SRC TR 89-5

Robust Optimal Input Profile Determination for Semi-Batch Processes

by

E. Zafiriou and
A. O. Bamgbose

Robust Optimal Input Profile Determination for Semi-Batch Processes*

Evanghelos Zafiriou
Adebola O. Bamgbose

Chemical and Nuclear Engineering Department
and Systems Research Center
University of Maryland
College Park, MD 20742

Presented at the 1988 AICHE Annual Meeting, Washington, D. C.

Abstract

The input profiles (trajectories) that are implemented during the operation of a semi-batch process are obtained by using the available model of the process. Since modeling error is always present, the perfomance could be very different when these inputs are applied to the actual plant. This paper examines the problem of determining the optimal input trajectory for semi-batch processes in spite of the presence of modeling error, by using information from previous batches to modify the trajectories that are applied to the subsequent ones. The proposed approach does not require the complex remodeling of the process, but instead it redetermines the input profile directly, so that a steady improvement is accomplished from batch to batch.

1 Introduction

Most semi-batch or fed-batch processes of interest are polymerization or biochemical reactions, in which on-line measurements are very hard to obtain. The usual approach for obtaining the input profile is to solve an open-loop control problem by defining an appropriate objective function and optimizing over the input profile. This optimization can in some cases be carried out analytically [6]. Also a numerical solution of the optimization problem is possible [4]. Often approximations of the optimal solution are obtained to avoid the complexities of finding the true optimum. One such approach [1], uses "fictitious" on-line PID controllers to obtain the approximations through repeated simulations.

Regardless of the method used and of whether the true optimal profile or an approximation is obtained, the fact is that the construction of this profile is based on some process model which can often be very different from the plant. The nonlinear optimal control theory is quite extensive (e.g., [3]), but it cannot handle the problem of modeling error which often causes very bad performance when the "optimal" profile is applied to the real plant.

To improve the performance, one should use information from previous batches to improve the operation of the next. To do so, one could simply try to identify more accurately the model parameters and solve again the open-loop nonlinear optimal control problem between batches. This, however, is a very difficult and occasionally impossible task, because of errors in the structure of the often empirical models.

Rather than attempting to redetermine the optimal in one single step between two batches, this paper proposes a new approach for modifying the input profile from batch to batch so that an improvement in the objective function, computed from the actual plant, is acomplished in every batch.

*Supported in part by the National Science Foundations's Engineering Research Centers Program : NSFD CDR 8803012.

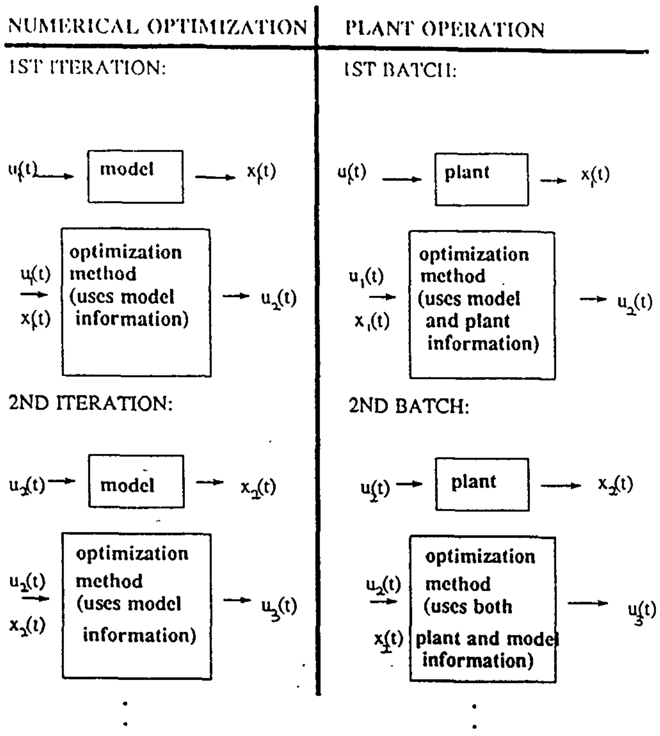
NUMERICAL OPTIMIZATION
PLANT OPERATION
1ST ITERATION:
1ST BATCH:
model
plant
optimization method (uses model information)
optimization method (uses model and plant information)
2ND ITERATION:
2ND BATCH:
optimization method (uses both plant and model information)
u₁(t)
x₁(t)
u₂(t)
x₂(t)
u₃(t)

Figure 1: Analogy between the iterations of the numerical optimization and the successive batches during the plant operation

2 A New Approach to the Problem

The approach followed in this paper is to directly modify the input profile $u(t)$ during the course of successive batches. In doing so, we exploit the ***analogy between the iterations during the numerical optimization of the appropriate cost functional (objective function) on one hand and the successive batches during the operation of the plant on the other.*** More specifically, one way to solve the open-loop control problem is to numerically minimize the appropriate functional by using, e.g., some gradient method [4]. However, if this minimization is based solely on the process model, the resulting input profile will not be optimal for the true plant because of model-plant mismatch, as described above. We propose that this iteration includes information from the actual plant, ***by corresponding each iteration to one of the successive batches.*** This information will come from the measurement of appropriate variables during the batch. Note that since these measurements are not used for on-line control, where almost instantaneous measurements are required, we are not limited in their selection. For example, samples of the product can be gathered during the course of the batch and sent for analysis. The results will be used for the computation of a better $u(t)$ for the next batches. In this way, in every successive batch we will have a performance improvement and finally the input profile will converge to the "optimal" for the true plant.

The starting function $u(t)$ applied to the first batch could be the optimal for the model or some other reasonable good profile. Note, however, that the matter is somewhat more complicated than just described above, because the previous batches cannot necessarily provide all the information needed for the numerical algorithm. Consequently, the process model will be used to provide part of the information required to find the next $u(t)$. Significant model error could then cause a failure to converge to the true optimal profile. Conceptually, this bears a similarity to the Internal Model Control structure for continuous processing systems, where information both from the plant (measurement) and the parallel model is used in computing the next input. The algorithm used in the numerical minimization of the cost functional corresponds to the IMC controller. The same conceptual similarity exists also with the Operator Control theory for continuous processing systems [2], where algorithms for the numerical solution of operator equations are used as compared to the use of algorithms for the numerical optimization of a functional in this project. Figure 1 gives a schematic representation of the analogy that was described in this session.

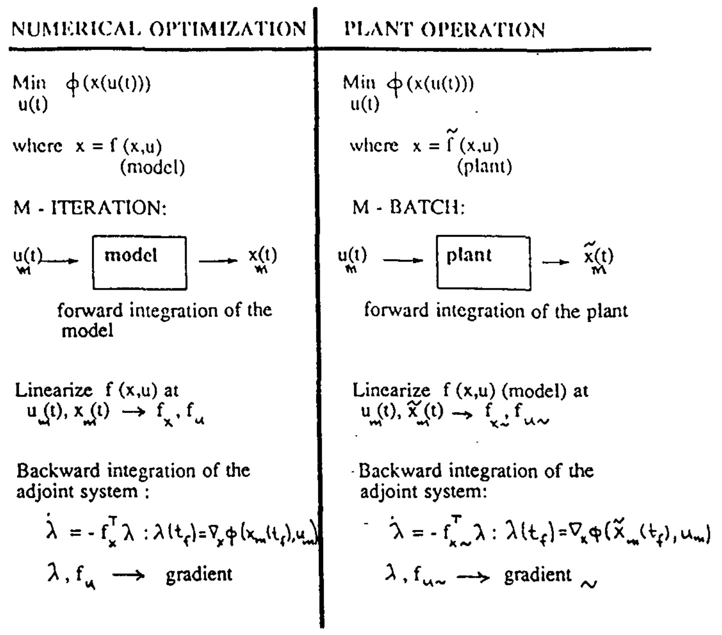
NUMERICAL OPTIMIZATION
PLANT OPERATION
Min φ(x(u(t))) u(t)
where x = f (x,u) (model)
M - ITERATION:
u(t) → model → x(t)
forward integration of the model
Linearize f (x,u) at u(t), x(t) → f_x, f_u
Backward integration of the adjoint system :
λ̇ = - f_x^T λ : λ(t_f)=∇_x φ(x_m(t_f),u_m)
λ, f_u → gradient
Min φ(x(u(t))) u(t)
where x = f̃ (x,u) (plant)
M - BATCH:
u(t) → plant → x̃(t)
forward integration of the plant
Linearize f (x,u) (model) at u(t), x̃(t) → f_x, f_u
Backward integration of the adjoint system:
λ̇ = - f_x^T λ : λ(t_f)=∇_x φ(x̃_m(t_f),u_m)
λ, f_u → gradient

Figure 2: Gradient Computation

3 Computational Issues

Let us now consider the type of information required by the optimization algorithm. When a gradient based method, like Steepest Decent or the Conjugate Gradient is used, the algorithm requires the knowledge of the gradient of the cost functional at the past input function $u_m(t)$. This computation involves two integrations. The first is forward integration of the differential equations describing the plant

$$\dot{x} = f(x, u_m) \tag{1}$$

in order to obtain $x_m(t)$, where to simplify the notation we assume that all of the variables in the vector x appear in the cost functional and they can either be measured or estimated from measurements, speed of measurement not being important. The values of $u_m(t)$ and $x_m(t)$ are then used to compute the derivatives of $f(x, u)$ along these trajectories. The second integration is the backward integration of

$$\dot{\lambda} = -f_x^T \lambda : \lambda(t_f) = \nabla_x \phi(x(t_f, u)) \tag{2}$$

where T means transpose, f_x denotes the derivative of f with respect to x and ϕ is the objective function. The gradient is then obtained from $\lambda(t)$ and f_u (the derivative of f with respect to u).

In the approach described in the previous section, the first forward integration is ***not*** carried out numerically, but rather ***it is carried by the actual plant itself.*** Its result $x_m(t)$ is the result of the previous batch. The remaining computations however require to compute f_x, f_u, at the trajectories of the previous batch $x_m(t)$, $u_m(t)$. This is accomplished by using the process model. Note that f_x, f_u define a linear time-varying system, obtained by linearization of the nonlinear model (1) along the trajectories $x_m(t)$, $u_m(t)$:

$$\dot{\xi} = f_x(t)\xi(t) + f_u(t)v(t) \tag{3}$$

The analogy between the computation of the gradient for numerical optimization, based on the model, and its computation during the plant operation, based both on model and plant operation, is shown schematically in Figure 2.

The exact expressions for the gradient depend on the type of the input $u(t)$. The input trajectories could, for example, be limited to being continuous, piecewise continuous, bang-bang or some other type. The type arising in most applications of interest is that of piecewise continuous:

$$u(t) = c(t) + \sum_{i=0}^{M} h_i[1(t - t_i) - 1(t - t_{i+1})] \tag{4}$$

where $c(t)$ is the continuous function, h_i is the height of the piecewise constant part of the input function in the interval $t_i < t < t_{i+1}$ and $1(t)$ is the unit step function. In this case the optimization is carried out over $c(t), h_0, \ldots, h_M, t_1, \ldots, t_M$. The gradient is given by [4]:

$$G(c,t,h) = \begin{bmatrix} f_u^T \lambda(t), t \in [t_0, t_f] \\ (f|_{t_1-} - f|_{t_1+})^T \lambda(t_1) \\ \vdots \\ (f|_{t_M-} - f|_{t_M+})^T \lambda(t_M) \\ \int_{t_0}^{t_1} f_u^T \lambda(t) dt \\ \vdots \\ \int_{t_M}^{t_f} f_u^T \lambda(t) dt \end{bmatrix} \tag{5}$$

4 Illustration

The following simple example is used to demonstrate that the method can provide an improvement in the objective function (computed for the actual plant) even though a mismatch between the model and the plant may exist. The starting point is chosen to be suboptimal but reasonably good as judged by model simulations. The approach is then applied to both the case of no model error and to that of mismatch. In both cases it seems that a local minimum is eventually reached, but an improvement is accomplished nevertheless in every batch.

The lysine fermentation model proposed by Ohno *et al* [6] is considered:

$$x = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} (\dot{XV}) \\ (\dot{SV}) \\ (\dot{PV}) \\ (\dot{V}) \end{bmatrix} = \begin{bmatrix} \mu x_1 \\ -\sigma x_1 \\ \pi x_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ S_F \\ 0 \\ 1 \end{bmatrix} F \tag{6}$$

$$\mu = 0.125S \tag{7}$$

$$\pi = -384\mu^2 + 134\mu \tag{8}$$

$$\sigma = \mu/0.135 \tag{9}$$

where S, X, P are the concentrations of the substrate, cell, and product respectively; V is the fermentor volume; F is the volumetric feed rate of substrate; S_F is the substrate feed concentration; and μ,σ,and π are the specific rates of growth, substrate consumption, and product formation, respectively. The objective that we used was to minimize the final product concentration $\phi(t_f) = -x_3(t_f)$ with a maximum volume $V_M = 20.00L$ and for a fixed final time $t_f = 35hrs$. The initial conditions were $x_0 = 0.02g/L$, $S_0 = 2.8wt\%$, $P_0 = 0g/L$, $V_0 = 5L$; S_F was $2.8wt\%$

The optimization algorithm used for this example is a modified Steepest Descent Direction Method; modified in the sense that three different stepsize coefficients were used, one for $c(t)$, one for h_0, h_1, h_2, and one for t_1, t_2. After a gradient has been computed, the input profiles for the subsequent batches are obtained by changing the stepsizes (increase or decrease in the powers of 2) until no further improvement can be obtained. Then a new gradient is computed and the procedure continues.

Figure 3 shows the suboptimal initial input profile. The switching times are $t_1 = 3.85hrs, t_2 = 21.83hrs$. A significant improvement in the objective function from batch to batch is observed and eventually, it seems that a local minimum is reached. Table 1 shows some of the values of the objective function obtained after certain batches and Fig. 4 shows the final input profile.

With model-plant mismatch, (obtained by 20% increase in the coefficient in (7) and with the same initial input profile (Figure 3), a similar final input profile (Figure 5) was obtained after three gradient computations. The same trend of improvement in the objective functional is observed (Table 2).

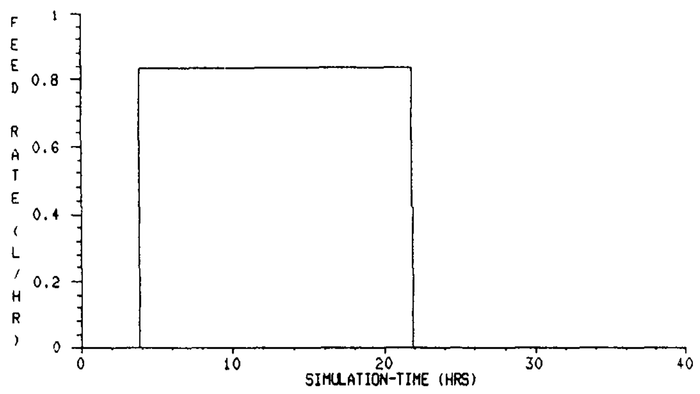

Figure 3: Suboptimal initial input profile

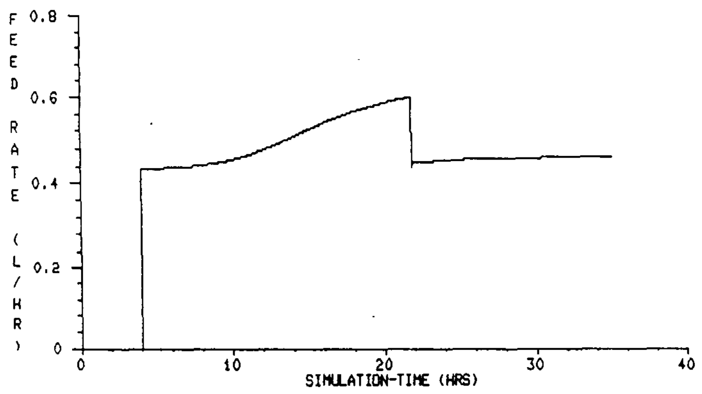

Figure 4: Final input profile (local minimum) for no model-plant mismatch

BATCH	OBJECTIVE FUNCTION
1	-634.414
6	-660.380
8	-660.380
20	-661.21
25	-661.21

Table 1: No model error

BATCH	OBJECTIVE FUNCTION
1	-612.666
6	-651.790
8	-651.790
20	-655.470
25	-655.470

Table 2: Model-plant mismatch

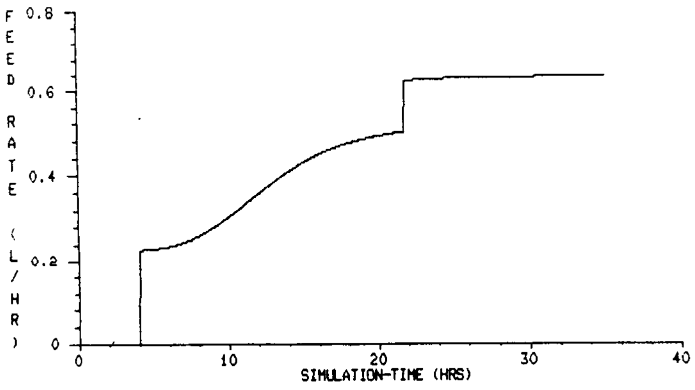

Figure 5: Final input profile (local minimum) for the case of model-plant mismatch

5 Research Issues

The simple example of the previous section demonstrates the validity of the approach that this paper proposes. The method seems to have a lot of promise but its successful practical implementation will depend on obtaining answers to a number of research questions.

The first question to come in mind is the selection of the optimization method. Selecting a particular method and parameter values is conceptually equivalent to selecting the controller for a classical feedback control problem. The presence of local minima as Section 4 illustrates, makes the question even more complex. Our main goal at this stage is to study how different algorithms behave when applied to a few selected examples of polymerization and biochemical semi-batch processes.

Another important objective is to quantify the effect of model uncertainty on the convergence (from batch to batch) properties of the method. The discussion in Section 2 indicates that it is the accuracy with which the linear time varying system described by (3) is known for the past input profile $u(t)$, that will affect the convergence properties. Two issues should be investigated. The one is the derivation of robustness conditions that guarantee convergence, if the error is norm-bounded by a known bound. This will allow the designer to examine the effectiveness of a specific algorithm and possibly introduce appropriate modifications. The other is the question of how one can utilize the information of past batches to obtain a more accurate estimate of (3). This question falls into the research area of identification for control purposes.

Finally, it should be emphasized that the proposed method is not supposed to be a stand-alone technique. It is meant to be used in a complementary fashion to the on-line regulatory control system that is in place to reject disturbances during the operation of the process. The performance of this regulatory controller can interfere with the convergence properties of the batch to batch input profile modification and it is another question that has to be addressed.

References

[1] K. Y. Choi and D. Butala, "Control of Batch Free Radical Polymerization Reactors", Proc. of the Amer. Control Conf., Minneapolis, MN, 1987.

[2] C. Economou, *An Operator Theory Approach to Nonlinear Controller Design*, Ph.D. Thesis, California Institute of Technology, 1985.

[3] R. V. Gamkrelidze, *Principles of Oprimal Control Theory*, Plenum Press: New York, 1978.

[4] L. Hasdorff, *Gradient Optimization and Nonlinear Control*, John Wiley and Sons: New York, 1976.

[5] H. Ohno, E. Nakanishi and T. Takamatsu, "Optimal Control of a Semi-Batch Fermentation", Biotechnol. Bioeng., **18**, pp. 847-864, 1976.

[6] S. J. Parelukar, J. M. Modak and H. C. Lim, "Optimal Control of Fed-Batch Bioreactors", Proc. of the Amer. Control Conf., p. 849, Boston, MA, 1985.