RUN-TO-RUN OPTIMIZATION VIA CONSTRAINT CONTROL

B. Srinivasan*, C. J Primus†, D. Bonvin*, and N. L. Ricker†

**Institut d'Automatique, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland*
†Department of Chemical Engineering, University of Washington, Seattle, Washington, USA

Abstract: Run-to-run optimization methodologies exploit the repetitive nature of batch processes to find the optimal operating policy in the presence of uncertainty. For the class of batch optimization problems where the solution is determined by terminal constraints, the update of the decision variables towards their optimal values is realized using a constraint control scheme. The methodology is adapted to improve the cost function from batch to batch without constraint violation. (*Copyright ©2000 IFAC*)

Keywords: Optimization, Batch processes, Run-to-Run optimization, Batch-to-Batch optimization, Constraint control

1. INTRODUCTION

Batch and semi-batch processes are of great importance to the fine chemicals industry. Three characteristics that differentiate batch processes from continuous processes are (i) unsteady-state operation, (ii) limited operating time, and (iii) the repetitive nature. Optimization of such processes have received increased attention since, in the face of increased competition, optimization is a natural choice for cutting production costs. In addition, industry sees the potential of process optimization with respect to safety, product quality and ease of scale-up (Bonvin, 1998).

Classically, optimal control theory has been utilized in the literature to calculate the input profile for specific batch processes (Ogunnaike and Ray, 1994). In most cases, the proposed implementation has been open-loop. However, open-loop implementation of the optimal input trajectory may not lead to optimal product formation due to uncertainty in initial conditions and model parameters, and to process disturbances.

Traditionally, batch processes have been operated with very little instrumentation. However, in the last two decades, sensor technology has improved considerably for the purpose of monitoring the production on-line (Nichols, 1988). For instance, the measurement of melt index and monomer concentration by IR spectroscopy is fairly standard in the polymer industry. Hence, these measurements can be used effectively to cope with uncertainty. This way, the focus of optimization is shifted from being a *model-based* framework to being a *measurement-based* framework.

Since batch processes are intended to be run repeatedly, it is logical to exploit this repeatable nature for process optimization. The goal of batch-to-batch optimization is to find iteratively the optimal operating conditions in the presence of uncertainty, while performing the fewest number of sub-optimal runs. There have been several proposals in the literature that take advantage of batch-

to-batch similarities for input profile optimization of batch processes (Filippi *et al.*, 1989; Zafiriou and Zhu, 1990). Run-to-run optimization is also of interest in the semiconductor and related industries (Adivikolanu and Zafiriou, 1998). On the one hand, model-free approaches such as evolutionary optimization (Box and Draper, 1987) or MultiSimplex (Guide, 1999) use many batch runs to converge to an optimal solution. On the other hand, model-based approaches could converge faster but they suffer from the lack of accuracy of the model, especially when the model needs to be identified in the presence of noise. As a compromise, a scheme that only necessitates a structurally-correct model will be proposed in this paper.

Constraints play an important role in optimization. In continuous processes, the optimal operating policy is often determined by constraints (Maarleveld and Rijnsdorp, 1970). The same is true for batch processes, but the dependence of the input profile on the constraints (especially terminal constraints) is considerably more complicated. The run-to-run optimization methodology proposed in this work is for batch processes whose optimal solution is governed by terminal constraints, i.e., constraints which depend only on the final condition of the batch. Typical terminal constraints arise from selectivity considerations, where the concentration of particular species (e.g. side products) must be less than specified values to facilitate down-stream processing or simply to avoid additional separation steps.

2. PRELIMINARIES

In this paper, the study will be restricted to a typical problem structure frequently encountered in batch process optimization. In most batch chemical processes, the manipulated inputs are flow rates that enter the system equations in an affine manner. Examples of typical manipulated inputs include flow rates of hot and cold fluids, and flow rates of reactants. Such systems are called affine-in-input systems or control-affine systems. Furthermore, the batch objective involves meeting certain specifications only at the *end* of the batch cycle. Typical cost functions are the maximization of yield or ratio between two products at final time. Thus, the objective function depends only on the final state. Without loss of generality, the final time, t_f, can be assumed to be fixed.

Let us assume that the parameters of the model, θ, are unknown but constant from batch to batch. To cope with this uncertainty, the run-to-run optimization utilizes measurements taken from one batch operation to improve the optimal operating policy in subsequent batches. The fact that the batch operation is repetitive is exploited to converge to the optimal operating policy over successive batch runs. The run-to-run terminal-cost optimization problem under uncertainty can be formulated as follows:

$$\min_{u^k(t)} J^k = \phi(x^k(t_f)) \tag{1}$$

$$\begin{aligned} s.t.\ \dot{x}^k &= f(x^k,\theta) + g(x^k,\theta)\, u^k + d^k \\ x^k(0) &= x_0 \\ y^k &= h(x^k,\theta) + v^k \\ S(x^k,u^k) &\le 0 \\ T(x^k(t_f)) &\le 0 \\ \text{given}\ \ y^j(i),\ \forall\, i,\ \forall j &\le (k-1) \end{aligned} \tag{2}$$

where J^k is the cost function, $x^k(t) \in \Re^n$ is the state, $u^k(t) \in \Re^m$ is the input, and $d^k(t)$ is the process noise of the k^{th} batch. Let $y(t) \in \Re^p$ be the combination of states that can be measured, $y^k(i)$ the i^{th} measurement in time taken during the k^{th} batch, and $v^k(t)$ the measurement noise. $f \in \Re^n$ and $g \in \Re^{n\times m}$ describe the system dynamics, $S \in \Re^\sigma$ the path constraints, $T \in \Re^\tau$ the terminal constraints, and $\phi \in \Re$ the cost function. The objective is to utilize the measurements from the previous $(k-1)$ batches to handle the uncertainty in θ and find the open-loop optimal input policy for the k^{th} batch.

The solution of terminal-cost optimization of control-affine systems has the following properties (Bryson and Ho, 1975):

- The input is in general discontinuous; yet, in between discontinuities, the input is analytic.
- Two types of intervals (nonsingular and singular) are possible between two switching instants; analytical expressions can be obtained for either type of interval.

The pieces described above are sequenced in an appropriate manner to obtain the optimal solution. The sequence of pieces and the switching times need to be determined. Though analytical expressions are available for the input in either type of intervals, for ease of implementation, the input is considered to be piecewise constant or linear. In summary, the sequence of arcs, the corresponding switching times, and the coefficients of the approximation completely parameterize the input.

It is reasonable to assume that the sequence of pieces is unaffected by the parametric uncertainty, while the switching times and the coefficients of the approximation depend on the uncertain parameters. Let these decision variables, i.e., the switching times and the coefficients of the approx-

imation, be represented by ν. Let ν^k be the set of decision variables for the k^{th} batch run. The goal of the run-to-run optimization is to choose the decision variables in such a manner that, as k increases, the computed solution approaches the (unknown) optimal solution of the real system.

Depending upon whether or not a model is used, various strategies for run-to-run optimization are possible.

Model-free Evolutionary Optimization:
In this approach, no model is used, and the performance of a proposed input is evaluated experimentally. The algorithm for evolutionary optimization is as follows:

(1) Parameterize the input using a finite number of decision variables ν and choose the corresponding initial values.
(2) Run the batch with the given input and compute the performance index and the path and terminal constraints from the measurements.
(3) Sequentially perturb every component of ν, each time re-running the batch in order to calculate the corresponding gradient, and thus the search direction.
(4) Use an optimization algorithm (such as steepest descent) to update ν. Repeat Steps 2-4 until the objective function is minimized.

As can be seen, $\dim(\nu) + 1$ batch runs are necessary for each optimization iteration. Also note that the optimization algorithms that do not use gradient information typically converge more slowly, thereby requiring even more process runs.

Evolutionary Optimization with Model-based Gradient:
The expensive part in the evolutionary optimization approach, in terms of the number of batch runs, is the calculation of the gradient (Step 3). The key idea in the model-based gradient approach is to use a dynamic model of the process, instead of an experimental run, to calculate the gradient. The model can be run as many times as the dimension of ν to obtain the gradient. Another possibility is to use the Hamiltonian formulation, where the gradient is calculated from the states and adjoint variables. The states are measured or inferred from an experimental batch run. The adjoint variables are obtained from the model of the system by integrating the adjoint equations backward in time (Zafiriou and Zhu, 1990).

Optimization via Model Refinement:
This approach uses a model of the process for optimization and refines it using information gathered from previous batches. An optimization problem is solved before each batch run with the refined model. The algorithm is as follows:

(1) Choose initial guesses for the parameters θ.
(2) Use the model and an optimization algorithm to obtain the optimal ν.
(3) Run the batch open-loop with the optimal ν.
(4) Use an identification algorithm and all the available measurements to obtain a new estimate for θ. Repeat Steps 2-4 until convergence.

The model-free evolutionary optimization has the drawback of using numerous batch runs to calculate the gradient, while, when a model is used, the accuracy of the model becomes crucial. If the model has to be refined, care should be taken to guarantee that the input is persistently exciting to uncover the parameters that have to be identified. This is normally not the case when the optimal input is implemented. Thus, there exists a clear conflict between identification and optimization (Roberts and Williams, 1981).

The scheme proposed in this paper attempts to resolve this conflict for batch processes whose optimal solution is governed by terminal constraints. The scheme lies in between the model-free evolutionary optimization and evolutionary optimization with model-based gradient. The gradient is obtained implicitly from the structure of the problem. Hence, no model is used for the implementation and no parameters need to be adapted. However, a *structural model* is necessary to devise the scheme.

3. OPTIMIZATION *VIA* CONSTRAINT CONTROL

Consider the class of problems where optimal operation, in the absence of terminal constraints, corresponds to constant operating conditions (input on one of the bounds with no switching, such as the batch mode in chemical reaction systems). This happens in controllable linear systems or when the input-cost linearization of the system is controllable. For these cases, which are quite frequent in batch processing, the optimization potential arises solely from selectivity considerations. Thus, the optimal solution is governed by terminal constraints. Without loss of generality, one can assume all terminal constraints active - the inactive constraints being simply removed from the optimization problem.

For a given parameterization of the input, the optimal solution consists in choosing ν such that all the terminal constraints are active. Hence, the deviations from the constraints represent a measure of non-optimality and also gives the direction to update ν. The algorithm is as follows:

(1) Use prior knowledge or a structural model to determine the active terminal constraints. Inactive constraints are removed from the optimization problem.
(2) Parameterize the input and choose an initial guess of parameters ν^1, $k = 1$.
(3) Run the batch using the input corresponding to ν^k. Compute $T(x^k(t_f))$ from the measurements at t_f.
(4) Update the input parameters using $\nu^{k+1} = \nu^k + K\ T(x^k(t_f))$, with K being an appropriate gain matrix. Set $k = k + 1$ and repeat Steps 3-4 until convergence.

This procedure is similar to evolutionary optimization to the difference that extra batch runs are not required to compute the gradient. The gradient is obtained directly from the deviations from the terminal constraints. Also, in the presence of disturbances, the gradient obtained from deviations from terminal constraints is more robust than the gradient calculated using finite differences. This is due to the fact that the gradient calculation using finite differences uses the difference of two quantities corrupted by noise, while that obtained from terminal constraints uses the difference between a quantity corrupted by noise and a fixed value. Thus, the constraint control method does not have to excite the system to get the gradient. Also, the optimization *via* constraint control considerably reduces the number of batch runs required for convergence. Another attractive feature of this approach is its model-free implementation. However, a *structural model* is necessary to determine the active constraints on which the solution will lie.

The update law $\nu^{k+1} = \nu^k + G\ T(x^k(t_f))$ can be easily implemented by a feedback controller. The system to be controlled is a *static* one represented by $\mathcal{S} : \nu \rightarrow T(x(t_f))$ and the reference to be tracked is $T_{ref} = 0$. Note that $\mathcal{S}$ does not represent the dynamic system (2), but just a static map between the input parameterization and the terminal constraints. In this controlled system, the independent variable is the batch index k and not the time t. The controller, which has to be dynamic to avoid algebraic inconsistency, is the only dynamic element in the control scheme (Figure 1). Figure 1 also indicates the effect of process noise on the terminal constraints, T_d, and the effect of the measurement noise, T_v.

The idea presented here is similar to that of tracking constraints using feedback for the sake of optimality (Maarleveld and Rijnsdorp, 1970). The particularities of the present work are: (i) *terminal* constraints are considered, and (ii) the independent variable of the control loop is the batch number. Thus, the solution is implemented open-loop within each batch, and the feedback works over the successive batches.

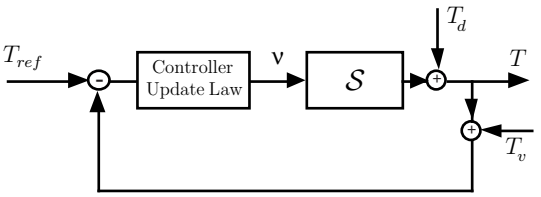

Fig. 1. Optimization *via* tracking of terminal constraints $T_{ref} = 0$

The update law (Step 4) represents an integral controller. To improve the rate of convergence, a proportional or a derivative term can be added, thus resulting in a PI or a PID controller, for the design of which standard controller design methodologies can be used. To reject the effect of the process noise T_d, a high-gain controller is required, while to be insensitive to the measurement noise T_v, the gain of the controller has to be as low as possible. Hence, a compromise has to be reached in the controller design.

The system $\mathcal{S}$ is in general multi-input, multi-output and non-square. However, if the elements of T can be controlled by the input $u(t)$, and if an appropriate parsimonious parameterization of the input is used, one needs only as many degrees of freedom in ν as the dimension of T, thereby leading to a square system $\mathcal{S}$. Note that the static map $\mathcal{S}$ varies considerably with the operating point, which complicates the controller design. Either a centralized or a decentralized controller can be used. In the latter case, a decoupling scheme can be useful. In addition, once the solution has converged to the optimum, one need not keep the control loop active and the adaptation can be stopped. In this case, convergence to the optimum can be tested by looking at the error $T_{ref} - T$ or the adaptation can be stopped after a fixed number of batch runs.

Since constraint violations may mean that the batch is wasted, it is better to be sub-optimal than to violate the constraints. Hence, heuristics can be used to first render the solution feasible and then approach the constraint from within the feasible region. In addition, conservatism needs to be introduced to account for disturbances (process and measurement noise). For example, one can backoff from the active constraint $T_{ref} = 0$ by defining T_{ref} negative so that T remains negative despite disturbances:

$$T_{ref} = -\max_z \left| \frac{T_d(z)}{1 + G(z)\ \mathcal{S}} \right| - \max_z \left| \frac{G(z)\ \mathcal{S}\ T_v(z)}{1 + G(z)\ \mathcal{S}} \right|$$

k_1	0.053	l/mol min
k_2	0.128	l/mol min
k_3	0.028	min^{-1}
k_4	0.001	l/mol min
$c_{b_{in}}$	5	mol/l
c_{a_0}	0.72	mol/l
c_{b_0}	0.05	mol/l
c_{c_0}	0.08	mol/l
c_{d_0}	0.01	mol/l
V_0	1	l

Table 1. Parameter values and Initial conditions

where $G(z)$ is the controller transfer function. Note that $T_d(z)$ is the z-transform of the noise sequence T_d^k, where k is the batch number.

4. EXAMPLE

The methodology proposed in this work will be applied to a semi-batch reactor system, specifically, the acetoacetylation of pyrrole with diketene. The reaction system considered is described below and, for a more detailed description of the process and model, the reader is referred to (Ruppen *et al.*, 1998):

$$\begin{aligned} A + B &\rightarrow C \\ 2\,B &\rightarrow D \\ B &\rightarrow E \\ C + B &\rightarrow F \end{aligned}$$

where A: pyrrole, B: diketene, C: 2-acetoacetyl pyrrole, D: dehydroacetic acid, E: oligomers and F: undesired by-products. The optimization problem is

$$\begin{aligned} \min_{u(t)} J &= -c_c(t_f)V(t_f) \\ s.t.\ \dot{c_a} &= -k_1 c_a c_b - (u/V)\, c_a \\ \dot{c_b} &= -k_1 c_a c_b - 2\, k_2 c_b^2 - k_3 c_b - k_4 c_b c_c \\ &\quad +(u/V)\,(c_{b_{in}} - c_b) \\ \dot{c_c} &= k_1 c_a c_b - k_4 c_b c_c - (u/V)\, c_c \\ \dot{c_d} &= k_2 c_b^2 - (u/V)\, c_d \\ \dot{V} &= u \\ 0 &\le u \le 0.002 \\ c_b(t_f) - 0.025 &\le 0, \quad c_d(t_f) - 0.15 \le 0 \end{aligned}$$

where c_a, c_b, c_c and c_d are the concentrations of A, B, C, and D in mol/l, respectively, in the k^{th} batch. The feed consists of only the species B with concentration $c_{b_{in}}$. The goal is to maximize the number of moles of C at the final time, $t_f = 250$ min, by manipulating the feedrate u in l/min, while satisfying the two terminal constraints on the concentrations of B and D. It is assumed that the concentrations of Species B and D are measured at final time.

The optimal solution for this problem, obtained numerically, can be characterized as having three intervals: (i) input at its maximum value, (ii) input being singular, approximated by an unknown constant value, and (iii) input at its minimum value. The maximum amount of C obtained is 0.4884. From the optimal solution, the natural parameterization corresponds to the following set: t_m, switching time between the maximum input and the singular interval, u_s, the feedrate during the singular interval, and t_s, the switching time between the singular interval and the minimum input.

Two degrees of freedom are necessary to meet the two terminal constraints. To check how much can be gained with the third decision variable, t_m was set to zero and the other two variables were manipulated. The maximum amount of C obtained in such a case is 0.4882, a deterioration of 0.05%, which is indeed negligible. Thus t_m could be set to zero, thereby leaving only two intervals, the singular one with feedrate u_s and the switching to the zero feed at time instant t_s. Hence the parameterization: $\nu = [u_s, t_s]^T$, $T(x(t_f)) = [c_b(t_f) - 0.025, c_d(t_f) - 0.15]^T$. The static map $\mathcal{S}$ obtained at the optimum solution was

$$\mathcal{S} = \begin{bmatrix} 1.2 \times 10^{-3} & 0.5 \times 10^{-3} \\ 0.2 \times 10^{2} & 1.6 \times 10^{2} \end{bmatrix}$$

The scale-independent diagonal dominance of $\mathcal{S}$ was tested using the relative gain array technique, which gave $\lambda = 1.054$ (Ogunnaike and Ray, 1994). This implies that two independent control loops could be constructed, with $c_b(t_f)$ being paired with t_s and $c_d(t_f)$ with u_s. PI-controllers tuned for the static map $\mathcal{S}$ were used for the $t_s \rightarrow c_b(t_f)$ and $u_s \rightarrow c_d(t_f)$ loops with proportional gains, $K_b = 100$ l-min/mol, $K_d = 2.5{\times}10^{-3}$ l^2/mol min, and integral time constants, $T_{i_b} = 0.33$ min and $T_{i_d} = 10$ min, respectively.

An initial input parameterization of $u_s = 0.001$ l/min and $t_s = 175$ min was used. The initial values were chosen in a conservative manner so that the optimal solution is approached from the safe side. While running this simulation, there was a 5% zero-mean gaussian multiplicative measurement noise added to the concentration measurements of B and D. The results of a typical series of runs are shown in Figures 2-4. It can be seen that neither constraint was violated during the series of 150 batches, therefore producing usable product each time. Thus, the introduced backoff

efficiently accounted for the measurement noise. Figure 4 demonstrates the motivation for tracking terminal constraints for optimal control: as the constrained variables are pushed closer to their limits, the objective function is improved, in this case, producing more of the desired product. It is important to note that the main improvement in the cost is realized in the first 20 batches. Clearly after 100 batch runs, the improvement in the cost is marginal and the adaptation can be stopped.

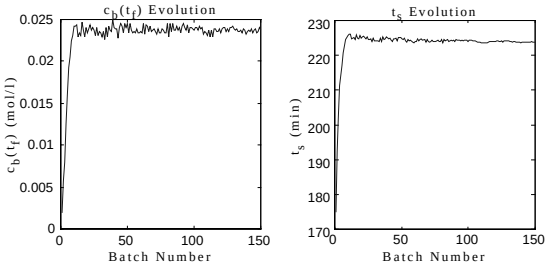

Fig. 2. Performance of $t_s \rightarrow c_b$ loop

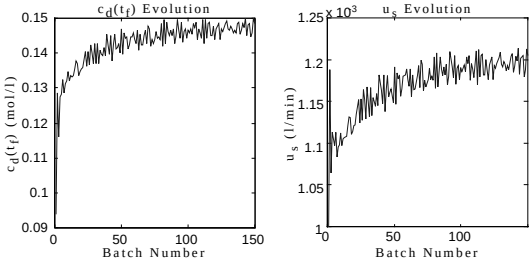

Fig. 3. Performance of $u_s \rightarrow c_d$ loop

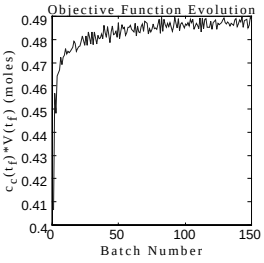

Fig. 4. Objective function evolution

5. CONCLUSION

This work has demonstrated the effectiveness of constraint control for the run-to-run optimization of a class of batch processes, where the solution is determined by terminal constraints. The update of the parameters required for optimization is realized using a simple control scheme. The update direction was obtained implicitly, thereby avoiding the need to excite the system for the estimation the gradient. The methodology has been adapted such that production of the desired product increases with each progressing batch, without constraint violations.

The design of the controller is a direction which merits further research. The gain matrix $\mathcal{S}$ varies with the operation point and hence between iterations. If this variation is large one needs more sophisticated control methodologies to avoid instability problems.

6. REFERENCES

Adivikolanu, S. and E. Zafiriou (1998). Robust run-to-run control for semiconductor manufacturing. In: *Proceedings of the American Control Conference*. Philadelphia. pp. 3687–3691.

Bonvin, D. (1998). Optimal operation of batch reactors - a personal view. *J. Process Contr.* **8**(5-6), 355–368.

Box, G. E. P. and N. R. Draper (1987). *Empirical Model-building and Response Surfaces*. John Wiley, New York.

Bryson, A. E. and Y. C. Ho (1975). *Applied Optimal Control*. John Wiley and Sons, Inc., New York.

Filippi, C., J. L. Greffe, J. Bordet, J. Villermaux, J. L. Barnay, B. Ponte and C. Georgakis (1989). Batch reactor optimization by use of tendency models. *Comp. Chem. Engng.* **13**, 35–47.

Guide, User's (1999). *User's Guide to MultiSimplex(r) 2.0*. MultiSimplex AB, Karlskrona, Sweden.

Maarleveld, A. and J. E. Rijnsdorp (1970). Constraint control of distillation columns. *Automatica* **6**, 51–58.

Nichols, G. D. (1988). *On-line Process Analyzers*. John Wiley, New York.

Ogunnaike, B. A. and W. H. Ray (1994). *Process Dynamics, Modeling and Control*. Oxford University Press.

Roberts, P. D. and T. W. C. Williams (1981). On an algorithm for combined system optimization and parameter estimation. *Automatica* **17**, 199–209.

Ruppen, D., D. Bonvin and D. W. T. Rippin (1998). Implementation of adaptive optimal operation for a semi-batch reaction system. *Comp. Chem. Engng.* **22**, 185–189.

Zafiriou, E. and J. M. Zhu (1990). Optimal control of semi-batch processes in the presence of modeling error. In: *Proceedings of the American Control Conference*. San Diego, CA. pp. 1644–1649.