Extension of Modifier Adaptation for Controlled Plants using Static Open-Loop Models

Extension of Modifier Adaptation for Controlled Plants using Static Open-Loop Models

G. François[a,b], S. Costello[a], A. G. Marchetti[a] and D. Bonvin[a]
[a] Laboratoire d'Automatique
École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland.
[b] Institute for Materials and Processes, School of Engineering
The University of Edinburgh. Edinburgh EH9 3FB, UK.

Abstract

Model-based optimization methods suffer from the limited accuracy of the available process models. Because of plant-model mismatch, model-based optimal inputs may be suboptimal or, worse, unfeasible for the plant. Modifier adaptation (MA) overcomes this obstacle by incorporating measurements in the optimization framework. However, the standard MA formulation requires that (1) the model satisfies adequacy conditions and (2) the model and the plant share the same degrees of freedom. In this article, three extensions of MA to problems where (2) does not hold are proposed. In particular, we consider the case of controlled plants for which the only a model of the open-loop plant is available. These extensions are shown to preserve the ability of MA to converge to the plant optimum despite disturbances and plant-model mismatch. The proposed methods are illustrated in simulation for the optimization of a CSTR.

1. INTRODUCTION

Process optimization consists in determining the values of input variables that maximize a given performance criterion (such as economic profit or product quality), while meeting all the safety, environmental and operational constraints. Although generally bounded, the values of these manipulated variables are typically not fixed at the design stage. The problem can be formulated mathematically as the following nonlinear program (NLP):

$$\begin{aligned} \mathbf{u}_\mathrm{p}^* := \arg\min_{\mathbf{u}} \; & \phi_\mathrm{p}(\mathbf{u}) \\ \text{subject to} \quad & \mathbf{g}_\mathrm{p}(\mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{1.1}$$

where $\mathbf{u}$ is the n_u-dimensional vector of inputs, ϕ_p is the cost function, and $\mathbf{g}_\mathrm{p}$ is the n_g-dimensional vector of process constraints. Here, the subscript $(\cdot)_\mathrm{p}$ indicates a quantity related to the plant, and this problem will be referred to as the *plant optimization* problem.

If a plant model is available, *numerical optimization* techniques can be used to compute a local [1] or even the global [2] solution. The problem to be solved then reads:

$$\begin{aligned} \mathbf{u}^*(\theta) := \arg\min_{\mathbf{u}} \; & \phi(\mathbf{u}, \theta) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{u}, \theta) \leq \mathbf{0}, \end{aligned} \tag{1.2}$$

where ϕ is the modeled cost function, $\mathbf{g}$ is the n_g-dimensional vector of modeled plant constraints, and θ is an n_θ-dimensional vector of model parameters. Clearly, if the model matches the plant perfectly, solving Problem (1.2) provides a solution to Problem (1.1). Unfortunately, this is rarely the case, since the structure of the model functions ϕ and $\mathbf{g}$ as well as the parameter values θ are likely to be incorrect. This structural and parametric mismatch implies that the model-based optimal inputs $\mathbf{u}^*(\theta)$ will probably not correspond to $\mathbf{u}_\mathrm{p}^*$.

When an accurate model is not available, one typically relies on plant measurements to help the optimization process, which is the field of *real-time optimization* (RTO). Various RTO techniques are available in the literature to solve Problem (1.1). These techniques can be classified in two broad families depending on whether a process model is used or not.

- If no model is available or if the model is too detailed to be used for numerical optimization in real time, *evolutionary techniques* can be utilized. With these techniques, the plant inputs are changed repeatedly based on observing the plant response, similarly to the way an operator would do it. Although the early works in this field date back to the 40's and 50's [3, 4, 5, 6, 7, 8], the resulting methods (such as the simplex algorithm, the steepest descent, and evolutionary operation) are still quite popular, mainly due to their simplicity. They basically follow the same successive stages: (i) initialize the inputs, (ii) apply the inputs to the plant, (iii) measure or estimate the plant cost and constraints ϕ_p and $\mathbf{g}_\mathrm{p}$, (iv) compute an educated modification of the inputs, and (v) go back to Step (ii) and repeat until convergence.

- On the other hand, *model-based RTO* methods apply a numerical optimization algorithm to a model of the plant to calculate a solution. The difference with offline numerical optimization is the fact that real-time measurements are incorporated in the optimization framework to compensate the effect of modeling errors and disturbances on both feasibility and performance. Hence, the model functions $\phi(\mathbf{u}, \theta)$ and $\mathbf{g}(\mathbf{u}, \theta)$ are modified, and the model-based optimization Problem (1.2) is solved iteratively. Model-based RTO can be seen as a combination of evolutionary operation and numerical optimization, as the advantage of using process measurements (which are representative of the actual behavior of the plant) is combined with numerical optimization and its ability to handle large, nonlinear and constrained systems. Measurements can be incorporated in the optimization framework in three distinct ways [9]: (i) adapt the process parameters and use the updated model for optimization (e.g. the two-step or two-stage approach [10]), (ii) add correction terms to the cost and constraint

functions and repeat the optimization, and (iii) directly adapt the manipulated inputs through an appropriate feedback strategy. This paper focuses on *modifier adaptation* (MA), a technique of Class (ii).

MA has received growing attention recently among the methods that do not adapt the model parameters, but modify the cost and constraint functions using measurements [11, 12, 13]. Typically, measurements are used to implement zeroth- and first-order corrections to the cost and constraint functions, while the model parameters are kept unchanged. The key feature of MA is to modify the necessary conditions of optimality (NCO) predicted by the model via input-affine corrections to the cost and constraint functions. As a result, the adequacy conditions [14] are much easier to meet than the corresponding conditions for the two-step approach [15], especially in the case of structural plant-model mismatch. This is a very valuable property since structural mismatch is almost invariably present in complex plants (i.e., there are always simplifying assumptions made during the modeling stage). However, experimental gradients must be estimated for the plant, an onerous task that has received much attention in the literature in recent years [16, 17].

Although MA has been designed to deal with plant-model mismatch, the *model* must still satisfy the following conditions:

- Condition 1: it is adequate for real-time optimization [15],
- Condition 2: it encompasses the same degrees of freedom as the plant that is used to generate the measurements.

Condition 1 can be enforced by the use of a convex approximation to the available model [18]. Note that the use of a model approximation is not a major limitation since, by definition, the model is corrected at each iteration by zeroth- and first-order modifier terms. In contrast, there are many reasons why Condition 2 may not hold, the most obvious one being the case where an open-loop plant model is available and a control system is implemented in the plant. In this case, the degrees of freedom are different entities, namely, the manipulated inputs for the model and the setpoints of the controlled system for the plant. More generally, it often happens that the model and the plant do not share the same inputs as illustrated in [19, 20].

The main contribution of this paper is to propose several extensions to the standard MA formulation, which can be applied when Condition 2 does not hold. Some of these extensions have already been mentioned in two conference articles [19, 20], but they are detailed and analyzed hereafter.

The paper is organized as follows. After a short review of the standard MA scheme, a motivating example highlighting the implications of Condition 2 is discussed in Section 2. Section 3 presents three extensions that can deal with the case where the plant and the model have different sets of decision variables. These methods are tested in simulation on a controlled continuous stirred-tank reactor in Section 4. It is shown that all methods are capable of converging to the plant optimum despite parametric uncertainty, plant-model mismatch and the use of an open-loop model. After brief concluding remarks in Section 5, the way one can construct a convex model approximation for the closed-loop system using the open-loop model is described in the Appendix.

2. Modifier Adaptation

2.1. Problem Formulation

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

$$\tilde{\phi}_{\mathrm{p},k} = \phi_{\mathrm{p},k} + e_k^{\phi} \tag{2.1}$$

$$\tilde{g}_{\mathrm{p},j,k} = g_{\mathrm{p},j,k} + e_k^{\mathrm{g}_j}, \qquad j = 1,\ldots,n_g, \tag{2.2}$$

where e_k^{ϕ} and $e_k^{\mathrm{g}_j}$ are realizations of a zero-mean random variable for the cost and the j^{th} constraint, respectively, with the corresponding variances σ_{ϕ}^2 and $\sigma_{g_j}^2$. These stochastic components represent high-frequency measurement noises. The plant measurements are used to iteratively modify the model-based Problem (1.2) in such a way that, upon convergence, the NCO of the *modified* problem match those of the plant Problem (1.1). This is made possible by using modifiers that, at each iteration, are computed as the differences between the measured and predicted values of the constraints and the measured and predicted values of the cost and constraint gradients. This forces the cost and constraints in the model-based optimization problem to locally match those of the plant. In its simplest form, the algorithm proceeds as given next.

Algorithm 2.1: Modifier Adaptation [13]

Initialize the n_g-dimensional vector of zeroth-order constraint modifiers $\varepsilon_0^{\mathrm{g}} = \mathbf{0}$ (if this leads to infeasibility, introduce backoffs by choosing $\varepsilon_0^{\mathrm{g}} > \mathbf{0}$), the n_u-dimensional vector of first-order cost modifiers $\lambda_0^{\phi} = \mathbf{0}$, and the n_u-dimensional vector of first-order modifiers for the j^{th} constraint $\lambda_0^{\mathrm{g}_j} = \mathbf{0}$.[1] Select the filter matrices $\mathbf{K}^{\varepsilon}$ of dimension $(n_g \times n_g)$, $\mathbf{K}^{\phi}$ and $\mathbf{K}^{\mathrm{g}_j}$ of dimension $(n_u \times n_u)$ as, typically, diagonal matrices with eigenvalues in the interval $(0,1]$. Also, set arbitrarily $\mathbf{u}_0 = \mathbf{u}_{nom}$, where $\mathbf{u}_{nom}$ is the nominal values of the inputs.

for $k = 1 \to \infty$

1. Solve the modified model-based optimization problem

$$\begin{aligned} \mathbf{u}_k := \underset{\mathbf{u}}{\operatorname{argmin}} \quad & \phi_{\mathrm{m},k-1}(\mathbf{u}) \\ \text{subject to} \quad & \mathbf{g}_{\mathrm{m},k-1}(\mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{2.3}$$

   where the modified cost and constraints are given by

$$\phi_{\mathrm{m},k-1}(\mathbf{u}) := \phi(\mathbf{u}) + (\lambda_{k-1}^{\phi})^T(\mathbf{u} - \mathbf{u}_{k-1}), \tag{2.4}$$

$$g_{\mathrm{m},j,k-1}(\mathbf{u}) := g_j(\mathbf{u}) + \varepsilon_{k-1}^{\mathrm{g}_j} + (\lambda_{k-1}^{\mathrm{g}_j})^T(\mathbf{u} - \mathbf{u}_{k-1}), \quad j = 1,\ldots,n_g. \tag{2.5}$$

[1] If $\varepsilon_0^{\mathrm{g}} = \mathbf{0}$, this initialization will lead to $\mathbf{u}_1 = \mathbf{u}^*(\theta)$, that is, to the solution of the unmodified problem.

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

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

3. Estimate the plant cost gradient, $\nabla_{\mathbf{u}}\phi_{\mathrm{E},k}$, and the plant constraint gradients, $\nabla_{\mathbf{u}}g_{\mathrm{E},j,k}$, $j = 1,\ldots,n_g$, at the *current* operating point $\mathbf{u}_k$. The gradients must be estimated using measurements collected at at least n_u different operating points close to $\mathbf{u}_k$.

4. Update the modifier terms using the following first-order filter equations:[2]

$$\boldsymbol{\varepsilon}^{\mathrm{g}}_k := (\mathbf{I}_{n_g} - \mathbf{K}^{\varepsilon})\boldsymbol{\varepsilon}^{\mathrm{g}}_{k-1} + \mathbf{K}^{\varepsilon}\left(\tilde{\mathbf{g}}_{\mathrm{p},k} - \mathbf{g}(\mathbf{u}_k)\right), \tag{2.6}$$

$$\lambda^{\phi}_k := (\mathbf{I}_{n_u} - \mathbf{K}^{\phi})\lambda^{\phi}_{k-1} + \mathbf{K}^{\phi}\left(\nabla_{\mathbf{u}}\phi_{\mathrm{E},k} - \frac{\partial\phi}{\partial\mathbf{u}}(\mathbf{u}_k)\right)^T, \tag{2.7}$$

$$\lambda^{\mathrm{g}_j}_k := (\mathbf{I}_{n_u} - \mathbf{K}^{\mathrm{g}_j})\lambda^{\mathrm{g}_j}_{k-1} + \mathbf{K}^{\mathrm{g}_j}\left(\nabla_{\mathbf{u}}g_{\mathrm{E},j,k} - \frac{\partial g_j}{\partial\mathbf{u}}(\mathbf{u}_k)\right)^T, \quad j = 1,\ldots,n_g. \tag{2.8}$$

end

2.2. Properties

If the scheme converges, then, under ideal circumstances, it will do so to a KKT point of the plant as indicated in the following theorem.

Theorem 2.1 (KKT matching)**.** *Let the filter matrices $\mathbf{K}^{\varepsilon}, \mathbf{K}^{\phi}$ and $\mathbf{K}^{\mathrm{g}_j}$ be nonsingular, and assume no measurement noise and perfect gradient estimates, that is, $\nabla_{\mathbf{u}}\phi_{\mathrm{E},k} = \frac{\partial\phi}{\partial\mathbf{u}}(\mathbf{u}_k)$, $\nabla_{\mathbf{u}}g_{\mathrm{E},j,k} = \frac{\partial g_j}{\partial\mathbf{u}}(\mathbf{u}_k)$, $j = 1,\ldots,n_g$. If Algorithm 2.1 converges, with $\mathbf{u}_\infty := \lim_{k\to\infty}\mathbf{u}_k$ being a KKT point of the modified Problem* (2.3)*, then $\mathbf{u}_\infty$ is also a KKT point for the plant optimization Problem* (1.1)*.*

Proof. See [13]. □

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

The KKT matching theorem guarantees that, *if* the MA algorithm converges, the resulting KKT point is also a KKT point for the plant. However, an important question remains: *Can* the algorithm converge to a (local) plant optimum? Or, in other words, what are the necessary conditions for such a convergence to be possible? In the field of RTO, this is referred to as the 'Model Adequacy' question [21]. For MA, the plant model $\{\phi(\mathbf{u}), \mathbf{g}(\mathbf{u})\}$ is called adequate if modifiers $\varepsilon^{\mathrm{g}}_\infty, \lambda^{\phi}_\infty$ and $\lambda^{\mathrm{g}_j}_\infty$ can be found such that a fixed point of Algorithm 2.1 coincides with a plant optimum $\mathbf{u}^*_{\mathrm{p}}$. This results in the following requirement for model adequacy.

[2] There is also a MA-variant that filters the inputs rather than the modifier terms [14].

Theorem 2.2 (Model Adequacy)**.** *Let* $\mathbf{u}_\mathrm{p}^*$ *be a plant optimum, which is assumed to be a regular point for the* n_g^a *active constraints. Furthermore, let* $L^a(\mathbf{u}) := \phi(\mathbf{u}) + (\nu^a)^T \mathbf{g}^a(\mathbf{u})$ *be the restricted Lagrangian function of Problem* (1.2)*. The plant model is adequate for use in MA if the reduced Hessian of* $L^a(\mathbf{u})$ *is positive definite at* $\mathbf{u}_\mathrm{p}^*$*, that is,*

$$\mathbf{Z}^T(\mathbf{u}_\mathrm{p}^*)\ \nabla^2 L^a(\mathbf{u}_\mathrm{p}^*)\ \mathbf{Z}(\mathbf{u}_\mathrm{p}^*) > \mathbf{0}, \tag{2.9}$$

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

Proof. See [13]. □

In particular, it has been shown that the model-adequacy condition is satisfied if the cost function is strictly convex and the active constraints are convex (for example linear) [18].

We have seen so far that (i) if MA converges, it will converge to a plant KKT point (provided a regularity condition is met), and (ii) it *can* converge to a plant optimum if the model-adequacy condition is met. But *will* the MA scheme converge at all? This is a difficult question to answer. Both necessary conditions and sufficient conditions have been proposed for the convergence of MA schemes [13, 22, 23]. Unfortunately, these conditions are not very satisfactory from a practical point of view, as they are generally impossible to verify/enforce in practice. The filter matrices $\mathbf{K}^\varepsilon, \mathbf{K}^\phi$ and $\mathbf{K}^{g_j}$ are the tuning parameters that affect convergence. These matrices are chosen with real, positive eigenvalues in the interval $(0, 1]$. Larger eigenvalues favor more rapid convergence, but may also cause oscillating behavior, or failure to converge at all. Smaller eigenvalues result in more cautious steps, making convergence more likely, but slower. Currently, the only viable option is to tune these filter matrices through simulation or experimental trials.

Finally, note that an innovative method named "Nested MA" has recently been proposed [24], which completely avoids the gradient estimation step. Instead, the gradient modifiers λ_k^ϕ and $\lambda_k^{g_j}$ are determined at each iteration by unconstrained gradient-free optimization, such as the simplex method. This optimization adjusts the gradient modifiers at each RTO iteration in order to optimize the observed plant performance. While this conveniently avoids gradient estimation, the drawback is that the gradient-free optimization algorithm must optimize the plant using $n_u(n_g + 1)$ decision variables. Due to this large number of decision variables, convergence to the plant optimum might be slow.

2.3. MA in the Context of Controlled Plants

We consider the case of controlled plants for which the degrees of freedom for optimization are the setpoints (or references) $\mathbf{r}$. The optimal setpoints $\mathbf{r}_\mathrm{p}^*$ are solution to the following NLP:

$$\begin{aligned} &\mathbf{r}_\mathrm{p}^* := \arg\min_{\mathbf{r}}\ \Phi_\mathrm{p}(\mathbf{r}) \\ &\text{subject to}\quad \mathbf{G}_\mathrm{p}(\mathbf{r}) \le \mathbf{0}, \end{aligned} \tag{2.10}$$

where Φ_{p} is the cost function, and $\mathbf{G}_{\mathrm{p}}$ is the n_g-dimensional vector of constraints for the controlled plant.

If only the open-loop model is available, Condition 2 will not be satisfied, since the degrees of freedom are (i) the setpoints $\mathbf{r}$ for the plant, and (ii) the manipulated inputs $\mathbf{u}$ for the model, as shown in Figure 1. For simplicity, we assume throughout that there are as many setpoints as there are outputs, thus $n_r = n_y$. One could compute the optimal *inputs* $\mathbf{u}^*$ on the basis of the open-loop model. However, since the plant is operated in closed loop, there is no direct way of implementing $\mathbf{u}^*$ to enforce optimality; this must be done via the *setpoints* $\mathbf{r}$. The open-loop model can be used to predict the optimal values of the controlled variables $\mathbf{y}(\mathbf{u}^*)$. However, choosing $\mathbf{r}^* = \mathbf{y}(\mathbf{u}^*)$ as the setpoints for the closed-loop plant will not result in optimal plant operation because neither $\mathbf{u}^*$ nor $\mathbf{r}^*$ would be optimal for the plant in the presence of plant-model mismatch and disturbances.

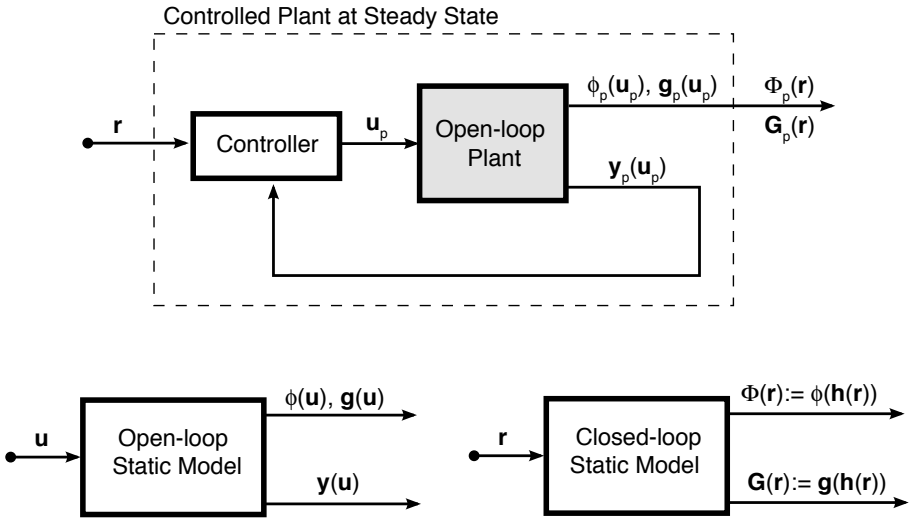

Figure 1: Controlled plant to be optimized and the static open-loop model that is available. The static closed-loop model is assumed to be unavailable.

In standard MA for open-loop systems, the inputs $\mathbf{u}$ are perturbed to estimate the gradients of the plant cost and constraints. This eventually leads to the optimal inputs $\mathbf{u}^*$. In contrast, for closed-loop systems, we are interested in determining the optimal setpoints $\mathbf{r}^*$. Furthermore, in closed-loop systems, one can typically estimate experimental gradients with respect to the setpoints $\mathbf{r}$ (which are independent variables) and not the manipulated inputs $\mathbf{u}_{\mathrm{p}}$.

If the static closed-loop model $\mathbf{u} = \mathbf{h}(\mathbf{r})$ was available, then one could compute the resulting closed-loop cost and constraints

$$\Phi(\mathbf{r}) := \phi\big(\mathbf{h}(\mathbf{r})\big) \qquad \text{and} \qquad \mathbf{G}(\mathbf{r}) := \mathbf{g}\big(\mathbf{h}(\mathbf{r})\big), \tag{2.11}$$

which are shown in Figure 1.

In practice, however, there are situations where only the open-loop model is available, as discussed below. The MA extensions presented in Section 3 can guarantee that the optimal setpoints $\mathbf{r}^*$ be reached upon convergence using an open-loop model.

2.4. MA when the Plant and the Model do not Share the Same Degrees of Freedom

As seen in the previous subsection, the degrees of freedom are different for the model and the plant optimization problems when the plant operates in closed-loop and only an open-loop model is available. Such a situation occurs relatively frequently in real-life applications.

In this paper, for the sake of conciseness, we write ϕ and $\mathbf{g}$ as explicit functions of $\mathbf{u}$. In practice, however, the open-loop model is of the implicit form:

$$\mathbf{F}(\mathbf{x}, \mathbf{u}) = \mathbf{0} \tag{2.12}$$

$$\mathbf{y} = \mathbf{H}(\mathbf{x}), \tag{2.13}$$

with $\mathbf{x}$ the n_x-dimensional state vector, $\mathbf{y}$ the n_y-dimensional output vector, and the cost and constraint functions given as $\tilde{\phi}(\mathbf{u}, \mathbf{y})$ and $\tilde{\mathbf{g}}(\mathbf{u}, \mathbf{y})$. Hence, we can solve (2.12) for $\mathbf{x}$ and substitute it in (2.13) to obtain $\phi(\mathbf{u}) := \tilde{\phi}(\mathbf{u}, \mathbf{y}(\mathbf{u}))$ and $\mathbf{g}(\mathbf{u}) := \tilde{\mathbf{g}}(\mathbf{u}, \mathbf{y}(\mathbf{u}))$.

In principle, the same open-loop model can be used to construct a model of the closed-loop process, that is, to build the functions $\Phi(\mathbf{r})$ and $\mathbf{G}(\mathbf{r})$. To rewrite this model in terms of $\mathbf{r}$, or equivalently in terms of $\mathbf{y}$ assuming that the closed-loop outputs follow their setpoints, one needs to solve the system (2.12)-(2.13) for the variables $\mathbf{x}$ and $\mathbf{u}$ in terms of $\mathbf{y}$. This implies using $n_x + n_y$ equations to compute $n_x + n_u$ variables, thus requiring $n_u \leq n_y$, which leads to:

$$\Phi(\mathbf{y}) := \tilde{\phi}\big(\mathbf{u}(\mathbf{y}), \mathbf{y}\big) \qquad \text{and} \qquad \mathbf{G}(\mathbf{y}) := \tilde{\mathbf{g}}\big(\mathbf{u}(\mathbf{y}), \mathbf{y}\big). \tag{2.14}$$

If the model inputs can be swapped this way to match the degrees of freedom of the plant, then there is no need for the MA extensions presented in this paper (see e.g. the application of MA to a controlled flotation column in [25]). Although very convenient, there are many situations where such a reformulation is not possible or is difficult to implement. In practice, swapping the degrees of freedom of the model to have them correspond to those of the plant may require rewriting the model and potentially changing the solution method. In some cases, the model is constructed so as to simplify the sequence of steps for performing process simulation, as for instance when artificial degrees of freedom are introduced during the modeling stage in order to reduce the number of algebraic equations to be solved simultaneously. A good example is when the plant includes recycles. In this case, it is often advantageous to treat the recycle streams as degrees of freedom and to iterate until convergence has been reached, that is, when steady-state equations and conservation equations at the recycling nodes are satisfied. This is a well-established approach, and methods exist for determining the minimal number of additional degrees of freedom, such as Motard's method [26]. This justifies the development of the three methods discussed in the subsequent section, whereby the model is kept unchanged, and only simple modifications in the formulation of the model-based optimization problem are required. A method for rewriting the model-based problem in terms of $\mathbf{r}$ is also proposed, which only requires offline model simulations and no model inversion.

3. Modifier-Adaptation Extensions

We show next how the standard MA scheme can be altered to optimize a closed-loop plant that does not have the same degrees of freedom (the setpoints $\mathbf{r}$) as the available open-loop model (the inputs $\mathbf{u}$).

Three algorithms will be presented. With all three, the actual degrees of freedom for RTO are the setpoints $\mathbf{r}$, with the estimated experimental gradients being $\nabla_{\mathbf{r}}\Phi_{\mathrm{E}}$, and $\nabla_{\mathbf{r}}G_{\mathrm{E},j}$, $j = 1,\ldots,n_g$. The algorithms will differ in (i) the choice of the decision variables (either $\mathbf{u}$ or $\mathbf{r}$), and (ii) the way the modifier terms are written (in terms of either $\mathbf{u}$ or $\mathbf{r}$). The algorithms will therefore be labeled Methods UR, UU and RR. In Method UR, for example, the optimization problem is solved for $\mathbf{u}$ and the modifier terms are written in terms of $\mathbf{r}$.

Methods UR and UU do not attempt to reconstruct $\Phi(\mathbf{r})$ and $\mathbf{G}(\mathbf{r})$ from $\phi(\mathbf{u})$, $\mathbf{g}(\mathbf{u})$ and $\mathbf{y}(\mathbf{u})$. Instead, they work directly with $\phi(\mathbf{u})$, $\mathbf{g}(\mathbf{u})$ and $\mathbf{y}(\mathbf{u})$ and solve the optimization problem for $\mathbf{u}$. Method UR computes the modifiers in the space of the setpoints $\mathbf{r}$, while Method UU computes them in the space of the manipulated inputs $\mathbf{u}$. Note that the model gradients $\frac{\partial\Phi}{\partial\mathbf{r}}$ and $\frac{\partial\mathbf{G}}{\partial\mathbf{r}}$ are not computed from $\Phi(\mathbf{r})$ and $\mathbf{G}(\mathbf{r})$, which are unkown, but rather by inverting the relationship $\frac{\partial\mathbf{y}}{\partial\mathbf{u}}$. It will be shown that, if Method UR converges, it will do so at a KKT point of the plant. Method UR requires the nonlinear term $\mathbf{y}(\mathbf{u})$ to be inserted into the optimization problem that is solved in real-time. Method UU addresses this issue by linearizing the problematic nonlinear term. It is demonstrated that, also for Method UU, a KKT point of the plant is reached upon convergence.

In contrast, Method RR constructs the model approximations $\Phi(\mathbf{r})$ and $\mathbf{G}(\mathbf{r})$ for the closed-loop plant. These model approximations are obtained without knowledge of the controller.

3.1. Method UR

The basic idea of this method is to express the model gradients with respect to the setpoints $\mathbf{r}$ by inverting the model $\mathbf{y}(\mathbf{u})$. The gradient $\frac{\partial\Phi}{\partial\mathbf{r}}$ of the unkown function $\Phi(\mathbf{r})$ is computed as $\frac{\partial\phi}{\partial\mathbf{u}}\left(\frac{\partial\mathbf{y}}{\partial\mathbf{u}}\right)^{+}$. This assumes that the controlled variables $\mathbf{y}$ follows their setpoints $\mathbf{r}$ with $n_u \geq n_y$.

Algorithm 3.1: Method UR

Initialize the n_g-dimensional vector of zeroth-order constraint modifiers $\varepsilon_0^{\mathrm{g}} = \mathbf{0}$ (if this leads to infeasibility, introduce backoffs by choosing $\varepsilon_0^{\mathrm{g}} > \mathbf{0}$), the n_y-dimensional vector of first-order cost modifiers $\lambda_0^{\phi,r} = \mathbf{0}$, and the n_y-dimensional vector of first-order modifiers for the j^{th} constraint $\lambda_0^{\mathrm{g}j,r} = \mathbf{0}$.[3] Select the filter matrices $\mathbf{K}^{\varepsilon}$ of dimension $(n_g \times n_g)$, $\mathbf{K}^{\phi}$ and $\mathbf{K}^{\mathrm{g}j}$ of dimension $(n_y \times n_y)$ as, typically, diagonal matrices with eigenvalues in the interval $(0,1]$. Also, set arbitrarily $\mathbf{u}_0 = \mathbf{u}_{nom}$.

for $k = 1 \rightarrow \infty$

[3] Again, if $\varepsilon_0^{\mathrm{g}} = \mathbf{0}$, this initialization will lead to $\mathbf{u}_1 = \mathbf{u}^*(\theta)$.

1. Solve the modified model-based optimization problem:

$$\begin{aligned} \mathbf{u}_k := \arg\min_{\mathbf{u}} \quad & \phi^r_{\mathrm{m},k-1}(\mathbf{u}), \\ \text{subject to} \quad & \mathbf{g}^r_{\mathrm{m},k-1}(\mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{3.1}$$

where the modified cost and constraints are given by

$$\phi^r_{\mathrm{m},k-1}(\mathbf{u}) := \phi(\mathbf{u}) + (\lambda^{\phi,r}_{k-1})^T \big(\mathbf{y}(\mathbf{u}) - \mathbf{r}_{k-1}\big), \tag{3.2}$$

$$g^r_{\mathrm{m},j,k-1}(\mathbf{u}) := g_j(\mathbf{u}) + \varepsilon^{\mathrm{g}_j,r}_{k-1} + (\lambda^{\mathrm{g}_j,r}_{k-1})^T \big(\mathbf{y}(\mathbf{u}) - \mathbf{r}_{k-1}\big), \quad j = 1,\ldots,n_g. \tag{3.3}$$

2. Apply the setpoints $\mathbf{r}_k := \mathbf{y}(\mathbf{u}_k)$ and measure $\tilde{\Phi}_{\mathrm{p},k}$ and $\tilde{\mathbf{G}}_{\mathrm{p},k}$ at steady state.

3. Estimate the plant cost gradient, $\nabla_{\mathbf{r}}\Phi_{\mathrm{E},k}$, and the plant constraint gradients, $\nabla_{\mathbf{r}}G_{\mathrm{E},j,k}$, $j = 1,\ldots,n_g$, at the current operating point $\mathbf{r}_k$. Note that these gradients are with respect to the setpoints $\mathbf{r}$.

4. Calculate the modifiers for the next iteration (for all $j \in [1,\ldots,n_g]$):

$$\varepsilon^{\mathrm{g}}_k := (\mathbf{I}_{n_g} - \mathbf{K}^{\varepsilon})\varepsilon^{\mathrm{g}}_{k-1} + \mathbf{K}^{\varepsilon}\left(\tilde{\mathbf{G}}_{\mathrm{p},k} - \mathbf{g}(\mathbf{u}_k)\right), \tag{3.4}$$

$$\lambda^{\phi,r}_k := (\mathbf{I}_{n_y} - \mathbf{K}^{\phi})\lambda^{\phi,r}_{k-1} + \mathbf{K}^{\phi}\left(\nabla_{\mathbf{r}}\Phi_{\mathrm{E},k} - \frac{\partial\phi}{\partial\mathbf{u}}(\mathbf{u}_k)\left(\frac{\partial\mathbf{y}}{\partial\mathbf{u}}(\mathbf{u}_k)\right)^{+}\right)^T, \tag{3.5}$$

$$\lambda^{\mathrm{g}_j,r}_k := (\mathbf{I}_{n_y} - \mathbf{K}^{\mathrm{g}_j})\lambda^{\mathrm{g}_j,r}_{k-1} + \mathbf{K}^{\mathrm{g}_j}\left(\nabla_{\mathbf{r}}G_{\mathrm{E},j,k} - \frac{\partial g_j}{\partial\mathbf{u}}(\mathbf{u}_k)\left(\frac{\partial\mathbf{y}}{\partial\mathbf{u}}(\mathbf{u}_k)\right)^{+}\right)^T. \tag{3.6}$$

with $(\cdot)^+$ indicating the Moore-Penrose pseudo inverse.

end

Remark 1. *The function $\Phi(\mathbf{r})$ is unknown, but we assume $\Phi(\mathbf{r}) = \Phi\big(\mathbf{y}(\mathbf{u})\big) = \phi(\mathbf{u})$, from which we can write $\frac{\partial\phi}{\partial\mathbf{u}} = \frac{\partial\Phi}{\partial\mathbf{r}}\frac{\partial\mathbf{y}}{\partial\mathbf{u}}$. It follows that, in order to be able to compute $\frac{\partial\Phi}{\partial\mathbf{r}}$ from $\frac{\partial\phi}{\partial\mathbf{u}}$, one needs to "invert" $\frac{\partial\mathbf{y}}{\partial\mathbf{u}}$, which requires $n_u \geq n_y$. Note that this condition has to hold for the model only.*

Remark 2. *Since the setpoints $\mathbf{r}$ are typically bounded, it might happen that, due to plant-model mismatch, the computed setpoints $\mathbf{r}_k = \mathbf{y}(\mathbf{u}_k)$ cannot be applied exactly. Hence, in order to converge to a KKT point, the bounds on $\mathbf{r}$ should be included as inequality constraints on $\mathbf{y}(\mathbf{u})$, and these constraints need to be included in the constraints $\mathbf{g}(\mathbf{u})$ in* (3.3)*.*

Remark 3. *The analytical expressions of the mapping $\mathbf{y}(\mathbf{u})$ and of the inverse mapping $\mathbf{u}(\mathbf{y})$ are not needed as Problem 3.1 is solved numerically. It is sufficient to be able to compute $\mathbf{y}(\mathbf{u})$ at each $\mathbf{u}$ given by the numerical optimizer. Similarly, $\left(\frac{\partial\mathbf{y}}{\partial\mathbf{u}}\right)^+$ is only*

needed at each iteration k. Indeed, $\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^+$ *is typically computed numerically by e.g. taking the pseudo-inverse of* $\frac{\partial \mathbf{y}}{\partial \mathbf{u}}$ *at each* $\mathbf{u}_k$*, while* $\frac{\partial \mathbf{y}}{\partial \mathbf{u}}$ *can be computed by finite differences at each* $\mathbf{u}_k$*. Hence, the model* $\mathbf{y}(\mathbf{u})$ *does not need to be in explicit form.*

A fixed point of this iterative procedure is a KKT point of the controlled plant, as stated by the following theorem.

Theorem 3.1 (Plant optimality for Method UR)**.** *Consider Method UR with no measurement noise and perfect estimates of the gradients* $\frac{\partial \Phi_\mathrm{p}}{\partial \mathbf{r}}$ *and* $\frac{\partial \mathbf{G}_\mathrm{p}}{\partial \mathbf{r}}$*. If Algorithm 3.1 converges, with* $\mathbf{u}_\infty := \lim_{k\to\infty}\mathbf{u}_k$ *being a regular point for the constraints, then* $\mathbf{r}_\infty = \mathbf{y}(\mathbf{u}_\infty)$ *is a KKT point for the closed-loop plant optimization Problem* (2.10)*.*

Proof: Consider the iterative scheme upon convergence, i.e. $\lim_{k\to\infty}\mathbf{u}_k = \mathbf{u}_\infty$. We will first derive relationships between $\phi^r_{\mathrm{m},k}$ and $\mathbf{g}^r_{\mathrm{m},k}$ and the controlled plant cost and constraints Φ_p and $\mathbf{G}_\mathrm{p}$.[4] Upon convergence, due to the assumption of perfect gradient estimates, $\nabla_\mathbf{r}\Phi_{\mathrm{E},\infty} = \frac{\partial \Phi_\mathrm{p}}{\partial \mathbf{r}}$, and thus:

$$(\lambda^{\phi,r}_\infty)^T = \frac{\partial \Phi_\mathrm{p}}{\partial \mathbf{r}} - \frac{\partial \phi}{\partial \mathbf{u}}\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+. \tag{3.7}$$

The gradient of the cost function $\phi^r_{\mathrm{m},\infty}$ in Problem (3.1) is

$$\frac{\partial \phi^r_{\mathrm{m},\infty}}{\partial \mathbf{u}} = \frac{\partial \phi}{\partial \mathbf{u}} + (\lambda^{\phi,r}_\infty)^T \frac{\partial \mathbf{y}}{\partial \mathbf{u}}, \tag{3.8}$$

which, using (3.7), gives:

$$\frac{\partial \phi^r_{\mathrm{m},\infty}}{\partial \mathbf{u}} = \frac{\partial \phi}{\partial \mathbf{u}} + \left(\frac{\partial \Phi_\mathrm{p}}{\partial \mathbf{r}} - \frac{\partial \phi}{\partial \mathbf{u}}\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+\right)\frac{\partial \mathbf{y}}{\partial \mathbf{u}}. \tag{3.9}$$

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

$$\frac{\partial \phi^r_{\mathrm{m},\infty}}{\partial \mathbf{u}}\left(\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+ \frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right) = \frac{\partial \Phi_\mathrm{p}}{\partial \mathbf{r}}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+ \frac{\partial \mathbf{y}}{\partial \mathbf{u}} = \frac{\partial \Phi_\mathrm{p}}{\partial \mathbf{r}}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}. \tag{3.10}$$

The same argument can be used to show that (for all $j \in [1,\ldots,n_g]$)

$$\frac{\partial g^r_{\mathrm{m},j,\infty}}{\partial \mathbf{u}}\left(\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+ \frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right) = \frac{\partial G_{\mathrm{p},j}}{\partial \mathbf{r}}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}. \tag{3.11}$$

The modified constraints in (3.3) become

$$g^r_{\mathrm{m},j,\infty}(\mathbf{u}_\infty) = g_j(\mathbf{u}_\infty) + \varepsilon^{g_j,r}_\infty + (\lambda^{g_j,r}_\infty)^T\big(\mathbf{y}(\mathbf{u}_\infty) - \mathbf{r}_\infty\big), \tag{3.12}$$

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

which, with $\mathbf{y}(\mathbf{u}_\infty) = \mathbf{r}_\infty$, the definition of $\varepsilon_k^{\mathrm{g}}$ in (3.4) and the assumption $\tilde{\mathbf{g}}_{\mathrm{p},\infty} = \mathbf{G}_{\mathrm{p}}(\mathbf{r}_\infty)$, gives:

$$g^r_{\mathrm{m},j,\infty}(\mathbf{u}_\infty) = g_j(\mathbf{u}_\infty) + G_{\mathrm{p},j}(\mathbf{r}_\infty) - g_j(\mathbf{u}_\infty) = G_{\mathrm{p},j}(\mathbf{r}_\infty). \tag{3.13}$$

The KKT conditions being necessary conditions under the regularity assumption, it follows that $\mathbf{u}_\infty$ is a KKT point of Problem (3.1). Thus, $\exists\ \nu \geq \mathbf{0}$ such that

$$\frac{\partial \phi^r_{\mathrm{m},\infty}}{\partial \mathbf{u}} + \nu^T \frac{\partial \mathbf{g}^a_{\mathrm{m},\infty}}{\partial \mathbf{u}} = \mathbf{0}\,. \tag{3.14}$$

Assuming rank $\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right) = n_y$, it follows from (3.10) and (3.11) that

$$\frac{\partial \Phi_{\mathrm{p}}}{\partial \mathbf{r}} + \nu^T \frac{\partial \mathbf{G}_{\mathrm{p}}}{\partial \mathbf{r}} = \mathbf{0}\,. \tag{3.15}$$

Hence, the dual feasibility KKT conditions are satisfied for the controlled plant. The KKT conditions for Problem (3.1) also state that $\mathbf{g}^r_{\mathrm{m},\infty} \leq \mathbf{0}$ and $\nu^T\, \mathbf{g}^r_{\mathrm{m},\infty} = 0$. It follows from $\mathbf{g}^r_{\mathrm{m},\infty}(\mathbf{u}_\infty) = \mathbf{G}_{\mathrm{p}}(\mathbf{r}_\infty)$ that,

$$\mathbf{G}_{\mathrm{p}}(\mathbf{r}_\infty) \leq \mathbf{0} \tag{3.16}$$

$$\nu^T\, \mathbf{G}_{\mathrm{p}}(\mathbf{r}_\infty) = 0\,. \tag{3.17}$$

Hence, both the primal feasibility and the complementary slackness KKT conditions are satisfied for the controlled plant. As all four KKT conditions are satisfied, $\mathbf{r}_\infty$ is a KKT point of the controlled plant optimization Problem (2.10). ■

Remark 4. *The model-adequacy condition for Method UR is affected by the way the modified optimization problem is formulated. With standard modifier adaptation, the modification only includes affine-in-input terms. This way, the reduced Hessian of* (2.9) *is the same for the modified and the original problems. The situation is different for Method UR, since the modification of the cost and constraints in* (3.2) *and* (3.3) *is no longer affine in* $\mathbf{u}$*. Hence, the resulting adequacy condition is affected by the second-order derivatives of* $\mathbf{y}(\mathbf{u})$.

3.2. Method UU

This method expresses the model gradients with respect to the manipulated inputs $\mathbf{u}$. In contrast to Method UR, the modifier terms are affine in the inputs $\mathbf{u}$. Like Method UR, the gradient $\frac{\partial \Phi}{\partial \mathbf{r}}$ is computed as $\frac{\partial \phi}{\partial \mathbf{u}} \left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+$, which assumes $\mathbf{y} = \mathbf{r}$ and $n_u \geq n_y$ under offset-free control and perfect output model.

Algorithm 3.2: Method UU

Initialize the n_g-dimensional vector of zeroth-order constraint modifiers $\varepsilon_0^{\mathrm{g}} = \mathbf{0}$ (if this leads to infeasibility, introduce backoffs by choosing $\varepsilon_0^{\mathrm{g}} > \mathbf{0}$), the n_u-dimensional vector of first-order cost modifiers $\lambda_0^{\phi,u} = \mathbf{0}$, and the n_u-dimensional vector of first-order

modifiers for the j^{th} constraint $\lambda_0^{g_j,u} = \mathbf{0}$.[5] Select the filter matrices $\mathbf{K}^{\varepsilon}$ of dimension $(n_g \times n_g)$, $\mathbf{K}^{\phi}$ and $\mathbf{K}^{g_j}$ of dimension $(n_u \times n_u)$ as, typically, diagonal matrices with eigenvalues in the interval $(0,1]$. Also, set arbitrarily $\mathbf{u}_0 = \mathbf{u}_{nom}$.

for $k = 1 \to \infty$

1. Solve the modified model-based optimization problem:

$$\begin{aligned} \mathbf{u}_k := \arg\min_{\mathbf{u}} \quad & \phi^u_{\mathrm{m},k-1}(\mathbf{u}), \\ \text{subject to} \quad & \mathbf{g}^u_{\mathrm{m},k-1}(\mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{3.18}$$

   where the modified cost and constraints are given by

$$\phi^u_{\mathrm{m},k-1}(\mathbf{u}) := \phi(\mathbf{u}) + (\lambda^{\phi,u}_{k-1})^T(\mathbf{u} - \mathbf{u}_{k-1}), \tag{3.19}$$

$$g^u_{\mathrm{m},j,k-1}(\mathbf{u}) := g_j(\mathbf{u}) + \varepsilon^{g_j,u}_{k-1} + (\lambda^{g_j,u}_{k-1})^T(\mathbf{u} - \mathbf{u}_{k-1}), \quad j = 1,\ldots,n_g. \tag{3.20}$$

2. Apply the setpoints $\mathbf{r}_k := \mathbf{y}(\mathbf{u}_k)$ to the plant to obtain $\tilde{\Phi}_{\mathrm{p},k}$ and $\tilde{\mathbf{G}}_{\mathrm{p},k}$.
3. Estimate the plant cost gradient, $\nabla_{\mathbf{r}}\Phi_{\mathrm{E},k}$, and constraint gradients, $\nabla_{\mathbf{r}}G_{\mathrm{E},j,k}$, at the current operating point $\mathbf{r}_k$.
4. Calculate the modifiers for the next iteration (for all $j \in [1,\ldots,n_g]$):

$$\varepsilon^{\mathrm{g}}_k := (\mathbf{I}_{n_g} - \mathbf{K}^{\varepsilon})\varepsilon^{\mathrm{g}}_{k-1} + \mathbf{K}^{\varepsilon}\left(\tilde{\mathbf{G}}_{\mathrm{p},k} - \mathbf{g}(\mathbf{u}_k)\right), \tag{3.21}$$

$$\lambda^{\phi,u}_k := (\mathbf{I}_{n_y} - \mathbf{K}^{\phi})\lambda^{\phi,u}_{k-1} + \mathbf{K}^{\phi}\left(\nabla_{\mathbf{r}}\Phi_{\mathrm{E},k}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial \phi}{\partial \mathbf{u}}(\mathbf{u}_k)\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^{+}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^T \tag{3.22}$$

$$\lambda^{g_j,u}_k := (\mathbf{I}_{n_y} - \mathbf{K}^{g_j})\lambda^{g_j,u}_{k-1} + \mathbf{K}^{g_j}\left(\nabla_{\mathbf{r}}G_{\mathrm{E},j,k}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial g_j}{\partial \mathbf{u}}(\mathbf{u}_k)\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^{+}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^T \tag{3.23}$$

end

Remark 5. *One interpretation of Method UU is that the gradients of the model cost and constraints are "corrected" only in those directions that locally influence* $\frac{\partial \mathbf{y}}{\partial \mathbf{u}}$*[27]. To this end, the post multiplication by* $\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^{+}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)$ *removes any components of* $\frac{\partial \phi}{\partial \mathbf{u}}(\mathbf{u}_k)$ *and* $\frac{\partial g_j}{\partial \mathbf{u}}(\mathbf{u}_k)$ *in the null space of* $\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)$.

Remark 6. *The gradient-modifier terms used in Method UU are affine approximations to those used in Method UR. This can be seen by performing a Taylor-series expansion of the gradient-modifier term of the cost used in Method UR:*

$$(\lambda^{\phi,r}_{k-1})^T\left(\mathbf{y}(\mathbf{u}) - \mathbf{r}_{k-1}\right) = (\lambda^{\phi,r}_{k-1})^T\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_{k-1})\right)(\mathbf{u} - \mathbf{u}_{k-1}) + O\left(\|\mathbf{u} - \mathbf{u}_{k-1}\|^2\right). \tag{3.24}$$

[5] Again, if $\varepsilon^{\mathrm{g}}_0 = \mathbf{0}$, this initialization will lead to $\mathbf{u}_1 = \mathbf{u}^*(\theta)$.

Noting that the gradient modifier for Method UU is given by:

$$\left(\lambda_{k-1}^{\phi,u}\right)^T = \left(\lambda_{k-1}^{\phi,r}\right)^T \left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_{k-1})\right), \tag{3.25}$$

it follows from (3.24) *that:*

$$\left(\lambda_{k-1}^{\phi,r}\right)^T \left(\mathbf{y}(\mathbf{u}) - \mathbf{r}_{k-1}\right) = \left(\lambda_{k-1}^{\phi,u}\right)^T (\mathbf{u} - \mathbf{u}_{k-1}) + O\left(\|\mathbf{u} - \mathbf{u}_{k-1}\|^2\right) \tag{3.26}$$

The same development can be carried out for the gradient-modifier terms of the constraints. Note that the elimination of the nonlinear terms $\mathbf{y}(\mathbf{u})$ *in the modified cost and constraints functions may facilitate the numerical solution and help avoid local minima.*

Remark 7. *In standard MA, the computed input* $\mathbf{u}_k$ *is applied to the plant at the* k^{th} *iteration. However, in Methods UR and UU, the computed* $\mathbf{u}_k$ *is not applied to the plant, since, in closed-loop operation,* $\mathbf{r}_k$ *is applied to the plant. In fact, the computed* $\mathbf{u}_k$ *does not agree with the inputs* $\mathbf{u}_{\mathrm{p},k}$ *reached by the controlled plant at steady state, not even upon convergence. In the case of Method UR, this does not pose any conceptual difficulty because the gradients are evaluated at* $\mathbf{r}_k$*, which are the current setpoints applied to the plant. However, in the case of Method UU, the gradients are evaluated at* $\mathbf{u}_k$*, which typically differs from the plant inputs* $\mathbf{u}_{\mathrm{p},k}$*.*

Remark 8. *Remarks 2 and 3 concerning the bounds on the setpoints and the absence of necessity of the availability of an explicit mapping* $\mathbf{y}(\mathbf{u})$ *also hold for Method UU.*

The following theorem shows that Method UU also benefits from the attractive property of converging to a plant KKT point.

Theorem 3.2 (Plant optimality for Method UU)**.** *Consider Method UU with no measurement noise and perfect estimates of the gradients* $\frac{\partial \Phi_{\mathrm{p}}}{\partial \mathbf{r}}$ *and* $\frac{\partial \mathbf{G}_{\mathrm{p}}}{\partial \mathbf{r}}$*. If Algorithm 3.2 converges, with* $\mathbf{u}_\infty := \lim_{k\to\infty} \mathbf{u}_k$ *being a regular point for the constraints, then* $\mathbf{r}_\infty = \mathbf{y}(\mathbf{u}_\infty)$ *is a KKT point for the closed-loop plant optimization Problem* (2.10)*.*

Proof: It follows from (3.8) and (3.25) that, upon convergence,

$$\frac{\partial \phi_{\mathrm{m},\infty}^{r}}{\partial \mathbf{u}} = \frac{\partial \phi}{\partial \mathbf{u}} + (\lambda_\infty^{\phi,r})^T \frac{\partial \mathbf{y}}{\partial \mathbf{u}} = \frac{\partial \phi}{\partial \mathbf{u}} + (\lambda_\infty^{\phi,u})^T = \frac{\partial \phi_{\mathrm{m},\infty}^{u}}{\partial \mathbf{u}}, \tag{3.27}$$

and, by the same logic, $\frac{\partial \mathbf{g}_{\mathrm{m},\infty}^{a}}{\partial \mathbf{u}} = \frac{\partial \mathbf{g}_{\mathrm{m},\infty}^{u}}{\partial \mathbf{u}}$. Hence, the gradients of the modified cost and constraints are identical for Methods UR and UU, which indicates that the optimality proof for Method UR, from Equation (3.9) onwards, also applies to Method UU. ■

Remark 9. *The model-adequacy condition is not affected by the way the modified optimization problem is formulated in Algorithm 3.2, since the modification only includes affine-in-input terms.*

3.3. Method RR

Method RR constructs the model approximations $\Phi(\mathbf{r})$ and $\mathbf{G}(\mathbf{r})$ from the open-loop models $\phi(\mathbf{u})$, $\mathbf{g}(\mathbf{u})$ and $\mathbf{y}(\mathbf{u})$. In fact, the model approximations are obtained for the functions $\Phi(\mathbf{y})$ and $\mathbf{G}(\mathbf{y})$, assuming $\mathbf{y} = \mathbf{r}$. The open-loop model equations are simulated for several different values of the model inputs, $\mathbf{u}_i$, $i = 1, \cdots, N$, and the corresponding values of the cost $\phi(\mathbf{u}_i)$, constraints $\mathbf{g}(\mathbf{u}_i)$ and outputs $\mathbf{y}(\mathbf{u}_i)$ are stored. Using these data, two approaches can be considered for constructing $\Phi(\mathbf{y})$ and $\mathbf{G}(\mathbf{y})$:

Approach RR1. Using the data $\{\mathbf{u}_i, \phi(\mathbf{u}_i), \mathbf{g}(\mathbf{u}_i), \mathbf{y}(\mathbf{u}_i)\}$, construct the approximate models $\Phi(\mathbf{y})$ and $\mathbf{G}(\mathbf{y})$ using for example polynomial functions.

Approach RR2. Using the data $\{\mathbf{u}_i, \mathbf{y}(\mathbf{u}_i)\}$, construct the model $\mathbf{u} = \mathbf{h}(\mathbf{y})$. This model approximates the inverse of $\mathbf{y}(\mathbf{u})$. Using this inverse model, the models $\Phi(\mathbf{y})$ and $\mathbf{G}(\mathbf{y})$ become $\Phi(\mathbf{y}) := \phi\big(\mathbf{h}(\mathbf{y})\big)$ and $\mathbf{G}(\mathbf{y}) := \mathbf{g}\big(\mathbf{h}(\mathbf{y})\big)$. This approach requires $n_y \geq n_u$ for model inversion.

Once the model approximations $\Phi(\mathbf{y})$ and $\mathbf{G}(\mathbf{y})$ are obtained, the variable $\mathbf{y}$ is replaced by $\mathbf{r}$, and these functions are used in the standard MA scheme to compute the optimal setpoints $\mathbf{r}_k$. Note that, if the assumption $\mathbf{y} = \mathbf{r}$ is not verified exactly, the difference will constitute another source of model mismatch, which can be handled by the MA approach. Then, with the constructed functions $\Phi(\mathbf{r})$ and $\mathbf{G}(\mathbf{r})$, one can apply the following standard MA algorithm.

Algorithm 3.3: Method RR

Initialize the n_g-dimensional vector of zeroth-order constraint modifiers $\varepsilon_0^{\mathrm{G}} = \mathbf{0}$ (if this leads to infeasibility, introduce backoffs by choosing $\varepsilon_0^{\mathrm{G}} > \mathbf{0}$), the n_y-dimensional vector of first-order cost modifiers $\lambda_0^{\Phi} = \mathbf{0}$, and the n_y-dimensional vector of first-order modifiers for the j^{th} constraint $\lambda_0^{\mathrm{G}_j} = \mathbf{0}$. Select the filter matrices $\mathbf{K}^{\varepsilon}$ of dimension $(n_g \times n_g)$, $\mathbf{K}^{\Phi}$ and $\mathbf{K}^{\mathrm{G}_j}$ of dimension $(n_u \times n_u)$ as, typically, diagonal matrices with eigenvalues in the interval $(0, 1]$. Also, set arbitrarily $\mathbf{r}_0 = \mathbf{r}_{nom}$, where $\mathbf{r}_{nom}$ is the nominal values of the setpoints.

for $k = 1 \to \infty$

1. Solve the modified model-based optimization problem

$$\begin{aligned} \mathbf{r}_k := \underset{\mathbf{r}}{\operatorname{argmin}} \quad & \Phi_{\mathrm{m},k-1}(\mathbf{r}) \\ \text{subject to} \quad & \mathbf{G}_{\mathrm{m},k-1}(\mathbf{r}) \leq \mathbf{0}, \end{aligned} \tag{3.28}$$

   where the modified cost and constraints are given by

$$\Phi_{\mathrm{m},k-1}(\mathbf{r}) := \Phi(\mathbf{r}) + (\lambda_{k-1}^{\Phi})^T(\mathbf{r} - \mathbf{r}_{k-1}), \tag{3.29}$$

$$G_{\mathrm{m},j,k-1}(\mathbf{r}) := G_j(\mathbf{r}) + \varepsilon_{k-1}^{\mathrm{G}_j} + (\lambda_{k-1}^{\mathrm{G}_j})^T(\mathbf{r} - \mathbf{r}_{k-1}), \quad j = 1, \ldots, n_g. \tag{3.30}$$

2. Apply the input $\mathbf{r}_k$ to the plant to obtain $\tilde{\Phi}_{\mathrm{p},k}$ and $\tilde{\mathbf{G}}_{\mathrm{p},k}$.

3. Estimate the plant cost gradient, $\nabla_{\mathbf{r}}\Phi_{\mathrm{E},k}$, and the plant constraint gradients, $\nabla_{\mathbf{r}}G_{\mathrm{E},j,k}$, $j = 1, \ldots, n_g$, at the *current* operating point $\mathbf{r}_k$.

4. Calculate the modifiers for the next iteration:

$$\varepsilon_k^{\mathrm{G}} := (\mathbf{I}_{n_g} - \mathbf{K}^{\varepsilon})\varepsilon_{k-1}^{\mathrm{G}} + \mathbf{K}^{\varepsilon}\left(\tilde{\mathbf{G}}_{\mathrm{p},k} - \mathbf{G}(\mathbf{r}_k)\right), \tag{3.31}$$

$$\lambda_k^{\Phi} := (\mathbf{I}_{n_y} - \mathbf{K}^{\Phi})\lambda_{j,k-1}^{\Phi} + \mathbf{K}^{\Phi}\left(\nabla_{\mathbf{r}}\Phi_{\mathrm{E},k} - \nabla_{\mathbf{r}}\Phi(\mathbf{r}_k)\right)^T, \tag{3.32}$$

$$\lambda_k^{\mathrm{G}_j} := (\mathbf{I}_{n_u} - \mathbf{K}^{\mathrm{G}_j})\lambda_{k-1}^{\mathrm{G}_j} + \mathbf{K}^{\mathrm{G}_j}\left(\nabla_{\mathbf{r}}G_{\mathrm{E},j,k} - \nabla_{\mathbf{r}}\mathbf{G}(\mathbf{r}_k)\right)^T, \quad j = 1,\ldots,n_g. \tag{3.33}$$

end

Remark 10. *Since Method RR implies constructing the models* $\Phi(\mathbf{r})$ *and* $\mathbf{G}(\mathbf{r})$*, it is recommended to aim for convex approximations (see Appendix). This way, Method RR would simultaneously meet Conditions 1 and 2.*

Remark 11. *In Methods UR and UU, the plant and the model have different degrees of freedom. In Method RR, thanks to the remodeling, the plant and the model have the same degrees of freedom* $\mathbf{r}$*.*

Remark 12. *In Method RR, the lower and upper bounds on the setpoints* $\mathbf{r}$ *pose no difficulty, as they can be included as lower and upper bounds on the decision variables of the optimization problem. In this case, the difficulty comes from the lower and upper bounds on the manipulated variables* $\mathbf{u}$*. If the bounds are reached, the controller will saturate, which might complicate gradient estimation. In order to converge to a KKT point, the lower and upper bounds on* $\mathbf{u}$ *should be included as inequality constraints in* $\mathbf{g}(\mathbf{u})$*.*

4. Simulated Example: Controlled Williams-Otto Reactor

Methods UR, UU and RR are illustrated on the Williams-Otto reactor [28]. We will use the model from [29], which has become a standard test problem for real-time optimization techniques [16]. Although the original problem is an open-loop reactor, the aim is here to optimize the reactor in closed-loop operation. The open-loop plant is an ideal continuous stirred-tank reactor with the following reactions:

$$A + B \xrightarrow{k_1} C, \qquad k_1 = k_{10}e^{-E_1/(RT_p)}, \tag{4.1}$$

$$C + B \xrightarrow{k_2} P + E, \quad k_2 = k_{20}e^{-E_2/(RT_p)}, \tag{4.2}$$

$$C + P \xrightarrow{k_3} G, \qquad k_3 = k_{30}e^{-E_3/(RT_p)}. \tag{4.3}$$

The manipulated inputs are $\mathbf{u}_{\mathrm{p}} = [F_{A,p}, F_{B,p}, T_p]^T$, that is, the feed rates of A and B, and the reactor temperature. However, the degrees of freedom of the controlled plant are the controller setpoints $\mathbf{r} = [X_{A,\mathrm{s}}, F_{B,\mathrm{s}}]^T$ for the mass fraction of A in the reactor and the feed rate of B. The desired products are P and E, and the reactor mass holdup is 2105 kg.

A rather poor controller adjusts the plant inputs $F_{B,p}$ and T_p in the following manner:

- $F_{B,p} = F_{B,\mathrm{s}} + 2$, that is, there is an offset between $F_{B,p}$ and $F_{B,\mathrm{s}}$.
- T_p is manipulated to regulate $X_{A,p}$; however, there is a large steady-state offset, $X_{A,p} = 0.75 X_{A,\mathrm{s}}$.

There is also an unknown (and unmodeled) ratio controller, which enforces $\frac{F_{A,p}}{F_{B,p}} = 2.4$, that is, $F_{A,p}$ is proportional to $F_{B,p}$.

The block diagram of the controlled CSTR is shown in Figure 2.

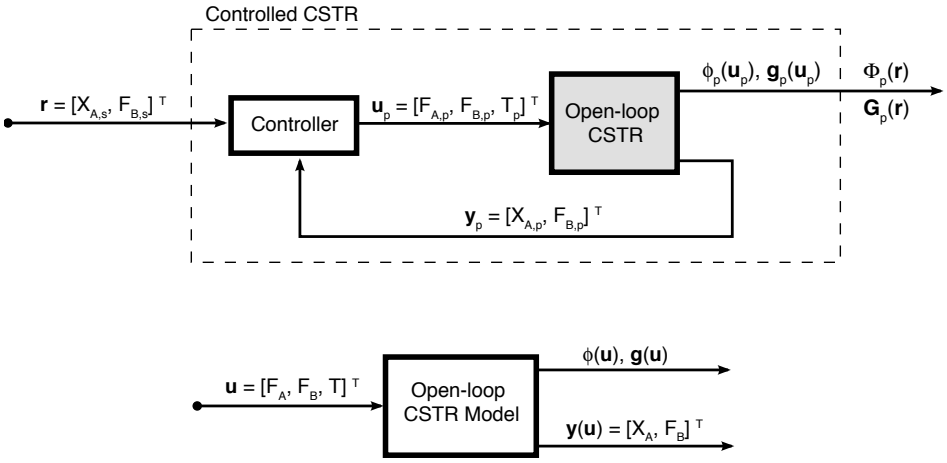

Figure 2: The controlled CSTR and the open-loop model.

Plant-model mismatch is simulated by choosing a reactor model that includes the following two-reaction approximation to the reaction system:

$$A + 2B \xrightarrow{k_1^*} P + E, \quad k_1^* = k_{10}^* e^{-E_1^*/(RT)}, \tag{4.4}$$

$$A + B + P \xrightarrow{k_2^*} G, \quad k_2^* = k_{20}^* e^{-E_2^*/(RT)}, \tag{4.5}$$

with the parameters k_{10}^*, k_{20}^*, E_1^* and E_2^*. Furthermore, two different models will be used by considering different values for the parameters E_1^* and E_2^* (the parameters k_{10}^* and k_{20}^* are fixed). Both the model and the plant consider that online measurements of X_A and F_B are available, that is $\mathbf{y} = [X_A, F_B]$ and $\mathbf{y}_\mathrm{p} = [X_{A,p}, F_{B,p}]$, respectively.

From the implementation point of view, the controller is considered to be *unknown*. In particular, no knowledge is available regarding the manner in which F_A is manipulated. The profit function to maximize is

$$\text{Profit} = 1143.38 X_P (F_A + F_B) + 25.92 X_E (F_A + F_B) - 76.23 F_A - 114.34 F_B, \tag{4.6}$$

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

$$X_A \leq 0.09 \quad \text{and} \quad \mathrm{X_G} \leq 0.6\,. \tag{4.7}$$

The plant cost Φ_p and the model cost function $\phi(\mathbf{u})$ are constructed by expressing the profit function using the plant and model equations, respectively. Similarly, the plant constraint values $\mathbf{G}_\mathrm{p}$ and the modeled constraints $\mathbf{g}(\mathbf{u})$ are constructed by expressing the above constraints using the plant and model equations, respectively. Finally, the modeled outputs are constructed from the model equations. In addition, the plant cost and constraint gradients $\nabla_\mathbf{r}\Phi_\mathrm{E}$ and $\nabla_\mathbf{r}\mathbf{G}_\mathrm{E}$ are computed from Φ_p and $\mathbf{G}_\mathrm{p}$, respectively. Table 1 gives the numerical values of the plant parameters and of the fixed model parameters k_{10}^* and k_{20}^*.

Table 1: Values of the plant parameters and the two fixed model parameters. The other model parameters are chosen differently for the two investigation cases, as shown in Table 2.

Parameter	Value	Unit
k_{10}	1.66×10^6	s^{-1}
k_{20}	7.21×10^8	s^{-1}
k_{30}	2.67×10^{12}	s^{-1}
E_1	5.543×10^4	$\mathrm{kJ\,mol}^{-1}$
E_2	6.928×10^4	$\mathrm{kJ\,mol}^{-1}$
E_3	9.238×10^4	$\mathrm{kJ\,mol}^{-1}$
k_{10}^*	6.716×10^4	s^{-1}
k_{20}^*	1.034×10^5	s^{-1}

Figures 3-5 show the performance of Methods UR, UU and RR for the two models given in Table 2. Each RTO scheme is initialized at the corresponding model optimum. Implementation specificities are given next:

- For Methods UR and UU, diagonal filter matrices with eigenvalues of 0.2 are used.
- For Model RR, *convex* model approximations are constructed using $11 \times 11 \times 11$ simulations performed offline for equally spaced values of the model inputs $\mathbf{u} = [F_A, F_B, T]^T$, respecting the input bounds $2 < F_A < 10$, $8.5 < F_B < 22$ and $78 < T < 92$. Algorithm 5.1 in the Appendix is applied with the lower bounds on the eigenvalues σ_Φ and σ_{G_j} being all set to 0.05. For MA, Algorithm 3.3 is used with the controller setpoints $\mathbf{r} = [X_{A,\mathrm{s}}, F_{B,\mathrm{s}}]^T$ as degrees of freedom. To avoid having to modify too strongly the optimization problem at the first iteration, the zeroth-order modifier terms are initialized at half the distance between the predicted and measured constraints. This is justified by the observation that a large initial modification of the constraints can lead to unfeasible inputs, which would then imply convergence to the plant optimum from the unfeasible side of the plant constraints. The filter matrices $\mathbf{K}^\varepsilon$, $\mathbf{K}^\phi$ and $\mathbf{K}^{\mathrm{G}_j}$ in Equations (2.6)-(2.8) are chosen as diagonal matrices with all elements equal to 0.85.

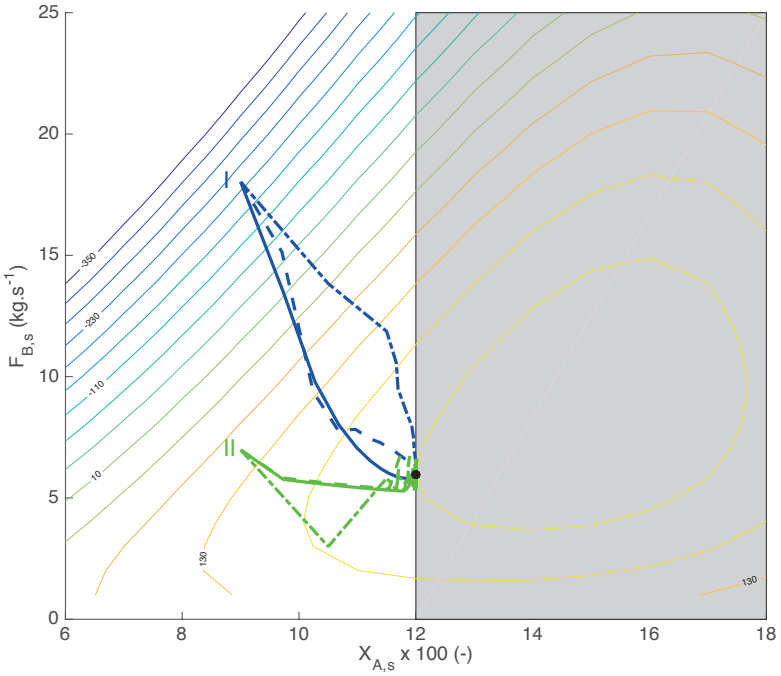

Figure 3: Evolution of the setpoints $\mathbf{r}$ during the first 30 iterations of three generalized MA schemes for Cases I (blue) and II (green). Solid = Method UR, Dashed = Method UU, Dash-Dotted = Method RR. The starting point corresponds to the model optimum and is marked by a roman numeral. The contour lines represent the plant profit. The shaded region is infeasible for the plant due to the constraint on X_A. The black dot represents the plant optimum.

The two trajectories labeled I and II correspond to the use of the models for Cases I and II in Table 2. All algorithms converge rapidly to the plant optimum, where the constraint on X_A is active ($X_A = 0.09$, which calls for $X_{A,s} = 0.12$). The main observation is that Methods UR and UU behave very similarly. Hence, although both algorithms can be used, Method UU is computationally advantageous. Method RR also works very well, with the convergence rate not being penalized by the use of a very crude convex model.

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

$$\nabla_{\mathbf{r}} \Phi_{\mathrm{E},k} = \left[\frac{\Phi_{\mathrm{p}}(\mathbf{r}_k + [\Delta X_{A,\mathrm{s}}, 0]^T) - \Phi_p(\mathbf{r}_k)}{\Delta X_{A,\mathrm{s}}} \qquad \frac{\Phi_{\mathrm{p}}(\mathbf{r}_k + [0, \Delta F_{B,\mathrm{s}}]^T) - \Phi_{\mathrm{p}}(\mathbf{r}_k)}{\Delta F_{B,\mathrm{s}}} \right]. \tag{4.8}$$

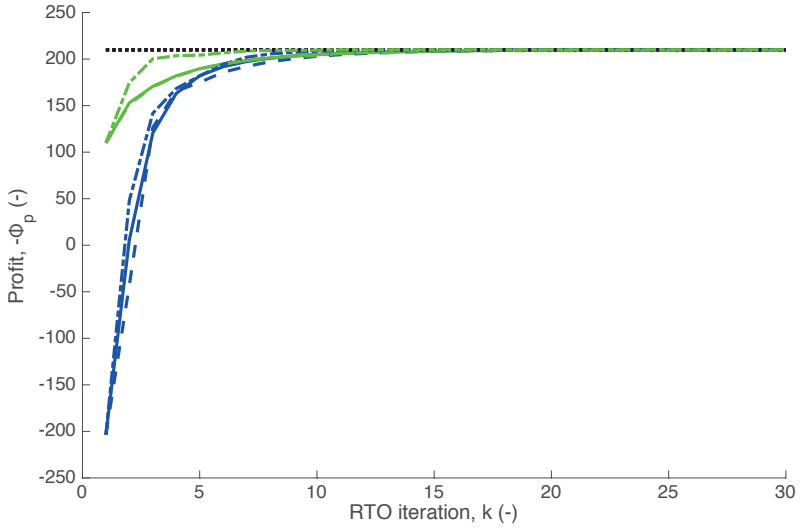

Figure 4: The profit as a function of the RTO iteration number k. Blue/green = Cases I/II. Solid = Method UR, Dashed = Method UU, Dash-Dotted = Method RR.

Table 2: Values of the activation energies for Cases I and II.

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

5. Conclusions

Modifier adaptation is a very appealing RTO method, whose principal strength lies in its capacity to converge to the plant optimum despite disturbances and plant-model mismatch. Although incorporating measurements in the optimization framework, MA still relies on a plant model, and it is typically assumed that the plant and the model have the same degrees of freedom. However, this assumption may not hold in practice, in particular in the context of controlled processes. Obtaining a closed-loop model with the same degrees of freedom as the controlled plant may not be feasible when the model is complex or difficult to reformulate.

Three extensions of MA have been presented. On the one hand, Methods UR and UU avoid remodeling the system, and this at no additional computational cost. On the other hand, Method RR constructs closed-loop model approximations using the available open-loop model, which can be achieved through simulation of the open-loop model and constrained least-squares fitting. Note that, in order to enforce the model-adequacy condition, Methods UR and RR can benefit from the availability of convex models for $\phi(\mathbf{u})$ and $\mathbf{g}(\mathbf{u})$ and for $\Phi(\mathbf{r})$ and $\mathbf{G}(\mathbf{r})$, respectively. In particular,

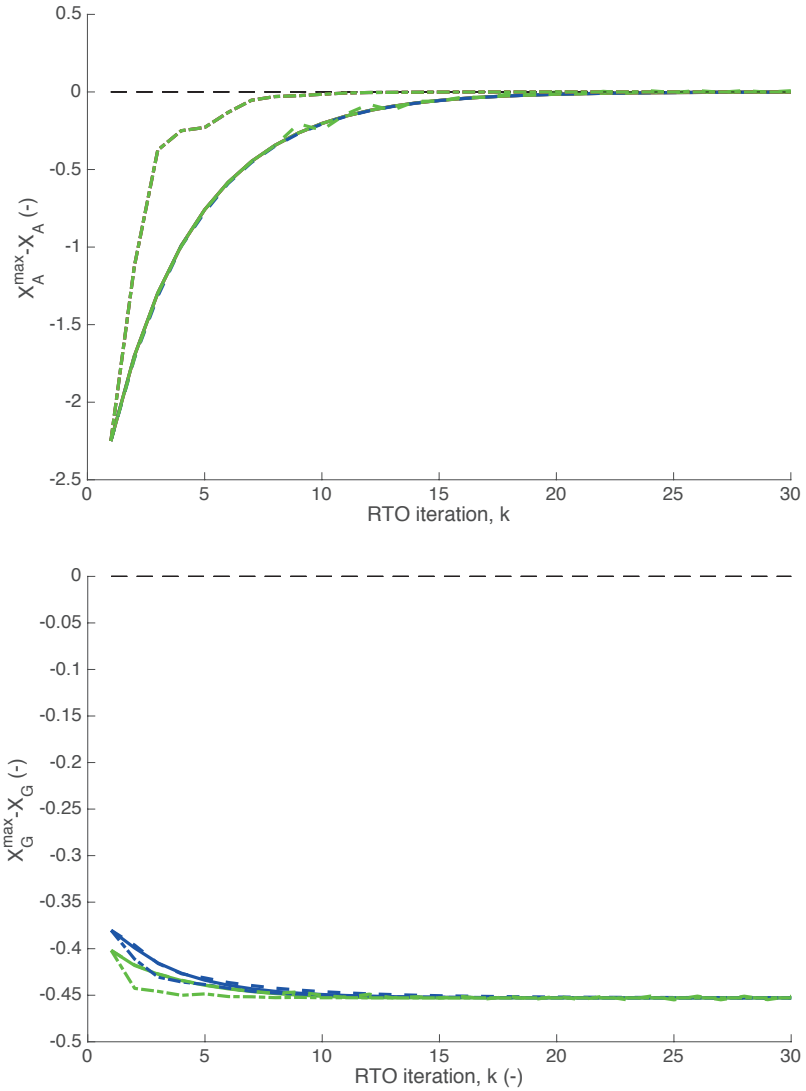

Figure 5: The constraints on X_A and X_G as a function of the RTO iteration number k. Blue/green = Cases I/II. Solid = Method UR, Dashed = Method UU, Dash-Dotted = Method RR. The black dotted lines indicate the plant constraints.

since Method RR proposes to construct model approximations for $\Phi(\mathbf{r})$ and $\mathbf{G}(\mathbf{r})$, it is possible to ensure that these models are convex, as shown in the Appendix.

The simulation results indicate that all three methods perform rather well. Method UU should therefore be preferred over Method UR, as it is computationally simpler. However, when model simulations are not too costly, Method RR with convex model approximations might well be the preferred approach, since it automatically enforces model adequacy. The choice will finally be case dependent, with the nice feature that all three methods preserve the valuable KKT matching property, that is, if convergence occurs, it will do so at a KKT point of the *plant*.

Appendix

Construction of Convex Model Approximations $\Phi_c(\mathbf{r})$ *and* $\mathbf{G}_c(\mathbf{r})$

This appendix describes a way of constructing the convex approximations $\Phi_c(\mathbf{r})$ and $\mathbf{G}_c(\mathbf{r})$ from the open-loop models $\phi(\mathbf{u})$, $\mathbf{g}(\mathbf{u})$ and $\mathbf{y}(\mathbf{u})$ without knowledge of the controller. Note that a similar approach is also possible for construction the convex model $\mathbf{u} = \mathbf{h}_c(\mathbf{y})$. The algorithm for constructing the convex model approximations is given next.

Algorithm 5.1: Construction of Convex Model Approximations $\Phi_c(\mathbf{r})$ and $\mathbf{G}_c(\mathbf{r})$

Initialize

1. Solve the model-based optimization Problem (1.2) and store the optimal values of the inputs $\mathbf{u}^* := \mathbf{u}^*(\theta)$, the cost $\phi^* := \phi(\mathbf{u}^*, \theta)$, the constraints $\mathbf{g}^* := \mathbf{g}(\mathbf{u}^*, \theta)$ and the outputs $\mathbf{y}^* := \mathbf{y}(\mathbf{u}^*)$.

2. Simulate the open-loop model equations for several different values of the model inputs, $\mathbf{u}_i$, $i = 1, \cdots, N$, and store the corresponding values of the cost $\phi(\mathbf{u}_i, \theta)$, the constraints $\mathbf{g}(\mathbf{u}_i, \theta)$ and the outputs $\mathbf{y}_i := \mathbf{y}(\mathbf{u}_i)$.

3. Construct the convex model approximations $\Phi_c(\mathbf{y})$ and $\mathbf{G}_c(\mathbf{y})$ using the data collected at the previous stage. The trick here is to use the values $\mathbf{y}_i$, which correspond to $\mathbf{r}_i$, rather than the values of $\mathbf{u}_i$. Many different methods can be used at this stage [30, 31, 18], and we propose here to modify the simple method of [18], Section IV, page 11622. For this purpose, the following $n_g + 1$ constrained least-squares regression problems are solved:

$$
\begin{aligned}
[\alpha_\Phi^*, \mathbf{Q}_\Phi^*] := \arg \min_{\alpha_\Phi, \mathbf{Q}_\Phi} & \left(\sum_{\mathrm{i}=1}^{\mathrm{n}} \| \Phi_{\mathrm{c}}(\mathbf{y}_{\mathrm{i}}) - \phi(\mathbf{u}_{\mathrm{i}}) \| \right) \\
\text{subject to:} \quad & \Phi_c(\mathbf{y}_i) = \phi^* + \alpha_\Phi (\mathbf{y}_i - \mathbf{y}^*) + \frac{1}{2} (\mathbf{y}_i - \mathbf{y}^*)^T \mathbf{Q}_\Phi (\mathbf{y}_i - \mathbf{y}^*) \\
& \mathrm{eig}(\mathbf{Q}_\Phi) > \sigma_\Phi > \mathbf{0}, \qquad (5.1)
\end{aligned}
$$

and, $\forall j \in [1, \ldots, n_g]$,

$$\left[\alpha^*_{\mathrm{G}_j}, \mathbf{Q}^*_{\mathrm{G}_j}\right] := \arg\min_{\alpha_{\mathrm{G}_j}, \mathbf{Q}_{\mathrm{G}_j}} \left(\sum_{\mathrm{i}=1}^{\mathrm{n}} \left\| \mathrm{G_{c,}}_j(\mathbf{y}_\mathrm{i}) - \mathrm{g}_j(\mathbf{u}_\mathrm{i}) \right\| \right)$$

$$\text{subject to:} \quad G_{\mathrm{c},j}(\mathbf{y}_i) = g^*_j + \alpha_{\mathrm{G}_j}(\mathbf{y}_i - \mathbf{y}^*) + \frac{1}{2}(\mathbf{y}_i - \mathbf{y}^*)^T \mathbf{Q}_{\mathrm{G}_j}(\mathbf{y}_i - \mathbf{y}^*)$$

$$\mathrm{eig}\left(\mathbf{Q}_{\mathrm{G}_j}\right) > \sigma_{\mathrm{G}_j} \geq \mathbf{0}, \tag{5.2}$$

where α_Φ, $\mathbf{Q}_\Phi$, α_{G_j} and $\mathbf{Q}_{\mathrm{G}_j}$ are as follows: (i) they are the parameters of the quadratic model for the cost and the j^{th} constraint, (ii) they are of dimensions $1 \times n_y$, $n_y \times n_y$, $1 \times n_y$ and $n_y \times n_y$, respectively, and (iii) the superscript * denotes their optimal values. σ_Φ and σ_{G_j} are lower bounds on the eigenvalues of the matrices $\mathbf{Q}_\Phi$ and $\mathbf{Q}_{\mathrm{G}_j}$, respectively.

end

Remark 13. *The assumption of perfect control at steady-state, that is* $\mathbf{y} = \mathbf{r}$*, is just another source of model-mismatch in case it is not true. Since MA is capable of handling plant-model mismatch provided Condition 1 holds, this additional modeling error will be handled just like any other error. It is therefore sufficient to consider the quadratic models:*

$$\Phi_c(\mathbf{r}) = \phi^* + \alpha^*_\Phi(\mathbf{r} - \mathbf{y}^*) + \frac{1}{2}(\mathbf{r} - \mathbf{y}^*)^T \mathbf{Q}^*_\Phi(\mathbf{r} - \mathbf{y}^*) \tag{5.3}$$

$$G_{\mathrm{c},j}(\mathbf{r}) = g^*_j + \alpha^*_{\mathrm{G}_j}(\mathbf{r} - \mathbf{y}^*) + \frac{1}{2}(\mathbf{r} - \mathbf{y}^*)^T \mathbf{Q}^*_{\mathrm{G}_j}(\mathbf{r} - \mathbf{y}^*), \quad \forall j. \tag{5.4}$$

Remark 14. *To enforce Condition 1, it is sufficient to have strict convexity of* Φ_c *and convexity of all* $G_{\mathrm{c},j}$ *[18]. Hence,* $\mathbf{Q}^*_{\mathrm{G}_j}$ *is not compulsory, and the constraints can be approximated by linear functions:*

$$\Phi_c(\mathbf{r}) = \phi^* + \alpha^*_\Phi(\mathbf{r} - \mathbf{y}^*) + \frac{1}{2}(\mathbf{r} - \mathbf{y}^*)^T \mathbf{Q}^*_\Phi(\mathbf{r} - \mathbf{y}^*) \tag{5.5}$$

$$G_{\mathrm{c},j}(\mathbf{r}) = g^*_j + \alpha^*_{\mathrm{G}_j}(\mathbf{r} - \mathbf{y}^*), \qquad \forall j. \tag{5.6}$$

[1] L. T. Biegler, Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes, MOS-SIAM Series on Optimization, 2010.

[2] C. A. Floudas, Deterministic Global Optimization: Theory, Methods and Applications, Nonconvex Optimization and its Applications, Book 37, Springer, 1999.

[3] H. Hotelling, Experimental determination of the maximum of a function, Ann. Math. Statist. 12 (1941) 20–45.

[4] G. E. P. Box, K. Wilson, On the experimental attainment of optimum conditions, J. Royal Stat. Society, Series B 13 (1951) 1–45.

[5] S. H. Brooks, A comparison of maximum-seeking methods, Oper. Res. 7 (1959) 430–457.

[6] S. H. Brooks, M. R. Mickey, Optimum estimation of gradient direction in steepest ascent experiments, Biometrics 17 (1961) 48–51.

[7] W. Spendley, G. R. Hext, F. R. Himsworth, Sequential application of simplex designs in optimisation and evolutionary operation, Technometrics 4 (1962) 441–461.

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

[9] G. François, D. Bonvin, Measurement-based real-time optimization of chemical processes, in: S. Pushpavanam (Ed.), Advances in Chemical Engineering, Vol. 43, Academic Press, 2013, pp. 1–50.

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

[11] P. Tatjewski, Iterative optimizing set-point control – The basic principle redesigned, in: Proc. IFAC World Congress, 2002, pp. 992–992.

[12] W. Gao, S. Engell, Iterative set-point optimization of batch chromatography, Comp. Chem. Eng. 29 (6) (2005) 1401–1409.

[13] A. G. Marchetti, B. Chachuat, D. Bonvin, Modifier-adaptation methodology for real-time optimization, Ind. Eng. Chem. Res. 48 (13) (2009) 6022–6033.

[14] A. G. Marchetti, Modifier-Adaptation Methodology for Real-Time Optimization, Ph.D. thesis, # 4449, EPFL, Lausanne (2009).

[15] J. F. Forbes, T. E. Marlin, J. F. MacGregor, Model adequacy requirements for optimizing plant operations, Comp. Chem. Eng. 18 (6) (1994) 497–510.

[16] A. G. Marchetti, B. Chachuat, D. Bonvin, A dual modifier-adaptation approach for real-time optimization, J. Process Control 20 (9) (2010) 1027–1037.

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

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

[19] S. Costello, G. François, G., D. Bonvin, Real-time optimization when the plant and the model have different inputs, in: Proc. IFAC Symp. DYCOPS, Mumbai, 2013, pp. 39–44.

[20] S. Costello, G. François, D. Bonvin, A. G. Marchetti, Modifier adaptation for constrained closed-loop systems, in: Proc. IFAC World Congress, Vol. 19, 2014, pp. 11080–11086.

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

[22] T. Faulwasser, D. Bonvin, On the use of second-order modifiers for real-time optimization, in: Proc. IFAC World Congress, Vol. 19, 2014, pp. 7622–7628.

[23] G. A. Bunin, On the equivalence between the modifier-adaptation and trust-region frameworks, Comp. Chem. Eng. 71 (2014) 154–157.

[24] D. Navia, L. Briceno, G. Gutiérrez, C. de Prada, Modifier-adaptation methodology for real-time optimization reformulated as a nested optimization problem, Ind. Eng. Chem. Res. 54 (48) (2015) 12054–12071.

[25] D. Navia, D. Villegas, I. Cornejo, C. de Prada, Real-time optimization for a laboratory-scale flotation column, Comp. Chem. Eng. 86 (2016) 62–74.

[26] R. Barkley, R. Motard, Decomposition of nets, The Chemical Engineering Journal 3 (1972) 265–275.

[27] A. Marchetti, P. Luppi, M. Basualdo, Real-time optimization via modifier adaptation integrated with model predictive control, in: Proc. IFAC World Congress, Vol. 18, 2011, pp. 9856–9861.

[28] T. J. Williams, R. E. Otto, A generalized chemical processing model for the investigation of computer control, Trans. of the American Inst. of Elec. Engineers, Part I: Communication and Electronics 79 (5) (1960) 458–473.

[29] P. D. Roberts, An algorithm for steady-state system optimization and parameter estimation, Int. J. Systems Sci. 10 (7) (1979) 719–734.

[30] C. Fleury, First- and second-order convex approximations strategies in structural optimization, Struct. Optim. 1 (1989) 3–10.

[31] W.-H. Zhang, C. Fleury, A modification of convex approximation methods for structural optimization, Comput. Struct. 64 (1997) 89–95.