and *consistently* solving Problem (1) to local optimality without violating the hard constraints in the process. As such, while one algorithm may perform well for a certain problem, it may perform quite poorly (or even be dangerous) for another. The result is an additional burden on the user, as even *conceptual* guarantees of feasibility and optimality are absent in these algorithms.

Our goal in this work is not to propose an algorithm that solves all of these issues – rather, we aim to propose a set of sufficient conditions that serve to ensure the desired properties while being applicable to any RTO algorithm that falls into the framework of Figure 1. In doing so, we hope to help lay the foundations for an RTO *theory* in a field that has often relied on heuristic approaches without rigorous guarantees. For convenience, we will refer to the full set of these conditions by the acronym SCFO (the "sufficient conditions for feasibility and optimality").

The basic idea of the SCFO is to ensure that the next applied iterate, $\mathbf{u}_{k+1}$, always fall into the local feasible descent space[1] and that the distance between $\mathbf{u}_k$ and $\mathbf{u}_{k+1}$ be small enough so as to ensure that the applied iterate will both be feasible and have an improved cost. The main contribution of this work is the theory and rigor behind this strategy (detailed in Theorems 2-5). The secondary contribution is the algorithm proposed for the basic SCFO implementation, which takes the input target provided by the RTO algorithm and projects it onto the local feasible descent space in such a manner so as to ultimately enforce the desired feasible-side convergence without introducing major performance drawbacks (i.e. very slow convergence speed).

In order to properly establish the aforementioned guarantees, we are forced to make several basic assumptions throughout this paper:

A1: The functions ϕ_p and $\mathbf{G}_p$ are twice continuously differentiable over the *relevant input space* $\mathcal{I} = \{\mathbf{u} : \mathbf{u}^L \preceq \mathbf{u} \preceq \mathbf{u}^U\}$.

A2: The initial point, $\mathbf{u}_0$, is strictly feasible with respect to the inequality constraints ($\mathbf{G}_p(\mathbf{u}_0) \prec \mathbf{0}$).

A3: For every function $g_{p,j}, j = 1, ..., n_g$, there exists an $\epsilon_{m,j} > 0$ such that, for any given iteration k, $0 \geq g_{p,j}(\mathbf{u}_k) \geq -\epsilon_{m,j} \Rightarrow \nabla g_{p,j}(\mathbf{u}_k) \neq \mathbf{0}$.

A4: All RTO algorithms considered will never yield a target outside of $\mathcal{I}$, i.e. $\mathbf{u}^*_{k+1} \in \mathcal{I}$ always.

Here, Assumption A1 is required for the existence of 1^{st}- and 2^{nd}-order upper bounds (see Section 2), as well as for the existence of the gradients of the cost

[1]The space where every choice of $\mathbf{u}$ is guaranteed to both decrease the cost and to satisfy the plant constraints locally.

and constraints, while Assumption A2 is required for feasibility (Section 3). Assumption A3 is required to formally avoid the (somewhat pathological) case where an active constraint has a gradient of zero, and is subsumed by the more standard linear independence constraint qualification (e.g. Fletcher, 1987, Ch. 9). Assumption A4 aids in simplifying some of the derivations and proofs in this work.

In this first part of the two-part contribution, the SCFO are stated and proven *conceptually*. As such, more importance is given to theory and less to practical aspects. Particularly, the following are assumed to be available:

- Knowledge of the gradients of ϕ_p and $\mathbf{G}_p$ at the current iteration k.
- Knowledge of global Lipschitz constants for $\mathbf{G}_p$.
- Knowledge of a global quadratic upper bound for ϕ_p.
- Knowledge of the exact values of $\mathbf{G}_p$ at the current iteration k (i.e. noise-free measurements, or perfect estimates, of the constraints).

The results presented in this first part are thus not directly applicable per se, and serve only to provide a theoretical base. Additionally, as our focus is solely on feasibility and optimality, the notion of convergence speed is also absent from our discussions. In the sequel paper (Bunin et al., 2013b), the practical implementation issues are treated in detail, considering the cases where the knowledge of the four items above is imperfect, as well as proposing modifications to the SCFO that may greatly speed up the convergence of a given algorithm.

The remainder of this paper is structured as follows. Following the presentation of the key mathematical concepts in Section 2, we dedicate Section 3 to the feasibility problem of meeting the hard constraints at every RTO iteration, which makes up the bulk of the paper and presents the core components of the theoretical contribution. Guaranteeing convergence to a Karush-Kuhn-Tucker (KKT) point via feasible iterates is then the subject of Section 4, and acts as a natural extension of the theory presented in the section before. Numerical examples are given in both sections to illustrate the methodology for a simple two-dimensional case with readily available geometric representations. Section 5 then serves to conclude the paper with a summary and discussion of the results and their potential significance in the RTO community.

2. Mathematical Preliminaries

Here, we present the tools and definitions that will act as the foundation for the results that follow in Sections 3 and 4.

A key necessity for the theory proposed in this work is to be able to upper bound the evolution (the worst-case possible growth) of twice continuously differentiable black-box functions, generalized here by $f : \mathbb{R}^{n_u} \to \mathbb{R}$, over the compact input space $\mathcal{I}$. For convenience and cohesion with results that come later, this is always done with respect to the input point $\mathbf{u}_k$, which may be thought of as the current iterate at the current RTO iteration k. The following two bounds, one linear and one quadratic, are proposed in Lemmas 1 and 2, respectively.

Lemma 1. (Linear Upper Bound on the Evolution of an Unknown Function)
Let f be continuously differentiable over $\mathcal{I}$, so that:

$$-\kappa_i < \frac{\partial f}{\partial u_i}\Big|_{\mathbf{u}} < \kappa_i, \quad \forall \mathbf{u} \in \mathcal{I}, \quad i = 1, ..., n_u, \tag{3}$$

where κ are the univariate Lipschitz constants of f.

Then, the evolution of f between any $\mathbf{u}_k, \mathbf{u}_{k+1} \in \mathcal{I} \setminus \{\mathbf{u}_k, \mathbf{u}_{k+1} : \mathbf{u}_k = \mathbf{u}_{k+1}\}$ may be strictly bounded as:

$$f(\mathbf{u}_{k+1}) - f(\mathbf{u}_k) < \sum_{i=1}^{n_u} \kappa_i |u_{k+1,i} - u_{k,i}|. \tag{4}$$

Proof. Start by defining:

$$\mathbf{\Delta}_i = \left[\begin{array}{cc} \mathbf{I}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0}_{n_u - i} \end{array} \right], \tag{5}$$

i.e. an $n_u \times n_u$ diagonal matrix with the first i diagonal elements equal to 1 and the remaining $n_u - i$ equal to 0. This allows us to write the evolution of f as the summation of its decoupled, univariate evolutions in the following compact form:

$$f(\mathbf{u}_{k+1}) - f(\mathbf{u}_k) = \sum_{i=1}^{n_u} \Big(f(\mathbf{u}_k + \mathbf{\Delta}_i(\mathbf{u}_{k+1} - \mathbf{u}_k)) - f(\mathbf{u}_k + \mathbf{\Delta}_{i-1}(\mathbf{u}_{k+1} - \mathbf{u}_k)) \Big). \tag{6}$$

From (3) and the definition of the Lipschitz constant for the univariate case, it follows that:

$$f(\mathbf{u}_k + \mathbf{\Delta}_i(\mathbf{u}_{k+1} - \mathbf{u}_k)) - f(\mathbf{u}_k + \mathbf{\Delta}_{i-1}(\mathbf{u}_{k+1} - \mathbf{u}_k)) \leq \kappa_i |u_{k+1,i} - u_{k,i}|, \tag{7}$$

with strict inequality when $u_{k+1,i} \neq u_{k,i}$. Since $\mathbf{u}_k \neq \mathbf{u}_{k+1}$, it follows that $\exists i : u_{k+1,i} \neq u_{k,i}$ and, summing (7) for $i = 1, ..., n_u$, we immediately have the result in (4). $\square$

Lemma 2. (Quadratic Upper Bound on the Evolution of an Unknown Function)
Let f be twice continuously differentiable over $\mathcal{I}$, so that:

$$-M_{ij} < \frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}} < M_{ij}, \quad \forall \mathbf{u} \in \mathcal{I}, \quad i, j = 1, ..., n_u. \tag{8}$$

Then, the evolution of f between any $\mathbf{u}_k, \mathbf{u}_{k+1} \in \mathcal{I}$ may be bounded as:

$$f(\mathbf{u}_{k+1}) - f(\mathbf{u}_k) \leq \nabla f(\mathbf{u}_k)^T (\mathbf{u}_{k+1} - \mathbf{u}_k) + \frac{1}{2}(\mathbf{u}_{k+1} - \mathbf{u}_k)^T \overline{\mathbf{Q}} (\mathbf{u}_{k+1} - \mathbf{u}_k), \quad (9)$$

where $\overline{\mathbf{Q}} \succ 0$ is the quadratic upper bound, a diagonal matrix with its diagonal elements defined as:

$$\overline{Q}_{ii} = \sum_{j=1}^{n_u} M_{ij}, \quad i = 1, ..., n_u. \quad (10)$$

Proof. We refer the reader to the appendix. □

Also common throughout the discussion that follows is the notion of descent halfspaces and approximately active (ϵ-active) constraints. We define these as follows.

Definition 1. (Local Descent Halfspace)
The strict local descent halfspace of f at $\mathbf{u}_k$ is defined as the set $\{\mathbf{u} : \nabla f(\mathbf{u}_k)^T (\mathbf{u} - \mathbf{u}_k) < 0\}$. Its nonstrict approximation is defined as $\{\mathbf{u} : \nabla f(\mathbf{u}_k)^T (\mathbf{u} - \mathbf{u}_k) \leq -\delta, \ \delta > 0\}$, with the quality of the approximation increasing as $\delta \to 0$.

Definition 2. (ϵ-Active Constraints)
A constraint $g_{p,j}$ is said to be ϵ-active at iteration k if, for $\epsilon_j > 0$, $0 > g_{p,j}(\mathbf{u}_k) \geq -\epsilon_j$. This is used to approximate an active constraint set by an ϵ-active set, with the quality of approximation increasing as $\epsilon_j \to 0$.

We now combine the concepts of the descent halfspace and the quadratic upper bound to derive a sufficient step size and step direction to guarantee strict descent for f. This is crucial both to guarantee that an algorithm can preserve feasibility without converging prematurely to a constraint (Section 3) and to guarantee that an algorithm can decrease the cost at every iteration (Section 4).

Lemma 3. (Guaranteed Descent Step)
Let $\mathbf{u}^*_{k+1}$ lie in the strict local descent halfspace of f at $\mathbf{u}_k$ and let K be the step size in the direction $\mathbf{u}^*_{k+1} - \mathbf{u}_k$, so that the next iterate is defined as in (2). A strict descent in the function value, i.e. $f(\mathbf{u}_{k+1}) < f(\mathbf{u}_k)$, is guaranteed if the following condition holds:

$$0 < K < -2 \frac{\nabla f(\mathbf{u}_k)^T (\mathbf{u}^*_{k+1} - \mathbf{u}_k)}{(\mathbf{u}^*_{k+1} - \mathbf{u}_k)^T \overline{\mathbf{Q}} (\mathbf{u}^*_{k+1} - \mathbf{u}_k)}, \quad (11)$$

with $\overline{\mathbf{Q}} \succ 0$ defined as in (10).

PROOF. The maximum value that $f(\mathbf{u}_{k+1})$ can take is given by Lemma 2:

$$f(\mathbf{u}_{k+1}) = f(\mathbf{u}_k) + \nabla f(\mathbf{u}_k)^T(\mathbf{u}_{k+1} - \mathbf{u}_k) + \frac{1}{2}(\mathbf{u}_{k+1} - \mathbf{u}_k)^T\overline{\mathbf{Q}}(\mathbf{u}_{k+1} - \mathbf{u}_k), \quad (12)$$

where substituting the update law of (2) readily yields:

$$f(\mathbf{u}_{k+1}) = f(\mathbf{u}_k) + K\nabla f(\mathbf{u}_k)^T(\mathbf{u}^*_{k+1} - \mathbf{u}_k) + \frac{K^2}{2}(\mathbf{u}^*_{k+1} - \mathbf{u}_k)^T\overline{\mathbf{Q}}(\mathbf{u}^*_{k+1} - \mathbf{u}_k), \quad (13)$$

which is quadratic in K and satisfies $f(\mathbf{u}_{k+1}) = f(\mathbf{u}_k)$ for:

$$K = 0, -2\frac{\nabla f(\mathbf{u}_k)^T(\mathbf{u}^*_{k+1} - \mathbf{u}_k)}{(\mathbf{u}^*_{k+1} - \mathbf{u}_k)^T\overline{\mathbf{Q}}(\mathbf{u}^*_{k+1} - \mathbf{u}_k)}, \quad (14)$$

with the latter forced to be strictly positive from the definition of a strict local descent halfspace (i.e. strictly negative numerator) and the strict positivity of the denominator.

As (13) must be strictly convex in K, it follows that $f(\mathbf{u}_{k+1}) < f(\mathbf{u}_k)$ for all values of K between the two bounds of (14). □

Finally, the algorithm that enforces the SCFO relies on projecting, at every iteration, the RTO target onto a local feasible descent space, which is characterized by an intersection of halfspaces. To prove that this projection will always be feasible unless a KKT point has been reached, the following theorem of alternatives will be needed.

Theorem 1. (Gale's Theorem)
Let $\mathbf{J}_k = [\nabla f_1(\mathbf{u}_k)...\nabla f_n(\mathbf{u}_k)]$ and consider the set $\{\mathbf{u} : \mathbf{J}_k^T(\mathbf{u} - \mathbf{u}_k) \preceq -\boldsymbol{\delta}\}$. This set is empty and does not admit a solution iff:

$$\exists \boldsymbol{\nu} \in \mathbb{R}^n_+ : \sum_{i=1}^{n} \nu_i \nabla f_i(\mathbf{u}_k) = \mathbf{0}, \quad -\boldsymbol{\nu}^T\boldsymbol{\delta} < 0. \quad (15)$$

PROOF. This is a well-known result in linear programming (Rockafellar, 1970, Th. 22.1). □

3. Feasibility Guarantees

The goal of this section is to focus on the guaranteed satisfaction of hard constraints, i.e. on constraints that, for safety reasons, must *never* be violated[2]. We start with a brief review of the existing methods for enforcing RTO feasibility, and then proceed to propose a new feasibility-guaranteeing condition based

[2]So as to work with the most limiting case, we will hereafter assume that all of the constraints in $\mathbf{G}_p$ are hard, i.e. it is required that $\mathbf{G}_p(\mathbf{u}_k) \preceq \mathbf{0}$, $\forall k$.

on the use of an *adaptive* filter gain in the input adaptation step (2), with the filter gain at iteration k a function of the constraint values at iteration k and the Lipschitz constants for those constraints. A drawback of the proposed approach is in its potentially premature convergence, and so we propose conditions that allow the algorithm to avoid this problem. Several examples are then given to illustrate the presented ideas.

3.1. Approaches to Guaranteeing Feasibility

Some of the more theoretically elegant approaches to enforcing the feasibility of (1) include the stochastic ("chance constraint") and worst-case robust approaches, which require a model of the constraints and are generally used in an initial design phase (Beyer & Sendhoff, 2007), though they may be used for RTO needs when the model is adapted (Flemming et al., 2007; Zhang et al., 2002). In all cases, parametric uncertainty in the model is assumed and quantified. As a simple illustration of what this entails, suppose that a single constraint $g_{p,j}$ is uncertain due to a single uncertain parameter θ, known to lie in the set Θ, and that we would like to guarantee the following:

$$g_{p,j}(\mathbf{u}_{k+1}) \leq 0, \tag{16}$$

by guaranteeing:

$$\sup_{\theta \in \Theta} g_j(\mathbf{u}_{k+1}, \theta) \leq 0, \tag{17}$$

where $g_j : \mathbb{R}^{n_u} \to \mathbb{R}$ is the parametric model.

The standard stochastic approach supposes a probability distribution for the possible values of θ and attempts to satisfy (17) with a specified probability, while the worst-case approach essentially attempts to do the same but with a probability of 1. The latter is generally considered as being very conservative since it accounts for all possible deviations, while the former suffers from the lack of robustness introduced by allowing the constraint to be violated, if only with a low probability.

While these methods are theoretically sound, there are several difficulties in applying them to RTO problems (Quelhas et al., 2013). The immediate issue that should come to mind is the initial assumption that the uncertainty is parametric, which will not hold in the general case. When structural uncertainties are present, it follows that both approaches lose their theoretical rigor, with the degree of the loss varying, naturally, with the amount of structural uncertainty. Additionally, even when the parametric uncertainty assumption is satisfied, obtaining the probability distributions or worst-case upper and lower bounds on the parameters may be a very challenging task. Finally, the problem may become computationally intractable, depending on the probability distribution and the way in which the parameters enter into the constraint model. This latter becomes particularly troublesome as the number of uncertain parameters and constraints grows. While it is sometimes proposed to solve this problem by linearizing the constraint model with respect to the uncertain parameters (Zhang et al., 2002), doing so may lead to significant inaccuracies when the process is nonlinear – even if there is no structural uncertainty.

Some simpler methods of enforcing feasibility involve the use of back-offs on the constraints (Govatsmack & Skogestad, 2005; Quelhas et al., 2013):

$$g_j(\mathbf{u}_{k+1}, \theta) + c \leq 0, \;\; c > 0, \tag{18}$$

or the use of limited input changes for each iteration (box constraints on the input step sizes (Cheng & Zafiriou, 2000; Gao & Engell, 2005)):

$$\mathbf{u}_k - \Delta \mathbf{u}_{max} \preceq \mathbf{u}_{k+1} \preceq \mathbf{u}_k + \Delta \mathbf{u}_{max}. \tag{19}$$

Both may be conservative, however, and are ultimately *ad hoc* approaches that do not come with the desired guarantees of feasibility.

In view of all this, Bunin et al. (2011) proposed a simple but robust approach that uses a variable filter gain K_k in the standard input filtering law (2):

$$\mathbf{u}_{k+1} = \mathbf{u}_k + K_k \left(\mathbf{u}^*_{k+1} - \mathbf{u}_k \right), \tag{20}$$

where the gain K_k is defined at iteration k in such a way so as to guarantee $g_{p,j}(\mathbf{u}_{k+1}) < 0$ provided that $g_{p,j}(\mathbf{u}_k) < 0$. The following theorem, largely taken from our previous work (Bunin et al., 2011), derives the adaptive upper bound on the filter gain K_k so as to guarantee recursive feasibility for any RTO scheme.

Theorem 2. (Sufficient Condition for Feasibility)
Suppose that $\mathbf{G}_p(\mathbf{u}_k) \prec \mathbf{0}$, and that an RTO algorithm provides a new optimal input target $\mathbf{u}^*_{k+1} \neq \mathbf{u}_k$ that is then filtered according to (20) to give $\mathbf{u}_{k+1}$. Then, enforcing the following upper bound on the adaptive filter gain:

$$K_k \leq \min_{j=1,\dots,n_g} \left[\frac{-g_{p,j}(\mathbf{u}_k)}{\displaystyle\sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}|} \right], \tag{21}$$

with $-\kappa_{ji} < \left. \frac{\partial g_{p,j}}{\partial u_i} \right|_{\mathbf{u}} < \kappa_{ji}, \forall \mathbf{u} \in \mathcal{I}$, *is sufficient to guarantee that* $\mathbf{G}_p(\mathbf{u}_{k+1}) \prec \mathbf{0}$.

Proof. Noting that $K_k \in [0,1]$ by definition, we treat two cases: $K_k = 0$ and $K_k > 0$. For the former, we have the trivial result that $\mathbf{u}_{k+1} = \mathbf{u}_k \Rightarrow \mathbf{G}_p(\mathbf{u}_{k+1}) = \mathbf{G}_p(\mathbf{u}_k) \preceq \mathbf{0}$, with feasibility guaranteed since the inputs are not adapted.

When $K_k > 0$ it follows that $\mathbf{u}_{k+1} \neq \mathbf{u}_k$, which allows us to use Lemma 1 to write:

$$g_{p,j}(\mathbf{u}_{k+1}) < g_{p,j}(\mathbf{u}_k) + \sum_{i=1}^{n_u} \kappa_{ji} |u_{k+1,i} - u_{k,i}|, \quad \forall j = 1, ..., n_g. \tag{22}$$

Substituting in the filter law (20) for $\mathbf{u}_{k+1}$ then leads to:

$$g_{p,j}(\mathbf{u}_{k+1}) < g_{p,j}(\mathbf{u}_k) + K_k \sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}|, \quad \forall j = 1, ..., n_g. \tag{23}$$

Let us now choose K_k in such a way so as to make the right-hand side less than or equal to 0, which will clearly guarantee the same (with strict inequality) for $g_{p,j}(\mathbf{u}_{k+1})$:

$$g_{p,j}(\mathbf{u}_k) + K_k \sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}| \leq 0, \quad \forall j = 1, ..., n_g. \tag{24}$$

Rearranging leads to:

$$K_k \leq \frac{-g_{p,j}(\mathbf{u}_k)}{\displaystyle\sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}|}, \quad \forall j = 1, ..., n_g. \tag{25}$$

Now, to guarantee that this holds for all of the constraints, we simply take the component-wise minimum, which leads to the expression in (21). □

Remarks

- We can now justify the initial supposition of $\mathbf{G}_p(\mathbf{u}_k) \prec \mathbf{0}$ *a posteriori*, since Assumption A2 calls for $\mathbf{G}_p(\mathbf{u}_0) \prec \mathbf{0}$ and recursive feasibility is enforced by (21) up to iteration k and beyond.
- The main concept of Theorem 2 is very simple. Namely, we bound the worst-case evolution of the constraints (via the Lipschitz constants), calculate how far we can move before one of them, in the worst case, reaches its boundary, and then define the filter gain as the adaptive parameter with which to control this distance. The benefits of the approach lie in its generality, as it does not require the uncertainty to have any particular structure – everything is simply lumped into the worst-case growth of the constraints. The adaptive element is also an advantage, in that the gain is always a function of the current measured/estimated values of the constraints. As such, when the plant is far away from the constraints, the gain is naturally larger, and is reduced as the constraints are approached. In the case of model-based RTO algorithms, this method also avoids introducing additional constraints into the original model-based optimization problem, thus guaranteeing that tractability is never lost. Figure 2 illustrates the main idea of Theorem 2 geometrically.

We will, for the remainder of this section, adopt the following heuristic choice of K_k:

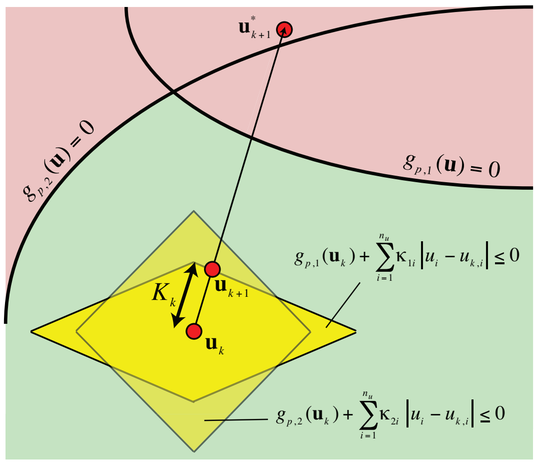

$\mathbf{u}^*_{k+1}$
$g_{p,2}(\mathbf{u}) = 0$
$g_{p,1}(\mathbf{u}) = 0$
$g_{p,1}(\mathbf{u}_k) + \sum_{i=1}^{n_u} \kappa_{1i} |u_i - u_{k,i}| \leq 0$
K_k
$\mathbf{u}_{k+1}$
$\mathbf{u}_k$
$g_{p,2}(\mathbf{u}_k) + \sum_{i=1}^{n_u} \kappa_{2i} |u_i - u_{k,i}| \leq 0$

Figure 2: Geometric illustration of adaptive input filtering as a means of guaranteeing feasibility from iteration to iteration. Here, two constraints are considered. The green region denotes the feasible space, with the yellow polytopes corresponding to the robust feasible areas that are generated by the Lipschitz constants for each constraint. The adaptive filter gain value may be any value that keeps all of the constraint values in their worst-case growth regions. Here, the first constraint proves to be limiting.

$$K_k := \min_{j=1,\dots,n_g} \left[\frac{-g_{p,j}(\mathbf{u}_k)}{\sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}|} \right], \tag{26}$$

as taking the largest allowable K_k generally leads to faster progress for the RTO algorithm.

3.2. Premature Convergence of the Adaptive Input Filter Method

By themselves, the input filtering law (20) and the upper bound on the filter gain (21) are sufficient for feasibility, provided that the initial point $\mathbf{u}_0$ is feasible. There is, however, a major algorithmic issue with using the inequality (21), which may be illustrated as follows. Suppose that a single constraint $g_{p,j}$ is active and its value is arbitrarily close[3] to 0. In looking at the expression for the adaptive filter gain in (21), it becomes clear that the bound on K_k will be arbitrarily close to 0 as well. This, in turn, implies that the input is not adapted and no further progress is made.

This issue has been pointed out previously (Bunin et al., 2011), and is illustrated in Figure 3 via a simple constructed example. Here, the RTO algorithm

[3]We insist on "arbitrarily close" since the Lipschitz bounds are strict and so the constraint never reaches 0 exactly with the input filtering scheme as proposed in this work.

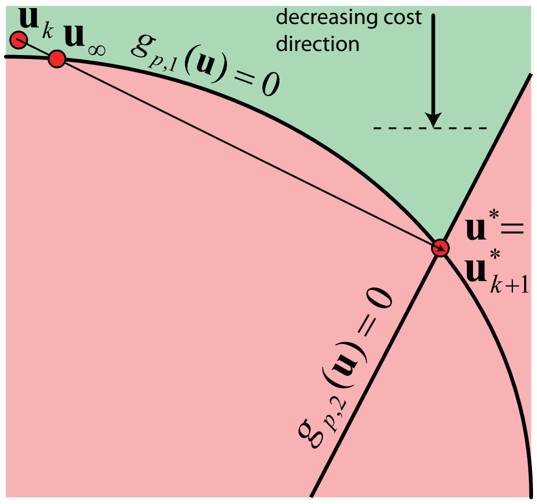

$\mathbf{u}_k$
$\mathbf{u}_\infty$
$g_{p,1}(\mathbf{u}) = 0$
decreasing cost direction
$\mathbf{u}^* = \mathbf{u}^*_{k+1}$
$g_{p,2}(\mathbf{u}) = 0$

Figure 3: An example illustrating premature convergence, where one of the constraints becomes active and brings the filter gain to an arbitrarily small value. As the segment $[\mathbf{u}_\infty, \mathbf{u}^*_{k+1}]$ is infeasible, the algorithm converges to $\mathbf{u}_\infty$ without being able to proceed further.

is perfect in that it provides the true plant optimum, $\mathbf{u}^*$, at each iteration. However, because of the concave nature of one of the plant constraints, it is clear that the iterates will eventually touch the boundary and remain arbitrarily close to it without being able to advance any further.

Depending on the particular problem, the losses incurred from such premature convergence may or may not be significant. In Bunin et al. (2011), it was argued that the majority of the optimality gains could be attained before one of the constraints became active, and this was shown through a numerical example. However, this may not hold always, and for processes that operate for a great number of iterations converging prematurely without any further improvement may not be an attractive result, even if feasibility is retained throughout. In the following section, we show how using the constraint *gradients* allows us to construct an algorithm that avoids premature convergence by "sliding" along any constraints that are close to active.

3.3. Avoiding Premature Convergence

Prior to presenting an improved version of the method, it is first necessary to characterize the degree to which the Lipschitz constants of the constraint functions are strict.

Definition 3. (Degree of Strictness of the Lipschitz Constants)

Denoting by $\tilde{\kappa}$ the nonstrict Lipschitz constants, which allows the nonstrict analogue to (3):

$$-\tilde{\kappa}_{ji} \leq \left.\frac{\partial g_{p,j}}{\partial u_i}\right|_{\mathbf{u}} \leq \tilde{\kappa}_{ji}, \quad \forall \mathbf{u} \in \mathcal{I}, \quad i = 1, ..., n_u, \;\; j = 1, ..., n_g, \tag{27}$$

the degree of strictness of the Lipschitz constants, γ_κ, is defined as:

$$\gamma_\kappa = \max_{\substack{i=1,...,n_u \\ j=1,...,n_g}} \frac{\tilde{\kappa}_{ji}}{\kappa_{ji}}. \tag{28}$$

We now give the sufficient conditions for maintaining K_k above a certain strictly positive value, thereby precluding the possibility of its becoming arbitrarily small, and thus avoiding premature convergence.

Theorem 3. (Sufficient Conditions for a Strictly Positive Adaptive Filter Gain)

Let K_k be chosen as in (26). If $\mathbf{u}^*_{k+1}$ lies in the strict local descent halfspace for all of the ϵ-active constraints:

$$\nabla g_{p,j}(\mathbf{u}_k)^T(\mathbf{u}^*_{k+1} - \mathbf{u}_k) \leq -\delta_{g,j}, \; \forall j : g_{p,j}(\mathbf{u}_k) \geq -\epsilon_j, \tag{29}$$

with $\delta_{g,j} > 0$, then:

$$K_k > \min_{j=1,\ldots,n_g} \left[\frac{\min\left((1-\gamma_\kappa)\epsilon_j, (1-\gamma_\kappa)\dfrac{K_{\epsilon,j}\delta_{g,j}}{\gamma_\kappa}, -g_{p,j}(\mathbf{u}_0) \right)}{\boldsymbol{\kappa}_j^T(\mathbf{u}^U - \mathbf{u}^L)} \right] > 0, \tag{30}$$

where:

$$K_{\epsilon,j} = \frac{2\delta_{g,j}}{(\mathbf{u}^U - \mathbf{u}^L)^T \overline{\mathbf{Q}}_j (\mathbf{u}^U - \mathbf{u}^L)}, \tag{31}$$

$$\boldsymbol{\kappa}_j^T = \left[\kappa_{j1} \; ... \; \kappa_{jn_u}\right], \tag{32}$$

and $\overline{\mathbf{Q}}_j \succ 0$ denotes the quadratic upper bound for constraint $g_{p,j}$ as defined in Lemma 2.

Proof. The theorem may be proven by exploiting the result of Lemma 3 and showing that enforcing Condition (29) makes it impossible for any constraint, and thereby K_k, to approach 0 since for a small enough step ($K_{\epsilon,j}$) one will be forced to push the constraint away from activity. We refer the reader to the appendix for the detailed proof. □

Several points of the theorem merit some discussion.

Geometric Interpretation of Theorem 3

The geometric interpretation of the above theorem is presented via two examples in Figure 4, and illustrates the role of Condition (29) in forcing the optimal input direction to lie in the strict local descent halfspace of an ϵ-active constraint. The top example in Figure 4 shows this for a single convex constraint. When Condition (29) is met and the optimal direction lies in the strict local descent halfspace (the lined area), it is easy to see that each optimal direction will result, even if very locally, in a descent direction. The degree of locality here depends on both the direction (how orthogonal it is to the hyperplane $\nabla g_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$) and the magnitude of the higher-order derivatives (in some sense, $\overline{\mathbf{Q}}$). Concerning the effect of direction, it is easy to see that directions orthogonal to the hyperplane would be steepest descent directions, with the constraint value going down substantially locally. With regard to the size of the nonlinear terms, we can imagine what would happen if, for example, the higher-order derivatives of the constraint were much larger – this would squeeze the feasible space in the top example in Figure 4 (shown via the dashed blue lines and the double-lined region) and lead to both much shorter feasible and descent steps if we were to follow the same optimal directions (arrowed lines). Both of these cases are reflected analytically in the upper bound (11) of Lemma 3 – steepest descent directions leading to a greater numerator and thus larger steps, and larger higher-order derivatives (greater $\overline{\mathbf{Q}}$) leading to a greater denominator and thus smaller steps. Finally, the illustration makes it obvious that any optimal direction that consistently points into the local *ascent* halfspace would lead to the constraint value approaching 0, thereby forcing the algorithm to converge prematurely.

Significance of $\overline{\mathbf{Q}}$

It has been supposed throughout, both in the proofs of Lemma 3 and Theorem 3, that $\overline{\mathbf{Q}} \succ 0$, as this represents the worst case with respect to the evolution of K_k. If the constraint is strictly convex, then its higher-order evolution will be positive definite and such worst-case behavior will be justified (indeed, this is the case in the top example of Figure 4). We now consider the alternative when $\overline{\mathbf{Q}} \preceq 0$, which could be a valid upper bound when the constraint is concave (e.g. a negative semidefinite quadratic). If we allow such a bound, then it is clear from the analysis in the proof of Lemma 3 that $g_{p,j}(\mathbf{u}_{k+1}) < g_{p,j}(\mathbf{u}_k)$ will hold for *all* strictly positive values K_k, and indeed it becomes clear why if we consider the interpretation in the bottom example of Figure 4. With a concave constraint, the hyperplane $\nabla g_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$ supports the entire infeasible side of the constraint, and the condition in (29) thereby guarantees that any direction taken will only lead to both feasible and smaller values, regardless of the filter gain. The strict descent halfspace is global in this case.

More generally speaking, the abstract purpose of $\overline{\mathbf{Q}}$ is to help provide the conceptual proof that Condition (29) will prevent K_k from going to 0 by always allowing a local descent direction whose length will depend, in large part, on $\overline{\mathbf{Q}}$. The fact that $\overline{\mathbf{Q}}$ may never be known in practice therefore does not hinder us – we only need it to exist. As such, one does not need to go through the trouble of estimating $\overline{\mathbf{Q}}$ and attempting to enforce the step size derived in Lemma 3, although being able to do so would provide a tool to guarantee immediate descent at iteration k.

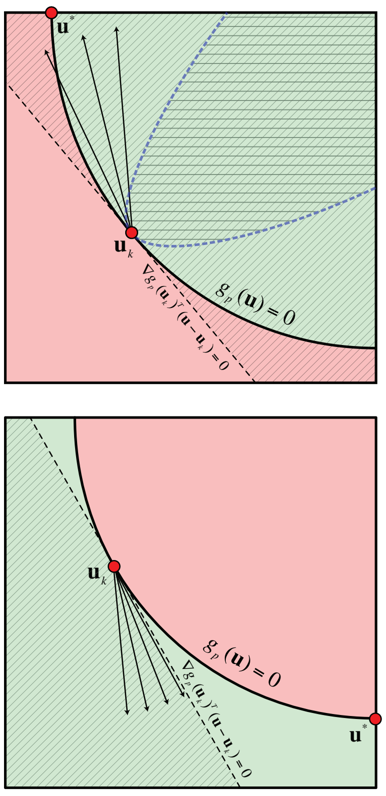

$\mathbf{u}^*$
$\mathbf{u}_k$
$\nabla g_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$g_p(\mathbf{u})=0$
$\mathbf{u}_k$
$g_p(\mathbf{u})=0$
$\nabla g_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$\mathbf{u}^*$

Figure 4: A geometric interpretation of the sufficient condition for avoiding premature convergence in the adaptive input filter scheme. In both examples, the lined area represents the local descent halfspace for the relevant constraint, with the arrows used to show potential directions that satisfy Condition (29) and for which sufficiently small step sizes will lead to true descent for the plant constraint. In the top case, a second lined area, corresponding to a hypothetical constraint with stronger nonlinear behavior, is also given, and it is clear how the same directions for this alternate case would require much smaller steps to guarantee true descent. The plant optimum, $\mathbf{u}^*$, is only given as a visual aid, and does not play any theoretical role here (as the focus is only on feasibility).

The Zero-Gradient Case

It is clear that the Condition (29) cannot possibly be satisfied if $\nabla g_{p,j}(\mathbf{u}_k) = \mathbf{0}$. Assumption A3 is used to avoid this case formally for a small enough choice of $\boldsymbol{\epsilon}$.

The Strictness of the Lipschitz Constants

Not surprisingly, the strictness of the Lipschitz constants κ is a *necessity*, as the absence of a strict linear bound (as in Lemma 1) allows the possibility of a constraint becoming active, with its value exactly at 0. This is reflected in the bound on K_k in (30), which approaches 0 as $\gamma_\kappa \to 1$.

We next discuss how these ideas may be enforced algorithmically.

3.4. Basic Implementation

In general, there is no RTO algorithm that satisfies Condition (29) implicitly. However, the target optimum $\mathbf{u}^*_{k+1}$ may be seen as a degree of freedom that could be manipulated to fulfill (29) – it is, in fact, the only degree of freedom, as the local gradient and current input are both fixed and cannot be modified. What we can do then is to project the optimum given by the RTO algorithm at iteration k in the appropriate manner[4]:

$$\begin{aligned} \bar{\mathbf{u}}^*_{k+1} = \arg \underset{\mathbf{u}}{\text{minimize}} \quad & \left\| \mathbf{u} - \mathbf{u}^*_{k+1} \right\|_2^2 \\ \text{subject to} \quad & \nabla g_{p,j}(\mathbf{u}_k)^T (\mathbf{u} - \mathbf{u}_k) \leq -\delta_{g,j} \\ & \forall j : g_{p,j}(\mathbf{u}_k) \geq -\epsilon_j \\ & \mathbf{u}^L \preceq \mathbf{u} \preceq \mathbf{u}^U \end{aligned} , \tag{33}$$

with $\bar{\mathbf{u}}^*_{k+1}$ then used in place of $\mathbf{u}^*_{k+1}$ in (20) and (21).

If (33) has a solution, then it is clear that Condition (29) will be satisfied by the resulting $\bar{\mathbf{u}}^*_{k+1}$. We now prove the feasibility of (33) for the majority of practical cases.

Theorem 4. (Feasibility of Projection (33))

Consider the Jacobian matrix, $\mathbf{J}_k \in \mathbb{R}^{n_u \times n_J}$, of a simplified constraint set of (33) that only consists of the gradients of the ϵ-active constraints and those box constraints that are active at iteration k ($i : u_{k,i} = u_i^L$ or $u_{k,i} = u_i^U$). Using $(\tilde{\cdot})$ to designate the inactive box constraints allows writing the constraint set of (33) as[5]*:*

[4]The norm in the projection may be chosen depending on user preference and any necessary scaling that may need to be taken into account (i.e. the inputs should be adjusted so that they are of comparable magnitudes and have comparable sensitivities in the objective norm function). For all of the examples that follow in this article, we use a squared 2-norm as the default for any projections, with the chosen inputs already well scaled.

[5]The (abuse of) notation $\tilde{\mathbf{u}}^L \preceq \mathbf{u} \preceq \tilde{\mathbf{u}}^U$ may be taken to mean generating the individual constraints of $\mathbf{u}^L \preceq \mathbf{u} \preceq \mathbf{u}^U$ and then removing those that are active at $\mathbf{u}_k$.

$$\begin{aligned} &\mathbf{J}_k^T(\mathbf{u}-\mathbf{u}_k) \preceq \begin{bmatrix} -\boldsymbol{\delta}_g \\ \mathbf{0} \end{bmatrix} \\ &\tilde{\mathbf{u}}^L \preceq \mathbf{u} \preceq \tilde{\mathbf{u}}^U \end{aligned}. \tag{34}$$

Let $\mathbf{J}_{k,1}, ..., \mathbf{J}_{k,n_J}$ represent the rows of $\mathbf{J}_k^T$.

If no rows of $\mathbf{J}_k$ are negatively spanned by other rows of $\mathbf{J}_k$:

$$\nexists a_2, ..., a_{n_J} \geq 0 : \mathbf{J}_{k,1} = -a_2\mathbf{J}_{k,2} - ... - a_{n_J}\mathbf{J}_{k,n_J}, \tag{35}$$

then there exists a vector $\boldsymbol{\delta}_g \succ \mathbf{0}$ for which Projection (33) has a solution.

Proof. We start by noting that the proof is trivial if there are no ϵ-active constraints (i.e. any point in $\mathcal{I}$ will be a feasible point).

For the general case with multiple ϵ-active constraints, we use Theorem 1 to state that the top set of inequalities in (34) is infeasible iff:

$$\exists \boldsymbol{\nu} \in \mathbb{R}_+^{n_J} : \sum_{i=1}^{n_J} \nu_i \mathbf{J}_{k,i} = \mathbf{0}, \quad \boldsymbol{\nu}^T[-\boldsymbol{\delta}_g \ \mathbf{0}] < 0, \tag{36}$$

which then implies:

$$\mathbf{J}_{k,1} = -\frac{\nu_2}{\nu_1}\mathbf{J}_{k,2} - ... - \frac{\nu_{n_J}}{\nu_1}\mathbf{J}_{k,n_J}, \tag{37}$$

i.e. that some row is negatively spanned by the others. It follows that the top set of inequalities of (34) is feasible unless there is a negative spanning.

Let $\bar{\mathbf{u}}_{k+1}^*$ represent a solution to this set, with:

$$\mathbf{J}_k^T(\bar{\mathbf{u}}_{k+1}^* - \mathbf{u}_k) \preceq \begin{bmatrix} -\boldsymbol{\delta}_g \\ \mathbf{0} \end{bmatrix}. \tag{38}$$

We must now show that:

$$\exists \bar{\mathbf{u}}_{k+1}^* : \tilde{\mathbf{u}}^L \preceq \bar{\mathbf{u}}_{k+1}^* \preceq \tilde{\mathbf{u}}^U, \tag{39}$$

or that some solution of (38) will hold for the inactive constraints as well. This may be restated as:

$$\tilde{\mathbf{u}}^L \preceq \mathbf{u}_k + (\bar{\mathbf{u}}_{k+1}^* - \mathbf{u}_k) \preceq \tilde{\mathbf{u}}^U. \tag{40}$$

Note, however, that scaling the right-hand side of (38) by a scalar $\alpha > 0$ allows for the constraints to be satisfied:

$$\mathbf{J}_k^T\alpha(\bar{\mathbf{u}}_{k+1}^* - \mathbf{u}_k) \preceq \begin{bmatrix} -\alpha\boldsymbol{\delta}_g \\ \mathbf{0} \end{bmatrix}, \tag{41}$$

for an arbitrarily small $\alpha(\bar{\mathbf{u}}_{k+1}^* - \mathbf{u}_k)$. This guarantees that a sufficiently small choice of α will allow:

$$\tilde{\mathbf{u}}^L \preceq \mathbf{u}_k + \alpha(\bar{\mathbf{u}}_{k+1}^* - \mathbf{u}_k) \preceq \tilde{\mathbf{u}}^U, \tag{42}$$

since $\tilde{\mathbf{u}}^L \prec \mathbf{u}_k \prec \tilde{\mathbf{u}}^U$, thereby completing the proof. □

All in all, this particular result should be intuitive, as it basically excludes the case where a descent direction for some of the constraints is an ascent direction for another. As the most basic example of a case where Projection (33) would be infeasible, consider two constraints, $g_{p,1}$ and $g_{p,2}$, with $\nabla g_{p,1}(\mathbf{u}_k) = -\nabla g_{p,2}(\mathbf{u}_k)$. This is a negative spanning and indeed matches the expected result – one cannot go in a local descent direction for one without increasing the other.

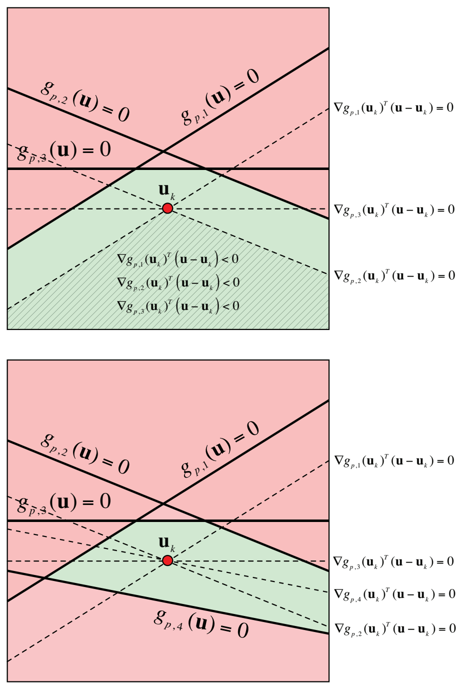
Figure 5 serves to illustrate this point further. In the top example, we have a simple linear case with three constraints. As their descent directions do not negatively span one another (we see, in fact, that $\nabla g_{p,1}(\mathbf{u}_k)$ and $\nabla g_{p,2}(\mathbf{u}_k)$ span $\nabla g_{p,3}(\mathbf{u}_k)$ *positively*), the projection problem is feasible (with the descent cone of all feasible projections denoted by the lined region). In the bottom case, an additional constraint, $g_{p,4}$, is added, whose descent direction lies opposite to that of the original three constraints and is spanned negatively by them. As should be clear from the figure, this renders the descent cone empty – there is no direction that will allow for all four constraints to be decreased simultaneously. The projection is therefore infeasible in this case.

The cases where a negative spanning occurs are believed to be of little practical concern, as they represent those cases where either: (a) the choice of $\boldsymbol{\epsilon}$ is too large, or (b) the RTO problem is ill-posed to begin with. Both can be illustrated by referring to the bottom case in Figure 5. For (a), we may assume that $\mathbf{u}_k$ is a point that is well within the feasible set, but, by an overly large choice of $\boldsymbol{\epsilon}$, we have told the projection to locally decrease all of the constraints at once. As the purpose of the projection is to avoid coming too close to the constraints that are becoming active, it is clear that such a choice is poor and unnecessary, as none of the constraints are close to active here. In this case, $\boldsymbol{\epsilon}$ should be reduced to make (33) feasible. For (b), suppose that $\boldsymbol{\epsilon} \approx \mathbf{0}$. If so, then the feasible space in Figure 5 is actually very, very small, with $\mathbf{u}_k$ being squeezed tightly between all four constraints, all of which are almost active – another way to visualize this is to imagine the bottom figure in Figure 5 as having a very large zoom factor. In this case, we are probably not interested in optimization anyway, since, as would be the case with satisfying unknown equality constraints, just remaining feasible is difficult enough. For these reasons, we state that Condition (29) may be satisfied for the vast majority of practical cases, due to (33) always having a solution in these cases provided a sufficiently low choice of $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}_g$.

The choice of projection parameters $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}_g$ deserves a few words. Perhaps the best way to present their respective roles would be by pointing out that the choice of $\boldsymbol{\epsilon}$ acts to identify the ϵ-active set (and thus the local descent cone $\{\mathbf{u} : \mathbf{J}_k^T(\mathbf{u} - \mathbf{u}_k) \prec \mathbf{0}\}$), while $\boldsymbol{\delta}_g$ decides just how deep into this cone one would like to step.

Choosing an $\boldsymbol{\epsilon}$ that is too large can have the following negative consequences:

- $\{\mathbf{u} : \mathbf{J}_k^T(\mathbf{u} - \mathbf{u}_k) \prec \mathbf{0}\} = \varnothing$, due to a negative spanning in $\mathbf{J}_k^T$.
- Enforcing the algorithm to stay too far away from the constraints (a local descent direction being enforced long before the constraint starts to become active).



$g_{p,2}(\mathbf{u})=0$
$g_{p,1}(\mathbf{u})=0$
$g_{p,3}(\mathbf{u})=0$
$\mathbf{u}_k$
$\nabla g_{p,1}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$\nabla g_{p,3}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$\nabla g_{p,2}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$\nabla g_{p,1}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)<0$
$\nabla g_{p,2}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)<0$
$\nabla g_{p,3}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)<0$
$g_{p,2}(\mathbf{u})=0$
$g_{p,1}(\mathbf{u})=0$
$g_{p,3}(\mathbf{u})=0$
$\mathbf{u}_k$
$\nabla g_{p,1}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$\nabla g_{p,3}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$\nabla g_{p,4}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$\nabla g_{p,2}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$g_{p,4}(\mathbf{u})=0$

Figure 5: An illustration of the cases where (33) is feasible (top) and infeasible (bottom) for a linear-constraint case. The lined region represents the space of descent directions for all of the constraints, but is empty in the bottom case due to a negative spanning between $g_{p,4}$ and the other three constraints.

On the contrary, choosing $\boldsymbol{\epsilon}$ to be too small will essentially cancel out the effect of Condition (29), as this condition will not take effect until the corresponding constraint is almost 0 (and K_k almost 0). Although K_k will still avoid approaching 0 asymptotically, it may become so small that the practical difference between its actual positive value and 0 will be negligible. In looser terms, this would mean "waiting too long" before enforcing a decrease in a given constraint value. The result in (30) reflects this analytically, where the lower bound on K_k clearly goes to 0 as $\boldsymbol{\epsilon} \to \mathbf{0}$.

For $\boldsymbol{\delta}_g$, we remark that an arbitrarily large choice has no effect on the existence of a solution for $\mathbf{J}_k^T(\mathbf{u} - \mathbf{u}_k) \preceq -\boldsymbol{\delta}_g$ alone, as this is linked purely to the non-negative spanning condition of the rows of $\mathbf{J}_k^T$. However, as demonstrated in the proof of Theorem 4, choosing $\boldsymbol{\delta}_g$ to be too large would be equivalent to scaling with a very large α, which may make satisfaction of the inactive box constraints impossible. For a small choice it is clear that the feasibility of (33) will be preserved, but, by contrast, performance issues may arise. As with $\boldsymbol{\epsilon}$, we see, from a quick analytical examination of (30), that the lower bound on K_k goes to 0 as $\boldsymbol{\delta}_g \to \mathbf{0}$, thereby allowing for very small steps and very slow progress.

A geometric illustration of the different choices of $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}_g$ for a linear-constraint case is given in Figure 6. We conclude by proposing that a moderate choice between the high and low extremes be used for both. However, as will be shown in Section 4, such a choice may be automated once optimality guarantees are included in the algorithm.

3.5. Illustrative Examples

The following RTO problem is considered:

$$\begin{array}{ll} \underset{u_1,u_2}{\text{maximize}} & u_2 \\ \text{subject to} & g_{p,1}(\mathbf{u}) = u_1^2 - 0.5u_1 + u_2 - 0.7 \leq 0 \\ & g_{p,2}(\mathbf{u}) = 2u_1^2 + 0.5u_1 + u_2 - 0.75 \leq 0 \\ & u_1 \in [-0.5, 0.5], u_2 \in [0, 0.8] \end{array}, \tag{43}$$

with an initial, feasible starting point of $\mathbf{u}_0 = [-0.4, 0.1]$. Here, we assume exact knowledge of the nonstrict Lipschitz matrix (the component-wise maximum of the absolute values in the Jacobian), $\tilde{\mathbf{K}}$, over the relevant space $\mathcal{I}$:

$$\tilde{\mathbf{K}} = \begin{bmatrix} 1.5 & 1 \\ 2.5 & 1 \end{bmatrix}, \tag{44}$$

which we then multiply by 1.1 to obtain the strict Lipschitz constants.

Three algorithms are applied:

Algorithm 1 – Fixed Target

By far the simplest, this "adaptation" involves providing $\mathbf{u}_{k+1}^*$ only once by selecting the target value of $[-0.2, 0.7]$ and keeping it there. Practically, this may correspond to a case where the initial input $\mathbf{u}_0$ has been applied for

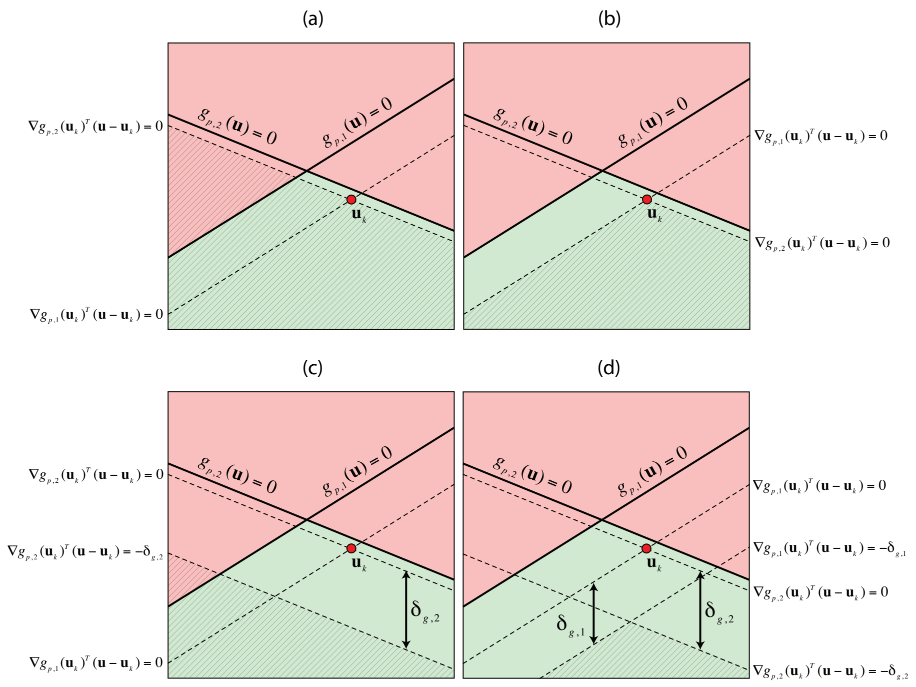

(a)
(b)
(c)
(d)
$g_{p,2}(\mathbf{u}) = 0$
$g_{p,1}(\mathbf{u}) = 0$
$\mathbf{u}_k$
$\nabla g_{p,2}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) = 0$
$\nabla g_{p,1}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) = 0$
$\nabla g_{p,2}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) = -\delta_{g,2}$
$\nabla g_{p,1}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) = -\delta_{g,1}$
$\delta_{g,1}$
$\delta_{g,2}$

Figure 6: An illustration of how the different choices of $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}_g$ affect the feasible space of Projection (33) (denoted by the lined region). (a) Small $\boldsymbol{\epsilon}$ and small $\boldsymbol{\delta}_g$ – only $g_{p,2}$ is deemed ϵ-active and only its local descent direction is enforced, (b) large $\boldsymbol{\epsilon}$ and small $\boldsymbol{\delta}_g$ – both $g_{p,1}$ and $g_{p,2}$ are deemed ϵ-active, (c) small $\boldsymbol{\epsilon}$ and large $\boldsymbol{\delta}_g$ – only $g_{p,2}$ is deemed ϵ-active and its local descent direction is enforced with a shift of $\delta_{g,2}$, (d) large $\boldsymbol{\epsilon}$ and large $\boldsymbol{\delta}_g$ – both $g_{p,1}$ and $g_{p,2}$ are deemed ϵ-active and their descent directions are enforced with shifts of $\delta_{g,1}$ and $\delta_{g,2}$. Note that the shifts $\boldsymbol{\delta}_g$ are plotted conceptually – they are not, in general, the exact distances by which a halfspace is shifted vertically/horizontally.

some time and has been deemed unsatisfactory, thus prompting a careful off-line analysis to choose a better operating point. However, since we do not trust to apply such an input change immediately, we would like to go there in steps, making sure that feasibility is retained throughout.

Algorithm 2 – Projected Gradient Descent with Diminishing Step

Here, we adapt by taking steps along the gradient of the objective function:

$$\mathbf{u}_{k+1}^* = \mathbf{u}_k - \frac{1}{k}\nabla\phi_p(\mathbf{u}_k), \tag{45}$$

where $\nabla\phi_p(\mathbf{u}_k) = [0 \ -1]^T$. A separate projection is applied following this step for the cases when $\mathbf{u}_{k+1}^*$ falls outside of $\mathcal{I}$.

Algorithm 3 – Initial Linear Model with Constraint Adaptation

A bit more involved, this algorithm employs a linear model of the constraints that is identified once in the vicinity of the initial point:

$$\begin{aligned} g_1(\mathbf{u}) &= -1.3u_1 + u_2 - 1.02 \\ g_2(\mathbf{u}) &= -1.1u_1 + u_2 - 1.39 \end{aligned} . \tag{46}$$

At each iteration, bias terms ε are defined as:

$$\varepsilon_{k,j} = g_{p,j}(\mathbf{u}_k) - g_j(\mathbf{u}_k), \tag{47}$$

and the adaptation is then performed by solving the model-based optimization with the bias correction terms added (Chachuat et al., 2008b):

$$\begin{aligned} \mathbf{u}_{k+1}^* = \arg \underset{u_1,u_2}{\text{maximize}} \quad & u_2 \\ \text{subject to} \quad & g_1(\mathbf{u}) + \varepsilon_{k,1} \leq 0 \\ & g_2(\mathbf{u}) + \varepsilon_{k,2} \leq 0 \\ & u_1 \in [-0.5, 0.5], u_2 \in [0, 0.8] \end{aligned} . \tag{48}$$

As nominal settings, we choose $\epsilon_1 = \epsilon_2 = 0.02$ and $\delta_{g,1} = \delta_{g,2} = 0.1$ for the projection.

We consider Algorithm 1 first, presenting results for the cases where no projection is done to keep a constraint from becoming active, and where this projection is applied using both the nominal and perturbed $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}_g$ values (Figure 7). Not enforcing Condition (29) leads to premature convergence, as the iterates go in a straight line towards the target point until the constraint is reached. When the projection is applied, the iterates approach the constraint but are then diverted to keep the constraint from becoming active, and the algorithm is able to reach (approximately) the closest feasible point to the target. In some sense, applying (29) allows us to reach a projection of the target onto the plant feasible space.

We see that varying the parameters $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}_g$ corresponds to the expectations outlined in the previous subsection. Using smaller $\boldsymbol{\delta}_g$ values clearly leads to weaker local descent – it is seen from comparing cases (b) and (c) that the backing off from an ϵ-active constraint is not as aggressive. Case (d) illustrates that choosing a larger $\boldsymbol{\epsilon}$ leads to the constraint being backed off from earlier, with the benefit of allowing larger iterate steps due to greater filter gains K_k. However, we see that such a choice also leads to a persistently large back-off that does not vanish.

The same cases are presented (in the same order) for Algorithms 2 and 3 in Figures 8 and 9, respectively. The same patterns as were noted for Algorithm 1 are seen here as well – not applying the projection leads to convergence as soon as a constraint is reached, while applying it leads to continuous movement between iterations.

3.6. Summary

So as to guarantee hard constraint satisfaction of an RTO scheme at all iterations, we have reviewed some of the standard approaches to satisfy uncertain constraints. As many of these are either difficult to apply or suffer from a lack of rigor, we chose to extend the theory of the adaptive input filter method (Bunin et al., 2011) as we believe this to be a more realizable alternative. We then proceeded to highlight the main algorithmic drawback of this approach, in that it leads to premature convergence whenever one constraint becomes active and approaches 0. So as to remedy this, we proposed a condition that would be sufficient to keep the algorithm moving by projecting the target optimum onto the strict local descent halfspace of any constraint that is approaching activity. We then provided an algorithmic way of guaranteeing this condition via a projection, realized by solving a quadratic programming problem, which we then showed to be feasible for a sufficiently small choice of projection parameters $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}_g$ for the vast majority of practical cases. The effects of varying these parameters were hypothesized and corroborated in simulation, where several RTO algorithms were tried and showed the importance of having the additional condition so as to avoid premature convergence.

In the next section, we go through a similar analysis for the cost function of the RTO problem, thereby allowing us to derive conditions that, when coupled with the feasibility conditions, form the full SCFO and may be used to enforce convergence to a KKT point of the RTO problem via strictly feasible iterates.

4. Optimality Guarantees and the Full SCFO

We start this section by first defining "optimality" and then reviewing what is available in terms of optimality guarantees for the different RTO algorithms in the literature. Noting that these guarantees are algorithm-specific and rarely account for feasibility guarantees (Section 3), we propose general sufficient conditions that are algorithm-independent and preserve feasibility. As was done in the previous section, we present the mathematical analysis first, stating the sufficient conditions and proving how their presence guarantees optimality, and then provide an algorithm for their enforcement. Conceptually, the approach is

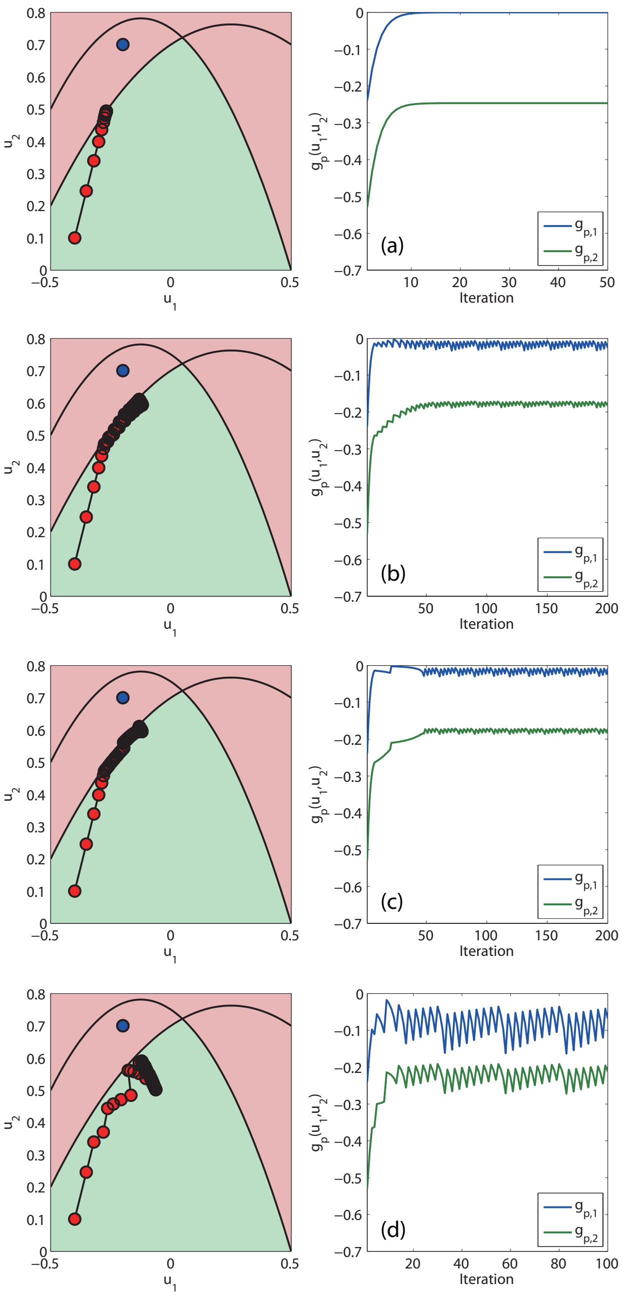


Figure 7: Feasible operation of Algorithm 1 with (a) no projection done, (b) projection with nominal $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}_g$ values, (c) projection with nominal $\boldsymbol{\epsilon}$ and $\delta_{g,1} = \delta_{g,2} = 0.01$, and (d) projection with $\epsilon_1 = \epsilon_2 = 0.1$ and nominal $\boldsymbol{\delta}_g$. The blue dot represents the target optimum.

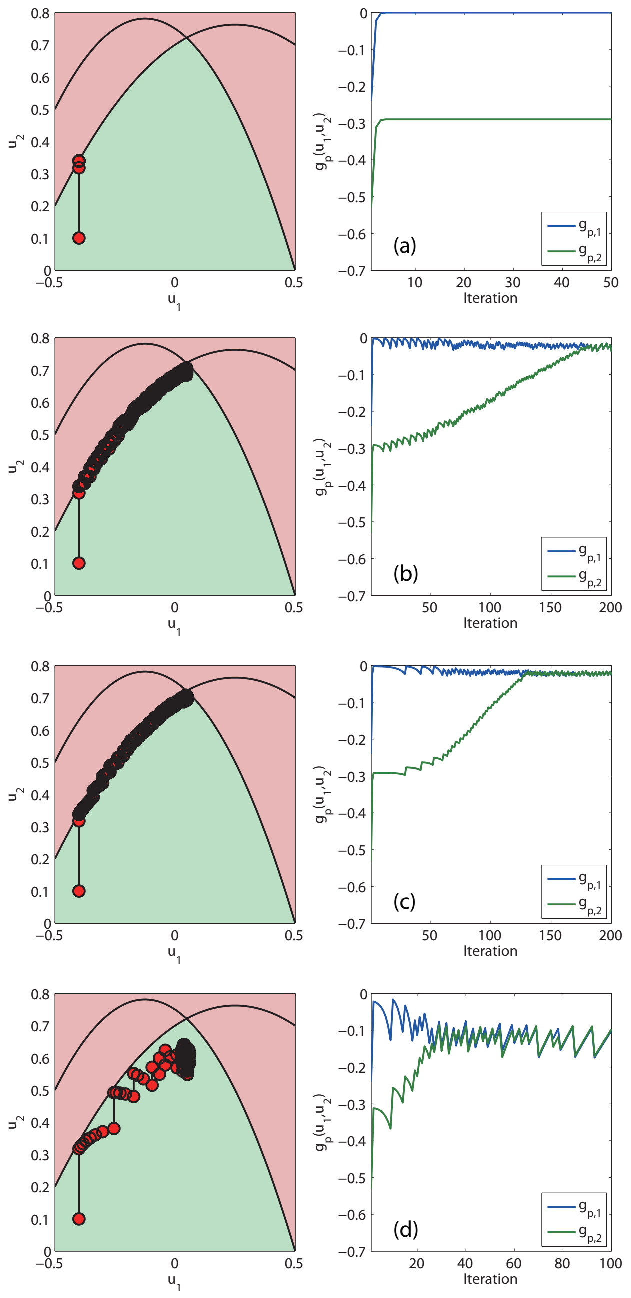


Figure 8: Feasible operation of Algorithm 2.

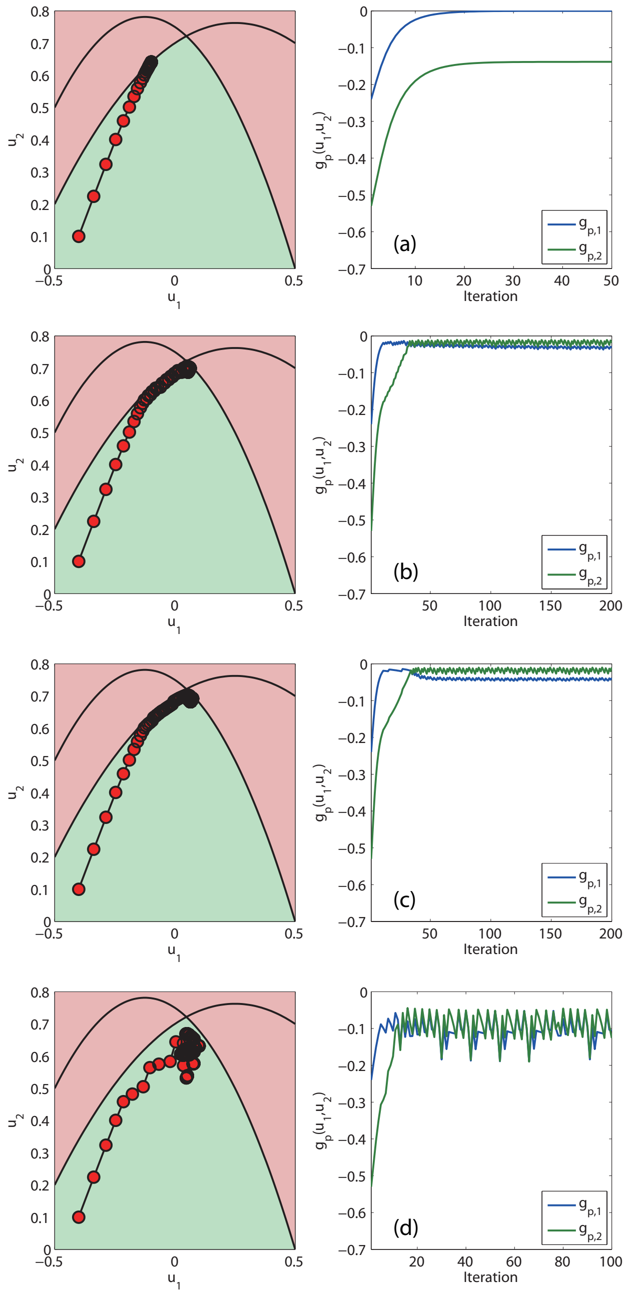


Figure 9: Feasible operation of Algorithm 3.

very similar to the one taken in the previous section – we simply enforce that the target optimum lie in the strict local descent halfspace of the objective function, and that the steps taken in this direction are small enough so as not to lose the descent (using, again, Lemma 3). We then finish by presenting a few examples to demonstrate the effectiveness of the theory.

4.1. KKT Convergence and Existing Guarantees

By "optimality", we simply mean KKT convergence, in that the algorithm always converges to a KKT point, $\mathbf{u}^*$, satisfying the necessary, first-order optimality conditions:

$$\begin{array}{l} \mathbf{G}_p(\mathbf{u}^*) \preceq \mathbf{0},\ \mathbf{u}^L \preceq \mathbf{u}^* \preceq \mathbf{u}^U \\ \mu_j g_{p,j}(\mathbf{u}^*) = 0,\ \zeta_i^L(u_i^L - u_i^*) = 0,\ \zeta_i^U(u_i^* - u_i^U) = 0 \\ \forall j = 1, ..., n_g, \forall i = 1, ..., n_u \\ \nabla \mathcal{L}(\mathbf{u}^*) = \nabla \phi_p(\mathbf{u}^*) + \displaystyle\sum_{j=1}^{n_g} \mu_j \nabla g_{p,j}(\mathbf{u}^*) - \boldsymbol{\zeta}^L + \boldsymbol{\zeta}^U = \mathbf{0} \end{array}, \tag{49}$$

with $\boldsymbol{\mu} \in \mathbb{R}_+^{n_g}$ and $\boldsymbol{\zeta}^L, \boldsymbol{\zeta}^U \in \mathbb{R}_+^{n_u}$ vectors of Lagrange multipliers corresponding to the uncertain and box constraints, respectively, and $\mathcal{L}(\mathbf{u}) : \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_u}$ the Lagrangian function that defines the KKT stationarity condition.

While this is not *sufficient* to guarantee that the KKT point is a local minimum (a second-order condition is also required), we limit ourselves to this definition as the KKT points that are not local minima (e.g. local maxima and saddle points) will not be algorithmically stable in any descent method. As the conditions proposed in this work enforce a descent method, we will consider (49) to be "practically sufficient" to guarantee local minimality, even though we are unable to guarantee this rigorously – while conceptually possible, such guarantees would require much stronger assumptions (knowledge of the plant Hessian) that may be extremely difficult to uphold in practice.

In considering different RTO algorithms, one usually sees a clear distinction between model-based and model-free methods with regard to KKT convergence – the conditions for the latter are usually much simpler and easier to realize (though the algorithms are slower and require more iterations), while the former, despite generally faster convergence, have conditions that require difficult assumptions on the model used and the potential plant-model error. The guarantees available for the different approaches, as well as the price to pay for such guarantees, are summarized as follows:

- The two-step parameter-identification-and-optimization approach, which is the natural and standard approach to optimize many processes, has very weak guarantees for converging to an actual KKT point of the plant (Brdys & Tatjewski, 2005; Forbes et al., 1994; Marchetti, 2009; Quelhas et al., 2013). In the general case, such convergence may only be guaranteed if there exists a set of parameters for which the model has the same KKT point as the plant (Biegler et al., 1985) and if such a set is identified by the parameter estimation method. When this is not the case, it is generally accepted that the converged point of the model optimization will not be a plant KKT point. In the rare case when structural uncertainty is not present and all of the uncertainty is described by the parameters, the convergence properties of the two-step approach are very simple – it is sufficient to identify the correct parameters, to optimize the model once, and to apply the result to the plant.

- Methods that employ measurements to correct the 0^{th}- and 1^{st}-order error of the model, known as *ISOPE* (Brdys & Tatjewski, 2005) or *modifier adaptation* (Marchetti et al., 2009), are guaranteed to be at a KKT point *upon convergence*, which is an attractive property but nevertheless requires that the scheme converge. Local necessary conditions for convergence have been detailed (Marchetti et al., 2009), but are difficult to implement as they require the knowledge of the first and second derivatives of the plant at the KKT point. Sufficient conditions for the case without uncertain inequality constraints are also available (Brdys & Tatjewski, 2005, Th. 4.1), but depend on quantities that may not be easy to compute (i.e. the global minimal eigenvalue of the model Hessian and its relation to the quadratic upper bound of the plant) and assume convexity in the constraints that are known. Sufficient conditions for the case with both uncertain cost and constraints have been proposed (Chachuat et al., 2008a), but are largely abstract and require multiple quadratic upper bounds that are linked to the plant-model error, as well as certain continuity assumptions on the model.

- Model-free methods such as the simplex, direct search, and gradient descent come with simple guarantees of KKT convergence that rely almost entirely on some sort of line search (Conn et al., 2009; Fletcher, 1987). These are easy to implement as they need only continued experimentation and sufficiently small steps, but have the obvious drawback of inefficiency and slow convergence as they do not use *a priori* knowledge.

It is important to note that none of these guarantees, to the authors' best knowledge, take hard constraints into account. As such, while guaranteeing convergence to a KKT point may be possible using the available theory for a specific algorithm, doing so in the presence of hard constraints is not something that has been addressed.

4.2. General Sufficient Conditions for Feasibility and Optimality

We begin by proposing a set of sufficient conditions for monotonic cost decrease in a general RTO algorithm.

Lemma 4. (Minimal Cost Decrease Between Iterations)

Let the following two conditions be satisfied at every iteration of the RTO algorithm:

$$
\begin{aligned}
&\nabla \phi_p(\mathbf{u}_k)^T(\mathbf{u}_{k+1}^* - \mathbf{u}_k) \leq -\delta_\phi \\
&0 < K_{min} \leq K_k \leq \\
&\qquad -2\frac{\nabla \phi_p(\mathbf{u}_k)^T(\mathbf{u}_{k+1}^* - \mathbf{u}_k)}{(\mathbf{u}_{k+1}^* - \mathbf{u}_k)^T \overline{\mathbf{Q}}_\phi (\mathbf{u}_{k+1}^* - \mathbf{u}_k)} - K_{min}
\end{aligned} , \tag{50}
$$

with $\delta_\phi > 0$, $\overline{\mathbf{Q}}_\phi \succ 0$ the quadratic upper bound of $\phi_p(\mathbf{u})$ as defined in Lemma 2, and $K_{min} > 0$ some minimal value achieved by the adaptive filter gain K_k.[6]

Then, the change in the cost from iteration to iteration will be bounded as:

$$
\begin{aligned}
&\phi_p(\mathbf{u}_{k+1}) - \phi_p(\mathbf{u}_k) \leq -K_{min}\,\delta_\phi + \\
&\qquad \frac{K_{min}^2}{2}(\mathbf{u}^U - \mathbf{u}^L)^T \overline{\mathbf{Q}}_\phi (\mathbf{u}^U - \mathbf{u}^L) < 0
\end{aligned} . \tag{51}
$$

Proof. Using Lemma 3, we know that the worst-case evolution (13), being strictly convex with respect to K_k, achieves its maximum at either $K_k = K_{min}$ or $K_k = -2\frac{\nabla \phi_p(\mathbf{u}_k)^T(\mathbf{u}_{k+1}^* - \mathbf{u}_k)}{(\mathbf{u}_{k+1}^* - \mathbf{u}_k)^T \overline{\mathbf{Q}}_\phi (\mathbf{u}_{k+1}^* - \mathbf{u}_k)} - K_{min}$ and has a value strictly less than 0.

Substituting either of these bounds into the worst-case evolution expression (13) for $\phi_p(\mathbf{u}_{k+1}) - \phi_p(\mathbf{u}_k)$ yields:

$$
\begin{aligned}
&\phi_p(\mathbf{u}_{k+1}) - \phi_p(\mathbf{u}_k) \leq K_{min} \nabla \phi_p(\mathbf{u}_k)^T(\mathbf{u}_{k+1}^* - \mathbf{u}_k) + \\
&\qquad \frac{K_{min}^2}{2}(\mathbf{u}_{k+1}^* - \mathbf{u}_k)^T \overline{\mathbf{Q}}_\phi (\mathbf{u}_{k+1}^* - \mathbf{u}_k)
\end{aligned} , \tag{52}
$$

which, with the worst-case values of the linear and quadratic terms, leads to the bound in (51). □

Conditions (50) are useful in that they give a guarantee of a minimal cost decrease between two consecutive iterations. As the cost is, by assumption, continuous over a compact domain and thereby bounded, it follows that meeting such conditions indefinitely is not possible. The following gives a global upper bound on the maximum number of iterations for which these conditions may be met.

Corollary 1. (Upper Bound on Number of Cost-Decreasing, Feasible Iterations)

Let $\phi_{p,min} = \min(\phi_p(\mathbf{u}) : \mathbf{u} \in \mathcal{I}, \mathbf{G}_p(\mathbf{u}) \preceq \mathbf{0})$ and let the conditions of Lemma 4 hold whenever possible for some fixed $\delta_\phi > 0$, with all iterates satisfying the hard inequality and box constraints. The number of iterations for which the conditions of Lemma 4 are satisfied cannot exceed:

$$
\frac{\phi_p(\mathbf{u}_0) - \phi_{p,min}}{K_{min}\,\delta_\phi - \frac{K_{min}^2}{2}(\mathbf{u}^U - \mathbf{u}^L)^T \overline{\mathbf{Q}}_\phi (\mathbf{u}^U - \mathbf{u}^L)} < \infty. \tag{53}
$$

[6]For (50) to be true, it is implicit that $K_{min} \leq -\frac{\nabla \phi_p(\mathbf{u}_k)^T(\mathbf{u}_{k+1}^* - \mathbf{u}_k)}{(\mathbf{u}_{k+1}^* - \mathbf{u}_k)^T \overline{\mathbf{Q}}_\phi (\mathbf{u}_{k+1}^* - \mathbf{u}_k)}$.

PROOF. This results directly from Lemma 4. As $\phi_p(\mathbf{u}_0) - \phi_{p,min}$ represents the greatest possible suboptimality gap between the initial point and the global minimum, the validity of Conditions (50) for more than this number of iterations would reduce the cost below its global minimum, which is not possible without losing feasibility. □

We now combine all of the results obtained so far to give the full SCFO.

Theorem 5. (Sufficient Conditions for Feasible-Side Convergence to a KKT Point)

Let an RTO algorithm satisfy Conditions (21), (29), and (50) for all iterations when it is possible to do so for some fixed choice of $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, $\delta_\phi \succ \mathbf{0}$, and yield $\mathbf{u}_{k+1} = \mathbf{u}_k$ otherwise. Defining the KKT error $\mathcal{E}$ as the minimal sum of squared errors in the stationarity and complementary slackness conditions of (49):

$$\begin{aligned}\mathcal{E}(\mathbf{u}) = \inf_{\boldsymbol{\mu},\boldsymbol{\zeta}^L,\boldsymbol{\zeta}^U \succeq \mathbf{0}} \Big(\nabla \mathcal{L}(\mathbf{u})^T \nabla \mathcal{L}(\mathbf{u}) + \sum_{j=1}^{n_g} \left[\mu_j g_{p,j}(\mathbf{u}) \right]^2 + \\ \sum_{i=1}^{n_u} \left[(\zeta_i^L (u_i^L - u_i))^2 + (\zeta_i^U (u_i - u_i^U))^2 \right] \Big)\end{aligned}, \tag{54}$$

it follows that such an algorithm will:

(i) Converge to a static point, $\mathbf{u}_\infty$, in a finite number of iterations.

(ii) Satisfy the plant constraints at every iteration.

(iii) Decrease the cost at every iteration until $\mathbf{u}_\infty$.

(iv) Have the KKT error at $\mathbf{u}_\infty$ go to 0 in the limit with respect to $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ:

$$\lim_{\boldsymbol{\epsilon},\boldsymbol{\delta}_g,\delta_\phi \to \mathbf{0}} \mathcal{E}(\mathbf{u}_\infty) = 0. \tag{55}$$

PROOF. We refer the reader to the appendix. □

Conditions (21), (29), and (50), together with $\mathbf{u}_{k+1}^* \in \mathcal{I}$, constitute the full SCFO – guaranteeing finite-time convergence to a stationary point with arbitrarily small KKT error as $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ are made arbitrarily small. We note that we cannot decouple optimality from feasibility, as we must have the latter to guarantee the former. However, it may be shown that convergence via infeasible iterates is also possible via an additional implementation technique – this will be addressed in the companion work (Bunin et al., 2013b).

We give a geometrical illustration of the combined effect of Conditions (29) and (50) in Figure 10 (as an extension of Figure 4), which allows for a much simpler interpretation – by enforcing that the RTO algorithm always move in a locally cost-decreasing and feasible direction, it is natural that it converge when there no longer exists such a direction to move in (i.e. to the geometric definition of a KKT point).

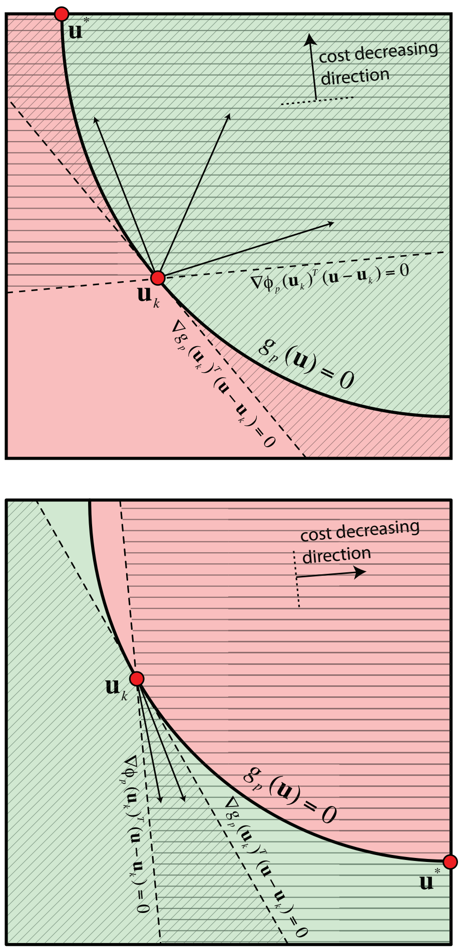

$\mathbf{u}^*$
cost decreasing direction
$\nabla\phi_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$\mathbf{u}_k$
$\nabla g_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$g_p(\mathbf{u})=0$
cost decreasing direction
$\mathbf{u}_k$
$\nabla\phi_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$g_p(\mathbf{u})=0$
$\nabla g_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k)=0$
$\mathbf{u}^*$

Figure 10: The geometric interpretation of the SCFO for the earlier examples of Figure 4 with a linear cost included. Here, the diagonally-lined regions represent the local descent halfspaces for the constraints, while the horizontally-lined regions represent the local descent halfspaces for the cost. The double-lined regions (cones) are therefore those that satisfy both Conditions (29) and (50), with the arrows showing potential directions that may then be taken by the RTO algorithm in accordance to the SCFO. It is not difficult to visualize that such a cone will always exist unless the current iterate $\mathbf{u}_k$ is already a KKT point.

4.3. Basic Implementation

We propose a similar implementation as we did for the feasibility-only case, and perform the following projection:

$$
\begin{array}{rll}
\bar{\mathbf{u}}_{k+1}^* = \arg \underset{\mathbf{u}}{\text{minimize}} & \left\|\mathbf{u} - \mathbf{u}_{k+1}^*\right\|_2^2 & \\
\text{subject to} & \nabla g_{p,j}(\mathbf{u}_k)^T(\mathbf{u} - \mathbf{u}_k) \leq -\delta_{g,j} & \\
& \forall j : g_{p,j}(\mathbf{u}_k) \geq -\epsilon_j & , \\
& \nabla \phi_p(\mathbf{u}_k)^T(\mathbf{u} - \mathbf{u}_k) \leq -\delta_\phi & \\
& \mathbf{u}^L \preceq \mathbf{u} \preceq \mathbf{u}^U &
\end{array}
\tag{56}
$$

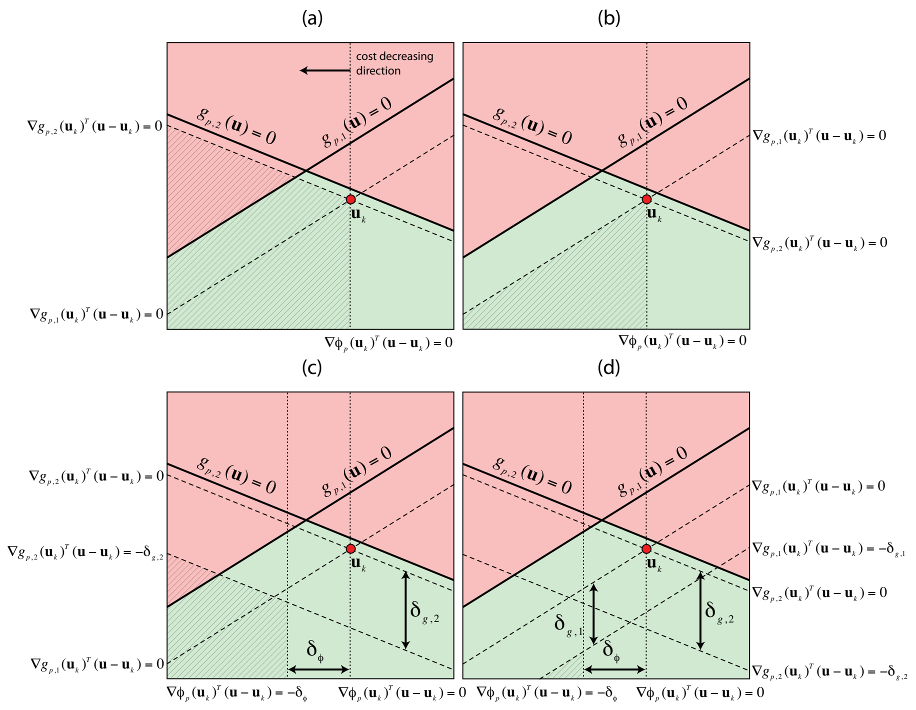
followed by applying (20) with respect to $\bar{\mathbf{u}}_{k+1}^*$, with the filter gain defined as:

$$
\begin{array}{l}
K_k := \min \left\{ \underset{j=1,\ldots,n_g}{\min} \left[\dfrac{-g_{p,j}(\mathbf{u}_k)}{\displaystyle\sum_{i=1}^{n_u} \kappa_{ji} |\bar{u}_{k+1,i}^* - u_{k,i}|} \right], \right. \\
\qquad\qquad \left. -1.99 \dfrac{\nabla \phi_p(\mathbf{u}_k)^T(\bar{\mathbf{u}}_{k+1}^* - \mathbf{u}_k)}{(\bar{\mathbf{u}}_{k+1}^* - \mathbf{u}_k)^T \overline{\mathbf{Q}}_\phi (\bar{\mathbf{u}}_{k+1}^* - \mathbf{u}_k)} \right\} \quad , \\
K_k > 1 \rightarrow K_k := 1
\end{array}
\tag{57}
$$

where we use 1.99 instead of 2 as an approximation of the inequality in (50).

As shown in Theorem 5, Problem (56) will have a solution for $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ sufficiently small unless $\mathbf{u}_k$ is already a KKT point. As might be expected, and as was already shown in both Figure 6 and the illustrative example in Section 3, the exact way in which the projection is done (i.e. the values of $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ) will influence the performance of the algorithm – we give an extension of the previous geometric illustration, adding a linear cost, in Figure 11. It is not clear what choices would lead to the best performance, but the following general trends may be expected and have been observed:

- Small constant values of $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ approximate the true SCFO well, i.e. the ϵ-active constraints are close to the true active constraints, and Conditions (29) and (50) are close to the strict local descent halfspace conditions (the shifts seen in Cases (c) and (d) of Figure 11 are negligible). This leads to "small" KKT error upon convergence as characterized analytically in Theorem 5. However, the conditions may be enforced "too late", with a constraint becoming close to active, and the amount of local descent may be small. This may result in very small values for K_k and therefore small steps, causing the algorithm to progress very slowly, especially when close to a constraint. The number of worst-case iterations as given by (53) also grows accordingly, and the cost improvement guaranteed at each step, via Lemma 4, lessens.



(a)
(b)
(c)
(d)
cost decreasing direction
$g_{p,2}(\mathbf{u}) = 0$
$g_{p,1}(\mathbf{u}) = 0$
$\mathbf{u}_k$
$\nabla g_{p,2}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) = 0$
$\nabla g_{p,1}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) = 0$
$\nabla \phi_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) = 0$
$\nabla g_{p,2}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) = -\delta_{g,2}$
$\nabla g_{p,1}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) = -\delta_{g,1}$
$\nabla \phi_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) = -\delta_\phi$
$\delta_{g,2}$
$\delta_{g,1}$
δ_ϕ

Figure 11: An illustration of how the different choices of $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ affect the feasible space of Projection (56) (shown via the lined regions). (a) Small $\boldsymbol{\epsilon}$ and small $\boldsymbol{\delta}_g$, δ_ϕ, (b) large $\boldsymbol{\epsilon}$ and small $\boldsymbol{\delta}_g$, δ_ϕ, (c) small $\boldsymbol{\epsilon}$ and large $\boldsymbol{\delta}_g$, δ_ϕ, (d) large $\boldsymbol{\epsilon}$ and large $\boldsymbol{\delta}_g$, δ_ϕ. Note that the shifts $\boldsymbol{\delta}_g$ and δ_ϕ are plotted conceptually – they are not, in general, the exact distances by which a halfspace is shifted vertically/horizontally.

- Large constant values of $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ have the opposite effect. A greater ϵ-active set is chosen due to larger $\boldsymbol{\epsilon}$ and thus the algorithm tries to stay away from more constraints and tries to stay away from them earlier, long before they become close to active. The amount of local descent in both the cost and ϵ-active constraints is also greater, and so it should be expected that the algorithm will aim for larger steps (this is seen clearly in Case (d) of Figure 11). However, the SCFO are poorly approximated and the worst-case KKT error upon convergence may be large. Clearly, large $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ all increase the chance of the conditions being impossible to satisfy – for example, were $\boldsymbol{\delta}_g$ and δ_ϕ increased further in Case (d) of Figure 11, the projection would clearly become infeasible as the small triangular feasible space would simply vanish.

As such, a sound approach would be to use larger values of $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ if possible, as this appears to lead to faster progress, and to use smaller values otherwise to enforce true KKT convergence, which is guaranteed only as $\boldsymbol{\epsilon}, \boldsymbol{\delta}_g, \delta_\phi \rightarrow \mathbf{0}$. To this end, we propose the following method to auto-select and auto-tune these parameters:

Initialization – Done Prior to RTO

1. The degrees of the different constraints' activity, as well as the amount of local descent in the constraints and cost, should be on a comparable scale. A simple way to ensure this is by scaling the constraints and cost with respect to their ranges (e.g. if $\min\limits_{\mathbf{u} \in \mathcal{I}} g_p(\mathbf{u}) = -100$, the scaled constraint may be re-defined as $g_p(\mathbf{u}) := 0.01 g_p(\mathbf{u})$).

   Setting upper and lower limits on the projection parameters, let $\overline{\boldsymbol{\epsilon}} = \overline{\boldsymbol{\delta}}_g = \mathbf{1}$, $\overline{\delta}_\phi = 1$, and choose $\underline{\boldsymbol{\epsilon}}$, $\underline{\boldsymbol{\delta}}_g$, and $\underline{\delta}_\phi$ sufficiently small (e.g. 10^{-6}) so that the approximation error of the active set by the $\underline{\epsilon}$-active set and of the strict local descent conditions by the nonstrict inequality conditions, with $-\underline{\boldsymbol{\delta}}_g$ and $-\underline{\delta}_\phi$ on the right-hand sides, is negligible.

Search for a Feasible Projection – Before Each RTO Iteration

2. Set $\boldsymbol{\epsilon} := \overline{\boldsymbol{\epsilon}}$, $\boldsymbol{\delta}_g := \overline{\boldsymbol{\delta}}_g$, and $\delta_\phi := \overline{\delta}_\phi$.
3. Check the feasibility of (56) for the given choice of $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ [7]:

[7]As the arguably best way to verify the feasibility of (56) is to solve a linear programming optimization problem, we write it in this form, with the "minimization of 0" used to denote that no optimization actually takes place.

$$\begin{array}{rl} \underset{\mathbf{u}}{\text{minimize}} & 0 \\ \text{subject to} & \nabla g_{p,j}(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) \leq -\delta_{g,j} \\ & \forall j : g_{p,j}(\mathbf{u}_k) \geq -\epsilon_j \\ & \nabla \phi_p(\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k) \leq -\delta_\phi \\ & \mathbf{u}^L \preceq \mathbf{u} \preceq \mathbf{u}^U \end{array} . \tag{58}$$

If (58) does not have a solution, set $\boldsymbol{\epsilon} := 0.5\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g := 0.5\boldsymbol{\delta}_g$, $\delta_\phi := 0.5\delta_\phi$, and attempt to re-solve (58). Otherwise, proceed to Step 4.

4. If $\boldsymbol{\epsilon} \succeq \underline{\boldsymbol{\epsilon}}$, $\boldsymbol{\delta}_g \succeq \underline{\boldsymbol{\delta}}_g$, or $\delta_\phi \geq \underline{\delta}_\phi$, solve (56) with $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ. Else, terminate (Step 5).

Termination – Declared Convergence to KKT Point

5. If $\boldsymbol{\epsilon} \prec \underline{\boldsymbol{\epsilon}}$, $\boldsymbol{\delta}_g \prec \underline{\boldsymbol{\delta}}_g$, and $\delta_\phi < \underline{\delta}_\phi$, terminate, with $\bar{\mathbf{u}}^*_{k+1} := \mathbf{u}_k$.

The basic philosophy of this method is in using larger values of $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ to both stay away from the constraints (large $\boldsymbol{\epsilon}$ and $\boldsymbol{\delta}_g$) and to significantly decrease the cost (large δ_ϕ) *when possible*. Once this becomes impossible due to feasibility issues in (56), these parameters are lowered until (56) becomes feasible. From Theorem 5, it is clear that infeasibility that persists even when $\boldsymbol{\epsilon} \prec \underline{\boldsymbol{\epsilon}}$, $\boldsymbol{\delta}_g \prec \underline{\boldsymbol{\delta}}_g$, and $\delta_\phi < \underline{\delta}_\phi$ implies that convergence to a KKT point has been achieved with very good accuracy, provided that $\underline{\boldsymbol{\epsilon}}$, $\underline{\boldsymbol{\delta}}_g$, and $\underline{\delta}_\phi$ are not too large.

We finish by remarking that the update laws of dividing $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ by 2 in Step 3 are purely heuristic. One could propose other, perhaps better performing, rules, but such an optimization of the reduction law for $\boldsymbol{\epsilon}$, $\boldsymbol{\delta}_g$, and δ_ϕ (itself a potential RTO problem) is outside the scope of the present work. As will be seen in the next subsection, the proposed law gives satisfactory results for the cases studied here.

4.4. Illustrative Examples

Consider the following RTO problem with one convex and two concave uncertain constraints:

$$\begin{array}{rl} \underset{u_1,u_2}{\text{minimize}} & \phi_p(\mathbf{u}) = (u_1-0.5)^2 + (u_2-0.4)^2 \\ \text{subject to} & g_{p,1}(\mathbf{u}) = -6u_1^2 - 3.5u_1 + u_2 - 0.6 \leq 0 \\ & g_{p,2}(\mathbf{u}) = 2u_1^2 + 0.5u_1 + u_2 - 0.75 \leq 0 \\ & g_{p,3}(\mathbf{u}) = -u_1^2 - (u_2-0.15)^2 + 0.01 \leq 0 \\ & u_1 \in [-0.5, 0.5], u_2 \in [0, 0.8] \end{array} , \tag{59}$$

for which the matrix of Lipschitz constants is defined as:

$$\mathbf{K} = 1.1 \begin{bmatrix} 9.5 & 1 \\ 2.5 & 1 \\ 1 & 1.3 \end{bmatrix}, \tag{60}$$

and $\overline{\mathbf{Q}}_\phi$ chosen as the Hessian of the (uncertain) cost function:

$$\overline{\mathbf{Q}}_\phi = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}. \tag{61}$$

A scaling of 4 : 2 : 1 : 1.5 is chosen for $g_{p,1} : g_{p,2} : g_{p,3} : \phi_p$, while $\overline{\boldsymbol{\epsilon}}$, $\overline{\boldsymbol{\delta}}_g$, and $\overline{\delta}_\phi$ are set to unity, and an approximation tolerance of 10^{-8} is used for $\underline{\boldsymbol{\epsilon}}$, $\underline{\boldsymbol{\delta}}_g$, and $\underline{\delta}_\phi$.

We note that the concave constraints serve a dual purpose. The first is the general role of uncertain hard constraints, while the second has more to do with their topological properties – it is particularly difficult for an algorithm to remain feasible and avoid premature convergence while navigating around constraints with concave properties (see, e.g., the example in Figure 3). Here, we construct a rather nasty case so as to highlight the benefits that enforcing the SCFO may bring to an algorithm – this corresponds to the cases where the algorithms are initialized from $\mathbf{u}_0 = [-0.5, 0.05]$. To be more – or, perhaps, less – realistic, we also consider a nicer case with the initial point $\mathbf{u}_0 = [0, 0.4]$ where these constraints do not really play a role. We also note that their presence does not change the fact that this problem only has a single stable KKT point at $\mathbf{u}^* = [0.35, 0.32]$, which is, in this case, the global minimum. An unstable KKT point at $\mathbf{u} = [-0.09, 0.11]$ is also present, however.

As all three constraints are defined as hard constraints, we apply the filter gain criterion (21) so as to ensure that the constraints are never violated. We also look at the cases where only Condition (29) is applied, as was done in Section 3. For these schemes, we do not auto-tune $\underline{\boldsymbol{\epsilon}}$ and $\underline{\boldsymbol{\delta}}_g$, simply setting them to $\underline{\boldsymbol{\epsilon}} := 0.01\overline{\boldsymbol{\epsilon}}$ and $\underline{\boldsymbol{\delta}}_g := 0.01\overline{\boldsymbol{\delta}}_g$.

Five different RTO algorithms are considered, of which only two are reported here in the interest of space, with the results for the remaining three relegated to the Supplementary Material. We describe the two algorithms reported here below:

Algorithm 1 – Two-Step Approach

A model of the plant, with a set of uncertain parameters $\boldsymbol{\theta}$, is available:

$$\begin{array}{l} \phi(\mathbf{u},\boldsymbol{\theta}) = \theta_1(u_1-0.3)^2 + \theta_2(u_2-0.3)^2 \\ g_1(\mathbf{u},\boldsymbol{\theta}) = -\theta_3 u_1^2 - 3.5u_1 + u_2^2 - 0.6 \leq 0 \\ g_2(\mathbf{u},\boldsymbol{\theta}) = \theta_4 u_1 + u_2 + \theta_5 \leq 0 \\ g_3(\mathbf{u},\boldsymbol{\theta}) = -u_1^2 - (u_2-0.15)^2 + 0.01 \leq 0 \end{array}, \tag{62}$$

where we allow the third constraint to be modeled perfectly.

The parameters are re-estimated at each iteration (via linear regression) and the updated model is optimized numerically to compute $\mathbf{u}^*_{k+1}$. This represents a relatively good model-based RTO algorithm that nevertheless fails to achieve KKT convergence due to structural mismatch between the parametric model and the plant.

Algorithm 2 – Random Step

Here, we draw the $\mathbf{u}_{k+1}^*$ out of a hat, using a uniform random number generator to provide an input such that $u_{k+1,1}^* \sim \mathcal{U}[-0.5, 0.5], u_{k+1,2}^* \sim \mathcal{U}[0, 0.8]$. While highly impractical and not at all advised, we use this method to underline the usefulness that enforcing the SCFO has, even when the RTO algorithm is completely haphazard.

Figures 12-13 provide the results. Leaving the algorithm-related remarks to the figure captions, we proceed to outline the noticeable general trends below:

- Feasibility is indeed preserved due to (21) for all cases.
- Premature convergence to one of the concave constraints is noted for all algorithms unless Condition (29) is enforced, in which case the iterates are able to "slide" around these obstacles.
- Enforcing the SCFO leads to feasible convergence to the optimum in every case. This holds even when the algorithm is completely haphazard (Algorithm 2).
- The SCFO always avoid the unstable KKT point at $\mathbf{u} = [-0.09, 0.11]$.
- Enforcing Condition (29) alone proves sufficient for certain algorithms (See Algorithms A1, A2, and A3 in the Supplementary Material) to reach a region very close to the optimum. However, the lack of an auto-tuning scheme for the projection parameters keeps these realizations oscillating with an offset.
- The steps are significantly smaller and progress significantly slower when the algorithm is close to a constraint. This is due to the numerator of Condition (21).
- Enforcing the SCFO for cases where the algorithm converges without them does affect performance slightly, in that more care is given with respect to the constraints, which affects the convergence trajectory (see examples in the Supplementary Material).
- The choice of algorithm does not appear to be crucial when the SCFO are enforced, but does affect performance. For the case with the second initial point, for example, one may see that Algorithm 2 has reduced performance when compared to the others, due to its somewhat erratic trajectory.
- Enforcing Condition (50) does indeed lead to monotonic improvement in the cost function.

4.5. Summary

The problem of convergence to a KKT optimum has been defined and the extents to which various RTO algorithm classes could guarantee this property have been reviewed, with the SCFO being proposed as a general set of conditions

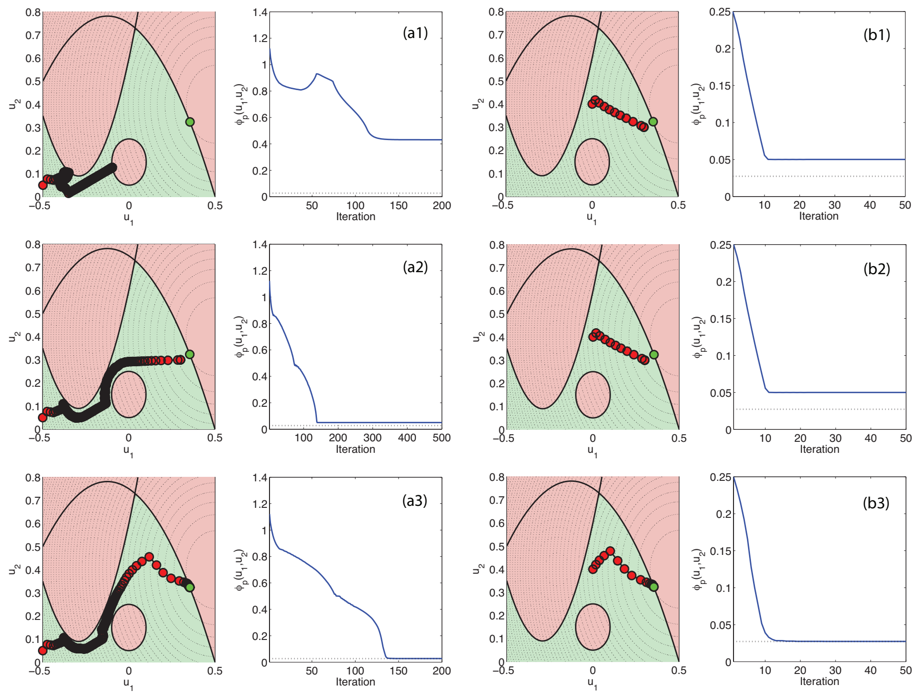


Figure 12: Performance of Algorithm 1 (two step) for the cases where (1) neither Conditions (29) nor (50) are enforced, (2) only Condition (29) is enforced, (3) the full SCFO are enforced, for (a) the first initial point and (b) the second initial point. Contours of the cost are given in black, with the optimal plant cost given by the dotted black lines in the right-hand figures. The plant optimum is plotted in green. We note that the SCFO are needed to enforce convergence to the optimum due to structural errors in the parametric model.

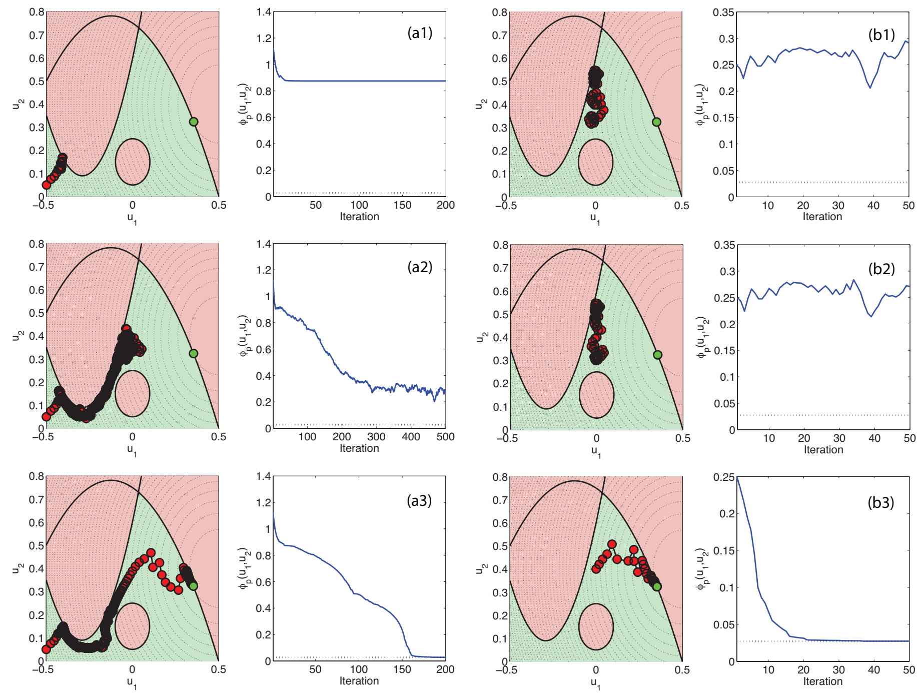


Figure 13: Performance of Algorithm 2 (random step) for the six cases. Not surprisingly, the algorithm does almost nothing desirable unless the SCFO are enforced.

to force this property for any algorithm – taking the feasibility conditions of the previous section and adding to them the requirement that the cost be locally decreasing at every iteration and that an additional constraint on the filter gain be satisfied to guarantee that the cost decrease for each step of the algorithm. Using Theorem 1, we showed that any algorithm meeting the SCFO would continue to decrease the cost until it became impossible to do so for a sufficiently low choice of projection parameters, which could only occur at a (practically stable) KKT point. This was demonstrated using several challenging examples.

5. Conclusions

The goal of the present paper has been to propose a set of sufficient conditions for convergence, via feasible iterates, to a plant KKT point in the context of the generalized RTO algorithm, and may be summarized via Figure 14 (as an extension of the structure in Figure 1). Sections 3 and 4 have presented the necessary theory with respect to feasibility and optimality, respectively, with the latter subsuming the former to comprise the full SCFO set. Testing the SCFO in simulation gave excellent performance, and showed that we could achieve convergence even for cases where the feasible set had unfavorable topological properties that made it impossible for most algorithms to get around unless the SCFO were applied.

It would be fair to note that, at least from a mathematical point of view, the proposed conditions are not terribly surprising, or even novel, in their structure – simply put, they restate the somewhat natural fact that local derivative information, when coupled with higher-order global bounds, allow us to manipulate any optimization algorithm to force it to converge to a KKT point regardless of its innate structure. This sort of idea cannot possibly be new in the context of numerical optimization. However, we believe that the work in this paper constitutes an important contribution to the RTO field, where the numerous challenges, as outlined in the introduction, have led to the creation of many methods but no general set of guarantees. It is hoped that the SCFO can fill an important gap by providing a theoretical foundation to guide the development and operation of future (and current) RTO algorithms.

Another point worth noting is how the *ad hoc* nature of the different algorithms is significantly reduced when placed into this framework. As we see from the proposed projection steps, the RTO algorithm still plays a role (by calculating the initial $\mathbf{u}_{k+1}^*$) in determining the general direction for the iterates to follow. As such, we naturally expect better, more accurate algorithms to get us to the optimum quicker, and are able to confirm these expectations in the simulated trials of Section 4. However, it is also important to note that the actual choice of algorithm is no longer crucial and is merely a preference – when the SCFO are enforced, feasible-side convergence to a KKT point is ensured regardless. We believe that this has the potential to remove a considerable burden from the practitioner, as no "optimal" algorithm exists and several valid approaches could be proposed depending on the specific problem at hand.

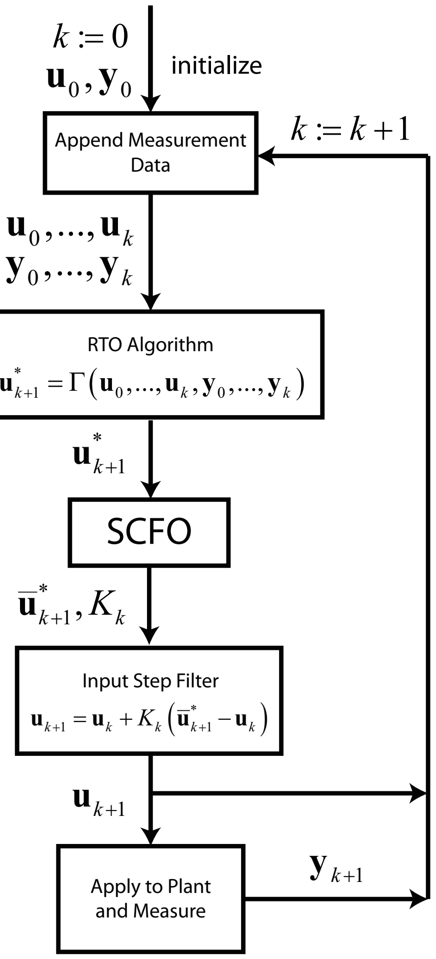

$k := 0$
$\mathbf{u}_0, \mathbf{y}_0$
initialize
Append Measurement Data
$k := k+1$
$\mathbf{u}_0, \ldots, \mathbf{u}_k$
$\mathbf{y}_0, \ldots, \mathbf{y}_k$
RTO Algorithm
$\mathbf{u}^*_{k+1} = \Gamma(\mathbf{u}_0, \ldots, \mathbf{u}_k, \mathbf{y}_0, \ldots, \mathbf{y}_k)$
$\mathbf{u}^*_{k+1}$
SCFO
$\bar{\mathbf{u}}^*_{k+1}, K_k$
Input Step Filter
$\mathbf{u}_{k+1} = \mathbf{u}_k + K_k(\bar{\mathbf{u}}^*_{k+1} - \mathbf{u}_k)$
$\mathbf{u}_{k+1}$
Apply to Plant and Measure
$\mathbf{y}_{k+1}$

Figure 14: Generalized feedback structure of RTO schemes with the SCFO included so as to enforce feasibility and KKT convergence.

Perhaps the immediate question introduced in this work is the following: should these conditions actually be enforced in practice as suggested in Figure 14? The short and immediate answer is that they probably should, as with a relatively compact set of assumptions they allow for very strong and useful guarantees, as well as for a general foundation for RTO analysis. A longer answer might involve the study of the impact of the employed projection on the convergence rate of an algorithm, as it is more than likely that there exist problems for which such a projection actually makes convergence slower. In these latter cases, it may be better to let the algorithm operate as it would normally, and then enforce the SCFO only if it fails to perform as desired. We have not attempted to carry out such a study here.

The other question – one of pressing importance and the subject of the companion paper – is whether or not it is possible to actually enforce these conditions for a real system, as they depend on the knowledge of exact local derivatives, global upper bounds, and noise-free measurements, none of which is generally *known* in application. It will be argued in the second paper that, while the SCFO cannot be applied as easily as they were here when the knowledge assumptions were made, they can still be robustly enforced for a number of practical realizations, and that doing so will generally lead to superior performance.

References

Alexandrov, N., Dennis, J., Lewis, R., & Torczon, V. (1997). *A Trust Region Framework for Managing the Use of Approximation Models in Optimization*. Technical Report Langley Research Center.

Beyer, H., & Sendhoff, B. (2007). Robust optimization - A comprehensive survey. *Comput. Methods Appl. Mech. Eng.*, *196*, 3190–3218.

Biegler, L., Grossmann, I., & Westerberg, A. (1985). A note on approximation techniques used for process optimization. *Comput. Chem. Eng.*, *9*, 201–206.

Box, G., & Draper, N. (1969). *Evolutionary Operation*. John Wiley & Sons.

Box, G., & Wilson, K. (1951). On the experimental attainment of optimum conditions. *J. Royal Stat. Society. Series B (Methodological)*, *13*, 1–45.

Brdys, M., & Tatjewski, P. (2005). *Iterative Algorithms for Multilayer Optimizing Control*. Imperial College Press.

Bunin, G. A., Fraire, F., François, G., & Bonvin, D. (2012a). Run-to-run MPC tuning via gradient descent. In I. D. L. Bogle, & M. Fairweather (Eds.), *22nd European Symposium on Computer Aided Process Engineering (London)* (pp. 927–931).

Bunin, G. A., François, G., & Bonvin, D. (2012b). Exploiting local quasiconvexity for gradient estimation in modifier-adaptation schemes. In *2012 American Control Conference (Montréal)* (pp. 2806–2811).

Bunin, G. A., François, G., & Bonvin, D. (2013a). Iterative controller tuning by real-time optimization. In *Dynamics and Control of Process Systems (DYCOPS) (Mumbai)*.

Bunin, G. A., François, G., & Bonvin, D. (2013b). Sufficient conditions for feasibility and optimality of real-time optimization schemes - II. Implementation issues. *arXiv [math.OC]*.

Bunin, G. A., François, G., Srinivasan, B., & Bonvin, D. (2011). Input filter design for feasibility in constraint-adaptation schemes. In *18th IFAC World Congress (Milan)* (pp. 5585–5590).

Chachuat, B., Marchetti, A., & Bonvin, D. (2008a). *Convergence of Modifier-Adaptation Schemes for Real-Time Optimization*. Technical Report EPFL.

Chachuat, B., Marchetti, A., & Bonvin, D. (2008b). Process optimization via constraints adaptation. *J. Process Control*, *18*, 244–257.

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

Cheng, J., & Zafiriou, E. (2000). Robust model-based iterative feedback optimization of steady state plant operations. *Ind. Eng. Chem. Res.*, *39*, 4215–4227.

Cheng, J., & Zafiriou, E. (2004). Robust measurement-based optimization with steady-state differential equation models. *Chem. Eng. Commun.*, *191*, 767–795.

Conn, A., Scheinberg, K., & Vicente, L. (2009). *Introduction to Derivative-Free Optimization*. Cambridge University Press.

Costello, S., François, G., Srinivasan, B., & Bonvin, D. (2011). Modifier adaptation for run-to-run optimization of transient processes. In *18th World Congress of the International Federation of Automatic Control (IFAC) (Milan)* (pp. 11471–11476).

Engell, S. (2007). Feedback control for optimal process operation. *J. Process Control*, *17*, 203–219.

Fatora, F., & Ayala, J. (1992). Successful closed loop real-time optimization. *Hydrocarbon Processing*, *71*, 65–68.

Flemming, T., Bartl, M., & Li, P. (2007). Set-point optimization for closed-loop control systems under uncertainty. *Ind. Eng. Chem. Res.*, *46*, 4930–4942.

Fletcher, R. (1987). *Practical Methods of Optimization*. Wiley and Sons.

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

François, G., & Bonvin, D. (2013). Use of convex model approximations for real-time optimization via modifier adaptation. *Ind. Eng. Chem. Res. (submitted)*.

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

Gao, W., & Engell, S. (2005). Iterative set-point optimization of batch chromatography. *Comput. Chem. Eng.*, *29*, 1401–1409.

Garcia, C., & Morari, M. (1984). Optimal operation of integrated processing systems. Part II: Closed-loop on-line optimizing control. *AIChE J.*, *30*, 226–234.

Georgakis, C. (2009). A model-free methodology for the optimization of batch processes: Design of dynamic experiments. In *7th IFAC International Symposium on Advanced Control of Chemical Processes (ADCHEM) (Istanbul)* (pp. 644–649).

Govatsmack, M., & Skogestad, S. (2005). Selection of controlled variables and robust setpoints. *Ind. Eng. Chem. Res.*, *44*, 2207–2217.

Hjarmarsson, H., Gevers, M., Gunnarsson, S., & Lequin, O. (1998). Iterative feedback tuning: Theory and applications. *Control Systems, IEEE*, *18*, 26–41.

Holmes, D. (2003). Model-free optimization in cement plants. In *Cement Industry Technical Conference* (pp. 159–173).

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

Kadam, J. V., Schlegel, M., Srinivasan, B., Bonvin, D., & Marquardt, W. (2007). Dynamic optimization in the presence of uncertainty: From off-line nominal solution to measurement-based implementation. *J. Process Control*, *17*, 389–398.

Karimi, A., Mišković, L., & Bonvin, D. (2004). Iterative correlation-based controller tuning. *Int. J. Adapt. Control Signal. Process.*, *18*, 645–664.

Killingsworth, N. J., & Krstić, M. (2006). PID tuning using extremum seeking: Online, model-free performance optimization. *Control Systems, IEEE*, *26*, 70–79.

Korn, G. A., & Korn, T. M. (2000). *Mathematical Handbook for Scientists and Engineers*. Dover Publications.

Magni, L., Forgione, M., Toffanin, C., Man, C., Kovatchev, B., De Nicolao, G., & Cobelli, C. (2009). Run-to-run tuning of model predictive control for type 1 diabetes subjects: In silico trial. *J. of Diabetes Science and Technology*, *3*, 1091–1098.

Marchetti, A. (2009). *Modifier-Adaptation Methodology for Real-Time Optimization (No. 4449)*. Ph.D. thesis EPFL.

Marchetti, A., Chachuat, B., & Bonvin, D. (2009). Modifier-adaptation methodology for real-time optimization. *Ind. Eng. Chem. Res.*, *48*, 6022–6033.

Marchetti, A., Chachuat, B., & Bonvin, D. (2010). A dual modifier-adaptation approach for real-time optimization. *J. Process Control*, *20*, 1027–1037.

Myers, R., Montgomery, D., & Anderson-Cook, C. (2009). *Response Surface Methodology*. John Wiley & Sons.

Naysmith, M., & Douglas, P. (1995). Review of real-time optimization in the chemical process industries. *Developments in Chemical Engineering and Mineral Processing*, *3*, 67–87.

Quelhas, A., Castro, N., & Pinto, J. (2013). Common vulnerabilities of RTO implementations in real chemical processes. *Can. J. Chem. Eng.*, *91*, 652–668.

Roberts, P. (1978). Algorithms for integrated system optimisation and parameter estimation. *Electron. Lett.*, *14*, 196–197.

Rockafellar, R. (1970). *Convex Analysis*. Princeton University Press.

Rodger, E. (2010). *Dual Modifier Adaptation Methodology For the On-line Optimization of Uncertain Processes*. Master's thesis McMaster University.

Srinivasan, B., Bonvin, D., Visser, E., & Palanki, S. (2003a). Dynamic optimization of batch processes: II. Role of measurements in handling uncertainty. *Comput. Chem. Eng.*, *27*, 27–44.

Srinivasan, B., Palanki, S., & Bonvin, D. (2003b). Dynamic optimization of batch processes: I. Characterization of the nominal solution. *Comput. Chem. Eng.*, *27*, 1–26.

Tadej, W., & Tatjewski, P. (2001). Analysis of an ISOPE-type dual algorithm for optimizing control and nonlinear optimization. *Int. J. Appl. Math. Comput. Sci.*, *11*, 429–457.

Tatjewski, P. (2008). Advanced control and on-line process optimization in multilayer structures. *Annual Reviews in Control*, *32*, 71–85.

Vugrin, K. (2003). *On the Effect of Numerical Noise in Simulation-Based Optimization*. Master's thesis Virginia Polytechnic Institute and State University.

Xu, G., & Cheng, S. (2008). Algorithm for steady-state optimizing control of industrial processes. *Control and Decision*, *23*, 619–625.

Zhang, Y., Monder, D., & Forbes, J. F. (2002). Real-time optimization under parametric uncertainty: A probability constrained approach. *J. Process Control*, *12*, 373–389.

Appendix

Proof of Lemma 2

We prove the lemma by considering the inequality along the line segment between an arbitrary pair $\mathbf{u}_k, \mathbf{u}_{k+1} \in \mathcal{I}$. The following one-dimensional parameterization is used:

$$f^u(\gamma) = f(\mathbf{u}(\gamma)), \tag{63}$$

with $\mathbf{u}(\gamma) = \mathbf{u}_k + \gamma(\mathbf{u}_{k+1} - \mathbf{u}_k),\ \gamma \in [0,1]$. As f is twice continuously differentiable, it follows that f^u is as well, which allows us to use the Taylor series expansion between $\gamma = 0$ and $\gamma = 1$, together with the mean-value theorem, to state (Korn & Korn, 2000, Section 4.10-4):

$$\begin{aligned} f^u(1) &= f^u(0) + \frac{df^u}{d\gamma}\Big|_{\gamma=0} + R_1(1,0) \\ R_1(1,0) &= \frac{1}{2}\frac{d^2 f^u}{d\gamma^2}\Big|_{\gamma=\tilde{\gamma}} \end{aligned}, \tag{64}$$

for some $\tilde{\gamma} \in (0,1)$. We proceed to define the first- and second-order derivatives in terms of the original function f. To do this we apply the chain rule:

$$\frac{df^u}{d\gamma}\Big|_{\gamma} = \sum_{i=1}^{n_u} \frac{\partial f}{\partial u_i}\Big|_{\mathbf{u}(\gamma)} \frac{du_i}{d\gamma}\Big|_{\gamma} = \nabla f(\mathbf{u}(\gamma))^T(\mathbf{u}_{k+1} - \mathbf{u}_k), \tag{65}$$

and then differentiate once more with respect to γ:

$$\frac{d^2 f^u}{d\gamma^2}\Big|_{\gamma} = \sum_{i=1}^{n_u} \frac{d}{d\gamma}\left(\frac{\partial f}{\partial u_i}\Big|_{\mathbf{u}(\gamma)} \frac{du_i}{d\gamma}\Big|_{\gamma} \right) = \sum_{i=1}^{n_u} \frac{d}{d\gamma}\left(\frac{\partial f}{\partial u_i}\Big|_{\mathbf{u}(\gamma)} \right) \frac{du_i}{d\gamma}\Big|_{\gamma}, \tag{66}$$

where we have ignored the terms corresponding to $d^2 u_i / d\gamma^2$ as all such terms are 0. Applying the chain rule again yields:

$$\begin{aligned} \frac{d^2 f^u}{d\gamma^2}\Big|_{\gamma} &= \sum_{i=1}^{n_u}\sum_{j=1}^{n_u} \frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\gamma)} \frac{du_j}{d\gamma}\Big|_{\gamma} \frac{du_i}{d\gamma}\Big|_{\gamma} \\ &= \sum_{i=1}^{n_u}\sum_{j=1}^{n_u} \frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\gamma)} (u_{k+1,i} - u_{k,i})(u_{k+1,j} - u_{k,j}) \end{aligned}. \tag{67}$$

Substituting the results of (65) and (67) into (64), and noting that $f^u(0) = f(\mathbf{u}_k)$ and $f^u(1) = f(\mathbf{u}_{k+1})$, leads to:

$$\begin{aligned} f(\mathbf{u}_{k+1}) &= f(\mathbf{u}_k) + \nabla f(\mathbf{u}_k)^T(\mathbf{u}_{k+1} - \mathbf{u}_k) + R_1(1,0) \\ R_1(1,0) &= \frac{1}{2}\sum_{i=1}^{n_u}\sum_{j=1}^{n_u} \frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\tilde{\gamma})} (u_{k+1,i} - u_{k,i})(u_{k+1,j} - u_{k,j}) \end{aligned}. \tag{68}$$

We complete the proof by deriving an upper bound on the remainder term. First, note that (by $ab \leq |a||b|$):

$$\begin{aligned}&\frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\tilde{\gamma})}(u_{k+1,i}-u_{k,i})(u_{k+1,j}-u_{k,j})\\&\leq \left|\frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\tilde{\gamma})}\right|\Big|(u_{k+1,i}-u_{k,i})(u_{k+1,j}-u_{k,j})\Big|\end{aligned}. \tag{69}$$

From the positivity of a quadratic (i.e. $0 \leq a^2 \pm 2ab + b^2 \Rightarrow 2|ab| \leq a^2 + b^2$), we have:

$$\begin{aligned}&\Big|(u_{k+1,i}-u_{k,i})(u_{k+1,j}-u_{k,j})\Big|\\&\quad\leq \frac{1}{2}(u_{k+1,i}-u_{k,i})^2 + \frac{1}{2}(u_{k+1,j}-u_{k,j})^2\end{aligned}, \tag{70}$$

from which it follows that:

$$\begin{aligned}&\frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\tilde{\gamma})}(u_{k+1,i}-u_{k,i})(u_{k+1,j}-u_{k,j})\\&\leq \frac{1}{2}\left|\frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\tilde{\gamma})}\right|\left[(u_{k+1,i}-u_{k,i})^2+(u_{k+1,j}-u_{k,j})^2\right]\end{aligned}. \tag{71}$$

Substituting this into (68) then yields the following bound on the remainder:

$$\begin{aligned}R_1(1,0) \leq &\frac{1}{4}\sum_{i=1}^{n_u}\sum_{j=1}^{n_u}\left|\frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\tilde{\gamma})}\right|(u_{k+1,i}-u_{k,i})^2\\&+\frac{1}{4}\sum_{i=1}^{n_u}\sum_{j=1}^{n_u}\left|\frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\tilde{\gamma})}\right|(u_{k+1,j}-u_{k,j})^2\end{aligned}. \tag{72}$$

By Clairaut's theorem, we have that:

$$\frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\tilde{\gamma})} = \frac{\partial^2 f}{\partial u_j \partial u_i}\Big|_{\mathbf{u}(\tilde{\gamma})}, \tag{73}$$

which, together with the interchangeability of the order of summation (Korn & Korn, 2000, Section 4.8-3), allows us to rewrite the second term on the right-hand side of (72) as:

$$\frac{1}{4}\sum_{j=1}^{n_u}\sum_{i=1}^{n_u}\left|\frac{\partial^2 f}{\partial u_j \partial u_i}\Big|_{\mathbf{u}(\tilde{\gamma})}\right|(u_{k+1,j}-u_{k,j})^2, \tag{74}$$

which is clearly equivalent to the first term (only the choice of indices differs). This allows us to combine the two to obtain:

$$R_1(1,0) \leq \frac{1}{2}\sum_{i=1}^{n_u}\sum_{j=1}^{n_u}\left|\frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}(\tilde{\gamma})}\right|(u_{k+1,i}-u_{k,i})^2. \tag{75}$$

To make this bound global for any choice of $\mathbf{u}_k, \mathbf{u}_{k+1} \in \mathcal{I}$ and all $\tilde{\gamma} \in (0,1)$, we use (8), which implies:

$$\left| \frac{\partial^2 f}{\partial u_i \partial u_j}\Big|_{\mathbf{u}} \right| < M_{ij}, \;\; \forall \mathbf{u} \in \mathcal{I}. \tag{76}$$

This globally bounds the remainder as:

$$R_1(1,0) \leq \frac{1}{2} \sum_{i=1}^{n_u} \sum_{j=1}^{n_u} M_{ij} (u_{k+1,i} - u_{k,i})^2, \tag{77}$$

or, in vector notation, as:

$$R_1(1,0) \leq \frac{1}{2} (\mathbf{u}_{k+1} - \mathbf{u}_k)^T \overline{\mathbf{Q}} (\mathbf{u}_{k+1} - \mathbf{u}_k), \tag{78}$$

with $\overline{\mathbf{Q}}$ a diagonal matrix with the diagonal elements defined as in (10). Substituting (78) into (68) and rearranging leads to the main result. □

Proof of Theorem 3

We first obtain a more global version of (21) that is independent of the RTO algorithm by remarking that:

$$\sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}| \leq \boldsymbol{\kappa}_j^T \left(\mathbf{u}^U - \mathbf{u}^L \right), \; \forall j = 1, ..., n_g, \tag{79}$$

which allows us to lower bound K_k as:

$$0 < \min_{j=1,...,n_g} \left[\frac{-g_{p,j}(\mathbf{u}_k)}{\boldsymbol{\kappa}_j^T (\mathbf{u}^U - \mathbf{u}^L)} \right] \leq \min_{j=1,...,n_g} \left[\frac{-g_{p,j}(\mathbf{u}_k)}{\displaystyle\sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}|} \right] = K_k. \tag{80}$$

This then leaves us with the simpler task of lower bounding K_k by lower bounding $-g_{p,j}(\mathbf{u}_k)$, since $\boldsymbol{\kappa}_j^T (\mathbf{u}^U - \mathbf{u}^L)$ is independent of iteration.

We proceed by stating that $-g_{p,j}(\mathbf{u}_k)$ may be lower bounded differently depending on its value. Specifically, we break the full set of possibilities into the following subcases:

1. $-g_{p,j}(\mathbf{u}_k) > \epsilon_j$
2. $-g_{p,j}(\mathbf{u}_k) \leq \epsilon_j$
   (a) $-g_{p,j}(\mathbf{u}_k) \geq \dfrac{K_{\epsilon,j} \delta_{g,j}}{\gamma_\kappa}$
   (b) $-g_{p,j}(\mathbf{u}_k) < \dfrac{K_{\epsilon,j} \delta_{g,j}}{\gamma_\kappa}$

where we will refer to the three subcases as Case 1, Case 2a, and Case 2b. We will now derive the lowest values that $-g_{p,j}$ can achieve for each of these three scenarios.

For Case 1, we start by noting that, from the definition of γ_κ in (28), we have that:

$$\tilde{\kappa}_{ji} \leq \gamma_\kappa \kappa_{ji}, \; \forall i = 1, ..., n_u, \;\; j = 1, ..., n_g, \tag{81}$$

with $\gamma_\kappa \in (0, 1)$ implicit from the strictness of κ and the fact that all of the $\tilde{\kappa}$ cannot be null simultaneously (each constraint being a function of the inputs and not simply a constant value). We now follow the same steps as in Lemma 1 and Theorem 2, this time for the case with the nonstrict Lipschitz constants, to obtain:

$$\begin{aligned} & g_{p,j}(\mathbf{u}_{k+1}) \leq g_{p,j}(\mathbf{u}_k) + \sum_{i=1}^{n_u} \tilde{\kappa}_{ji} |u_{k+1,i} - u_{k,i}| \\ & \Rightarrow g_{p,j}(\mathbf{u}_{k+1}) \leq g_{p,j}(\mathbf{u}_k) + \gamma_\kappa \sum_{i=1}^{n_u} \kappa_{ji} |u_{k+1,i} - u_{k,i}| \end{aligned} . \tag{82}$$

Since

$$\begin{aligned} g_{p,j}(\mathbf{u}_k) + \sum_{i=1}^{n_u} \kappa_{ji} |u_{k+1,i} - u_{k,i}| \leq 0 \\ \Leftrightarrow \sum_{i=1}^{n_u} \kappa_{ji} |u_{k+1,i} - u_{k,i}| \leq -g_{p,j}(\mathbf{u}_k) \end{aligned} \tag{83}$$

is enforced by (21) at all iterations, it follows that:

$$\begin{aligned} g_{p,j}(\mathbf{u}_{k+1}) \leq g_{p,j}(\mathbf{u}_k) - \gamma_\kappa g_{p,j}(\mathbf{u}_k) = (1 - \gamma_\kappa) g_{p,j}(\mathbf{u}_k) \\ \Leftrightarrow -g_{p,j}(\mathbf{u}_{k+1}) \geq (1 - \gamma_\kappa)(-g_{p,j}(\mathbf{u}_k)) \end{aligned} . \tag{84}$$

It is evident that the lowest value that $-g_{p,j}(\mathbf{u}_{k+1})$ can achieve while $-g_{p,j}(\mathbf{u}_k) > \epsilon_j$ is:

$$-g_{p,j}(\mathbf{u}_{k+1}) > (1 - \gamma_\kappa)\epsilon_j, \tag{85}$$

after which Case 1 would no longer be pertinent and we would shift our analysis to Cases 2a and 2b.

For these cases, we employ the result of Lemma 3, which states that:

$$0 < K_k < -2 \frac{\nabla g_{p,j}(\mathbf{u}_k)^T (\mathbf{u}^*_{k+1} - \mathbf{u}_k)}{(\mathbf{u}^*_{k+1} - \mathbf{u}_k)^T \overline{\mathbf{Q}}_j (\mathbf{u}^*_{k+1} - \mathbf{u}_k)} \Rightarrow -g_{p,j}(\mathbf{u}_{k+1}) > -g_{p,j}(\mathbf{u}_k). \tag{86}$$

Using Condition (29), together with the fact that $\overline{\mathbf{Q}}_j \succ 0$, allows:

$$K_{\epsilon,j} \leq -2 \frac{\nabla g_{p,j}(\mathbf{u}_k)^T (\mathbf{u}^*_{k+1} - \mathbf{u}_k)}{(\mathbf{u}^*_{k+1} - \mathbf{u}_k)^T \overline{\mathbf{Q}}_j (\mathbf{u}^*_{k+1} - \mathbf{u}_k)}, \tag{87}$$

as (31) represents the minimal value that the right-hand side could possibly take for any iteration where $g_{p,j}$ is ϵ-active – this is done by maximizing the (negative) numerator and maximizing the (positive) denominator. This allows us to make (86) independent of the RTO algorithm and to state:

$$K_k < K_{\epsilon,j} \Rightarrow -g_{p,j}(\mathbf{u}_{k+1}) > -g_{p,j}(\mathbf{u}_k). \tag{88}$$

Using (80), we may extend this statement further:

$$\min_{\hat{j}=1,\dots,n_g} \left[\frac{-g_{p,\hat{j}}(\mathbf{u}_k)}{\sum_{i=1}^{n_u} \kappa_{\hat{j}i} |u^*_{k+1,i} - u_{k,i}|} \right] < K_{\epsilon,j} \Rightarrow -g_{p,j}(\mathbf{u}_{k+1}) > -g_{p,j}(\mathbf{u}_k), \tag{89}$$

where we make a temporary change of indices so as to distinguish between the terms in the minimum operator (indexed by $\hat{j}$) from the single constraint whose evolution is being considered (indexed by j). However, since satisfying the inequality for any arbitrary j also satisfies it for the minimum:

$$\frac{-g_{p,j}(\mathbf{u}_k)}{\sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}|} < K_{\epsilon,j} \Rightarrow \min_{\hat{j}=1,\dots,n_g} \left[\frac{-g_{p,\hat{j}}(\mathbf{u}_k)}{\sum_{i=1}^{n_u} \kappa_{\hat{j}i} |u^*_{k+1,i} - u_{k,i}|} \right] < K_{\epsilon,j}, \tag{90}$$

we may simplify (89) to:

$$\frac{-g_{p,j}(\mathbf{u}_k)}{\sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}|} < K_{\epsilon,j} \Rightarrow -g_{p,j}(\mathbf{u}_{k+1}) > -g_{p,j}(\mathbf{u}_k). \tag{91}$$

In order to make this statement independent of the RTO algorithm, we exploit the ϵ-activity of $g_{p,j}$ and rewrite the condition in (29) as:

$$\nabla g_{p,j}(\mathbf{u}_k)^T (\mathbf{u}_k - \mathbf{u}^*_{k+1}) = \sum_{i=1}^{n_u} \frac{\partial g_{p,j}}{\partial u_i}\Big|_{\mathbf{u}_k} \left(u_{k,i} - u^*_{k+1,i} \right) \geq \delta_{g,j}, \tag{92}$$

from which it readily follows that:

$$\begin{aligned} &\sum_{i=1}^{n_u} \tilde{\kappa}_{ji} |u^*_{k+1,i} - u_{k,i}| \geq \sum_{i=1}^{n_u} \frac{\partial g_{p,j}}{\partial u_i}\Big|_{\mathbf{u}_k} \left(u_{k,i} - u^*_{k+1,i} \right) \geq \delta_{g,j} \\ &\Rightarrow \gamma_\kappa \sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}| \geq \delta_{g,j} \Leftrightarrow \sum_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}| \geq \frac{\delta_{g,j}}{\gamma_\kappa} \end{aligned}, \tag{93}$$

thereby allowing us to further chain the implication of (90):

$$\frac{-\gamma_\kappa g_{p,j}(\mathbf{u}_k)}{\delta_{g,j}} < K_{\epsilon,j} \Rightarrow \frac{-g_{p,j}(\mathbf{u}_k)}{\sum\limits_{i=1}^{n_u} \kappa_{ji} |u^*_{k+1,i} - u_{k,i}|} < K_{\epsilon,j}, \tag{94}$$

to obtain:

$$\begin{array}{l} \dfrac{-\gamma_\kappa g_{p,j}(\mathbf{u}_k)}{\delta_{g,j}} < K_{\epsilon,j} \Rightarrow -g_{p,j}(\mathbf{u}_{k+1}) > -g_{p,j}(\mathbf{u}_k) \\ \Leftrightarrow -g_{p,j}(\mathbf{u}_k) < \dfrac{K_{\epsilon,j}\delta_{g,j}}{\gamma_\kappa} \Rightarrow -g_{p,j}(\mathbf{u}_{k+1}) > -g_{p,j}(\mathbf{u}_k) \end{array}. \tag{95}$$

We are now equipped to make statements about Cases 2a and 2b. Note first that Case 2a may not exist – this occurs if

$$\epsilon_j < \frac{K_{\epsilon,j}\delta_{g,j}}{\gamma_\kappa}, \tag{96}$$

since Case 1 would remain dominant. If this is so, then only Case 1 and Case 2b need consideration. Assuming that Case 2a can occur, it follows that we can exploit the bound in (84) to calculate the lowest value that $-g_{p,j}(\mathbf{u}_{k+1})$ can achieve while $-g_{p,j}(\mathbf{u}_k) > \dfrac{K_{\epsilon,j}\delta_{g,j}}{\gamma_\kappa}$:

$$-g_{p,j}(\mathbf{u}_{k+1}) > (1-\gamma_\kappa)\frac{K_{\epsilon,j}\delta_{g,j}}{\gamma_\kappa}, \tag{97}$$

after which we would shift to Case 2b.

In this final scenario, we have, by (95), that the value cannot decrease and that $-g_{p,j}(\mathbf{u}_{k+1}) > -g_{p,j}(\mathbf{u}_k)$. It thus only remains to lower bound $-g_{p,j}(\mathbf{u}_k)$ in this bound to obtain the ultimate lower bound on $-g_{p,j}$ for all iterations. To do this, one needs to analyze how one can arrive at Case 2b. The first means is directly from Case 1, in which case we may shift the indices (by applying $k := k+1$) and note that:

$$-g_{p,j}(\mathbf{u}_k) > (1-\gamma_\kappa)\epsilon_j. \tag{98}$$

The second means is to go to Case 2b from Case 2a, where, using the same logic, we have:

$$-g_{p,j}(\mathbf{u}_k) > (1-\gamma_\kappa)\frac{K_{\epsilon,j}\delta_{g,j}}{\gamma_\kappa}. \tag{99}$$

Finally, there is also the possibility that the RTO algorithm is initialized directly in Case 2b, with $-g_{p,j}(\mathbf{u}_0) < \dfrac{K_{\epsilon,j}\delta_{g,j}}{\gamma_\kappa}$. In this case, we have:

$$-g_{p,j}(\mathbf{u}_k) > -g_{p,j}(\mathbf{u}_0). \tag{100}$$

To account for all of these possibilities, which are comprehensive and account for all that could occur, we may proceed to obtain the global lower bound on $-g_{p,j}$ by taking the minimum of the three:

$$-g_{p,j}(\mathbf{u}_k) > \min\left((1-\gamma_\kappa)\epsilon_j, (1-\gamma_\kappa)\frac{K_{\epsilon,j}\delta_{g,j}}{\gamma_\kappa}, -g_{p,j}(\mathbf{u}_0)\right). \tag{101}$$

This allows us to restate (80) as:

$$\begin{aligned} 0 < \min_{j=1,\ldots,n_g} & \left[\frac{\min\left((1-\gamma_\kappa)\epsilon_j, (1-\gamma_\kappa)\frac{K_{\epsilon,j}\delta_{g,j}}{\gamma_\kappa}, -g_{p,j}(\mathbf{u}_0)\right)}{\boldsymbol{\kappa}_j^T(\mathbf{u}^U - \mathbf{u}^L)}\right] \\ & < \min_{j=1,\ldots,n_g}\left[\frac{-g_{p,j}(\mathbf{u}_k)}{\boldsymbol{\kappa}_j^T(\mathbf{u}^U - \mathbf{u}^L)}\right] \leq K_k \end{aligned}, \tag{102}$$

which proves the desired result. □

Proof of Theorem 5

(i) (21) and (29) guarantee the existence of $K_{min} > 0$, by Theorem 3, which in turn allows the application of (50) and the guarantee that the conditions may only be satisfied for a finite number of iterations, as upper bounded in (53). As the algorithm is entirely deterministic, it follows that the inputs remain at $\mathbf{u}_\infty$ indefinitely following the first instance where the conditions can no longer be satisfied.

(ii) Feasibility follows from (21).

(iii) Monotonic cost improvement until $\mathbf{u}_\infty$ follows from (50).

(iv) We must characterize the point $\mathbf{u}_\infty$ where the three sets of Conditions (21), (29), and (50) cannot be satisfied. First, we drop the conditions on the filter gain (Condition (21) and the second condition in (50)) from the discussion, noting that these can always be satisfied for $K_k = K_{min}$ and are not a concern. We thus turn to the inequality constraints of Conditions (29) and (50), as well as to the box constraints. Using the same notation as in Theorem 4, these may be rewritten as:

$$\begin{gathered} \begin{bmatrix} \nabla\phi_p(\mathbf{u}_\infty)^T \\ \mathbf{J}_\infty^T \end{bmatrix}(\mathbf{u} - \mathbf{u}_\infty) \preceq \begin{bmatrix} -\delta_\phi \\ -\boldsymbol{\delta}_g \\ \mathbf{0} \end{bmatrix} \\ \tilde{\mathbf{u}}^L \preceq \mathbf{u} \preceq \tilde{\mathbf{u}}^U \end{gathered}. \tag{103}$$

Equivalently, we may consider the scaled version:

$$\begin{aligned} &\begin{bmatrix} \nabla \phi_p(\mathbf{u}_\infty)^T \\ \mathbf{J}_\infty^T \end{bmatrix} \alpha(\mathbf{u} - \mathbf{u}_\infty) \preceq \begin{bmatrix} -\alpha\delta_\phi \\ -\alpha\boldsymbol{\delta}_g \\ \mathbf{0} \end{bmatrix} . \\ &\tilde{\mathbf{u}}^L \preceq \mathbf{u}_\infty + \alpha(\mathbf{u} - \mathbf{u}_\infty) \preceq \tilde{\mathbf{u}}^U \end{aligned} \tag{104}$$

Since $\tilde{\mathbf{u}}^L \prec \mathbf{u}_\infty \prec \tilde{\mathbf{u}}^U$, it follows that there exists a choice $\alpha^L > 0$, for any $\mathbf{u}$, so that the inactive box constraints will be satisfied. Since $\alpha \to 0 \Rightarrow \alpha\boldsymbol{\delta}_g, \alpha\delta_\phi \to \mathbf{0}$, it follows that there exist small enough $\boldsymbol{\delta}_g^L = \alpha^L \boldsymbol{\delta}_g \succ \mathbf{0}$ and $\delta_\phi^L = \alpha^L \delta_\phi > 0$ so that the inactive box constraints may be satisfied in the limit for this choice of $\boldsymbol{\delta}_g$ and δ_ϕ. We have therefore proven that the inactive box constraints cannot cause infeasibility in the conditions as $\boldsymbol{\delta}_g, \delta_\phi \to \mathbf{0}$, and proceed to consider the infeasibility of the simpler case:

$$\begin{bmatrix} \nabla \phi_p(\mathbf{u}_\infty)^T \\ \mathbf{J}_\infty^T \end{bmatrix} \alpha^L(\mathbf{u} - \mathbf{u}_\infty) \preceq \begin{bmatrix} -\delta_\phi^L \\ -\boldsymbol{\delta}_g^L \\ \mathbf{0} \end{bmatrix} , \tag{105}$$

which is equivalent with respect to feasibility to the original unscaled set:

$$\begin{bmatrix} \nabla \phi_p(\mathbf{u}_\infty)^T \\ \mathbf{J}_\infty^T \end{bmatrix} (\mathbf{u} - \mathbf{u}_\infty) \preceq \begin{bmatrix} -\delta_\phi \\ -\boldsymbol{\delta}_g \\ \mathbf{0} \end{bmatrix} . \tag{106}$$

We analyze the infeasibility of this set as $\boldsymbol{\epsilon} \to \mathbf{0}$. From Theorem 1, we know that the system in (106) is infeasible iff:

$$a_\phi \nabla \phi_p(\mathbf{u}_\infty) + a_1 \mathbf{J}_{\infty,1} + ... + a_{n_J} \mathbf{J}_{\infty,n_J} = \mathbf{0}, \tag{107}$$

with at least one a coefficient corresponding to either the cost or the uncertain constraints strictly positive.

Since (107) must be true at $\mathbf{u}_\infty$ for $\mathbf{0} \prec \boldsymbol{\delta}_g \preceq \boldsymbol{\delta}_g^L, 0 < \delta_\phi \leq \delta_\phi^L$, we proceed to analyze the several cases that satisfy (107). The first corresponds to the occurrence of a trivial negative spanning where one of the elements in the summation is $\mathbf{0}$. This may occur with either the cost or the uncertain constraints, as the box constraints cannot have a zero gradient. If this is true for the cost, with $\nabla \phi_p(\mathbf{u}_\infty) = \mathbf{0}$, then the KKT conditions for an unconstrained KKT point are satisfied and the KKT error is 0. For the constraints, we have, from Assumption A3, that there exists $\epsilon_{m,j} > 0$ for every uncertain constraint $g_{p,j}$ so that the gradient $\nabla g_{p,j}(\mathbf{u}_\infty)$ cannot be $\mathbf{0}$ as $\boldsymbol{\epsilon} \to \mathbf{0}$.

We consider the cases of nontrivial negative spanning next, where at least two of the a coefficients are strictly positive. There are two cases of interest – one where $a_\phi = 0$ (the cost descent condition does not contribute to infeasibility) and one where $a_\phi > 0$ (the cost does contribute). Suppose first that $a_\phi = 0$ as $\boldsymbol{\epsilon} \to \mathbf{0}$. If this is true, then the relevant constraint gradients span each other negatively even as the ϵ-active set becomes an arbitrarily good approximation of the true active set. This, in turn, means that there is no direction that locally decreases all of these constraints, i.e. a locally feasible direction. This then implies that $\mathbf{u}_\infty$ is a singleton in the set defined by the active constraints, thereby allowing us to ignore this case. As stated before, such problems are ill-posed and do not require RTO. We are therefore left with the case of $a_\phi > 0$ for a sufficiently small $\boldsymbol{\epsilon}^L \succ \mathbf{0}$ (we need $\boldsymbol{\epsilon}^L$ to be small enough so as to ensure that there is no negative spanning between the ϵ-active constraints).

We now proceed to derive an upper bound on the KKT error that is valid for $\mathbf{0} \prec \boldsymbol{\epsilon} \preceq \boldsymbol{\epsilon}^L$, $\mathbf{0} \prec \boldsymbol{\delta}_g \preceq \boldsymbol{\delta}_g^L$, and $0 < \delta_\phi \leq \delta_\phi^L$ and show how it must tend to 0 as $\boldsymbol{\epsilon} \to \mathbf{0}$.

For this case, we may, without loss of generality, scale to set $a_\phi = 1$ and, changing the notation on the coefficients, write (107) in a form that is analogous to that of the Lagrangian in (49):

$$\nabla \tilde{\mathcal{L}}(\mathbf{u}_\infty) = \nabla \phi_p(\mathbf{u}_\infty) + \sum_{j=1}^{n_g} \tilde{\mu}_j \nabla g_{p,j}(\mathbf{u}_\infty) - \tilde{\boldsymbol{\zeta}}^L + \tilde{\boldsymbol{\zeta}}^U = \mathbf{0}, \tag{108}$$

with $\tilde{\mu}_j = 0,\ \forall j : g_{p,j}(\mathbf{u}_\infty) < -\epsilon_j$, $\tilde{\zeta}_i^U = 0,\ \forall i : u_{\infty,i} < u_i^U$, $\tilde{\zeta}_i^L = 0,\ \forall i : u_{\infty,i} > u_i^L$ establishing equivalence with (107).

Using (108), we may express the gradient of the true Lagrangian as:

$$\begin{array}{l} \nabla \mathcal{L}(\mathbf{u}_\infty) = \nabla \mathcal{L}(\mathbf{u}_\infty) - \nabla \tilde{\mathcal{L}}(\mathbf{u}_\infty) = \\ \qquad \displaystyle\sum_{j=1}^{n_g} (\mu_j - \tilde{\mu}_j) \nabla g_{p,j}(\mathbf{u}_\infty) - (\boldsymbol{\zeta}^L - \tilde{\boldsymbol{\zeta}}^L) + (\boldsymbol{\zeta}^U - \tilde{\boldsymbol{\zeta}}^U) \end{array}. \tag{109}$$

Noting that $\nabla \mathcal{L}(\mathbf{u}_\infty) = \mathbf{0}$ for the choice $\boldsymbol{\mu} = \tilde{\boldsymbol{\mu}}$, $\boldsymbol{\zeta}^L = \tilde{\boldsymbol{\zeta}}^L$, $\boldsymbol{\zeta}^U = \tilde{\boldsymbol{\zeta}}^U$, we upper bound the KKT error as follows:

$$
\begin{aligned}
&\inf_{\boldsymbol{\mu},\boldsymbol{\zeta}^L,\boldsymbol{\zeta}^U\succeq\mathbf{0}} \Bigg(\nabla\mathcal{L}(\mathbf{u}_\infty)^T\nabla\mathcal{L}(\mathbf{u}_\infty)+\sum_{j=1}^{n_g}\left[\mu_j g_{p,j}(\mathbf{u}_\infty)\right]^2+ \\
&\qquad\qquad \sum_{i=1}^{n_u}\left[(\zeta_i^L(u_i^L-u_{\infty,i}))^2+(\zeta_i^U(u_{\infty,i}-u_i^U))^2\right]\Bigg) \\
&\le \inf_{\boldsymbol{\mu}=\tilde{\boldsymbol{\mu}},\boldsymbol{\zeta}^L=\tilde{\boldsymbol{\zeta}}^L,\boldsymbol{\zeta}^U=\tilde{\boldsymbol{\zeta}}^U} \Bigg(\nabla\mathcal{L}(\mathbf{u}_\infty)^T\nabla\mathcal{L}(\mathbf{u}_\infty)+\sum_{j=1}^{n_g}\left[\mu_j g_{p,j}(\mathbf{u}_\infty)\right]^2+ \\
&\qquad\qquad \sum_{i=1}^{n_u}\left[(\zeta_i^L(u_i^L-u_{\infty,i}))^2+(\zeta_i^U(u_{\infty,i}-u_i^U))^2\right]\Bigg) \qquad , \qquad (110) \\
&= \sum_{j=1}^{n_g}\left[\tilde{\mu}_j g_{p,j}(\mathbf{u}_\infty)\right]^2+ \\
&\qquad\qquad \sum_{i=1}^{n_u}\left[(\tilde{\zeta}_i^L(u_i^L-u_{\infty,i}))^2+(\tilde{\zeta}_i^U(u_{\infty,i}-u_i^U))^2\right] \\
&= \sum_{j=1}^{n_g}\left[\tilde{\mu}_j g_{p,j}(\mathbf{u}_\infty)\right]^2 \le \sum_{j=1}^{n_g}(\tilde{\mu}_j\epsilon_j)^2
\end{aligned}
$$

where the steps may be justified as follows:

- Setting the Lagrange multipliers equal to their $\tilde{(\cdot)}$ analogues leads to taking an infimum over a set with tighter constraints, which cannot lower the value of the infimum.
- The infimum may be evaluated by substituting in the $\tilde{(\cdot)}$ values and noting that the gradient of the Lagrangian is $\mathbf{0}$ for this choice, leaving only the complementary slackness terms.
- All of the terms corresponding to the box constraints may be removed since the constraints are either active with a value of 0 or have $\tilde{\zeta}$ coefficients equal to 0 by definition.
- As all of the ϵ-active constraints must have values that are smaller, in absolute value, than the corresponding $\boldsymbol{\epsilon}$ values, the bound may be restated further in terms of $\boldsymbol{\epsilon}$ (the ϵ-inactive constraints have corresponding $\tilde{\mu}$ values of 0 by definition).

Clearly, this upper bound approaches 0 as $\boldsymbol{\epsilon}\to\mathbf{0}$ provided that the relevant $\tilde{\mu}_j$ are finite for any realization of the algorithm. This is guaranteed by the finite nature of (108), which may only have infinite coefficients for those uncertain constraints with a zero gradient. However, this has already been ruled out by Assumption A3 as $\boldsymbol{\epsilon}\to\mathbf{0}$. Since the upper bound in (110) tends to 0, it follows that the error in (54) does as well, thereby proving (55). □