Modifier Adaptation for Run-to-Run Optimization of Transient Processes

Sean Costello, * Grégory François, * Bala Srinivasan, **
Dominique Bonvin *

* Laboratoire d'Automatique, École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland
(e-mail: dominique.bonvin@epfl.ch).
** Department of Chemical Engineering, École Polytechnique de Montréal, Canada, H3C 3A7 (e-mail: bala.srinivasan@polymtl.ca).

Abstract: Dynamic optimization can be used to determine optimal input profiles for dynamic processes. Due to plant-model mismatch and disturbances, the optimal inputs determined through model-based optimization will, in general, not be optimal for the plant. Modifier adaptation is a methodology that uses measurements to achieve optimality in the presence of uncertainty. Modifier-adaptation schemes have been developed for the real-time optimization of plants operating at steady state. In this paper, the concept of modifier adaptation is extended to transient plants such as batch processes. Two different schemes are proposed, and their performance is illustrated via the simulation of a semi-batch reaction system.

Keywords: Dynamic optimization, Real-time optimization, Run-to-run optimization, Modifier adaptation, Batch processes.

1. INTRODUCTION

Dynamic optimization provides a framework for increasing the productivity of dynamic processes, without violating safety and quality constraints. The problem is that most optimization techniques are model-based, and reliable models are rarely available at the industrial level. This is particularly true for batch processes, where the effort required for accurate modeling is often incompatible with short product life-time and the necessity for fast time-to-market (Bonvin (1998)). The reactions in batch processes are often more complex than those encountered in continuous processes, thus making the development of detailed models an extremely challenging task (Filippi-Bossy et al. (1989)). In addition to modeling uncertainty, disturbances occur due to variations in the initial conditions and the operating conditions.Luckily, batch processes are usually repeated, and often measured quantities are available at the end of each batch. Run-to-run measurement-based optimization uses these measurements to reduce the uncertainty and recover optimal operation for future batches.

Measurement-based optimization techniques for transient processes can be separated into two categories, namely fixed-rule methods and repeated-optimization methods. The distinction is based upon how the available measurements are used. Fixed-rule methods try to improve the input profiles without solving an optimization problem between batches. NCO tracking (François et al. (2005)), interpolation-based algorithms (Krothapally et al. (1999)), and measurement-based gradient search methods (Ge et al. (2000), Zafiriou and Zhu (1990)) fall into this category. Repeated-optimization methods, on the other hand, modify the optimization problem and recompute the input profiles. Typically the so-called ‘two-step’ approach is used. The available measurements are used in a first step to re-identify the dynamic model at the end of each batch, and this refined model is used in a subsequent step to optimize for the next batch. An exhaustive review of measurement-based optimization techniques, including run-to-run schemes, is given in Srinivasan et al. (2003). Although recent publications elaborate on some aspects of these techniques (Kadam et al. (2007)), to the best of the authors’ knowledge, no fundamentally different approaches have since been published.

The two-step repeated-optimization approach has the advantage of being particularly transparent, and as such it is more easily accepted in industry. It has been well documented in the literature (Clarke-Pringle and MacGregor (1998), Cruse et al. (2006)). It suffers, however, from two main drawbacks: (i) The inputs determined through optimization may not provide sufficient excitation to identify the model parameters, and (ii) in the case of structural plant-model mismatch, it is very unlikely that plant optimality can be achieved (Forbes et al. (1994)).

The same problems occur when continuous processes are optimized in real-time, which is static optimization. Extensive work has shown that, rather than trying to identify the plant, it often makes more sense to directly modify the cost and the constraints of the optimization problem (Roberts (1979), Gao and Engell (2004), Marchetti et al. (2010)). In the literature, this is often referred to as modifier adaptation. However, when the process is transient in nature, a level of complexity is added to the problem. This paper extends the modifier-adaptation concept to transient (batch) processes. A first attempt was made by Marchetti et al. (2007) using a continuous-time formulation and correction terms on the constraint functions.
In contrast, this work uses a discrete-time formulation. In this context, two approaches are possible. The correction terms may be placed on either the dynamic equations or the cost and constraints. Either way, the cost and the constraints of the optimization problem are modified, whether it is indirectly or directly. It is shown that the two methods share many similarities, yet differ considerably regarding the measurements required for their implementation. The first method requires that the constraints and the cost be measured, while the second method assumes full state measurements (or estimation). The merits of each method are discussed, although the question of which one is more useful is left open.

The paper is organized as follows. Section 2 formulates the problem and presents two schemes for its solving. The performance of these schemes is illustrated in Section 3 via the simulation of a semi-batch reaction system. Conclusions are drawn in Section 4.

2. RUN-TO-RUN OPTIMIZATION WITH MODIFIER ADAPTATION

2.1 Problem statement

Using a discrete-time formulation, the optimal input profiles for a batch process are obtained by solving the following model-based dynamic optimization problem:

\[
U^* = \arg\min_U \phi(x[j])
\]

\[
\text{s.t. } \begin{align*}
x[j+1] &= F(x[j], u[j]), \quad x[0] = x_0, \\
g(X, U) + \epsilon_k &\leq 0,
\end{align*}
\]  

where the superscript \((\cdot)^*\) indicates a quantity calculated via optimization, \(x\) is the \(n_x\)-dimensional vector of states, \(u\) is the \(n_u\)-dimensional vector of inputs, \(\phi\) is the terminal cost, \(F\) is the discrete dynamic model, \(g\) is a vector of constraints, and \(j\) is the fixed final sampling instant. Let us denote \(U = [u[0]^T, \ldots, u[j-1]^T]^T, X = [x[0]^T, \ldots, x[j]^T]^T\).

The difficulty is that \(U^*\) determined this way is optimal for the model, and not necessarily for the plant. In practice, if \(U^*\) is applied to the plant, the resulting performance will probably be sub-optimal and may even be infeasible. This is because the plant behaves differently to the model, i.e. \(F_p \neq F\), where the subscript \((\cdot)_p\) denotes a quantity related to the plant, in this case the dynamic equations.

Fortunately, batch processes are typically repeated many times, and measurements are generally available at the end of each batch. In run-to-run optimization, the idea is to iteratively use these measurements to achieve optimality and feasibility for the plant in as few runs as possible.

Optimization problem (1) can be modified at the end of each batch, either via the cost and the constraints, or through the discrete dynamic equations, as discussed next.

2.2 Method A - Cost and constraint modification

Let \(U^*_k\) and \(X^*_k\) be the optimal trajectories for the \(k\)th model-based optimization problem. \(U^*_k\) is applied to the plant, with the resulting state profile, \(X_{p,k}\). The model-based optimization problem for the \((k+1)\)th batch is defined as:

\[
U^*_k+1 = \arg\min_U \phi(x[j]) + \lambda^\phi_k(U - U^*_k)
\]

\[
\text{s.t. } \begin{align*}
x[j+1] &= F(x[j], u[j]), \quad x[0] = x_0, \\
g(X, U) + \epsilon_k + \lambda^g_k(U - U^*_k) &\leq 0,
\end{align*}
\]

where the modifier terms are defined as:

\[
\epsilon_k = g(X_{p,k}, U^*_k) - g(X^*_k, U^*_k),
\]

\[
\lambda^\phi_k = \frac{d\phi(x_{p,k}[j])}{dU} - \frac{d\phi(x^*_k[j])}{dU},
\]

\[
\lambda^g_k = \frac{dg(X_{p,k}, U^*_k)}{dU} - \frac{dg(X^*_k, U^*_k)}{dU}.
\]

Note that only the cost and the constraints are modified. It can be shown that this approach is equivalent to standard modifier adaptation for the static case. The derivative operator \(\frac{d}{dU}\) represents the derivatives with respect to \(U\), taking into account that \(X\), and only \(X\), is determined by \(U\).

The \(0^{th}\)-order term, \(\epsilon_k\), matches the values of the constraints for the plant and the model at \(U^*_k\). The \(1^{st}\)-order terms, \(\lambda^\phi_k\) and \(\lambda^g_k\), match the gradients of the cost and the constraints for the plant and the model, also at \(U^*_k\). As we will see in Section 2.4, this can be used to ensure optimality upon convergence.

The modifier \(\epsilon_k\) is determined by measuring the values of the constraints for the plant. If only the \(0^{th}\)-order term is used, i.e. \(\lambda^\phi_k = 0\) and \(\lambda^g_k = 0\) for all \(k\), we will refer to the method as Method A-0. The method is then equivalent to that proposed by Marchetti et al. (2007). If the \(1^{st}\)-order correction terms are used, the method is referred to as Method A-1.

2.3 Method B - Modification of the difference equations

An alternative to correcting the cost and the constraints is to add modifier terms to the discrete dynamic equations, \(F\). At the end of the \(k^{th}\) batch, the model-based optimization problem for the \((k+1)\)th batch then reads:
Fig. 2. Method B-0, which modifies the dynamic equations.

\[
\begin{align*}
U_{k+1}^{*} &= \min_{U}\phi(x(f)) \\
\text{s.t.} & \quad x[j+1] = F(x[j], u[j]) + c_k[j] \\
& \quad + C_k[j]\left[ x[j] - x_{p,k}[j]\right], \quad x[0] = x_0, \\
& \quad g(X, U) \leq 0.
\end{align*}
\]  

where the modifier terms are defined as:

\[c_k[j] = x_{p,k}[j+1] - x_{m,k}[j+1],\]  
\[C_k[j] = [A_{p,k}[j] - A_k[j], B_{p,k}[j] - B_k[j]],\]

with \(x_{m,k}[j+1] = F(x_{p,k}[j], u_k[j])\), the one-step-ahead state prediction that the modified dynamic equation would have made. \(A_{p,k}[j]\) and \(B_{p,k}[j]\) are the partial derivatives of the dynamic equations for the plant:

\[
A_k[j] = \frac{\partial F_p(x_{p,k}[j], u_k[j])}{\partial x[j]}, \quad j = 0, \ldots, f - 1. \tag{9}
\]
\[B_k[j] = \frac{\partial F_p(x_{p,k}[j], u_k[j])}{\partial u[j]}, \quad j = 0, \ldots, f - 1. \tag{10}
\]

It is particularly interesting to examine the effect of the \(0^{th}\)-order modifier, \(c_k[j]\). We call this a \(0^{th}\)-order correction because the correction involves only the values of the states. For any \(x_{p,k}[j]\) and \(u_k[j]\) from the \(k^{th}\) batch, \(c_k[j]\) is the difference between what the model would have predicted for the states at the sampling instant \(t_{j+1}\) and the states which actually occurred, \(x_{p,k}[j+1]\). Hence, \(c_k[j]\) can be seen as a one-step-ahead prediction correction. This ensures that the modified dynamic equation for the \((k+1)^{st}\) batch have no prediction error for the input trajectory \(U_k^{*}\), that is:

\[
F(x_{p,k}, u_k) + c_k[j] = F_p(x_{p,k}, u_k) \quad \forall j.
\]

If only the \(0^{th}\)-order term \(c_k[j]\) is used, i.e. \(C_k[j] = 0\), the method will be referred to as Method B-0. If the \(1^{st}\)-order term, \(C_k[j]\), is also used, the method will be referred to as Method B-1.

2.4 Properties upon convergence

The attraction of Methods A-0, A-1, B-0 and B-1 is that, upon convergence, it is possible to prove that either feasibility (Methods A-0 and B-0), or feasibility and optimality (Methods A-1 and B-1) will be achieved for the plant.

**Theorem 1.** If either Method A-0 or Method B-0 converges, then it will converge to an input trajectory that is feasible for the plant.

**Theorem 2.** If either Method A-1 or Method B-1 converges, then it will converge to an input trajectory that satisfies the \(1^{st}\)-order necessary conditions of optimality for the plant.

Outlines of the proofs are given in Appendix A.
3. SIMULATED EXAMPLE

3.1 Plant

Methods A-0, A-1, B-0 and B-1 are tested on the simulated example of an isothermal semi-batch reactor (Ruppen et al. (1998)). Two reactions occur: $A + B \rightarrow C$ and $2B \rightarrow D$. The objective is to maximize the production of $C$ at a fixed final time. The manipulated variable is the feed rate of $B$. There are limits on the concentrations of $B$ and $D$ at final time.

If we consider the state vector $[c_A, c_B, V]^T$, where $c_A$ and $c_B$ are the concentrations of $A$ and $B$ and $V$ is the volume of material in the reactor, the plant is governed by the continuous dynamic equations:

$$
\begin{align*}
\dot{c}_A &= -k_1 c_A c_B V + \frac{1}{V} \left( \frac{c_A}{c_B} - c_B \right) u, \quad (12) \\
\dot{c}_B &= -k_1 c_A c_B V - 2k_2 c_B V + \frac{1}{V} \left( \frac{c_B}{c_A} - c_A \right) u.
\end{align*}
$$

The concentrations of $C$ and $D$ can be expressed as:

$$
c_C = \frac{1}{V} (c_{A0} V_0 - c_A V), \\
c_D = \frac{1}{V} ((c_A + c_{Bin} - c_B) V - (c_{A0} + c_{Bin} - c_{Bo})). \quad (13)
$$

The numerical values used in this simulation are given in Table 1. We consider a piecewise-constant input, with the sampling period $h = 15.625$ min between each switching instant.

3.2 Model

The discrete-time nominal model is:

$$
\begin{align*}
\begin{bmatrix}
x_1[n+1] \\
x_2[n+1] \\
x_3[n+1]
\end{bmatrix} &= \begin{bmatrix}
x_1[n] \\
x_2[n] \\
x_3[n]
\end{bmatrix} + h \begin{bmatrix}
-k_{1,0} x_1[n] x_2[n] \\
-k_{1,0} x_1[n] x_2[n] - 2k_{2,0} x_2[n] \kappa_n \\
\end{bmatrix}
+ \begin{bmatrix}
\kappa_n x_2[n] - x_3[n] \\
0
\end{bmatrix} \cdot u[n], \\
\end{align*}
$$

where $k_{1,0}$, $k_{2,0}$ and $\kappa_n$ are the nominal values of the uncertain parameters, and the concentrations $c_c$ and $c_D$ are given by equations (13) and (14). The model-based optimization problem (without modifiers) is formulated as:

$$
U^* = \arg \max_U x_3[n] c_C[n] \quad (16)
$$

s.t. $x[n+1] = F(x[n], u[n])$, $x[0] = x_0$, equations (13) - (14)

$$
\begin{align*}
&u_{\min} \leq u[n] \leq u_{\max} \\
&c_B[n] \leq c_{Bf,\max} \\
&c_D[n] \leq c_{Df,\max}.
\end{align*}
$$

Notice that all the uncertainty in the model-based optimization problem is contained in $F$.

![Fig. 3. Optimal inputs: plant in solid, nominal model in dashed.](image)

| $k_1$ | 0.053 | 1/mol min |
| $k_2$ | 0.128 | 1/mol min |
| $k_{B_0}$ | 0.03 | 1/mol min |
| $\kappa$ | 2 | . |
| $c_{Bin}$ | 5 mol/l | |
| $u_{\min}$ | 0 | 1/min |
| $c_{Bf,\max}$ | 0.001 | 1/min |
| $c_{Df,\max}$ | 0.15 | mol/l |
| $c_{D_0}$ | 0.05 | mol/l |
| $V_0$ | 1 l | |
| $f$ | 16 | |

### Table 1. Parameters, operating bounds and initial conditions.

3.3 Simulation of A-0 and B-0

Figure 3 displays the optimal input for the plant and for the nominal model. The convergence towards feasibility for Methods A-0 and B-0 is shown in Figure 4. Both methods also converge to near optimality. Figures 5 and 6 indicate that the convergence path is infeasible for both methods. Indeed, there is no guarantee that either of the methods will converge following a feasible path. Constraint violation could be avoided by including an inner controller to monitor constraints. Another possibility would be to add a "back-off" (negative bias) to the constraint.

3.4 Simulation of A-1 and B-1

Figure 4 also shows that both Methods A-1 and B-1 converge to optimality, although by different paths. The path by which Method A-1 converges results in violation of the constraint on $B$, while the path by which Method B-1 converges results in violation of the constraint on $D$. This again highlights that, for a practical implementation,
It is assumed that a method is available for calculating suitable inputs, such that the gradients required in Method B-1 could be determined. Given this assumption, Method A-1 requires 17 batches to compute each set of gradients, while Method B-1 requires only 5 batches.

It is interesting to note that the performance of Methods A-0 and B-0 is similar to that of Methods A-1 and B-1. This is because, for this particular problem, achieving optimality is mainly dependent on meeting the active constraints. Methods A-1 and B-1 are expected to outperform Methods A-0 and B-0 for problems where optimality is mainly dependent on driving the cost sensitivity to zero.

It is worth noting that the convergence of the proposed algorithms can be significantly altered by filtering the modifier terms. For example, a low-pass filter generally results in slower, yet less oscillatory convergence.

4. CONCLUSIONS

In the presence of modeling uncertainty, resolving the model-based optimization problem will, in general, not lead to optimality for the plant. Modifier adaptation is a methodology for using plant measurements to achieve optimality despite modeling uncertainty. Until now, this has only been applied to processes operating at steady state. In this paper, we have successfully extended the modifier-adaptation concept to transient (batch) processes.

Two measurement-based optimization methodologies have been proposed. Method A modifies the cost and the constraints in the model-based optimization problem, while Method B modifies the dynamic equation. Each methodology has two variants, depending on whether gradients are used, giving a total of four methods.

Both methodologies guarantee feasibility upon convergence, and also optimality if gradient terms are used. An attractive aspect of the methods is that this holds even in the case of structural mismatch between the nominal model and the plant. This is because, unlike the standard two-step approach, the corrections are aimed at correcting the necessary conditions of optimality, rather than the (structurally incorrect) model of the process.

The methods were compared in simulation, which showed that there is little difference in terms of speed of convergence. The main difference between the two methodologies lies in the measurements required to obtain the necessary modifiers.

This paper raises an important question: Is it better to modify the cost and constraints (Method A) or the dynamic equations (Method B)? A preliminary discussion on this subject in Section 2.5 shows that Method B has the potential to be more useful, yet has greater implementation difficulties.

**REFERENCES**


### Appendix A. OUTLINE OF CONVERGENCE PROOFS

#### A.1 Method A

We assume that there exists a unique solution to the difference equations $F$, which can be written as an explicit function of the control variables:

$$x[j] = S_j(U).$$

The optimization problem (1) can be written as:

$$U^* = \text{arg min}_{U} \Phi(U)$$

s.t. $$G(U) \leq 0,$$

where $\Phi(U) = \phi(S_j(U))$ and $G(U) = g(S(U), U)$. From this formulation we can see that Method A is equivalent to modifier adaptation for the static case. The proofs regarding optimality and feasibility upon convergence are given in Marchetti et al. (2010).

#### A.2 Method B

If convergence occurs, with $\lim_{k \to \infty} U_k^* = U_\infty^*$, then, due to the $0^\text{th}$-order correction terms, there is no prediction error, and:

$$G_\infty(U_\infty^*) = G_p(U_\infty^*),$$

where $G_p$ are the plant constraints. Hence, feasibility is obtained upon convergence.

The effect of the $1^\text{st}$-order correction term is a little less obvious. We will first show that $\nabla \Phi(U)$ can be expressed as a function of the partial derivatives of $F$ and $\phi$. We have:

$$\nabla_{u[r]} \Phi(U) = \frac{\partial \phi(x[j])}{\partial x[f]} \frac{\partial S_f(U)}{\partial u[r]},$$

Differentiating $S_f$ using the chain rule gives:

$$\frac{\partial S_f(U)}{\partial u[r]} = \sum_{i=1}^{r+1} \frac{\partial F(x[i], u[i], i)}{\partial x[i]} \frac{\partial F(x[r], u[r], r)}{\partial u[r]}.$$

Combining this with equation (A.4) shows that $\nabla \Phi(U)$ depends on the partial derivatives of $F$ and $\phi$. In the same manner, we can express $\nabla G(U_p)$ as a function of the partial derivatives of $g$ and $F$. Such expressions are used in numerical optimization to compute functional derivatives via forward sensitivity analysis (Rossenwasser and Yusupov (2000)). We have assumed that the only uncertainty in the model-based optimization problem comes from $F$. The above development shows that by correcting the gradient of $F$ we also correct the gradients of $\Phi$ and $G$. Hence, we have:

$$\nabla \Phi_\infty(U_\infty^*) = \nabla \Phi_p(U_\infty^*),$$

$$\nabla G_\infty(U_\infty^*) = \nabla G_p(U_\infty^*),$$

where $\Phi_p$ is the cost for the plant. From this, it is easily shown that, if $U_\infty^*$ is a KKT point for the model, it is also a KKT point for the plant.