Submitted to *Processes*. Pages 1 - 35.

OPEN ACCESS

Article

A Real-Time Optimization Framework for the Iterative Controller Tuning Problem

Gene A. Bunin, Grégory François and Dominique Bonvin*

Laboratoire d'Automatique, Ecole Polytechnique Fédérale de Lausanne, CH-1015, Lausanne, Switzerland

* Author to whom correspondence should be addressed; dominique.bonvin@epfl.ch, Tel.: +41 21 6933843, Fax: +41 21 6932574.

Version August 23, 2013 submitted to Processes. *Typeset by LaTeX using class file mdpi.cls*

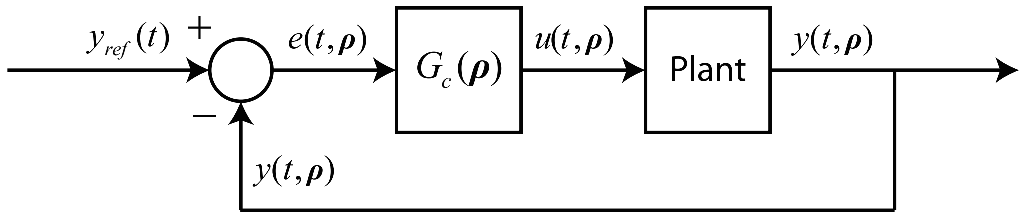
Abstract: We investigate the general iterative controller tuning (ICT) problem, where the task is to find a set of controller parameters that optimize some user-defined performance metric when the same control task is to be carried out repeatedly. Following a repeatability assumption on the system, we show that the ICT problem may be formulated as a real-time optimization (RTO) problem, thus allowing for the ICT problem to be solved in the RTO framework, which is both very flexible and comes with strong theoretical guarantees. In particular, we propose the use of a recently released RTO solver and outline a simple procedure for how this solver may be configured to solve ICT problems. The effectiveness of the proposed method is illustrated by successfully applying it to four case studies – two experimental and two simulated – that cover the tuning of model-predictive, general fixed-order, and PID controllers, as well as a system of controllers working in parallel.

Keywords: controller autotuning; real-time optimization; data-driven tuning methods

1. Introduction

The typical task of a controller consists in tracking a user-specified trajectory as closely as possible while observing certain additional specifications, such as stability, the satisfaction of safety limits, and the minimization of expensive control action when it is not needed. Mathematically, we may define such a controller by the mapping $G_c(\boldsymbol{\rho})$, where $\boldsymbol{\rho} \in \mathbb{R}^{n_\rho}$ denote the parameters that dictate the controller's behavior and represent decision variables (the "tuning parameters") for the engineer intending to

Figure 1. Qualitative schematic of a single-input-single-output system with the controller $G_c(\boldsymbol{\rho})$. Elements such as disturbances and sensor dynamics, as well as any controller-specific requirements, are left out for simplicity. We use the notation $y(t, \boldsymbol{\rho})$ to mark the (implicit) dependence of the control output on the tuning parameters $\boldsymbol{\rho}$ (likewise for the input and the error).



implement the controller in practice. In the simplest scenario, this often leads to a closed-loop system that may be described by the schematic in Figure 1. No assumptions are made on the nature of G_c, which may represent such controllers as the classical PID, the general fixed-order controller, or even the more advanced MPC (model-predictive control). To be even more general, G_c may represent an entire system of such controllers – one would need, in this case, to simply replace $y_{ref}(t)$, $y(t, \boldsymbol{\rho})$, $u(t, \boldsymbol{\rho})$, and $e(t, \boldsymbol{\rho})$ by their vector equivalents.

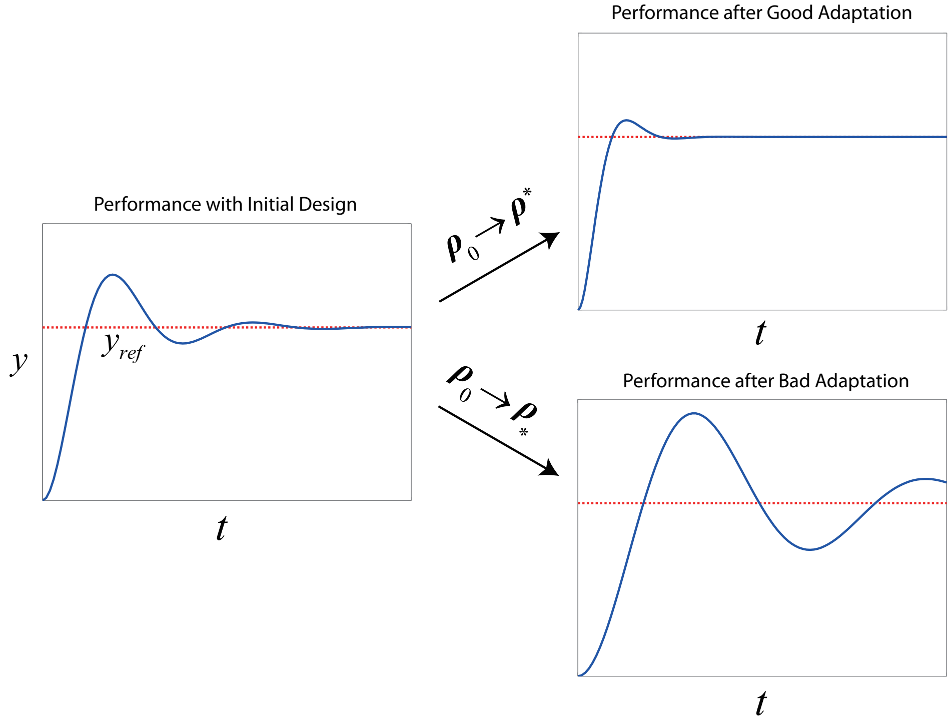
As with any set of decision variables, it should be clear that there are both good and bad choices of $\boldsymbol{\rho}$, and in every application some sort of design phase preceeds the actual implementation and acts to choose a set of $\boldsymbol{\rho}$ that is expected to track the reference y_{ref} "well" while meeting any additional specifications. The classic example for PID controllers is the Ziegler-Nichols tuning method [1], with methods such as model-based direct synthesis [2] and virtual reference feedback tuning [3] acting as more advanced alternatives. Though not as developed, both theoretical and heuristic approaches exist for the design of MPC [4] and general fixed-order controllers [5,6] as well.

In the majority of cases, the set of controller parameters obtained by these design methods will not be the best possible with respect to control performance. There are many reasons for this, with some of the common ones being:

- assumptions on the plant, such as linearity or time invariance, that are made at the design stage,
- modeling errors and simplifications,
- conservatism in the case of a robust design,
- time constraints and/or deadlines that give preference to a simpler design over an advanced one.

To improve the closed-loop performance of the system, some sort of data-driven adaptation of the parameters from their initial designed values, denoted here by $\boldsymbol{\rho}_0$, may be done online following the acquisition of new data. These are generally classified as "indirect" and "direct" adaptations [7] depending on what is actually adapted – the model (followed by a model-based re-design of the controller) in the indirect variant, or the controller parameters directly in the direct one. This paper

Figure 2. The basic idea of iterative controller tuning. Here, a step change in the setpoint represents the repetitive control task. We use $\boldsymbol{\rho}_*$ to denote a sort of "anti-optimum" that might be achieved with a bad adaptation algorithm.



investigates *direct methods that attempt to optimize control performance by establishing a direct link between the observed closed-loop performance and the controller parameters*, with the justification that such methods may be forced to converge – at least, theoretically – to a locally optimal choice $\boldsymbol{\rho}^*$ regardless of the quality or the availability of the model, which cannot be said for indirect schemes [8].

Many of these schemes attempt to minimize a certain user-defined performance metric (e.g., the tracking error) for a given run or batch by playing with the controller parameters as one would in an iterative optimization scheme – i.e., by changing the parameters between two consecutive runs, trying to discover the effect that this change has on the closed-loop performance (estimating the performance derivatives), and then using the derivative estimates to adapt the parameters further in some gradient-descent manner [9,10,11,12,13]. This is essentially the *iterative controller tuning* (ICT) *problem*, whose goal is to bring the initial suboptimal set $\boldsymbol{\rho}_0$ to the locally optimal $\boldsymbol{\rho}^*$ via *iterative experimentation on the closed-loop system*, all the while avoiding that the system become dangerously unstable from the adaptation (a qualitative sketch of this idea is given in Figure 2). A notable limitation of such methods, though rarely stated explicitly, is that the control task for which the controller is being adapted must be identical (or very similar) from one run to the next – otherwise, the concept of optimality may simply not exist since what is optimal for one control task (e.g., the tracking of a step change) need not be so for another (e.g., the tracking of a ramp). A closely-related problem where the assumption of a repeated control task is made formally is that of iterative learning control [14], although what is adapted in that case is the *open-loop* input trajectory rather than the parameters of a controller dictating the closed-loop system.

We observe that, as the essence of these tuning methods consists in iteratively minimizing a performance function that is *unknown* due to the lack of knowledge of the plant, the ICT problem is actually a *real-time optimization* (RTO) problem as it must be solved by iterative experimentation[1]. Recent work by the authors [19,20,21] has attempted to unify different RTO approaches and to standardize the RTO problem as any problem having the following canonical form:

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

where $\mathbf{v} \in \mathbb{R}^{n_v}$ denote the RTO variables[2] (RTO inputs) forced to lie in the relevant RTO input space defined by the lower and upper limits $\mathbf{v}^L$ and $\mathbf{v}^U$, ϕ_p denotes the cost function to be minimized, and $\mathbf{G}_p$ and $\mathbf{G}$ denote the sets of individual constraints $g_p, g : \mathbb{R}^{n_v} \rightarrow \mathbb{R}$ (i.e., safety limitations, performance specifications) to be respected. We use the symbol $\preceq$ to denote *componentwise inequality*.

The subscript p (for "plant") is used to indicate those functions that are unknown, or "uncertain", and can only be evaluated by applying a particular $\mathbf{v}_k$ and conducting a single experiment (with k denoting the experiment/iteration counter), from which the corresponding function values may then be measured or estimated:

$$\begin{aligned} \hat{\phi}_p(\mathbf{v}_k) &= \phi_p(\mathbf{v}_k) + w_{\phi,k} \\ \hat{g}_p(\mathbf{v}_k) &= g_p(\mathbf{v}_k) + w_{g,k} \end{aligned}, \tag{2}$$

with some additive stochastic error w. Conversely, the absence of p indicates that the function is easily evaluated by algebraic calculation without any error present.

Owing to the generality of (1), casting the ICT problem in this form is fairly straightforward and, as will be shown in this work, has numerous advantages as it allows for a fairly systematic and flexible approach to controller tuning in a framework where strong theoretical guarantees are available. The main contribution of our work is thus to make this generalization formally and to argue for its advantages while cautioning the potential user of both its apparent and hypothetical pitfalls.

Our second contribution lies in proposing a concrete method for solving the ICT problem in this manner. Namely, we advocate the use of the recently released open-source SCFO ("**s**ufficient **c**onditions for **f**easibility and **o**ptimality") solver that has been designed for solving RTO problems with strong theoretical guarantees [21]. While this choice is undoubtedly biased, we put it forward as it is, to the best of our knowledge, the only solver released to date that solves the RTO problem (1) *reliably*, which is to say that it consistently converges to a local minimum without violating the safety constraints in theoretical settings and that it is fairly robust in doing the same in practical ones. Though quite simple

[1] This is the definition of "real-time optimization" as it is used in the process systems engineering community (see, e.g., [15,16]) to indicate gradual improvement of an economic objective – the function ϕ_p in (1) – by iterative experimentation. We warn the reader to not confuse this with the more recent use of "real-time optimization" in the fast, on-line computation context (see, e.g., [17,18]), where an optimization problem is solved subject to real-time constraints for a purpose other than the optimization of an economic objective (e.g., for control or estimation).

[2] It is more common to denote RTO inputs by $\mathbf{u}$ in the literature, but we use $\mathbf{v}$ here so as not to create confusion with the control input $u(t)$.

to apply, the SCFO framework and the solver itself need to be properly configured, and so we guide the potential user through how to configurate the solver for the ICT problem.

Finally, as the theoretical discussion alone should not be sufficient to convince the reader that there is strong potential for solving the ICT problem as an RTO one, we finish the paper with a total of four case studies, which are intended to cover a diverse range of experimental and simulated problems and to demonstrate the general effectiveness of the proposed method, the difficulties that are likely to be encountered in application, and any weak points where the methodology still needs to be improved. Specifically, the four studies considered all solve the ICT problem for:

- the tracking of a temperature profile in a laboratory-scale stirred tank by an MPC controller,
- the tracking of a periodic setpoint for a laboratory-scale torsional system by a general fixed-order controller with a controller stability constraint,
- the PID tracking of a setpoint change for various linear systems (previously examined in [13,22]),
- the setpoint tracking and disturbance rejection for a five-input, five-output multi-loop PI system with imperfect decoupling and a hard output constraint.

In each case, we do our best to concretize the theory discussed earlier by showing how the resulting ICT problem may be formulated in the RTO framework, followed by the application of the SCFO solver with the proposed configuration.

2. The RTO Formulation of the Iterative Controller Tuning Problem

In this section, we go through the different components of the RTO problem (1) and state their ICT analogues, together with any assumptions necessary to make the links between the two clean. We then finish by reviewing the benefits and limitations of this approach.

2.1. The Cost Function $\phi_p \rightarrow$ The Control Performance Metric

The intrinsic driving force behind iteratively tuning a controller so that it performs "better" is the somewhat natural belief that there is some sort of deterministic link between the parameters and the observed performance. We qualify this via the following assumption, which was originally stated in the MPC context in [23] and then extended to the general controller in [24].

Assumption 1 (Repeatability)**.** *Let $\boldsymbol{\rho} \in \mathbb{R}^{n_\rho}$ denote the tuning parameters of a controller and J_k the observed value of the user-defined performance metric at run k for a fixed control task that is identical from run to run. The closed-loop process is* repeatable with respect to performance *if*

$$J_k = J(\boldsymbol{\rho}_k) + \delta_k, \tag{3}$$

where $\boldsymbol{\rho}_k$ are the parameters of the controller at run k, $J : \mathbb{R}^{n_\rho} \rightarrow \mathbb{R}$ is a purely deterministic relation between the performance metric and the parameters, and δ_k is the "non-repeatability noise", a purely stochastic element that is independent of $\boldsymbol{\rho}_k$.

In layman's terms, the (unknown) function J is precisely the intrinsic link that we believe in, while δ_k is a representation of reality, which most often manifests itself by means of measurement noise and differs unpredictably from run to run. The discussion of the validity of such an assumption is deferred to the end of the section.

Comparing (2) and (3), both of which involve a deterministic function that is sampled with additive noise, we establish our first RTO $\rightarrow$ ICT connection:

$$\underset{\mathbf{v}}{\text{minimize}} \quad \phi_p(\mathbf{v}) \rightarrow \underset{\boldsymbol{\rho}}{\text{minimize}} \quad J(\boldsymbol{\rho}). \tag{4}$$

A common general performance metric, given here in continuous form for the single-input-single-output (SISO) case, may be defined as:

$$J_k := \lambda_1 \int_0^{t_b} \left[y_{ref}(t) - y(t, \boldsymbol{\rho}_k) \right]^2 dt + \lambda_2 \int_0^{t_b} u^2(t, \boldsymbol{\rho}_k) dt + \lambda_3 \int_0^{t_b} \dot{y}^2(t, \boldsymbol{\rho}_k) dt + \lambda_4 \int_0^{t_b} \dot{u}^2(t, \boldsymbol{\rho}_k) dt, \tag{5}$$

where t_b denotes the total length of a single run and where the weights $\boldsymbol{\lambda} \succeq \mathbf{0}$ may be set as needed to trade off between giving preference to tracking error, the control action, the smoothness of the output response, and the aggressiveness of the controller. Modifications that include other criteria, such as frequency weighting [10], or that modify the time interval for which the performance is analyzed by adding a "mask" [22], are of course possible as well.

2.2. *The Uncertain Inequality Constraints* $\mathbf{G}_p \rightarrow$ *Safety and Economic Constraints*

Many control applications may have strict safety specifications that require a given output $y(t, \boldsymbol{\rho})$ to remain within a certain zone, defined by $\underline{y}$ and $\overline{y}$, throughout the length of the run:

$$\underline{y} \leq y(t, \boldsymbol{\rho}) \leq \overline{y}, \quad \forall t \in [0, t_b]. \tag{6}$$

While it is not difficult to propose methods to enforce such behavior for the general controller, many of which would likely try to incorporate the constraints as setpoint objectives, such approaches remain largely *ad hoc*. This drawback has shifted particular emphasis to MPC as being the advanced controller to be able to deal with output constraints systematically [25], but even here no rigorous conditions for satisfying (6) are available for the general case where any amount of plant-model mismatch is admissible.

Since rigorous theoretical conditions *are* available for satisfying $\mathbf{G}_p(\mathbf{v}) \preceq \mathbf{0}$ in the RTO framework [19], we may exploit this advantage by casting the hard output constraints for the ICT problem in RTO form. To do this, we start by replacing the two semi-infinite constraints of (6) by their equivalent finite versions:

$$\begin{array}{c} \underline{y} \leq \min\limits_{t \in [0, t_b]} y(t, \boldsymbol{\rho}) \\ \max\limits_{t \in [0, t_b]} y(t, \boldsymbol{\rho}) \leq \overline{y} \end{array}.$$

At this point, we need to apply a version of Assumption 1 for the constraints. For a particular run k, we assume that *the closed-loop process is repeatable with respect to the control output range*:

$$\begin{array}{c} \min\limits_{t \in [0, t_b]} y(t, \boldsymbol{\rho}_k) = y_{min}(\boldsymbol{\rho}_k) + \delta_{min,k} \\ \max\limits_{t \in [0, t_b]} y(t, \boldsymbol{\rho}_k) = y_{max}(\boldsymbol{\rho}_k) + \delta_{max,k} \end{array}, \tag{7}$$

i.e., that the minimum and maximum values of the trajectory $y(t, \boldsymbol{\rho}_k)$ observed for a given run k are (unknown) deterministic functions (y_{min}, y_{max}) of the parameters plus a stochastic element ($\delta_{min,k}$, $\delta_{max,k}$).

Making the link with (2), we may now restate the hard output constraints in RTO form as:

$$\mathbf{G}_p(\mathbf{v}) \preceq \mathbf{0} \rightarrow \begin{array}{c} -y_{min}(\boldsymbol{\rho}) + \underline{y} \leq 0 \\ y_{max}(\boldsymbol{\rho}) - \overline{y} \leq 0 \end{array}, \tag{8}$$

where the function values y_{min} and y_{max} can be measured for a given $\boldsymbol{\rho}$ with the additive errors δ_{min} and δ_{max}.

Alternatively, it may occur that there are economic constraints with respect to the inputs. As an example, consider a reactor where one of the control inputs is the feed rate of a reagent. While effective for the purposes of control, the reagent may be expensive and so only a limited amount may be allotted per batch, with the constraint

$$\int_0^{t_b} u(t, \boldsymbol{\rho}) dt \leq \overline{u}_T$$

imposed, where $\overline{u}_T$ is some user-defined limit. Following the same steps as above, we suppose that

$$\int_0^{t_b} u(t, \boldsymbol{\rho}_k) dt = u_T(\boldsymbol{\rho}_k) + \delta_{u,k},$$

with u_T the deterministic component and δ_u the non-repeatability noise, and make the connection:

$$\mathbf{G}_p(\mathbf{v}) \preceq \mathbf{0} \rightarrow u_T(\boldsymbol{\rho}) - \overline{u}_T \leq 0.$$

It should be clear that extension to multiple-input-multiple-output (MIMO) cases is trivial, as this only adds more elements to $\mathbf{G}_p$.

2.3. *The Certain Inequality Constraints* $\mathbf{G} \rightarrow$ *Controller Specifications and Stability Considerations*

In some controllers, analytically known inequality relations may need to be satisfied. One such example is the case of the MPC controller, where one may tune both the control and prediction horizons (m and n, respectively) with the built-in rule [25]:

$$m \leq n, \tag{9}$$

which, if we define $\rho_1 \triangleq m$ and $\rho_2 \triangleq n$, leads to the following link with (1):

$$\mathbf{G}(\mathbf{v}) \preceq \mathbf{0} \rightarrow \rho_1 - \rho_2 \leq 0.$$

As another example, we may want to adapt the parameters of the discrete fixed-order controller:

$$G_c(\boldsymbol{\rho}) = \frac{\rho_1 z^2 + \rho_2 z + \rho_3}{z^2 + \rho_4 z + \rho_5}, \tag{10}$$

but would like to limit our search to stable controllers only. Employing the Jury stability criterion [26], we generate the first four rows of the Jury table for the denominator of $G_c(\boldsymbol{\rho})$:

$$\begin{array}{lccc} \text{row } 1: & 1 & \rho_4 & \rho_5 \\ \text{row } 2: & \rho_5 & \rho_4 & 1 \\ \text{row } 3: & 1-\rho_5^2 & \rho_4 - \rho_4\rho_5 & 0 \\ \text{row } 4: & \rho_4 - \rho_4\rho_5 & 1-\rho_5^2 & 0 \end{array},$$

from which the sufficient conditions for controller stability are obtained as:

$$\begin{array}{l} |\rho_5| < 1 \\ |\rho_4 - \rho_4\rho_5| < |1-\rho_5^2| \end{array} \rightarrow \begin{array}{l} |\rho_5| \leq 1-\varepsilon \\ |\rho_4 - \rho_4\rho_5| \leq |1-\rho_5^2| - \varepsilon \end{array}, \tag{11}$$

with the constraint set on the right representing an implementable nonstrict version with negligible conservatism for $\varepsilon > 0$ small. Controller stability may now be ensured in RTO form with the correspondance:

$$\mathbf{G}(\mathbf{v}) \preceq \mathbf{0} \rightarrow \begin{array}{l} |\rho_5| - 1 + \varepsilon \leq 0 \\ |\rho_4 - \rho_4\rho_5| - |1-\rho_5^2| + \varepsilon \leq 0 \end{array}. \tag{12}$$

Finally, we note that *nominal* closed-loop stability constraints may also be incorporated in this manner. As a simple example, consider the unstable plant that is modeled as

$$G(s) = \frac{1}{s-1},$$

and that is to be controlled by a PD controller with ρ_1 and ρ_2 the proportional and derivative gains, respectively:

$$G_c(\boldsymbol{\rho}) = \rho_1 + \rho_2 s.$$

From the analysis of the characteristic equation $1 + GG_c = 0$, we have the stability condition, together with its implementable version:

$$\frac{1-\rho_1}{1+\rho_2} < 0 \rightarrow \frac{1-\rho_1}{1+\rho_2} \leq -\varepsilon,$$

which, again, allows the correspondance:

$$\mathbf{G}(\mathbf{v}) \preceq \mathbf{0} \rightarrow \frac{1-\rho_1}{1+\rho_2} + \varepsilon \leq 0. \tag{13}$$

Extensions to robust nominal stability follow easily, and would simply involve a greater number of constraints.

2.4. The Box Constraints $\mathbf{v}^L \preceq \mathbf{v} \preceq \mathbf{v}^U \rightarrow$ Controller Parameter Limits

Given the RTO-ICT correspondence of $\mathbf{v} \rightarrow \boldsymbol{\rho}$, the box constraints of the RTO problem are simply the lower and upper limits, $\boldsymbol{\rho}^L$ and $\boldsymbol{\rho}^U$, on the adapted controller parameters. We note that certain limits will be obvious for certain controllers – e.g., the integral time should be superior to 0 in a PI controller, while the prediction and control horizons of an MPC controller should be equal to or greater than 1. In other cases, one may have to think a little before deciding on appropriate limits. In Section 3, an easy way to set parameter bounds for the general controller will be provided.

2.5. The ICT Problem in RTO Form: Summary

Having now gone through all the components of Problem (1) and having provided their ICT analogues, we may make certain remarks and observations.

To start with the positive, almost all of the possible desired specifications in a standard ICT problem are easily stated in RTO terms, although this is not surprising given the generality of (1). Of particular interest with regard to this point are the constraint terms, as the flexibility of the RTO formulation has allowed for us to include limits on the control outputs and inputs, as well as any controller specifications, very easily. To the best of the authors' knowledge, constraints are generally avoided in the majority (though not all [27]) of direct tuning formulations. This is likely because the most commonly used method – the gradient descent – is not well-equipped to deal with them (apart from certain simple kinds, such as the box constraints [23]). Casting the ICT problem in the RTO framework therefore allows us to ignore this limitation.

The other big advantage is that no assumptions are needed on the nature of the controller (or their number, if a system of controllers is considered) – it simply has to be something that can be parametrically tuned, and so one could adapt just about anything. Likewise, the standard restricting assumption of linearity is also not needed, even formally, as the black-box nature of the RTO formulation does not make use of such assumptions since it ignores the actual dynamic behavior of the closed loop and only considers the RTO inputs (the tuning parameters) and RTO outputs (performance, proximity to constraints), both of which are static quantities with a static map between them. As such, the methodology applies just as readily to nonlinear systems as it does to linear ones.

There are, however, points to be contested. The key linking element between RTO and ICT is Assumption 1, which is, at best, only an approximation and merits justification. The driving force behind this assumption is the fact that any deterministic[3] controller with fixed tuning parameters, when applied repeatedly to a closed-loop process to perform the same control task, should always yield the exact same performance (and the exact same input/output trajectories) in the absence of non-repeatable effects such as *input/measurement noise, process degradation, and disturbances*. Indeed, the absence of such effects implies that the δ_k term in (3) is equal to 0, and that the repeatability assumption holds exactly. For cases when these effects are minor and do not influence controller behavior significantly, we expect that a given controller will yield the same performance "more or less", with variations being lumped into δ_k and the major deterministic trends being described by J. This neat way of decoupling the deterministic and stochastic components may not be valid when the non-repeatable effects become large and exert a significant influence on the controller behavior, however. As such, we may view this assumption as an approximation of reality that tends to perfection as the magnitude of the noise/degradation/disturbances in the closed-loop system tends to 0.

There is, as well, the issue of stability. Even with the direct incorporation of constraints in the RTO problem formulation (e.g., via (12) or (13)), there is no true way to incorporate a real constraint on closed-loop stability, as stability is not a real-numbered value that can be measured following a closed-loop experiment (if it were, it would be trivial to include it as an uncertain constraint in the set $\mathbf{G}_p$). Unfortunately, this is a much bigger problem that is not limited to just ICT – one cannot, for the general unknown plant, *ever* guarantee stability via *any* means without making additional assumptions on the nature of the plant. The bright side is that any of the standard stability-guaranteeing methods are easily incorporated into the RTO formulation as certain constraints $\mathbf{G}$, and may be used to limit the adaptations to those controllers that are at least nominally stable. Other workarounds could also be proposed – if the fear of having an unstable closed-loop system stems from having some control output leave its safe

[3] While almost all applied controllers are deterministic, we acknowledge the possibility of stochastic controllers (e.g., those employing Monte Carlo techniques [28]), for which the methods discussed in this work may not be applicable.

operating range, then one could simply introduce an output constraint on that quantity, which, as already shown, is easily integrated into the RTO formulation as $\mathbf{G}_p$.

3. The SCFO Solver and Its Configuration

Having now presented the formulation of the ICT problem as an RTO one, we go on to describe how Problem (1) may be solved. Although (1) is posed like a standard optimization problem, the reader is warned that it is *experimental* in nature and must be solved by iterative closed-loop experiments on the system – i.e., one cannot simply solve (1) by numerical methods since evaluations of functions ϕ_p and $\mathbf{G}_p$ require experiments. A variety of RTO (or "RTO-like") methodologies, all of which are appropriate for solving (1), have been proposed over the years and may be characterized as being model-based (see, e.g., [29,30,31,32]), model-free [33,34], or as hybrids of the two [35,36]. In this work, we opt to use the SCFO solver recently proposed and released by the authors [19,20,21], as it is the only tool available to theoretically guarantee that:

- the RTO scheme converges arbitrarily close to a Karush-Kuhn-Tucker (KKT) point that is, in the vast majority of practical cases, a local minimum,
- the constraints $\mathbf{G}_p(\mathbf{v}) \preceq \mathbf{0}$ and $\mathbf{G}(\mathbf{v}) \preceq \mathbf{0}$ are *never* violated,
- the objective value is consistently improved, with $\phi_p(\mathbf{v}_{k+1}) < \phi_p(\mathbf{v}_k)$ always,

with these properties enforced approximately in practice.

The basic structure of the solver may be visualized as follows:

$$\begin{array}{ccc} \text{Solver Configuration} & & \\ \downarrow & & \\ \text{SCFO Solver} & \longleftarrow & \text{Measurements} \\ \downarrow & & \\ \mathbf{v}_{k+1} & & \end{array}$$

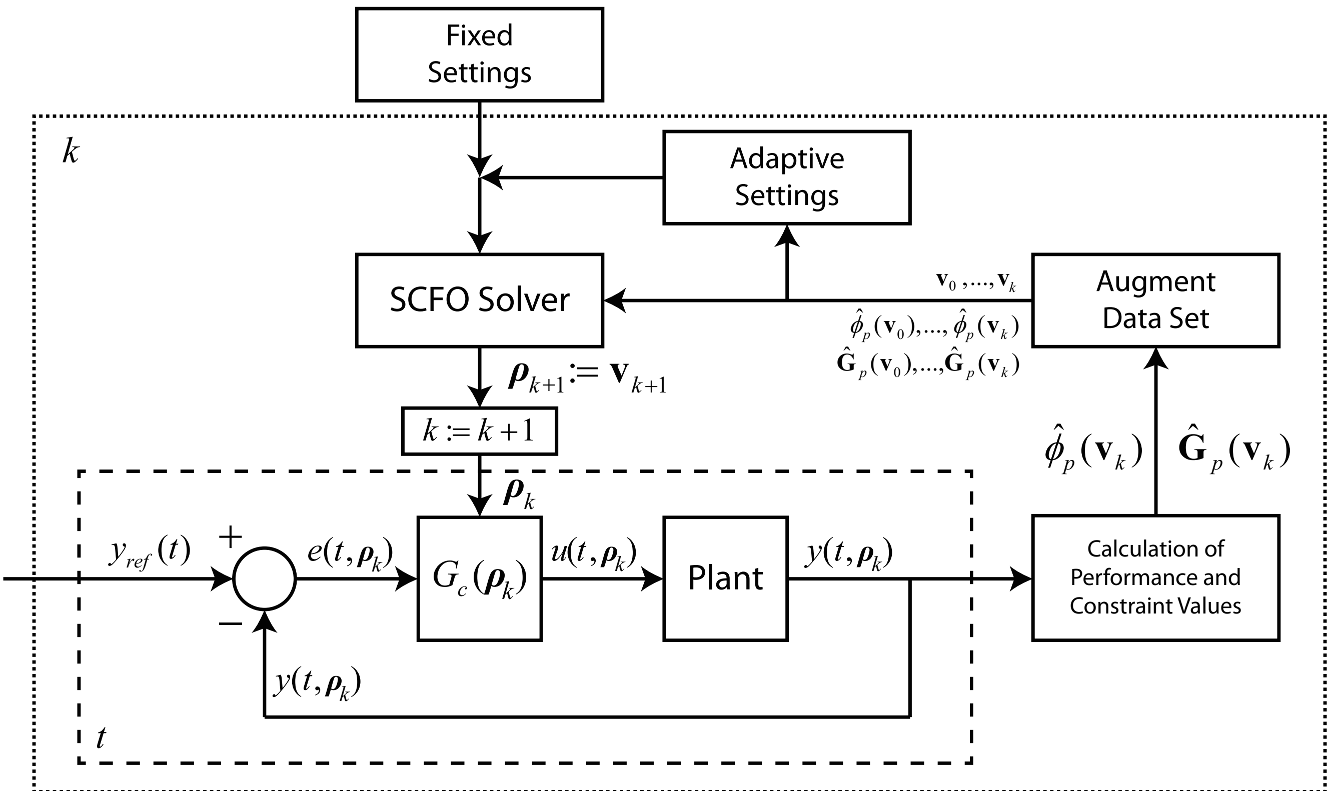
where the majority of the configuration is fixed once and for all while the measurements act as the true iterative components, with the full set of measured data being fed to the solver at each iteration, after which it does all of the necessary computations and outputs the next RTO input to be applied. This is illustrated for the ICT context in Figure 3 (as an extension of Figure 1).

The natural price to pay for such simplicity of implementation is, not surprisingly, the complexity of configuration. Table 1 provides a summary of all of the configuration components, how they are set, and the justifications for these settings. Noting that most of these settings are relatively simple and do not merit further discussion, we now turn our focus to those that do.

3.1. Solver Initialization

Prior to attempting to solve Problem (1), it is strongly recommended that the problem be well-scaled with respect to both the RTO inputs and outputs. For the former, this means that:

Figure 3. The iterative tuning scheme, where the results obtained after each closed-loop experiment on the plant (denoted by the dashed lines) are sent to the RTO loop (denoted by the dotted box), which then appends these data to previous data and uses the full data set to prompt the SCFO solver, as well as to update any data-driven adaptive settings (we refer the reader to Table 1 for which settings are fixed and which are adaptive).



$$v_1^U - v_1^L \approx v_2^U - v_2^L \approx ... \approx v_{n_v}^U - v_{n_v}^L \approx 1,$$

where "≈" may be read as "on the same order of magnitude as". For the RTO outputs, it is advised that both the cost and constraint functions are such that their values vary on the magnitude of 10^0. Once this is done, one may proceed to initialize the data set.

As the solver needs to compute gradient estimates directly from measured data, it is usually needed to generate the $n_v + 1$ measurements (whose corresponding RTO input values should be well-poised for linear regression) necessary for a rudimentary (linear) gradient estimate (see, e.g., [30,37]). In the case that previous measurements are already available (e.g., from experimental studies carried out prior to optimization), then one may be able to avoid this step partially or entirely.

We will, for generality, assume the case where no previous data are available. We will also assume that the initial point, $\mathbf{v}_0 := \boldsymbol{\rho}_0$, has been obtained by some sort of controller design technique. In addition, we require that the initial design satisfy $\mathbf{G}_p(\mathbf{v}_0) \prec \mathbf{0}$, $\mathbf{G}(\mathbf{v}_0) \prec \mathbf{0}$, and $\mathbf{v}^L \prec \mathbf{v}_0 \prec \mathbf{v}^U$ – this is expected to hold intrinsically since one would not start optimizing performance prior to having at least one design that is known to meet the required constraints with at least some safety margin. The next step is then to generate n_v additional measurements, i.e., to run n_v (n_ρ) closed-loop experiments on the plant.

Table 1. Summary of SCFO configuration settings for the ICT problem.

Solver Setting	**Chosen As**	**Justification**	**Type**
Initialization	$n_\rho + 1$ closed-loop experiments	See Section 3.1	–
Optimization target	Scaled gradient descent	See Section 3.2	Adaptive
Noise statistics	Initial experiments at $\boldsymbol{\rho}_0$	See Section 3.3	Fixed
Constraint concavity	None assumed	No reason for assuming this property in ICT context	Fixed
Constraint relaxations	None assumed	For simplicity (should be added if some constraints are soft)	Fixed
Cost certainty	Cost function is uncertain	The performance metric is an unknown function of $\boldsymbol{\rho}$	Fixed
Structural assumptions	Locally quadratic structure	Recommended choice for general RTO problem [21]	Fixed
Minimal-excitation radius	$0.01\left(\rho_1^U - \rho_1^L\right)$	Recommended choice for general RTO problem [21]	Fixed
Lower and upper limits, $\mathbf{v}^L$ and $\mathbf{v}^U$	Controller-dependent or set adaptively	See Section 3.4	Fixed/ Adaptive
Lipschitz and quadratic bound constants	Initial data-driven guess followed by adaptive setting	See Section 3.5	Fixed/ Adaptive
Scaling bounds	Problem-dependent; easily chosen	See [21]	Fixed
Maximal allowable adaptation step, $\Delta\mathbf{v}_{max}$	$0.1\left(\boldsymbol{\rho}^U - \boldsymbol{\rho}^L\right)^T$	Recommended choice for general RTO problem [21]	Fixed

A simple initialization method would be to perturb each controller parameter one at a time, as this would produce a well-poised data set with sufficient excitation in all input directions, thereby making the task of estimating the plant gradient possible. However, such a scheme could be wasteful, especially for ICT problems with many parameters to be tuned. One alternative would be to use smart, model-based initializations [30], but this would require having a plant model. In the case of no model, we propose to use a "smart" perturbation scheme that attempts to begin optimizing performance during the initialization phase, and refer the reader to the appendix for the detailed algorithm.

3.2. The Optimization Target

The target $\mathbf{v}^*_{k+1}$ represents a nominal optimum provided by any standard RTO algorithm that is coupled with the SCFO solver, and as such actually represents the choice of algorithm. This choice is important as it affects performance, with some of the results in [20] suggesting that coupling the SCFO with a "strong" RTO algorithm (e.g., a model-based one) can lead to faster convergence to the optimum. However, the choice is *not* crucial with respect to the reliability of the overall scheme, and so one does not need to be overly particular about what RTO algorithm to use, but should prefer one that generally guides the adaptations in the right direction.

For the sake of simplicity, the algorithm adopted in this work is the (scaled) gradient descent with a unit step size:

$$\mathbf{v}_{k+1}^* = \mathbf{v}_k - \mathbf{H}_k^\dagger \nabla \hat{\phi}_p(\mathbf{v}_k), \tag{14}$$

where both $\mathbf{H}_k$ and $\nabla \hat{\phi}_p(\mathbf{v}_k)$ are data-driven estimates. We refer the reader to the appendix for how these estimates are obtained.

3.3. The Noise Statistics

Obtaining the statistics (i.e., the probability distribution function, or PDF) for the stochastic error terms δ in (3) and (7) is particularly challenging in the ICT context. One reason for this is that these terms do not have an obvious physical meaning, as both (3) and (7), which model the observed performance/constraint values as a sum of a deterministic and stochastic component, are approximations. Furthermore, even if this model were correct, the actual computation of an accurate PDF would likely require a number of closed-loop experiments on the plant that would be judged as excessive in practice.

As will be shown in the first two case studies of Section 4, some level of engineering approximation becomes inevitable in obtaining the PDF for an experimental system. The basic procedure advocated here is to carry out a certain (economically allowable) number of repeated experiments for $\mathbf{v}_0 := \boldsymbol{\rho}_0$ *prior to the initialization step*. In the case where each experiment is expensive (or time consuming) and the total acceptable number is low, one may approximate the δ term by modeling the observed values by a zero-mean normal distribution with a standard deviation equal to that of the data. If the experiments are cheap and a fairly large number (e.g., a hundred or more) is allowed, then the observed data may be offset by its mean and then fed directly into the solver (as the solver builds an approximate PDF directly from the fed noise data).

3.4. Lower and Upper Input Limits

Providing proper lower and upper limits $\mathbf{v}^L$ and $\mathbf{v}^U$ can be crucial to solver performance. As already stated, for the ICT problem these are simply $\mathbf{v}^L := \boldsymbol{\rho}^L$ and $\mathbf{v}^U := \boldsymbol{\rho}^U$, but, as these values may not be obvious for certain controller designs, the user may use adaptive limits that are redefined at each iteration k:

$$\begin{array}{l} \boldsymbol{\rho}_k^L := \boldsymbol{\rho}_k - \mathbf{0.5} \\ \boldsymbol{\rho}_k^U := \boldsymbol{\rho}_k + \mathbf{0.5} \end{array}. \tag{15}$$

As the solver can never actually converge to an optimum that touches these limits, the resulting problem is essentially unconstrained with respect to them, thereby allowing us to configure the solver without affecting the optimality properties of the problem. We note that, while one could use very conservative choices and not adapt them (e.g., $\boldsymbol{\rho}^L := -\mathbf{1000}$ and $\boldsymbol{\rho}^U := \mathbf{1000}$), this is not recommended as it would introduce scaling issues into the solver's subroutines.

3.5. Lipschitz and Quadratic Bound Constants

The solver requires the user to provide the Lipschitz constants (denoted by κ) for all of the functions ϕ_p, $\mathbf{G}_p$, and $\mathbf{G}$. These are implicitly defined as:

$$\underline{\kappa}_{\phi,i} < \left.\frac{\partial \phi_p}{\partial v_i}\right|_{\mathbf{v}} < \overline{\kappa}_{\phi,i}, \quad \underline{\kappa}_{p,i} < \left.\frac{\partial g_p}{\partial v_i}\right|_{\mathbf{v}} < \overline{\kappa}_{p,i}, \quad \underline{\kappa}_{i} < \left.\frac{\partial g}{\partial v_i}\right|_{\mathbf{v}} < \overline{\kappa}_{i},$$

for all $\mathbf{v} \in \{\mathbf{v} : \mathbf{v}^L \preceq \mathbf{v} \preceq \mathbf{v}^U\}$. Quadratic bound constants (denoted by M) on the cost function are also required, and are implicitly defined as:

$$\underline{M}_{ij} < \left.\frac{\partial^2 \phi_p}{\partial v_i \partial v_j}\right|_{\mathbf{v}} < \overline{M}_{ij}, \quad \forall \mathbf{v} \in \{\mathbf{v} : \mathbf{v}^L \preceq \mathbf{v} \preceq \mathbf{v}^U\}.$$

For $\mathbf{G}$, which is easily evaluated numerically, we note that the choice is simple since one can, in many cases, compute these values prior to any implementation.

For $\kappa_{\phi,i}$, $\kappa_{p,i}$, and M_{ij}, the choice is a very difficult one. This is especially true for the ICT problem where such constants have no physical meaning, a trait that may make them easier to estimate for some RTO problems [20]. When a model of the plant is available, one may proceed to compute these values numerically for the modeled closed-loop behavior, and then make the estimates more conservative (e.g., by applying a safety-factor scaling) to account for plant-model mismatch.

For the pure model-free case, we have no choice but to resort to heuristic approaches. As a choice of $\kappa_{\phi,i}$, we thus propose the following (very conservative) estimate based on the gradient estimate for the initial $n_v + 1$ points (23):

$$\overline{\kappa}_{\phi,i}, \underline{\kappa}_{\phi,i} := \pm 10 \|\nabla \hat{\phi}_p\|_\infty, \; i = 1, ..., n_v,$$

as we expect these bounds to be valid unless $\|\nabla \hat{\phi}_p\|_\infty$ is small, which, however, would indicate that we are probably close to a zero-gradient stationary point already, and would have little to gain by trying to optimize performance further if this point were a minimum.

A similar rule is applied to estimate $\kappa_{p,i}$, with:

$$\overline{\kappa}_{p,i}, \underline{\kappa}_{p,i} := \pm 2 \|\nabla \hat{g}_p\|_\infty, \; i = 1, ..., n_v,$$

where the estimate $\nabla \hat{g}_p$ is obtained in the same manner as (23). The choice of 2, as opposed to 10, is made for performance reasons, as making $\kappa_{p,i}$ too conservative can lead to very slow progress in improving performance – this is expected to scale linearly, i.e., if the choice of $\pm 2\|\nabla \hat{g}_p\|_\infty$ leads to a realization that converges in 20 runs, the choice of $\pm 10\|\nabla \hat{g}_p\|_\infty$ may lead to one that converges in 100. Note, however, that this way of defining the Lipschitz constants does not have the same natural safeguard as it does for the cost, and it may happen that $\|\nabla \hat{g}_p\|_\infty \approx 0$ at the initial point even though the gradient may be quite large in the neighborhood of the optimum. When this is so, an alternate heuristic choice is to set:

$$\overline{\kappa}_{p,i}, \underline{\kappa}_{p,i} := \pm 2 \frac{-\underline{g}_p}{v_i^U - v_i^L}, \; i = 1, ..., n_v,$$

where $\underline{g}_p$ denotes the smallest value that the constraint can take in practice, with $\underline{g}_p \leq g_p(\mathbf{v}), \; \forall \mathbf{v} \in \{\mathbf{v} : \mathbf{v}^L \preceq \mathbf{v} \preceq \mathbf{v}^U\}$. Combining the two, one may then use the heuristic rule:

$$\overline{\kappa}_{p,i}, \underline{\kappa}_{p,i} := \pm 2 \max\left(\|\nabla \hat{g}_p\|_\infty, \frac{-\underline{g}_p}{v_i^U - v_i^L} \right), \; i = 1, ..., n_v.$$

However, it may still occur that this choice is not conservative enough. This lack of conservatism may be proven if a given constraint $g_p(\mathbf{v}) \leq 0$ is violated for one of the runs, since sufficiently conservative Lipschitz estimates will usually guarantee that this is not the case (provided that the noise statistics are sufficiently accurate). As such, the following adaptive refinement of the Lipschitz constants is proposed to be done online when/if the constraint is violated with sufficient confidence:

$$g_p(\mathbf{v}_k) \geq 3\sigma_g \rightarrow \overline{\kappa}_{p,i} := 2\overline{\kappa}_{p,i},\ \underline{\kappa}_{p,i} := 2\underline{\kappa}_{p,i},$$

where σ_g represents the estimated standard deviation of the non-repeatability noise term δ for g_p.

For the quadratic bound constants M, which represent lower and upper bounds on the second derivatives of ϕ_p, we propose to use the estimate of the Hessian $\mathbf{H}_k$ as obtained in Section 3.2 (see Appendix), together with a safety factor, η, to define the bounds at each iteration k as:

$$\overline{M}_{ij}, \underline{M}_{ij} := H_{k,ij} \pm \eta |H_{k,ij}|, \tag{16}$$

with η initialized as 1. Since such a choice may also suffer from a lack of conservatism, an adaptive algorithm for η is put into place. Since a common indicator of choosing M values that are not conservative enough is the failure to decrease the cost between consecutive iterations, the following law is proposed for any iterations where the solver applied the SCFO conditions but increased (with sufficient confidence) the value of the cost [21]:

- If $\hat{\phi}_p(\mathbf{v}_k) - 4\sigma_\phi \geq \min\limits_{i=0,\ldots,k-1} \hat{\phi}_p(\mathbf{v}_i)$, then set $\eta := \eta + 1$,
- otherwise, set $\eta := \eta - 0.5$, with $\eta < 0 \rightarrow \eta := 0$,

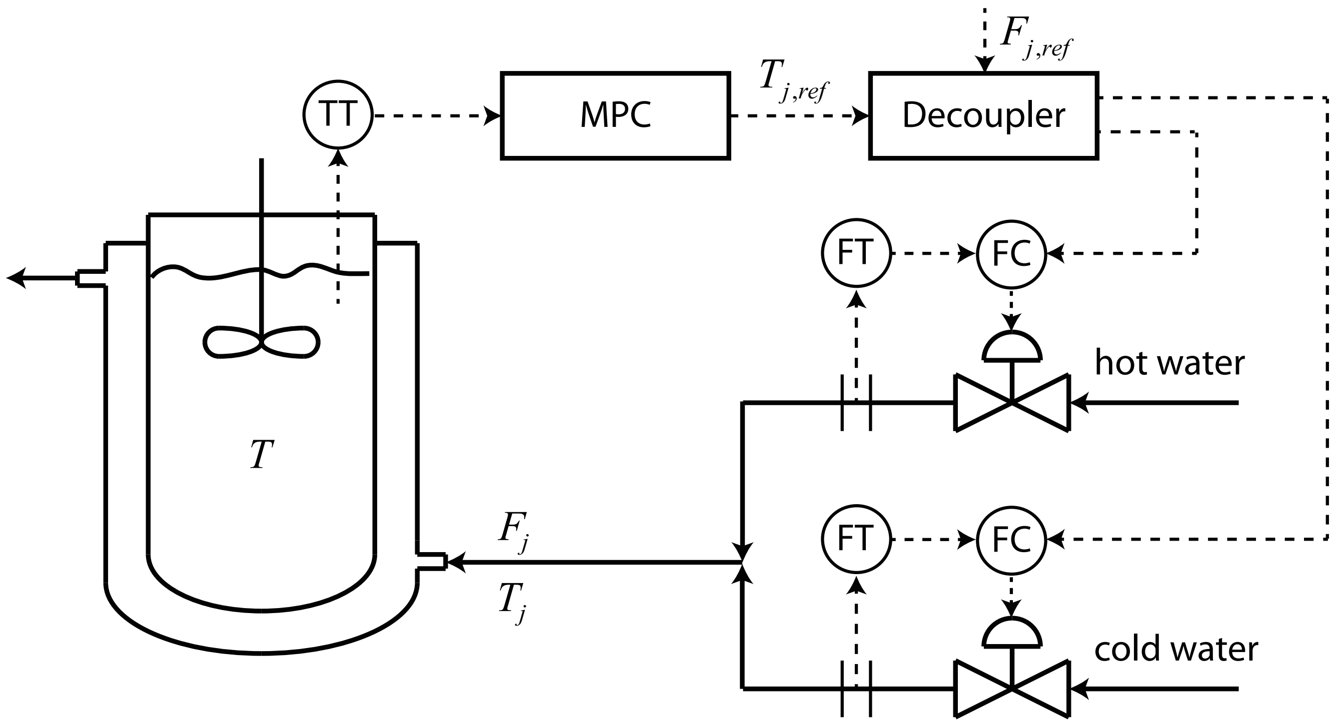
where σ_ϕ is the estimated standard deviation for the non-repeatability noise term in the measurement of J_k. The essence of this update law is to make M more conservative (by increasing η) whenever the performance is statistically likely to have increased in the recent adaptation, and to relax the conservatism otherwise, though at only twice the rate that it would be increased. Such a scheme essentially ensures that the M constants become conservative enough to continually guarantee improved performance with an increasing number of iterations.

4. Case Studies

The proposed method was applied to four different problems, of which the first two are of particular interest as they were carried out on experimental systems and demonstrate the reliability and effectiveness of the proposed approach when applied in settings where neither the plant nor the non-repeatability noise terms are known. Of these two, the first represents a typical batch scenario with fairly slow dynamics and time-consuming, expensive experiments for which an MPC controller is employed (Section 4.1), while the latter represents a much faster mechanical system where the optimization of the controller parameters for a general fixed-order controller must be carried out quickly due to real-time constraints, but where a single run is inexpensive (Section 4.2).

The last two studies, though lacking the experimental element, are nevertheless of interest as they make a link with similar work carried out by other researchers (Section 4.3) and generalize the method to *systems* of controllers with an additional challenge in the form of an output constraint (Section 4.4).

Figure 4. Schematic of the jacketed stirred tank and the cascade control system used to control the water temperature inside the tank. The reference ($F_{j,ref}$) for the water flow to the jacket (F_j) was fixed at 2 L/min.



In both of these cases, we have chosen to simplify things by assuming to know the noise statistics of the relevant δ terms and to let the repeatability assumption hold exactly.

In each of the four studies, we have used the configuration proposed in Section 3 and so will not repeat these details here. However, we will highlight those components of the configuration that are problem-dependent and will explain how we obtained them for each case.

4.1. Batch-to-Batch Temperature Tracking in a Stirred Tank

The plant in question is a jacketed stirred water tank, where a cascade system is used to control the temperature inside the tank by having an MPC controller manipulate the setpoint temperature of the jacket inlet, which is in turn tracked by a decoupled system of two PI controllers that manipulate the flow rates of the hot and cold water streams that mix to form the jacket inlet (Figure 4). As this system is essentially identical to what has been previously reported [38], we refer the reader to the previous work for all of the implementation details.

As the task of tracking an "optimal" temperature profile is fairly common in batch processes and the failure to do so well can lead to losses in product quality, a natural ICT problem arises in these contexts as it is desired that the temperature stay as close to the prescribed optimal setpoint trajectory as possible. In this particular case study, the controller that is tasked with this job is the MPC controller whose tunable parameters include:

- the output weight that controls the trade-off between controller aggressiveness and output tracking,
- the bias update filter gain, which acts to ensure offset-free tracking,

- the control and prediction horizons that dictate how far ahead the MPC attempts to look and control,

all of which act to change the objective function at the heart of the MPC controller [38]. For this problem, we decided to vary the output weight between 0.1 and 10 (i.e., covering three orders of magnitude) and defined its logarithm as the first tunable variable ρ_1. Our reason for choosing the logarithm, instead of the actual value, was due to the sensitivity of the performance being more uniform with respect to the magnitude difference between the priorities given to controller aggressiveness and output tracking (e.g., changing the output weight from 0.1 to 1.0 was expected to have a similar effect as changing it from 1.0 to 10). The bias update filter gain, defined as the second variable ρ_2, was forced to vary between 0 and 1 by definition. The control and prediction horizons, m and n, were both allowed to vary anywhere between 2 and 50 and, as this variance was on the magnitude of 10^2, were divided by 100 so as to have comparable scaling with the other parameters, with $\rho_3 \triangleq m/100$ and $\rho_4 \triangleq n/100$. We note as well that the horizons were constrained to be integers, whereas the solver provided real numbers, and so any answer provided by the solver had to be rounded to the nearest integer to accommodate these constraints.

As this system was fairly slow/stable and controller aggressiveness was not really an issue, and as there was no strong preference between using hot or cold water, the performance metric simply consisted of minimizing the tracking error (i.e., the general metric in (5) with $\lambda_1 := 1$ and $\lambda_2 := \lambda_3 := \lambda_4 := 0$) over a batch time of $t_b = 40$ minutes. The setpoint trajectory to be tracked consisted of maintaining the temperature at 52°C for 10 minutes, cooling by 4°C over 10 minutes, and then applying a quadratic cooling profile for the remainder of the batch. Each batch was initialized by setting the jacket inlet to 55°C and starting the batch once the tank temperature rose to 52°C.

The certain inequality constraint $\rho_3 \leq \rho_4$ was enforced as this was needed by definition – see (9) – thereby contributing to yield the following ICT problem in RTO form:

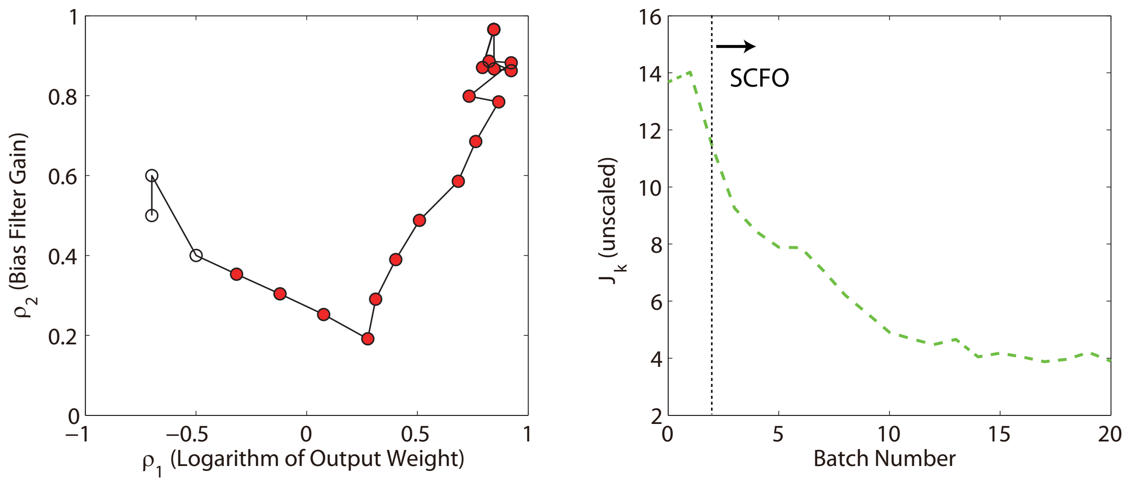
$$
\begin{array}{lll}
\underset{\boldsymbol{\rho}}{\text{minimize}} & \dfrac{1}{J_0}\displaystyle\int_0^{40} \left[T_{ref}(t) - T(t,\boldsymbol{\rho})\right]^2 dt & \big\}\ \phi_p(\mathbf{v}) \\
\text{subject to} & \rho_3 - \rho_4 \leq 0 & \big\}\ \mathbf{G}(\mathbf{v}) \preceq \mathbf{0} \\
& -1 \leq \rho_1 \leq 1 & \\
& 0 \leq \rho_2 \leq 1 & \\
& 0.02 \leq \rho_3 \leq 0.50 & \Big\}\ \mathbf{v}^L \preceq \mathbf{v} \preceq \mathbf{v}^U \\
& 0.02 \leq \rho_4 \leq 0.50 &
\end{array}, \qquad (17)
$$

where we scaled the performance metric by dividing by its initial value (thereby giving us a base performance metric value of 1, which was then to be lowered). We also note that in practice measurements were collected every 3 seconds, and so the integral of the squared error was evaluated discretely. The initial parameter set was chosen, somewhat arbitrarily, as $\boldsymbol{\rho}_0 := [-0.7\ 0.5\ 0.3\ 0.3]^T$.

Prior to solving (17), we first solved an easier problem where ρ_3 and ρ_4 were fixed at their initial values and only ρ_1 and ρ_2 were optimized over (these two parameters being expected to be the more influential of the four):

$$
\begin{array}{lll}
\underset{\rho_1,\rho_2}{\text{minimize}} & \dfrac{1}{J_0}\displaystyle\int_0^{40} \left[T_{ref}(t) - T(t,\rho_1,\rho_2)\right]^2 dt & \big\}\ \phi_p(\mathbf{v}) \\
\text{subject to} & -1 \leq \rho_1 \leq 1 & \Big\}\ \mathbf{v}^L \preceq \mathbf{v} \preceq \mathbf{v}^U \\
& 0 \leq \rho_2 \leq 1 &
\end{array}. \qquad (18)
$$

Figure 5. The parameter adaptation plot (left) and the measured performance metric (right) for the solution of Problem (18). Hollow circles on the left indicate batches that were carried out as part of the initialization (prior to applying the solver). Likewise, the dotted vertical line on the right shows the iteration past which the parameter adaptations were dictated by the SCFO solver.

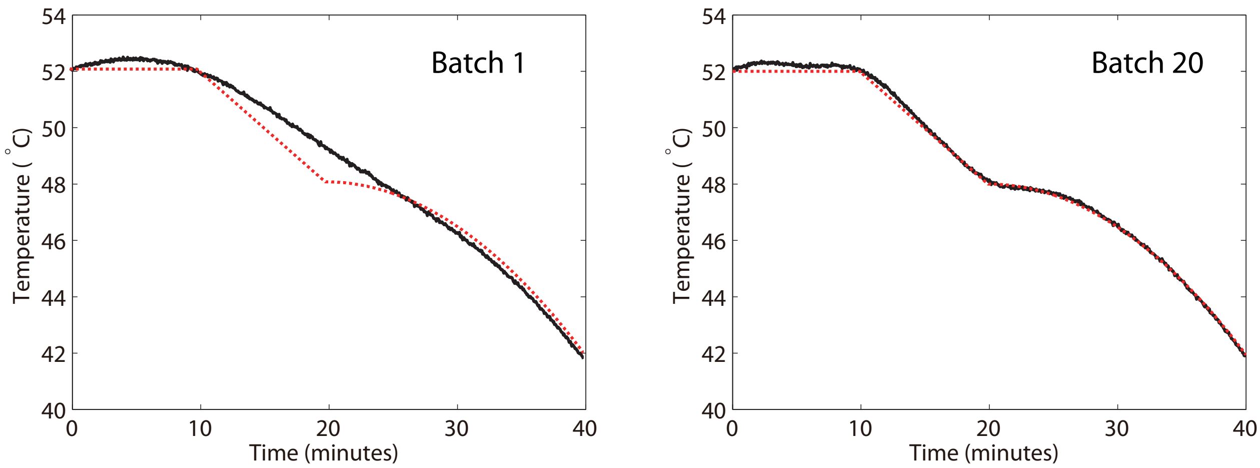


In order to approximate the non-repeatability noise term for the performance, a total of 8 batches were run with the initial parameter set $\boldsymbol{\rho}_0$, with the (unscaled) performance metric values obtained for those experiments being: 13.45, 13.31, 13.46, 14.25, 13.80, 13.44, 13.72, 13.98 (their mean then being taken as the scaling term J_0). Rather than attempt to run more experiments, which, though it could have improved the accuracy of our approximation, would have required even more time (each batch already requiring 40 minutes, with an additional 20-30 minutes of inter-batch preparation), we chose to approximate the statistics of the non-repeatability noise term by a zero-mean normal distribution with the standard deviation of the data, i.e., 0.32.

The map of the parameter adaptations and the values of the measured performance metric are given in Figure 5, with a visual comparison of the tracking before and after optimization given in Figure 6. It is seen that the majority of the improvement is obtained by about the tenth batch, with only minor improvements afterwards, and that monotonic improvement of the control performance is more or less observed.

We also note that the solution obtained by the solver is very much in line with what an engineer would expect for a system with slow dynamics such as this one, in that one should increase both the output weight so as to have better tracking and set the bias update filter gain close to its maximal value (both of these actions could have potentially negative effects for faster, less stable systems, however). As such, the solution is not really surprising, but it is still encouraging that a method with absolutely no knowledge embedded into it has been able to find the same in a relatively low number of experiments. It is also interesting to note that the non-repeatability noise in the measured performance metric originally puts us on the wrong track, as increasing the bias update filter gain does *not* improve the observed performance for Batch 1, though it probably should, and so the solver then spends the first 6 adaptations decreasing the bias filter gain in the belief that doing so should improve performance. However, it is able to recover by Batch 7 and to go in the right direction afterwards – this is likely due to the internal

Figure 6. The visual improvement in the temperature profile tracking from Batch 1 to Batch 20. The dotted (red) lines denote the setpoint, while the solid (black) lines denote the actual measured temperature.



gradient estimation algorithm of the solver having considered all of the batches and thereby decoupled the effects of the two parameters.

Problem (17) was then solved by similar means, though we used all of the data obtained previously to help "warm start" the solver. As the results were similar to what was obtained for the two-parameter case, we only give the measured performance metric values and the temperature profile at the final batch in Figure 7. We also note that the parameter values at the final batch were $\boldsymbol{\rho}_{30} = [0.89\ 0.95\ 0.07\ 0.12]^T$, from which we see that, while all four variables were clearly adapted and the solver chose to lower both the control and prediction horizons, any extra performance gains from doing this (if any) appear to have been marginal when compared to the simpler two-parameter problem. This is also in line with our intuition (i.e., that the output weight and bias filter gain are more important) and reminds us of a very important RTO concept: just because one has many variables that one *can* optimize over does not mean that one *should*, as RTO problems with more optimization variables are generally expected to converge slower and, as seen here, may not be worth the effort.

4.2. Periodic Setpoint Tracking in a Torsional System

In this study, we consider the three-disk torsional system shown in Figure 8 (the technical details of which may be found in [39]). Here, the control input is defined as the voltage of the motor located near the bottom of the system, with the control output taken as the angular position of the top disk.

To define an ICT problem, we generalize the idea of a "run" or a "batch" as seen in the previous example and consider instead a "window" of a periodic sinusoidal trajectory defined by:

$$y_{ref}(t) = -2\cos\frac{\pi t}{6},$$

with t given in seconds. As the same trajectory is repeated every 12 seconds, we can essentially consider each 12-second window as a "run" (or a "batch") as shown in Figure 9 and adapt the relevant controller parameters in the sampling time period between two consecutive windows.

Figure 7. The measured performance metric for the solution of Problem (17), together with the tracking obtained for the final batch.

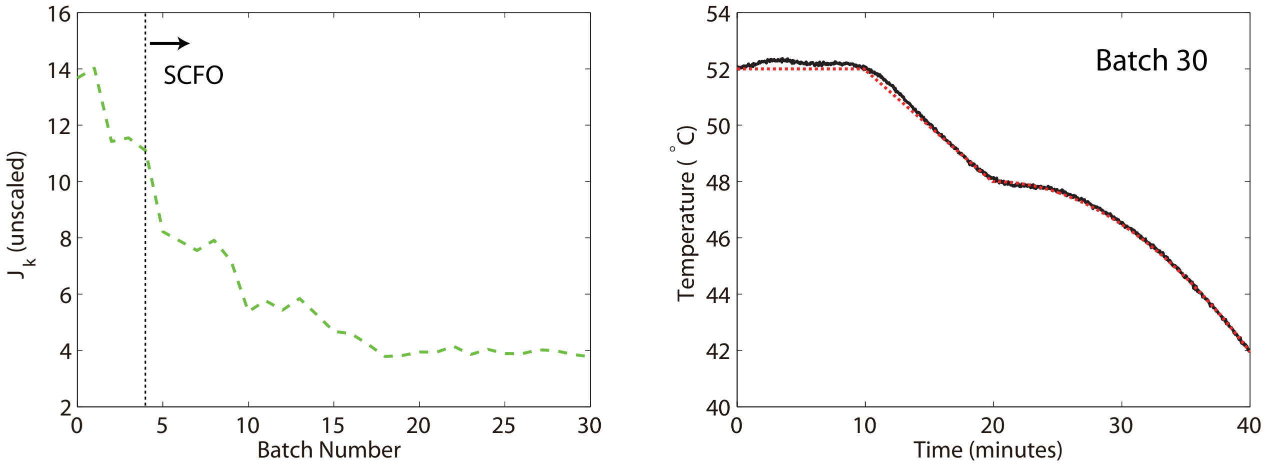
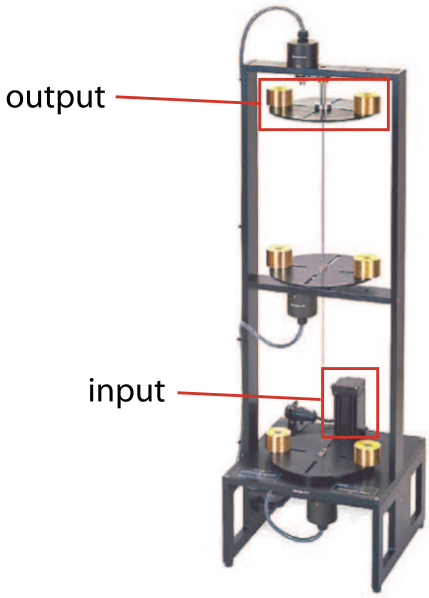
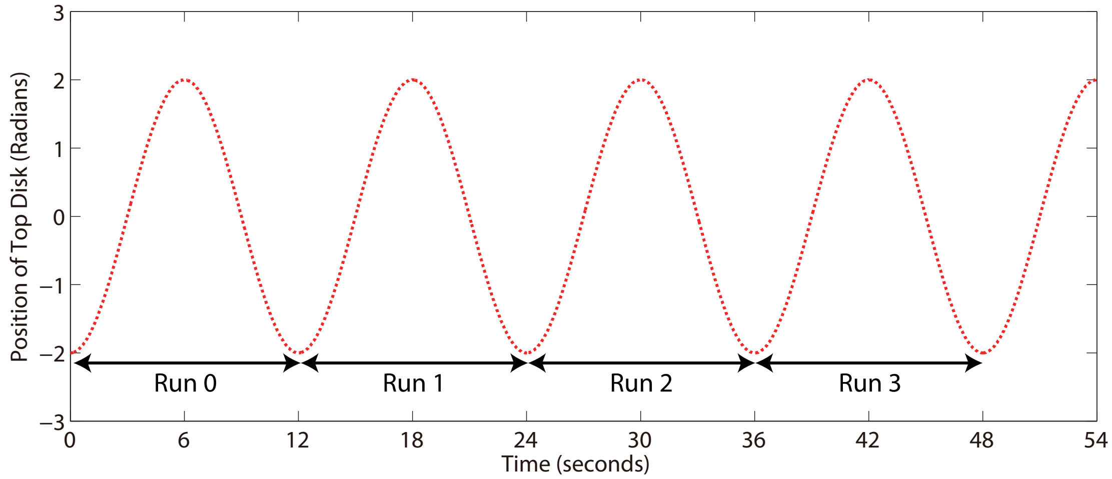


Figure 8. The ECP 205 torsional system.



Not surprisingly, this presents a computational challenge, as the sampling time for this system is only 60 milliseconds, which is, with the current version of the solver, insufficient – the solver needing at least a few seconds to provide a new choice of parameters. While a much simpler implementation that satisfies this real-time constraint has already been successfully carried out on the same system [24], we choose to apply the methodology presented in this paper by using a wait-and-synchronize approach. Here, the solver takes all of the available data and starts its computations, with no adaptation of the parameters being done until the solver's computations are finished. Afterwards, the solver waits until these new parameters are applied and the results for the corresponding run obtained, after which the new data is fed into the solver and the cycle restarts. The noted drawback of this approach is that we have to wait, on average, 2-3 runs (24-36 seconds) for an adaptation to take place, although the positive side of this is that the resulting data is generally less noisy due to the repeated experiments.

Figure 9. The generalization of "run-to-run" tuning to a system with a periodic setpoint trajectory. Only the setpoint is given here.



The controller employed is the discrete fixed-order controller given in (10), with the numerator and denominator coefficients being the (five) tuned parameters. The performance metric used is again a case of the general metric (5), but this time equal priority is given to tracking, controller aggressiveness, and the smoothness of the output trajectory, with $\lambda_1 := \lambda_3 := \lambda_4 := 1$ and $\lambda_2 := 0$.

As the poles of the controller are also being adapted (due to the adaptation of the denominator coefficients), controller stability constraints, as already derived in (11) and (12), are added to the ICT problem (with a tolerance of $\varepsilon := 0.01$):

$$\begin{array}{l} |\rho_5| - 0.99 \leq 0 \\ |\rho_4 - \rho_4\rho_5| - |1 - \rho_5^2| + 0.01 \leq 0 \end{array},$$

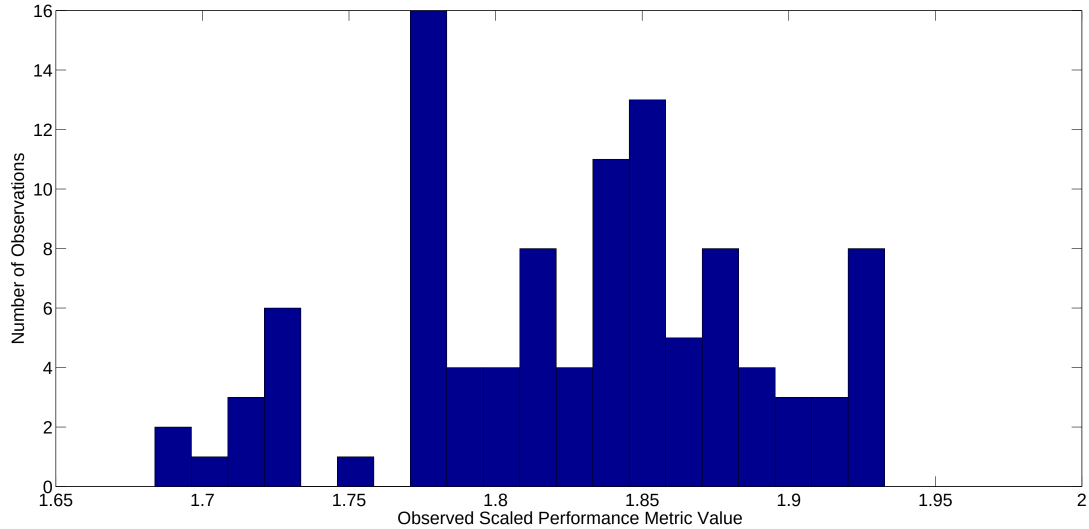
and are recast into differentiable form (as the solver requires $\mathbf{G}$ to be differentiable):

$$\begin{array}{l} \rho_5 - 0.99 \leq 0 \\ -\rho_5 - 0.99 \leq 0 \\ \rho_4 - \rho_4\rho_5 - (1 - \rho_5^2) + 0.01 \leq 0 \\ -\rho_4 + \rho_4\rho_5 - (1 - \rho_5^2) + 0.01 \leq 0 \end{array},$$

where we have used $|\rho_5| \leq 0.99 \Rightarrow |1 - \rho_5^2| = 1 - \rho_5^2$ in the reformulation of the second set.

The adaptive limits of (15) are used to constrain the individual parameters, thereby leading to the (adaptive) ICT-RTO problem:

Figure 10. A twenty-bin histogram representation of the observed scaled performance metric values for a hundred runs with the initial parameter set (Problem (19)).



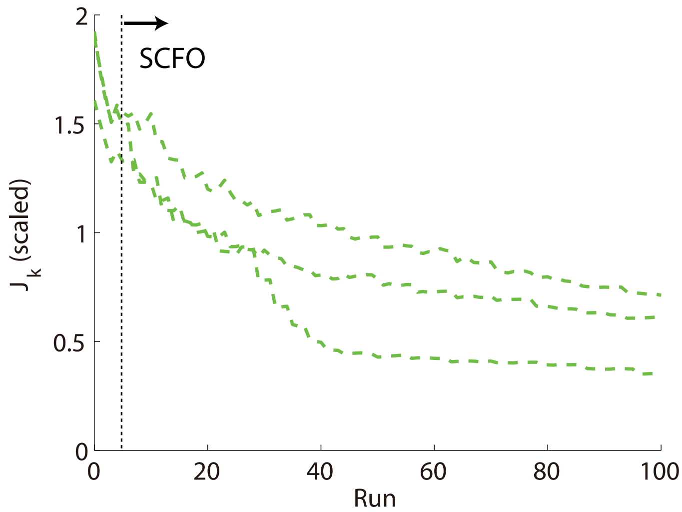
$$
\begin{array}{rll}
\underset{\boldsymbol{\rho}}{\text{minimize}} & \frac{1}{100}\int_0^{12}\left([y_{ref}(t)-y(t,\boldsymbol{\rho})]^2+\dot{u}^2(t,\boldsymbol{\rho})+\dot{y}^2(t,\boldsymbol{\rho})\right)dt & \Big\}\ \phi_p(\mathbf{v}) \\
\text{subject to} & \rho_5-0.99\leq 0 & \\
& -\rho_5-0.99\leq 0 & \\
& \rho_4-\rho_4\rho_5-(1-\rho_5^2)+0.01\leq 0 & \\
& -\rho_4+\rho_4\rho_5-(1-\rho_5^2)+0.01\leq 0 & \Big\}\ \mathbf{G}(\mathbf{v})\preceq\mathbf{0} \\
& \rho_{k,1}-0.5\leq\rho_1\leq\rho_{k,1}+0.5 & \\
& \rho_{k,2}-0.5\leq\rho_2\leq\rho_{k,2}+0.5 & \\
& \rho_{k,3}-0.5\leq\rho_3\leq\rho_{k,3}+0.5 & \\
& \rho_{k,4}-0.5\leq\rho_4\leq\rho_{k,4}+0.5 & \\
& \rho_{k,5}-0.5\leq\rho_5\leq\rho_{k,5}+0.5 & \Big\}\ \mathbf{v}^L\preceq\mathbf{v}\preceq\mathbf{v}^U
\end{array}
\quad , \qquad (19)
$$

where we scale the performance metric by 10^2 so as to make it vary on the magnitude of 10^0.

An initial parameter set of $\boldsymbol{\rho}_0 := [1.00\ 2.77\ \ -2.60\ 1.00\ 0.50]^T$ was chosen and corresponds to an *ad hoc* initial design found by a mix of both simulation and hand tuning. To estimate the noise statistics of the non-repeatability noise term in the performance metric, the system was operated at $\boldsymbol{\rho}_0$ for 20 minutes, which produced a total of 100 performance metric measurements (see Figure 10). These were then offset by their mean to generate the estimated noise samples, with the latter being fed directly into the solver, which would then build an approximate distribution function for them.

Problem (19) was solved a total of three times for 20 minutes of operation (100 runs), with the performance improvements for the three trials given in Figure 11 and the visual improvement for the middle case ("middle" with regard to the final performance metric value) given in Figure 12. We note the variability in convergence behavior for the three cases (both in terms of speed and the performance achieved after 100 runs), which was largely caused by the solver converging to different minima, but

Figure 11. Performance improvement over 100 runs of operation for three different trials (dashed lines) of Problem (19).



note as well that all three follow the same "reliable" trend, in that performance is always improved with a fairly consistent decrease in the metric value over the course of operation.

4.3. PID Tuning for a Step Setpoint Change

We consider the problem previously examined in [13,22], where the parameters of a PID controller are to be tuned for the closed-loop system given by:

$$Y(s) = \frac{G_{ref}(s)G_p(s)}{1+G_y(s)G_p(s)}Y_{ref}(s),$$

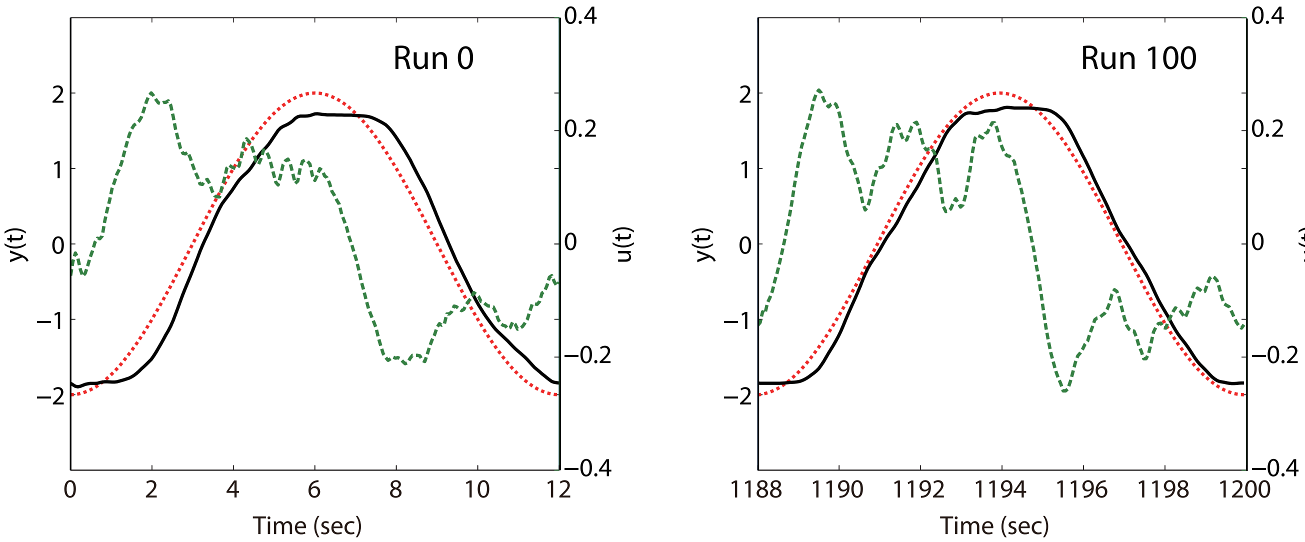
with the PID parameters K_p, τ_I, and τ_D being used to define $G_{ref}(s)$ and $G_y(s)$ as:

$$\begin{array}{c} G_{ref}(s) = K_p\left(1+\dfrac{1}{\tau_I s}\right) \\ G_y(s) = K_p\left(1+\dfrac{1}{\tau_I s}+\tau_D s\right) \end{array},$$

and $G_p(s)$ being the plant, whose definition will be varied for study purposes. The case of a setpoint step change ($Y_{ref}(s) = 1/s$) is considered, with only the tracking error to be minimized over a "masked" operating length, where a mask of t_m is applied so as not to penalize for errors on the interval $t \in [0, t_m]$, as proposed in [22].

Since the controller gain, K_p, is expected to vary on a magnitude of about 10^0, it does not need scaling and so we define $\rho_1 \triangleq K_p$. For both τ_I and τ_D we assume the possibility of greater variations, on the magnitude of 10^1 (as has been suggested in both [22] and [13]), and thus define the scaled second and third parameters as $\rho_2 \triangleq \tau_I/10$ and $\rho_3 \triangleq \tau_D/10$. Since we do not know *a priori* what $\boldsymbol{\rho}^L$ and $\boldsymbol{\rho}^U$ for a PID controller should be, but do realize that both τ_I and τ_D should be positive, the adaptive definition of the lower and upper limits with the positivity constraints respected is chosen to yield the ICT problem in RTO form:

Figure 12. Difference in control input and output profiles between the first and final runs of Problem (19), with the dashed green line used to denote the input (motor voltage) values.



$$
\left.\begin{array}{ll}
\underset{\boldsymbol{\rho}}{\text{minimize}} & \dfrac{1}{J_0}\displaystyle\int_{t_m}^{t_b}\left[y_{ref}(t)-y(t,\boldsymbol{\rho})\right]^2 dt
\end{array}\right\} \phi_p(\mathbf{v})
$$
$$
\left.\begin{array}{ll}
\text{subject to} & \rho_{k,1}-0.5 \leq \rho_1 \leq \rho_{k,1}+0.5 \\
& \max(\rho_{k,2}-0.5,0.01) \leq \rho_2 \leq \rho_{k,2}+0.5 \\
& \max(\rho_{k,3}-0.5,0.01) \leq \rho_3 \leq \rho_{k,3}+0.5
\end{array}\right\} \mathbf{v}^L \preceq \mathbf{v} \preceq \mathbf{v}^U, \qquad (20)
$$

where we scale the cost function by dividing by the performance metric value for the original parameter set.

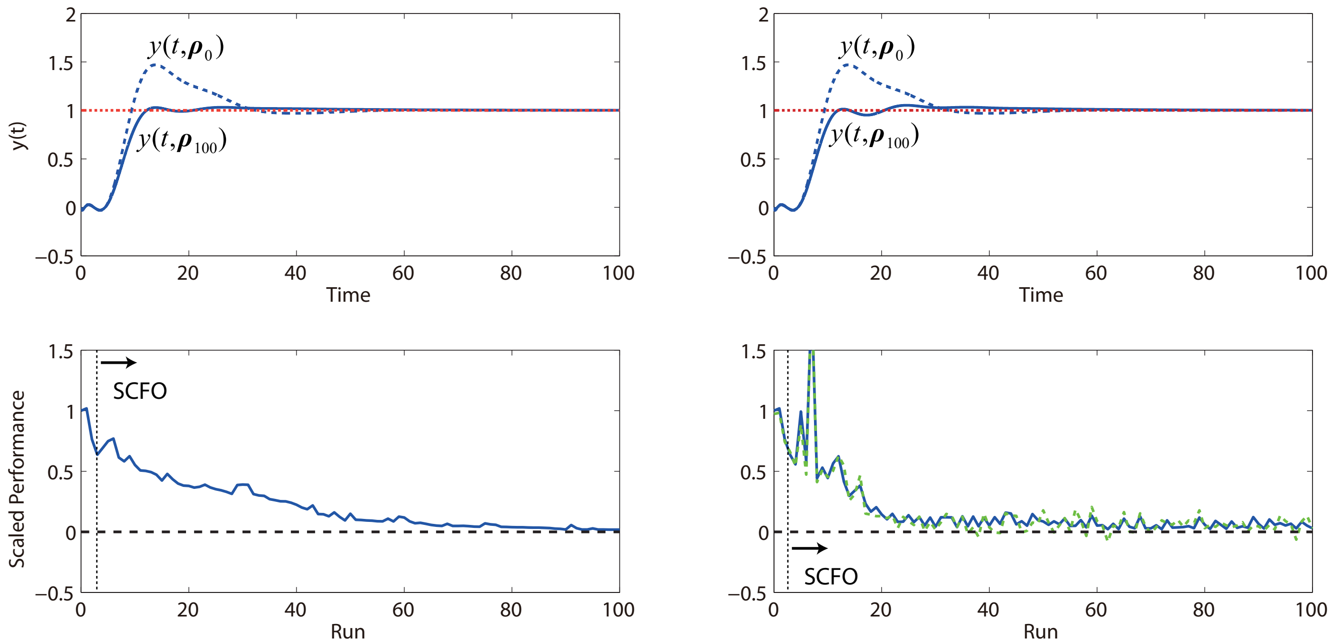
As done in [13], the original parameter set is chosen as the set found by Ziegler-Nichols tuning. The following three studies are considered here:

$$
\begin{array}{llll}
\text{Study 1}: & G_p(s)=\dfrac{1}{1+20s}e^{-5s}, & t_m := 10,\ t_b=100, & \boldsymbol{\rho}_0 := [4.06\ 0.93\ 0.23]^T \\
\text{Study 2}: & G_p(s)=\dfrac{1}{1+20s}e^{-20s}, & t_m := 50,\ t_b=300, & \boldsymbol{\rho}_0 := [1.33\ 3.10\ 0.65]^T \\
\text{Study 3}: & G_p(s)=\dfrac{1}{(1+10s)^8}, & t_m := 140,\ t_b=500, & \boldsymbol{\rho}_0 := [1.10\ 7.59\ 1.90]^T
\end{array}.
$$

So as to study the effect of non-repeatability noise, each observed performance metric value is corrupted with an additive error from $\mathcal{N}(0,(0.05J_0)^2)$, i.e., by an additive error with a standard deviation that is chosen as 5% of the original performance metric value (assumed known for solver configuration). Noiseless scenarios were simulated as well.

The results for the three studies are provided in Figures 13-15. On the whole, we see that the solver reliably optimizes control performance in both the noiseless and noisy scenarios, even though we note that the rate of improvement can vary from problem to problem. For the noisy cases, we generally see more "bumps" in the convergence trajectory, which should not be surprising given (a) the added difficulty for the solver in estimating local derivatives and (b) the reduced conservatism in the estimation of the quadratic bound constants M, for which the safety factor η in (16) is generally augmented less frequently when noise is present. However, for the latter point, we see that there is an upside with regard

Figure 13. Performance obtained by iterative tuning for both the noiseless (left) and noisy (right) cases of Study 1 of Problem (20), with the solid blue line used to denote the "true" performance of the closed-loop system and the green dashed line used to denote what is actually observed (and provided to the solver). In both cases, the SCFO solver brings the closed-loop performance metric value close to its global minimum of 0 (marked by the black dashed line in the lower plots).



to convergence speed. Because the values of M tend to be less conservative in the presence of noise, the algorithm tends to take larger steps and progresses quicker towards the optimum, as is witnessed in both Figures 13 and 15. We do note the occasional danger of performance worsening due to tuning, but this is almost always restricted to the earlier runs when the solver is relatively "data-starved".

4.4. Tuning a System of PI Controllers for Setpoint Tracking and Disturbance Rejection

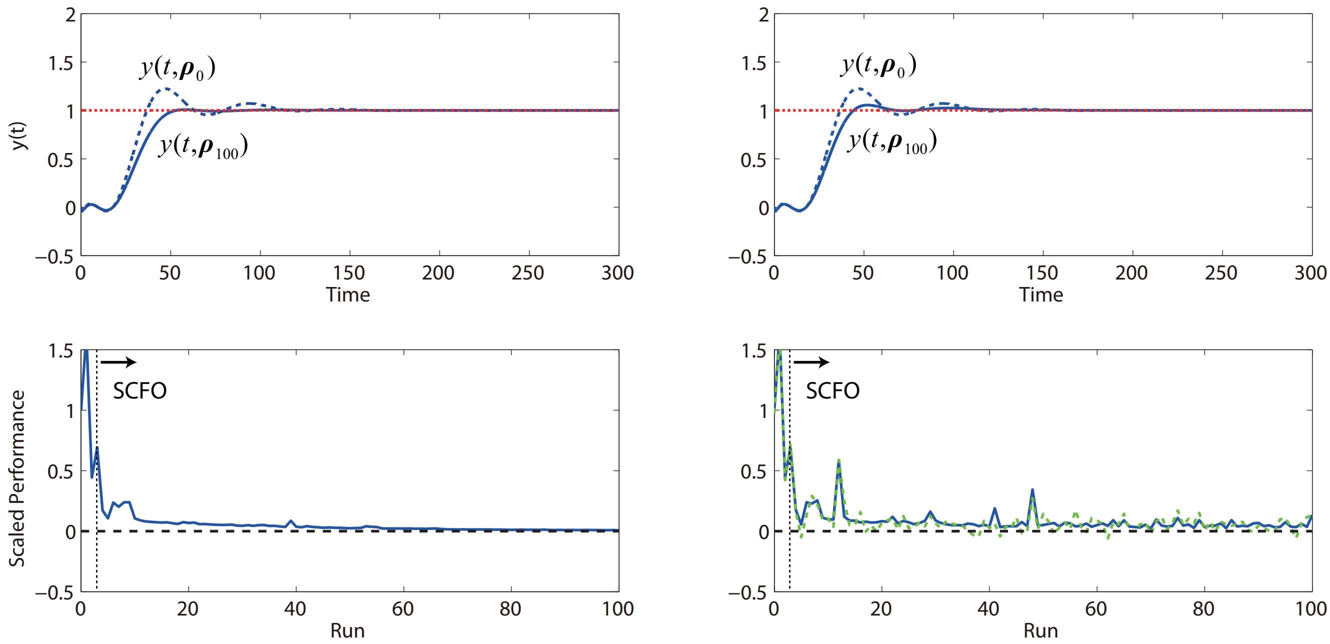
Here, we consider the following 5-input, 5-output dynamical system:

$$\begin{array}{rcl} \ddot{y}_1(t)+\dot{y}_1(t)+y_1(t) & = & u_1(t)-0.033u_3(t)-0.067u_4(t)-0.1u_5(t) \\ \ddot{y}_2(t)+0.1\dot{y}_2(t)+y_2(t) & = & 0.1u_1(t)+2u_2(t)+0.033u_3(t)-0.033u_5(t) \\ \ddot{y}_3(t)+5\dot{y}_3(t)+y_3(t) & = & 0.167u_1(t)+0.133u_2(t)+3u_3(t)+0.067u_4(t)+0.033u_5(t) \\ \ddot{y}_4(t)+2\dot{y}_4(t)+y_4(t) & = & 0.233u_1(t)+0.2u_2(t)+0.167u_3(t)+4u_4(t)+0.1u_5(t) \\ \ddot{y}_5(t)+3\dot{y}_5(t)+y_5(t) & = & 0.3u_1(t)+0.267u_2(t)+0.233u_3(t)+0.2u_4(t)+5u_5(t) \end{array} . \quad (21)$$

While the user cannot be assumed to know the plant (21), we will assume that they have been able to properly decouple the system with the input-output pairings of $u_i \rightarrow y_i,\ i=1,...,5$ (as this is evidently the superior choice if one considers the relative gains). A system of five PI controllers is used for the pairings:

$$u_i(t)=K_{p,i}\left(e_i(t)+\frac{1}{\tau_{I,i}}\int_0^t e_i(t)dt\right), \quad i=1,...,5,$$

Figure 14. Performance obtained by iterative tuning for Study 2 of Problem (20).



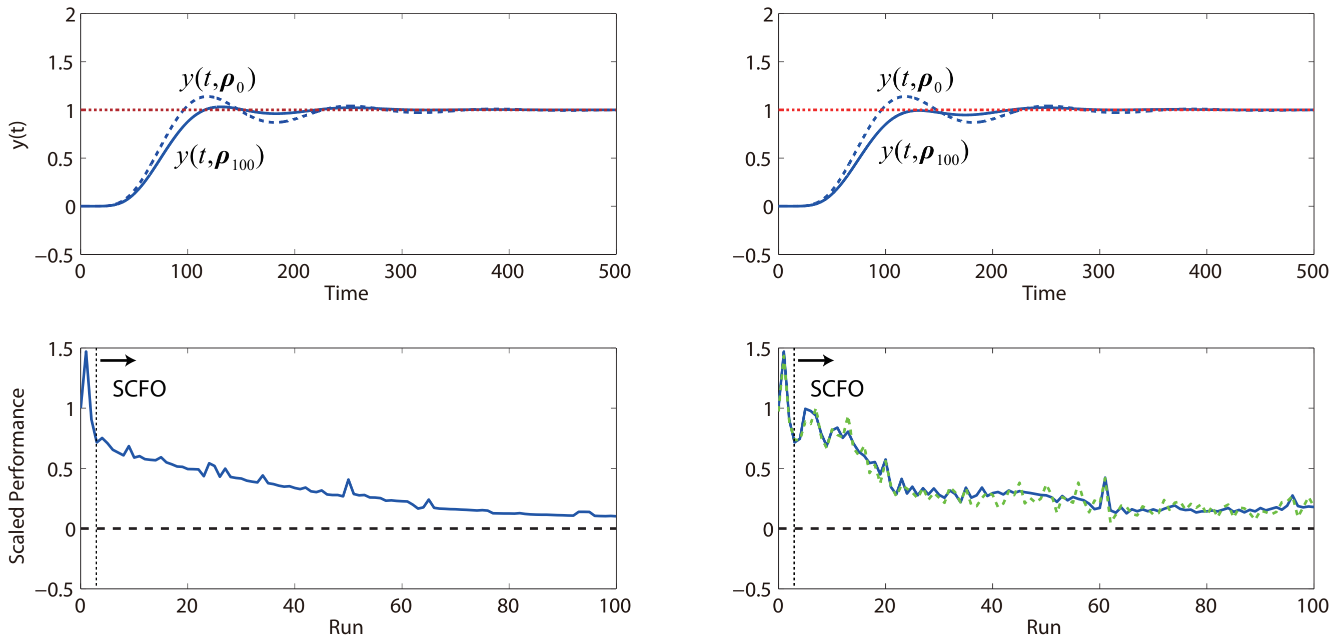
which, of course, is not perfect since the decoupling is not either, and so what one controller does will inevitably affect the others.

The ICT problem that we define for this system consists of starting with all $y_i(0) = 0$ and defining the setpoints of y_1, y_3, and y_5 as 1 (which makes this a tracking problem with respect to these outputs) and the setpoints of y_2 and y_4 as 0 (which makes it a disturbance rejection problem with respect to these two outputs). The total sum of squared tracking errors for all of the outputs is used as the performance metric, with the interval of $t \in [2, 15]$ being considered in the metric computation (a "mask" of 2 time units being employed).

The first five tuning parameters are simply defined as the controller gains, with $\rho_i \stackrel{\Delta}{=} K_{p,i},\ i = 1, ..., 5$. As in the previous example, we use a scaled version of the integral times to define the rest, with $\rho_{i+5} \stackrel{\Delta}{=} \tau_{I,i}/10,\ i = 1, ..., 5$. Once again, as we do not know *a priori* what lower and upper limits should be set on these parameters (save the positivity of the $\tau_{I,i}$), adaptive inputs with the positivity limitation (as shown in the previous case study) are used.

Furthermore, we suppose the existence of a safety limitation in the form of a maximal value that y_1 is allowed to take, with the constraint $y_1(t) \leq 1.2$ to be met at all times. Using the reformulation shown in Section 2.2, we may proceed to state this problem in RTO form as:

Figure 15. Performance obtained by iterative tuning for Study 3 of Problem (20).

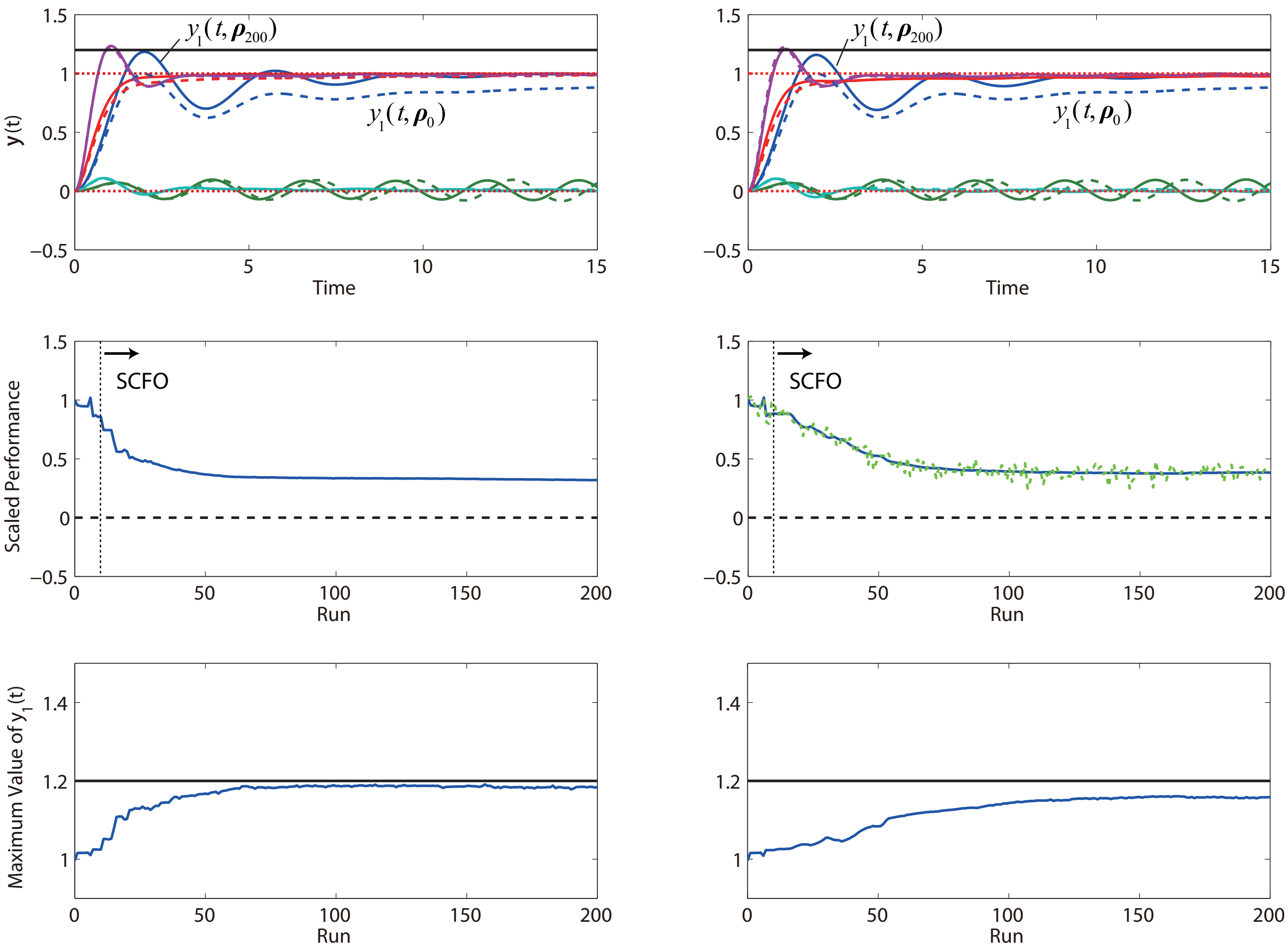


$$
\left.
\begin{array}{ll}
\underset{\boldsymbol{\rho}}{\text{minimize}} & \dfrac{1}{J_0}\displaystyle\sum_{i=1}^{5}\int_{2}^{15}\left[y_{i,ref}(t)-y_i(t,\boldsymbol{\rho})\right]^2 dt \\
\text{subject to} & y_{1,max}(\boldsymbol{\rho})-1.2\leq 0 \\
 & \rho_{k,1}-0.5\leq\rho_1\leq\rho_{k,1}+0.5 \\
 & \rho_{k,2}-0.5\leq\rho_2\leq\rho_{k,2}+0.5 \\
 & \rho_{k,3}-0.5\leq\rho_3\leq\rho_{k,3}+0.5 \\
 & \rho_{k,4}-0.5\leq\rho_4\leq\rho_{k,4}+0.5 \\
 & \rho_{k,5}-0.5\leq\rho_5\leq\rho_{k,5}+0.5 \\
 & \max(\rho_{k,6}-0.5,0.01)\leq\rho_6\leq\rho_{k,6}+0.5 \\
 & \max(\rho_{k,7}-0.5,0.01)\leq\rho_7\leq\rho_{k,7}+0.5 \\
 & \max(\rho_{k,8}-0.5,0.01)\leq\rho_8\leq\rho_{k,8}+0.5 \\
 & \max(\rho_{k,9}-0.5,0.01)\leq\rho_9\leq\rho_{k,9}+0.5 \\
 & \max(\rho_{k,10}-0.5,0.01)\leq\rho_{10}\leq\rho_{k,10}+0.5
\end{array}
\right\}
\begin{array}{l}
\phi_p(\mathbf{v}) \\
\mathbf{G}_p(\mathbf{v})\preceq\mathbf{0} \\
\mathbf{v}^L\preceq\mathbf{v}\preceq\mathbf{v}^U
\end{array}
. \qquad (22)
$$

We note that this problem is a bit more challenging than the ones considered in the previous three studies due to the increased number of tuning parameters, and point out that, were the problem perfectly decoupled, we would be able to solve it as five 2-parameter RTO problems in parallel. However, seeing as all of the parameters are intertwined, we have no choice but to optimize over all ten simultaneously – the expected price to pay being a slower rate of performance improvement obtained by the solver. Alternate strategies that are based on any additional engineering knowledge, such as optimizing only the parameters of specific controllers or optimizing only the controller gains, could of course be proposed and are highly recommended.

As a somewhat arbitrary design, the initial set is chosen as $\boldsymbol{\rho}_0 := [2\ 2\ 2\ 2\ 2\ 1\ 1\ 1\ 1\ 1]^T$. Like with the previous example, an additive measurement noise of $\mathcal{N}(0,(0.05J_0)^2)$ is added to corrupt the performance metric value that is observed for a given choice of tuning parameters. An additive measurement noise

Figure 16. Performance obtained by iterative tuning for the system of PI controllers in Problem (22) – the noiseless case is given on the left and the noisy case on the right. For the output profiles, we note that the initial profiles are given as dashed lines, with the final profiles given by solid lines of the same color.

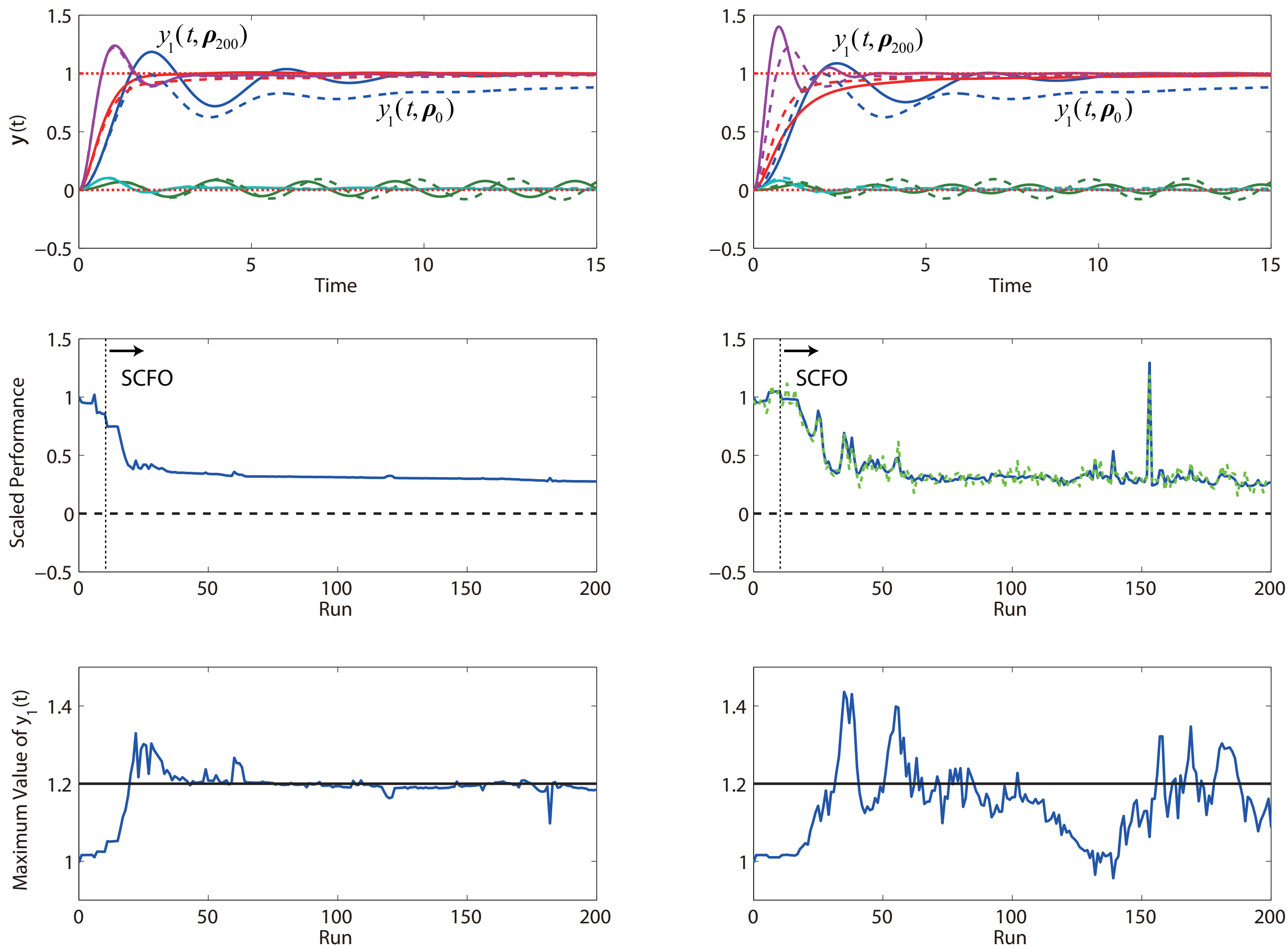


of $\mathcal{N}(0, 10^{-4})$ is added to corrupt the observed values of $y_{1,max}$. Both sets of statistics are assumed to be known for the purposes of SCFO solver configuration. As before, the noiseless scenarios are also considered.

We present the results in Figure 16, which show that the solver is able to obtain significant performance improvements within 50 iterations for both the noiseless and noisy cases without once violating the output constraint on y_1. In this case, we see that the noise has the effect of slowing down convergence, which may be explained by the fact that the solver must take even more cautious steps so as not to violate the output constraint. Additionally, the performance that is observed after 200 iterations is a bit worse for the noisy case, which may be seen as being due to the back-off from the output constraint being larger (to account for the noise).

To test the effect of this constraint and to see if it is even necessary, we also run a simulation where the constraint is lifted from the problem statement. The results for this study are given in Figure 17, and show that not having the constraint in place certainly leads to runs where it is violated. This is not surprising, given that a lot of the performance improvement is obtained by tracking the setpoint of y_1 faster, which

Figure 17. Performance obtained by iterative tuning for the system of PI controllers in Problem (22) (without an output constraint).



is easier to do once there is no constraint on its overshoot. It is also seen that the performance obtained after 200 iterations is generally better than what would be obtained with the constraint – this is, again, not surprising, as removing a limiting constraint should allow for greater performance gains. We do note that the noisy case is more bumpy without the constraint, which is expected as there is less to limit the adaptation steps and more "daring" adaptations become possible. While some of the bumps may be quite undesired (particularly, the one noted just after the 150$^{\text{th}}$ run), the algorithm remains, on the whole, reliable as it keeps the performance metric at low values for the majority of the runs despite significant noise corruption.

5. Concluding Remarks

The goal of this paper has been to propose the idea of posing the iterative controller tuning (ICT) problem in the real-time optimization (RTO) framework, and it has been shown how one can easily formulate most ICT problems as RTO ones with the use of a repeatability assumption that, though it is only an approximation of reality in the presence of noise/degradation, appears to suffice for application purposes (at least, in the two experimental case studies considered here). A major advantage of this reformulation is that a number of previously unaddressed challenges in ICT, the majority of which take the form of constraints in the performance metric minimization problem, may be addressed in a fairly straightforward manner. To make the message more concrete, we have also shown how the ICT problem may be solved by the SCFO real-time optimization solver, and have provided the reader with the necessary solver settings to do so. Four case studies have shown the method to work very well for a diverse range of problems.

Though we hope to have convinced the reader that the method proposed makes for a strong candidate for solving general ICT problems in practice, its potential drawbacks should be clear:

- No solution has been proposed for how to treat the case where the repeatability assumption is not a good approximation of reality. Instead of hoping that the approximation suffice in practice, it would be beneficial to propose alternatives that would still allow one to use the RTO framework to deal with the problem. In particular, one could attempt to make the repeatability assumption on the input and output trajectories rather than making it directly on the performance metric. This could allow one to establish a closer link between the lack of repeatability and the input/output noise in the control system.
- Although the proposed configuration has been shown to be largely successful here, many of the elements involved still remain heuristic in nature. Either improving on these heuristics or finding ways to avoid them are desired.
- The method is currently limited to solving ICT problems where the control task remains the same, which may significantly limit its domain of applicability. It would be interesting to attempt to extend it to cases where the control tasks were similar, rather than identical, and then somehow penalize the method based on the degree of similarity (e.g., one could attempt to lump non-similarity into the noise element δ of the repeatability assumption).

We finish by noting that an abstract advantage of the RTO-ICT formulation is that we are now able to attack the ICT problem from two directions – that of control and that of RTO. For the former, we note that the proposed method applies very few control principles (unlike other direct tuning methods [9,10], which make heavy use of control theory). While this is, in some sense, an advantage – as it allows us to use the proposed method to tune almost any controller for almost any system – there is undoubtedly something lost due to the "black-box veil" that the RTO formulation places on the problem, and incorporating additional knowledge for specific controllers would very likely allow for further improvements to the techniques discussed here. At the same time, the RTO methods themselves are in a fairly nascent stage theoretically. Many improvements to both RTO theory and solution methods are expected to appear in the coming years, which could only improve on the results presented here and to make the solution of the ICT problem both faster and more reliable.

Acknowledgements

The authors would like to acknowledge Fernando Fraire Tirado, for having contributed to the birth and development of the idea of using RTO approaches to solve the iterative controller tuning problem via a number of student projects, the results of which ultimately led to this publication. We would also like to thank Prof. Christos Georgakis, Prof. Kyongbum Lee, and Emily Edwards, as well as others at the Chemical and Biological Engineering Department of Tufts University for allowing the first author to spend two weeks there and to use the laboratory equipment to obtain the results for the first case study. Finally, we want to thank Dr. Timm Faulwasser for several useful discussions.

References

[1] Ziegler, J.G.; Nichols, N.B. Optimum settings for automatic controllers. *Trans. ASME* **1942**, *64*, 759–765.

[2] Ogunnaike, B.A.; Ray, W.H. *Process Dynamics, Modeling, and Control*; Oxford University Press, 1994; pp. 648–665.

[3] Campi, M.C.; Lecchini, A.; Savaresi, S.M. Virtual reference feedback tuning: A direct method for the design of feedback controllers. *Automatica* **2002**, *38*, 1337–1346.

[4] Garriga, J.; Soroush, M. Model predictive control tuning methods: A review. *Ind. Eng. Chem. Res.* **2010**, *49*, 3505–3515.

[5] Burke, J.V.; Henrion, D.; Lewis, A.S.; Overton, M.L. HIFOO - a MATLAB package for fixed-order controller design and H∞ optimization. Fifth IFAC Symposium on Robust Control Design (Toulouse), 2006.

[6] Khatibi, H.; Karimi, A.; Longchamp, R. Fixed-order controller design for systems with polytopic uncertainty using LMIs. *IEEE Trans. Automatic Control* **2008**, *53*, 428–434.

[7] Landau, I.D.; Lozano, R.; M'Saad, M.; Karimi, A. *Adaptive Control*; Springer, 2011; pp. 11–18.

[8] Hjalmarsson, H.; Gunnarsson, S.; Gevers, M. Optimality and sub-optimality of iterative identification and control design schemes. Proceedings of the American Control Conference (Seattle), 1995, pp. 2559–2563.

[9] Hjalmarsson, H.; Gevers, M.; Gunnarsson, S.; Lequin, O. Iterative feedback tuning: theory and applications. *Control Systems, IEEE* **1998**, *18*, 26–41.

[10] Kammer, L.C.; Bitmead, R.R.; Bartlett, P.L. Direct iterative tuning via spectral analysis. *Automatica* **2000**, *36*, 1301–1307.

[11] Hjalmarsson, H. Iterative feedback tuning - An overview. *Int. J. Adapt. Control Signal Process.* **2002**, *16*, 373–395.

[12] Karimi, A.; Mišković, L.; Bonvin, D. Iterative correlation-based controller tuning. *Int. J. Adapt. Control Signal Process.* **2004**, *18*, 645–664.

[13] Killingsworth, N.J.; Krstić, M. PID tuning using extremum seeking: online, model-free performance optimization. *Control Systems, IEEE* **2006**, *26*, 70–79.

[14] Wang, Y.; Gao, F.; Doyle III, F.J. Survey on iterative learning control, repetitive control, and run-to-run control. *J. Process Control* **2009**, *19*, 1589–1600.

[15] Cutler, C.R.; Perry, R.T. Real time optimization with multivariable control is required to maximize profits. *Comput. Chem. Eng.* **1983**, *7*, 663–667.

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

[17] Diehl, M.; Bock, H.G.; Schlöder, J.P.; Findeisen, R.; Nagy, Z.; Allgöwer, F. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *J. Process Control* **2002**, *12*, 577–585.

[18] Mattingley, J.; Boyd, S. Real-time convex optimization in signal processing. *IEEE Signal Proc. Mag.* **2010**, *27*, 50–61.
[19] Bunin, G.A.; François, G.; Bonvin, D. Sufficient conditions for feasibility and optimality of real-time optimization schemes - I. Theoretical foundations. *arXiv:1308.2620v1 [math.OC]* **2013**.
[20] Bunin, G.A.; François, G.; Bonvin, D. Sufficient conditions for feasibility and optimality of real-time optimization schemes - II. Implementation issues. *arXiv:1308.2625v1 [math.OC]* **2013**.
[21] Bunin, G.A.; François, G.; Bonvin, D. *The SCFO Real-Time Optimization Solver: Users' Guide (version 0.9)*. Ecole Polytechnique Fédérale de Lausanne, 2013. http://infoscience.epfl.ch/record/186672.
[22] Lequin, O.; Bosmans, E.; Triest, T. Iterative feedback tuning of PID parameters: Comparison with classical tuning rules. *Contr. Eng. Pract.* **2003**, *11*, 1023–1033.
[23] Bunin, G.A.; Fraire, F.; François, G.; Bonvin, D. Run-to-run MPC tuning via gradient descent. *Comput. Aided Chem. Eng.* **2012**, *30*, 927–931.
[24] Bunin, G.A.; François, G.; Bonvin, D. Iterative controller tuning by real-time optimization. Dynamics and Control of Process Systems (DYCOPS) (Mumbai), 2013.
[25] Maciejowski, J.M. *Predictive Control : With Constraints*; Pearson, 2002; pp. 1–32.
[26] Gopal, M. *Digital Control Engineering*; New Age International, 1988; pp. 79–81.
[27] Åkerblad, M.; Hansson, A.; Wahlberg, B. Automatic tuning for classical step-response specifications using iterative feedback tuning. Proceedings of the 39th IEEE Conference on Decision and Control (Sydney), 2000, Vol. 4, pp. 3347–3348.
[28] Kantas, N.; Maciejowski, J.M.; Lecchini-Visintini, A., Nonlinear Model Predictive Control; Springer-Verlag Berlin Heidelberg, 2009; chapter Sequential Monte Carlo for Model Predictive Control, pp. 263–273.
[29] Jang, S.; Joseph, B.; Mukai, H. On-line optimization of constrained multivariable chemical processes. *AIChE J.* **1987**, *33*, 26–35.
[30] Brdys, M.; Tatjewski, P. *Iterative Algorithms for Multilayer Optimizing Control*; Imperial College Press, 2005.
[31] Gao, W.; Engell, S. Iterative set-point optimization of batch chromatography. *Comput. Chem. Eng.* **2005**, *29*, 1401–1409.
[32] Marchetti, A.; Chachuat, B.; Bonvin, D. Modifier-adaptation methodology for real-time optimization. *Ind. Eng. Chem. Res.* **2009**, *48*, 6022–6033.
[33] Box, G.; Draper, N. *Evolutionary Operation: A Statistical Method for Process Improvement*; John Wiley & Sons, 1969.
[34] Conn, A.; Scheinberg, K.; Vicente, L. *Introduction to Derivative-Free Optimization*; Cambridge University Press, 2009.
[35] Alexandrov, N.; Dennis, J.; Lewis, R.; Torczon, V. A trust region framework for managing the use of approximation models in optimization. Technical report, Langley Research Center, 1997.
[36] Myers, R.; Montgomery, D.; Anderson-Cook, C. *Response Surface Methodology*; John Wiley & Sons, 2009.

[37] Marchetti, A.; Chachuat, B.; Bonvin, D. A dual modifier-adaptation approach for real-time optimization. *J. Process Control* **2010**, *20*, 1027–1037.
[38] Bunin, G.A.; Lima, F.V.; Georgakis, C.; Hunt, C.M. Model predictive control and dynamic operability studies in a stirred tank: Rapid temperature cycling for crystallization. *Chem. Eng. Commun.* **2010**, *197*, 733–752.
[39] Educational Control Products. *Manual for Model 205/205a: Torsional Control System*, 2008.

Appendix

Description of the Initialization Scheme

The algorithm used to initialize the SCFO solver is as follows:

1. Initialize $\mathbf{P} \in \mathbb{R}^{n_v \times n_v}$ as a diagonal matrix with $P_{11} := 1$ and all other elements set to 0. Set $k := 1$. Define by $\Delta\mathbf{v}_{pert} \in \mathbb{R}^{n_v}_{++}$ the perturbation vector and set $\Delta\mathbf{v}_{pert} := \Delta\mathbf{v}_{max}$.

2. Define $\mathbf{v}_k := \mathbf{v}_0 + \mathbf{P}\Delta\mathbf{v}_{pert}$ and compute the following matrix:

$$\Delta\mathbf{V} := \begin{bmatrix} (\mathbf{v}_0 - \mathbf{v}_1)^T \\ (\mathbf{v}_1 - \mathbf{v}_2)^T \\ \vdots \\ (\mathbf{v}_{k-1} - \mathbf{v}_k)^T \end{bmatrix}.$$

   If the condition number of $\Delta\mathbf{V}$ is greater than 50, re-define $\mathbf{v}_k$ as $\mathbf{v}_k := \mathbf{v}_{k-1} + \mathbf{R}_k\Delta\mathbf{v}_{pert}$, where $\mathbf{R}_k$ is a diagonal matrix of zeros with the sole k^{th} diagonal element equal to 1.

3. Obtain the corresponding $\hat{\phi}_p(\mathbf{v}_k) := J_k$ by running a closed-loop experiment with the controller parameters $\boldsymbol{\rho}_k := \mathbf{v}_k$. Define:

$$\Delta\Phi := \begin{bmatrix} \hat{\phi}_p(\mathbf{v}_0) - \hat{\phi}_p(\mathbf{v}_1) \\ \hat{\phi}_p(\mathbf{v}_1) - \hat{\phi}_p(\mathbf{v}_2) \\ \vdots \\ \hat{\phi}_p(\mathbf{v}_{k-1}) - \hat{\phi}_p(\mathbf{v}_k) \end{bmatrix},$$

   and compute:

$$\nabla\hat{\phi}_p := (\Delta\mathbf{V})^\dagger \, \Delta\Phi, \tag{23}$$

   with † denoting the Moore-Penrose pseudoinverse.

4. Re-define $\mathbf{P}$ as a diagonal matrix with the diagonal elements set as:

$$P_{ii} := \begin{cases} 1, & \nabla\hat{\phi}_{p,i} \leq 0 \text{ and } i \leq k \\ -1, & \nabla\hat{\phi}_{p,i} > 0 \text{ and } i \leq k \\ 1, & i = k+1 \\ 0, & i > k+1 \end{cases},$$

   where $\nabla\hat{\phi}_{p,i}$ denotes the i^{th} element of $\nabla\hat{\phi}_p$.

5. Set $k := k + 1$. If $k > n_v$, terminate. Otherwise, return to Step 2.

We make the following remarks:

- This scheme starts like the simple perturbation scheme where only one parameter is perturbed at a time (only ρ_1 is perturbed for the first experiment), but adapts based on the results of the perturbation. For example, if we see that setting $\rho_{1,1} := \rho_{0,1} + \Delta v_{pert,1}$ improves performance, then we will maintain this perturbation while additionally perturbing ρ_2 in the following experiment. On the other hand, if we see that this perturbation leads to worse control performance, then we simply negate it for the following experiment, with this experiment being defined by the perturbations $\rho_{2,1} := \rho_{0,1} - \Delta v_{pert,1}$ and $\rho_{2,2} := \rho_{0,2} + \Delta v_{pert,2}$. The (partial) linear estimate (23) of the gradient acts as a guide for which directions to perturb in.
- Due to the pseudo-inversion of $\Delta\mathbf{V}$, it follows that we also require an additional safeguard to ensure that the matrix remains well-conditioned, as not doing this could lead to a poor estimate of the gradient (assuming the inputs $\mathbf{v}$ to be well-scaled, which we do). Since the perturbation scheme alone does not ensure this, an override is introduced where only a single input is perturbed once the condition number goes over a certain threshold (chosen here as 50). This essentially ensures that the conditioning does not get any worse as it forces $\Delta\mathbf{V}$ to be block diagonal.
- The choice of $\Delta\mathbf{v}_{pert} := \Delta\mathbf{v}_{max}$ is only a recommendation, as the recommended definition for $\Delta\mathbf{v}_{max}$ as given in Table 1 (i.e., $0.1(\boldsymbol{\rho}^U - \boldsymbol{\rho}^L)$) tends to provide sufficient excitation without perturbing "too far". However, if there is a fear that applying perturbations of this size will violate some of the problem constraints or destabilize the system, then $\Delta\mathbf{v}_{pert}$ should be reduced accordingly.

Data-Driven Estimations of the Performance Gradient and Hessian

Estimates of the gradient and Hessian are obtained via response-surface modeling as follows:

- If $k < 2n_v + 1$, fit a linear model to all of the available data:

$$\phi_p(\mathbf{v}) \approx a_0 + \sum_{i=1}^{n_v} a_i v_i,$$

and define:

$$\left.\frac{\partial \hat{\phi}_p}{\partial v_i}\right|_{\mathbf{v}_k} := a_i, \quad H_{k,ij} := \begin{cases} 0.5\kappa_{\phi,i}, & i = j \\ 0, & i \neq j \end{cases},$$

i.e., the gradient is estimated as the coefficients of the linear model and the Hessian, in the absence of more measurements, is defined as a diagonal matrix whose diagonals are equal to half of the Lipschitz constants of the cost (we note that $\kappa_{\phi,i} = \overline{\kappa}_{\phi,i} = -\underline{\kappa}_{\phi,i}$ here – see Section 3.5 for how these are chosen). The latter choice is justified as it (a) does not affect the relative scaling of the different RTO input directions (the Lipschitz constants being equal for all inputs in this case – see Section 3.5), and (b) yields a fairly small step size due to the expected conservatism of $\kappa_{\phi,i}$

(which may be desired since $\nabla\hat{\phi}_p(\mathbf{v}_k)$ is unlikely to be small for earlier runs). In the case where the data are not well-poised for linear regression and the coefficients of the linear model are poorly estimated, the following control step is applied to trim potentially bad estimates:

$$\begin{array}{l} \left.\frac{\partial\hat{\phi}_p}{\partial v_i}\right|_{\mathbf{v}_k} > \kappa_{\phi,i} \rightarrow \left.\frac{\partial\hat{\phi}_p}{\partial v_i}\right|_{\mathbf{v}_k} := \kappa_{\phi,i} \\ \left.\frac{\partial\hat{\phi}_p}{\partial v_i}\right|_{\mathbf{v}_k} < -\kappa_{\phi,i} \rightarrow \left.\frac{\partial\hat{\phi}_p}{\partial v_i}\right|_{\mathbf{v}_k} := -\kappa_{\phi,i} \end{array}. \tag{24}$$

- If $2n_v + 1 \leq k < 2n_v + 1 + \sum_{i=1}^{n_v-1} i$, fit a diagonal quadratic model to the data (quadratic without interaction terms):

$$\phi_p(\mathbf{v}) \approx a_0 + \sum_{i=1}^{n_v} a_i v_i + \sum_{i=1}^{n_v} a_{ii} v_i^2,$$

  and define:

$$\left.\frac{\partial\hat{\phi}_p}{\partial v_i}\right|_{\mathbf{v}_k} := a_i + 2a_{ii}v_{k,i}, \quad H_{k,ij} := \begin{cases} 2a_{ii}, & i = j \\ 0, & i \neq j \end{cases},$$

  where the trimming (24) is applied, as well as:

$$\begin{array}{l} H_{k,ij} > 0.5\kappa_{\phi,i} \rightarrow H_{k,ij} := 0.5\kappa_{\phi,i} \\ H_{k,ij} < -0.5\kappa_{\phi,i} \rightarrow H_{k,ij} := -0.5\kappa_{\phi,i} \end{array}, \tag{25}$$

  where the latter supposes a certain degree of "flatness" in ϕ_p by supposing that no second derivative should ever be greater in magnitude than half of the maximal first derivative.

- If $k \geq 2n_v + 1 + \sum_{i=1}^{n_v-1} i$, fit a full quadratic model to the data:

$$\phi_p(\mathbf{v}) \approx a_0 + \sum_{i=1}^{n_v} a_i v_i + \sum_{i=1}^{n_v}\sum_{j=1}^{n_v} a_{ij} v_i v_j,$$

  where $a_{ij} = a_{ji}$. Define:

$$\left.\frac{\partial\hat{\phi}_p}{\partial v_i}\right|_{\mathbf{v}_k} := a_i + \sum_{j=1}^{n_v} a_{ij}v_{k,j}, \quad H_{k,ij} := 2a_{ij},$$

  and apply the trimmings (24) and (25) if necessary.

We note that while this scheme is not guaranteed to generate a positive-definite Hessian, the consequences of failing to do so are not expected to be very detrimental in our context, since the optimization target is, again, only a guide and does not affect the general reliability of the solver.