



Continuous optimization using a dynamic simplex method

Qiang Xiong, Arthur Jutan*

Department of Chemical and Biochemical Engineering, The University of Western Ontario, London, Ont., Canada N6A 5B9

Received 20 August 2001; received in revised form 27 November 2001; accepted 11 February 2003

Abstract

Most chemical processes are operated under continuously changing conditions and thus the optimal operating conditions change with time. On-line optimization techniques of various types which are used to track these kinds of moving optimum, have sparked significant interest in recent years. However, most of these strategies deal only with static optimum or optimum that move so slowly that they can be considered static.

The model-based optimization algorithms, such as SQP, use first-principle models to optimize the process in a sequential steady-state mode, and hence rely on accurate process models. But for many circumstances, a process model is either too complex or expensive to obtain or changes quickly with time. This can limit the model-based approach and points to the 'direct search' optimization methods, since for these methods no explicit model is required. Moreover, direct search methods need very few new measurements to calculate a new movement toward the optimum. However, little work has been done on optimizing dynamic systems with moving optima using direct search methods. In this work, the traditional Nelder–Mead simplex method is modified and extended to allow tracking of moving optima, which results in a so-called dynamic simplex algorithm. Various simulation examples demonstrate the capability and flexibility of this new direct search algorithm in tracking moving optima in multiple dimensions.

© 2003 Elsevier Ltd. All rights reserved.

Keywords: Optimization; Moving optimum; Dynamic simplex method

1. Introduction

Most chemical processes are operated under continuously changing conditions due to two main reasons. One is the physical drifting of the process itself, such as fouling of a heat exchanger, deactivation of the catalyst in a reactor, temperature and flow-rate fluctuation of the feed, etc. A second reason is a change in market demands and economic conditions, which may result in change of product specifications and plant schedules. In these circumstances real-time (or on-line) optimization (RTO) techniques are required to account for these changes and drive the process continuously toward its optimal operating point. RTO has enjoyed significant industrial interest in the past several decades because of its capability of boosting the profitability of plants (Cutler & Perry, 1983; Darby & White, 1988; Marlin & Hrymak, 1997). Selection of the correct set of operating conditions by an on-line optimization is thought to be worth about 3–5% of the economic value of a process (Cutler & Perry,

1983). Usually, two phases of optimization are required to increase the profit of a chemical plant during its lifetime, one is the optimal design of the plant and the other is optimal operation of the plant. The optimal design is a one-time problem but the optimal operation remains as a continuous challenge throughout the lifetime of the plant. Furthermore, considering the many existing chemical plants, a few percent marginal improvement can lead to huge profit increases.

The various dynamic on-line optimization techniques can be grouped into the two categories of model-based approaches and direct search approaches, respectively (Garcia & Morari, 1981).

A model-based real-time optimization procedure consists of the following four steps (Darby & White, 1988) as shown in Fig. 1:

1. *Data validation:* In this step the input and output data are validated using data reconciliation techniques to remove the gross errors, e.g., significant mass or heat unbalance.
2. *Process model updating:* Recursive least-squares algorithm is used to update the process model when new input and output data are available.

* Corresponding author. Tel.: +1-519-661-2111x88322; fax: +1-519-661-3498.

E-mail address: ajutan@uwo.ca (A. Jutan).

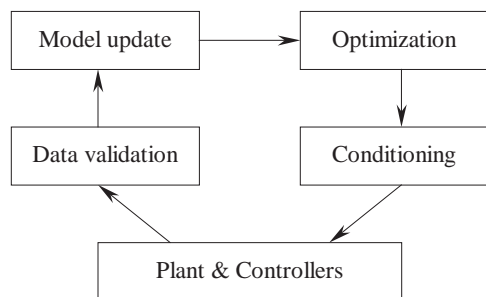


Fig. 1. Typical procedure of the model-based optimization.

3. *Model-based optimization*: An optimization problem based on an adaptive model is set up and solved to obtain the optimal operating point.
4. *Optimizer command conditioning*: Post-optimality analysis is carried out and the validated optimal set-point operating condition is passed to local controllers to implement the calculated changes.

There are a set of decisions associated with each step which have a significant effect on the eventual success of the overall real-time optimization system. Some of the more important decisions include measurement selection, partitioning model parameters into those that will be updated on-line and those that will be fixed, model structure, model updating method and post-optimality analysis.

In this paper, we focus our attention on the second and third step since they form the core of a real-time optimization strategy. The process model used in the model-based approach is usually a first-principle model. An objective function is chosen and a nonlinear optimization problem is constructed and solved using one of several methods. The most popular of these is the sequential quadratic programming (SQP) method. The success of this approach is highly dependent on the availability of a good process model. This method optimizes the process from one steady state to the next. Some chemical processes are too difficult and/or too expensive to model and even if a model is available it may be difficult to update due to its complexity and nonlinearity.

An alternate approach to the development of a sophisticated first-principle model, is to use an empirical black-box model. For instance, in response surface methodology (RSM) (Box, 1954), which has been used successfully in practice, an empirical response surface is fitted which describes the characteristics of the process near a certain operating point. A move to the optimum is carried out using this fitted surface. The RSM technique is effective when the process optimum is static, and difficulties arise when applying RSM to dynamic processes with a moving optimum. Edwards and Jutan (1997) have tried to extend the traditional RSM to a so-called dynamic response surface methodology (DRSM) to allow tracking of moving optimum. Some useful results were obtained but several

problems remained. The algorithm is extremely sensitive to the choice of parameters and suffered large excursions from the true optimum. The difficulties in DRSM may be partially due to its trying to estimate a quadratic surface when the process moves close to the true optimum where the surface becomes somewhat flat. The tracking algorithm tends to lose its direction under these conditions. Another more generic problem with RSM based methods is the “curse of dimensionality”: A large number of process measurements are required to estimate empirical constants required for multi-input models. The simplex method (Nelder & Mead, 1965) is an exception here, in that only one new measurement point per dimension is required to form a new simplex. This is a very important consideration for real-time optimization applications. In an earlier effort to use empirical models, Bamberger and Isermann (1978) presented an adaptive optimization technique to solve on-line optimization problems for processes with slow dynamics. In their approach, a Hammerstein model is used to model the dynamic process and the model parameters are updated online using recursive least-squares (RLS) method. Optimization is carried out using the predicted steady state based on the Hammerstein model. The main advantage of this approach is that there is no need to wait for the process to reach steady state at each iteration, thus significantly reducing the time for optimization. This algorithm is also attractive because it uses empirical models instead of first-principle models and thus can be applied under situations when first-principle models are hard to obtain, as is true for complex chemical and biochemical processes. Several applications have been reported in the literature on different chemical processes, such as CSTR (Garcia & Morari, 1981; McFarlane & Bacon, 1989), FCC (McFarlane & Bacon, 1989) and fixed bed reactor (Lee & Lee, 1985). Hammer and Richenberg (1988) argued that adaptive optimization using recursively identified dynamic models is a feasible approach but convergence rates in practice often fall below those predicted by simulation study. They studied the influence of excitation signals for identification and the frequency at which the optimization is performed on the overall performance of the adaptive optimization. Several improvements were used to speed up convergence, reject noise and tolerate highly nonlinear processes. Difficulties still remain in modelling the processes because of the high nonlinearities within the wide operating range during optimization. Golden and Ydstie (1989) proposed a hybrid approach to incorporate a priori process models to encounter the modeling difficulties.

For either first-principle or empirical model-based optimization algorithms, process modeling remains to be a difficult task. Thus we turn our attention to direct search methods, which requires no process model in order to perform the optimization. The most notable direct search method is the Nelder–Mead simplex method (Nelder & Mead, 1965), which was proposed in 1965 and has found wide use in optimization practice. This popular algorithm (which requires far fewer measurements than RSM-based

approaches) has never been extended to dynamic systems with moving optimum. The extension of this method to dynamic processes is the main motivation of this paper.

This new approach is closest to the EVOP method (Box, 1954) (which is nevertheless a static method), but does not suffer from the large number of experimental measurements required to produce a movement to the next step. This is very important in cases where the measurements are expensive to make or the process moves continuously. RTO, as practiced in industry, is really suited to processes that are expected to settle and have a steady state. Even if the optimization is updated on a greater frequency, the process is still considered to move from one steady state to the next. The processes we consider here, are more of a challenge in that they are continuously evolving and the algorithm is required to follow this. However, if the process does settle our algorithm will also settle. There is also the class of open-loop optimization methods based on the maximum principle or on dynamic programming, which represent quite a different problem to the one considered here and is more directed to batch processes. This last class of problems usually have integral objectives, whereas ours is restricted to algebraic objectives.

The paper is organized as follows: Section 2 gives the description of the problem. In Section 3 the direct search method is introduced and a novel dynamic simplex algorithm is developed by extension from the traditional Nelder–Mead simplex method. Section 4 demonstrates the promising performance of the dynamic simplex method using several examples and it is compared with the DRSM approach. The conclusion as well as some thoughts for future study are given in the last section.

2. Problem formulation

A general nonlinear time-varying system and its objective function can be described by

$$y = f(x, \theta_t, t) + \phi, \quad (1)$$

$$J = g(y, x), \quad (2)$$

where f and g are a nonlinear functions, x is the process inputs, θ_t is the time-varying parameter vector; y is the process output, ϕ is the noise, and J is the objective function to be minimized. This process has a moving optimum due to the time-varying parameters θ_t and t . f may also represent the solution to a differential equation. In simulation mode, the new value of y is calculated by evaluating the function f . In a real-time model, y is obtained as a measurement. Function value and measurement are used interchangeably in this paper. Each measurement of y takes some finite time to complete and has a certain measurement cost associated with it, which may be expensive. Thus, the problem we are faced with is to optimize the function y using as few function evaluations (measurements) as possible.

One example of this kind of process would be a catalytic plug flow reactor, where the catalyst is decaying with time causing the production rate to fall. Other variables, such as reactor inlet flow may also be changing with time due to upstream conditions. This production rate can be corrected by adjusting, say, the reactor wall temperature to compensate. A model of the reaction network may not be available and it may be costly to obtain process measurements which may also be of poor quality. Our algorithm would seek to measure the production and then continuously adjust the wall temperature to compensate for this falling production.

3. Direct search method

The name *direct search* refers to the fact that the search of the optimum is carried out directly without the help of a process model. This further implies that the direct approach needs less measurement data since data that are required to build and update process models become unnecessary. Thus the direct search method is recommended when the process has one or more of the following properties (Wright, 1995):

1. Process model is difficult or expensive to obtain.
2. Process exhibits discontinuities.
3. Measurement of the process data is expensive or time-consuming.
4. The measurement data are contaminated by significant noise.

3.1. Nelder–Mead simplex method

After it was first proposed in 1965, the Nelder–Mead simplex method (Nelder & Mead, 1965) has been extensively used in practice because of its simplicity and efficiency.

A simplex S is defined as a convex hull with $N + 1$ vertices $\{x_j\}_{j=1}^{N+1}$ in an N -dimensional space \mathcal{R}^N . These vertices satisfy the nondegeneracy condition that the volume of the simplex hull is nonzero.

The Nelder–Mead simplex method is an iterative algorithm starting from an initial simplex. Iteration k begins by ordering and labeling the current set of vertices as $\{x_j^{(k)}\}_{j=1}^{N+1}$ such that

$$g_1^{(k)} \geq g_2^{(k)} \geq \dots \geq g_{N+1}^{(k)}, \quad (3)$$

where $g_j^{(k)} = g(x_j^{(k)})$ is the objective function value. Because we are going to minimize the objective function g we refer to $x_{N+1}^{(k)}$ as the *best* point, $x_2^{(k)}$ the *next to the worst* point and $x_1^{(k)}$ the *worst* point. In the following steps, the worst point is discarded and several ‘better’ trial points are generated and function values are evaluated at these points. A new simplex with $N + 1$ vertices is then constructed using rules which favor the minimization of the function value.

A Nelder–Mead simplex iteration needs only one function evaluation when the iteration terminates after reflection,

or two function evaluations when termination occurs after expansion or contraction, and $N + 2$ function evaluations if a shrinkage step is taken. Hence the Nelder–Mead algorithm is particularly parsimonious in function evaluations (measurements in real-time applications) on a per iteration basis compared to other direct search methods (see, for example, evolutionary operation or other RSM-based methods (Box & Draper, 1969)) and certainly for model-based methods.

The Nelder–Mead simplex algorithm has been used extensively in many fields, especially in chemistry and chemical engineering. The reference *sequential simplex optimization* (Walters, Parker, Morgan, & Deming, 1991), contains a chronological bibliography showing the rapid growth of the number of applications using the Nelder–Mead algorithm.

Though widely used in practice, there is no firm theoretical basis of the Nelder–Mead simplex method until recently. Torczon (1997) gives sufficient conditions for convergence for problems in n -dimensions using fixed simplex. Proof of convergence for dimension 1 and various limited convergence results for dimension 2 using Nelder–Mead simplex method is given in Lagarias, Reeds, Wright, and Wright (1998). In addition to weak theoretical basis, there are some concerns about its reliability and expandability (Wright, 1995). In spite of this, it has been used in an extraordinary wide variety of contexts, especially in chemistry, chemical engineering and medicine (Wright, 1995).

3.2. Dynamic simplex method

Applications of the Nelder–Mead simplex method are limited to problems with a static optimum. Also the Nelder–Mead simplex algorithm has a well-known problem, in that it locks itself up when the measurement is contaminated by noise or the process optimum moves with time (see the example below). Kuznetsov and Rykov (1990) studied the application of the simplex method proposed by Rykov (1983) to a problem with a linear drifting optimum and the relationship between the drifting speed of the optimum and simplex size was obtained. One major problem with Kuznetsov and Rykov's (1990) algorithm is that the number of function evaluations required in each iteration soars as the dimension of the problem increases. It searches all the possible directions of single-point and multiple-point reflections and chooses the direction with the greatest improvement of the function value. This makes it unrealistic for use in real-time optimization where each function evaluation (measurement) is associated with a high cost.

Our objective is to develop a simplex-based algorithm that can treat a real-time optimization problem which has a moving optimum and noise contaminated measurement data. Nelder–Mead simplex method forms the basis of our approach because of its parsimony in function evaluations and algorithmic simplicity.

3.2.1. Strategies for developing a dynamic simplex method

3.2.1.1. Re-measure the best point. When the standard Nelder–Mead simplex method is used directly to optimize a function with a moving optimum, it tends to lock itself until there is no tracking at all. When dealing with dynamic problems with moving optima, we can never sample all points of the simplex instantaneously. We assume that the sample time is chosen so that it is much smaller than the dominant time constant of the process or the time constant of disturbances. But since the process keeps moving continuously, the measurements at a certain point may be valid for only a certain period of time and an older measurement can mislead the tracking algorithm. In the standard simplex algorithm, the best point in one iteration will always be kept in the next iteration since the objective surface is fixed. For a moving surface, however, all data may become invalid after a certain period of time. However, the best point in the current iteration will have the greatest influence on the search direction. It is thus advantageous to re-measure that best point at the end of each iteration and recalculate a new direction to allow better tracking for a moving optimum. Of course, it would be even better to re-measure all the points, but this would eliminate one of the main advantages of the simplex method: Its parsimonious requirement for few measurements. This way a compromise is reached.

3.2.1.2. Fixed simplex size vs. variable size simplex. No generality will be lost if we consider the initial simplex a regular simplex since we can scale the variables to regularize it.

We require the simplex to expand in the full dimensional factor space due to the changing nature of the process and noise. If a variable size simplex is used, the simplex may become nearly degenerate and remain in a hyper-plane of the full factor space. If the process optimum moves in the direction perpendicular to that hyper-plane, the simplex cannot adjust itself quickly enough to track an optimum that is moving away. In addition to this sluggish movement in a hyper-plane, the variable simplex method can also cause aggressive movement if some edges of the simplex become very large after several expansion operations. Based on these observations, our dynamic simplex algorithm uses a fixed size simplex and no operations other than reflection are used. Fixing the size of the simplex represents a compromise in that the accuracy of the optimal estimate is reduced. However since the optimum is continuously changing, we are more concerned with tracking this change. Also, jacking software could easily be added to detect a stationary condition and adjust the simplex size down, so that the algorithm can home in on the optimum if it stops moving.

Another reason we want to keep the simplex regular and fixed during the optimization procedure is the presence of noise. Since the reflection direction is a choice based on the difference of measurements at the vertices of the simplex,

it is necessary that the difference of measurements between vertices is larger than the noise level to make the direction change meaningful. Thus, the size of the simplex must be large enough to extract the useful information in the presence of the noise.

3.2.2. Dynamic simplex—the algorithm

Based on the Nelder–Mead simplex method and the above considerations, the dynamic simplex method is developed as an iterative algorithm with iteration k consisting of the following steps:

1. The start simplex in iteration k is $S_0 = \{x_j\}_{j=1}^{N+1}$ and the objective function values on these vertices are $\{g_j\}_{j=1}^{N+1}$.
2. Sort the vertices of the simplex such that

$$g_1 \geq g_2 \geq \dots \geq g_{N+1}. \quad (4)$$

3. Successive reflection.

A single point reflection of the first (*worst*) point of the simplex S_0 , i.e., x_1 , is operated

$$x_{N+2} = \frac{2}{N} \sum_{j=2}^{N+1} x_j - x_1 \quad (5)$$

and evaluate $g_{N+2} = g(x_{N+2})$. Thus a new simplex $S_1 = \{x_j\}_{j=2}^{N+2}$ is formed. Reflect the first point of this new simplex S_1 , i.e., x_2 ,

$$x_{N+3} = \frac{2}{N} \sum_{j=3}^{N+2} x_j - x_2 \quad (6)$$

and obtain the evaluation $g_{N+3} = g(x_{N+3})$. Thus we have the second new simplex $S_2 = \{x_j\}_{j=3}^{N+3}$. Keep reflecting the first point of the newly formed simplex and after the reflection is repeated M times we have $S_M = \{x_j\}_{j=M+1}^{N+M+1}$ and a series of new simplices $\{S_p\}_{p=1}^M$.

4. Choose the start simplex for iteration $k + 1$.

Calculate the average function value of these simplices $\{S_p\}_{p=1}^M$

$$\bar{g}_{S_p} = \frac{1}{N+1} \sum_{j=p+1}^{N+p+1} g_j, \quad p = 1, 2, \dots, M. \quad (7)$$

Select the simplex S_q that satisfies

$$\bar{g}_{S_q} = \min\{\bar{g}_{S_p}\}_{p=1}^M. \quad (8)$$

5. Remeasure the response at point x_{N+1} .

6. Set as S_q the new start simplex for the next iteration.

Unlike the traditional Nelder–Mead simplex method and other optimization algorithms, there is no step to test for convergence. For our case, where we have to deal with a moving optimum, compromise has to be made on the convergence of the algorithm in order to ensure that the step size remains large enough to follow a continuously moving objective. The noise estimates are useful for adjusting

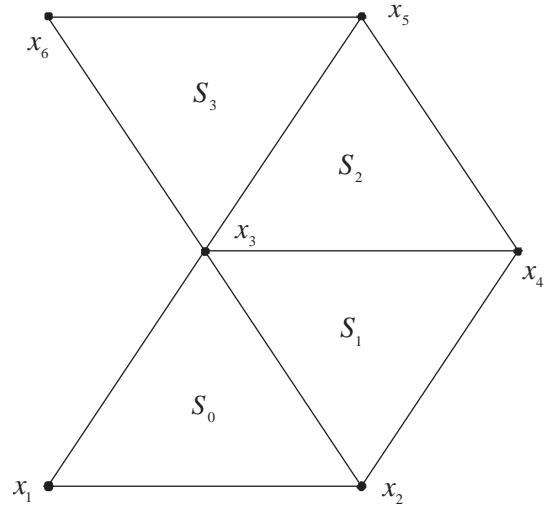


Fig. 2. Successive reflection in two-dimensional space: $x_1 \rightarrow x_4$, $x_2 \rightarrow x_5$, $x_3 \rightarrow x_6$.

convergence in the static case, but we found that the more important difficulty was keeping up with the moving surface. The simplest approach is to ensure a minimum simplex size, but no doubt more sophisticated heuristics are possible here.

A demonstration of the reflection scheme in a two-dimensional case is shown in Fig. 2. The worst point x_1 of the start simplex S_0 is reflected to x_4 and a new simplex S_1 is formed. Then x_5 and x_6 are obtained by reflecting x_2 and x_3 , respectively.

The objective of successive reflection in step 3 is to search a portion of the space to find the best direction in which to move. The size of the direction search space depends on the value of M . In the Nelder–Mead simplex method and in the multidirectional search method (Dennis & Torczon, 1991), there is only one possible search direction.

In step 4, the choice of the new start simplex in the next iteration, is determined only by selecting the minimum average function value of newly generated simplices, irrespective of its average compared to the start simplex S_0 . This is different from the way we choose the next simplex in the Nelder–Mead algorithm, because here our process moves continuously and we cannot always expect a decrease of the average function value each time. What we are trying to track is a moving optimum, which sometimes may have an increased function value (see examples below).

Step 5 implements the idea of remeasuring the old ‘best’ point, which may no longer qualify as the best point for a moving process. Of course this is true, to some extent, for all the measurements, but the oldest point is the most outdated and has the greatest need for an update measurement.

3.2.3. Choice of the parameters in dynamic simplex algorithm

There are two parameters that effect the performance of the dynamic simplex method. They are the simplex size s

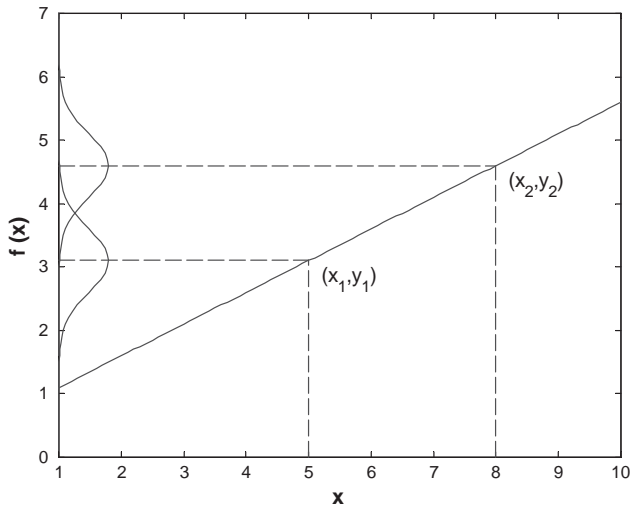


Fig. 3. Simplex size must be large enough to allow differentiation in the presence of measurement noise.

and the number of successive reflections M , in each iteration. A small value of M will suffice in most cases. In the examples presented below, M is chosen as 2. The step size should be large enough to track the move speed of the optimum and also avoid a poor directional choice caused by high noise to measurement ratio. A conflicting requirement is that s should not be too large so that the algorithm can still achieve adequate accuracy in converging to the optimum.

To track the moving optimum, the simplex size s depends on the number of successive reflections M , the measurement sampling time T and the maximum movement speed of the optimum, v . The number of measurements in each iteration is the number of reflections plus one, which includes the re-measurement of the last best point. Thus the minimum step size s_{\min} required to catch the moving optimum is given by

$$s_{\min} > (M + 1)vT \quad (9)$$

which is exactly how far the optimum moves during one iteration. It must be noted that this minimum step size of the simplex cannot guarantee the success of the dynamic simplex method, since the reflection direction is not continuous in the search space and the simplex will only move in the approximate direction as the true optimum moves. Thus a step size larger than s_{\min} is needed to ensure the success of the algorithm.

Another aspect we should consider when choosing the simplex size is the noise level. Simplex vertices must be sufficiently separated to ensure a significant measured difference between vertices. Fig. 3 shows a function $f(x)$ and two measurements y_1 and y_2 , taken at x_1 and x_2 , respectively. The measurements are contaminated by Gaussian noise as shown by the two distribution curves. To distinguish the relative height of these two measurements, y_1 and

y_2 must lie far away enough from each other. This in turn requires there must be enough distance between x_1 and x_2 , which is the simplex size. The distance required to tell the two distribution curves apart depends on the variance of the noise, i.e. the width of the distribution. The larger the variance, the wider the distribution, and the larger the distance required to distinguish between the two measurements y_1 and y_2 .

4. Examples

To test the newly developed dynamic simplex method, several simulation examples are studied. First a few mathematical functions with either linear or nonlinear drifting optimum are examined. Finally the simulated Williams–Otto CSTR is considered as an RTO example.

4.1. 2-D tracking

A two-variable problem with a linear drifting optimum is chosen as the first example. Similar examples were tested by Kuznetsov and Rykov (1990) using their simplex-based algorithm and by Edwards and Jutan (1997) using the dynamic response surface method. The objective function is

$$J(t) = (x_1 - 10 - \alpha t)^2 + (x_2 - 10 - \beta t)^2 + 5 + e(t), \quad (10)$$

where x_1 and x_2 are two input variables, $J(t)$ is the function value at time t , α and β are the two parameters which jointly specify a linear drifting speed and direction of the true optimum of this function; e is white noise with standard deviation σ and mean zero. In this example, the following parameters are used:

$$\alpha = 0.1, \quad \beta = 0.05, \quad \sigma = 0.1.$$

This function has a time-varying optimum $(10 + 0.1t, 10 + 0.05t)$, which moves along a straight line and the optimal function value is 5. The start point is chosen at $x_0 = (13, 15)$. The minimum step size required to catch the moving optimum is $s_{\min} = \sqrt{\alpha^2 + \beta^2}(M + 1)T$. By choosing $M = 2$ and the sampling time $T = 1$, the step minimum step size is $s_{\min} = 0.236$ and the actual one used here is $s = 0.5$. First the Nelder–Mead simplex algorithm is applied and it can be seen from the result shown in Fig. 4 that the standard Nelder–Mead algorithm cannot handle this kind of problem. It locks itself up after it reaches the optimum and loses tracking ability. The regular Nelder–Mead locks up since it assumes as soon as it has a successful move that it is climbing up the objective function in the right direction. It then contracts its size to reflect this success. However, the regular Nelder–Mead does not take into account the fact that the surface has moved and that now, at the new time, it is actually further away from the optimum and that a contraction was not a good strategy. This eventually leads to a complete lock up of the algorithm as the surface moves away completely.

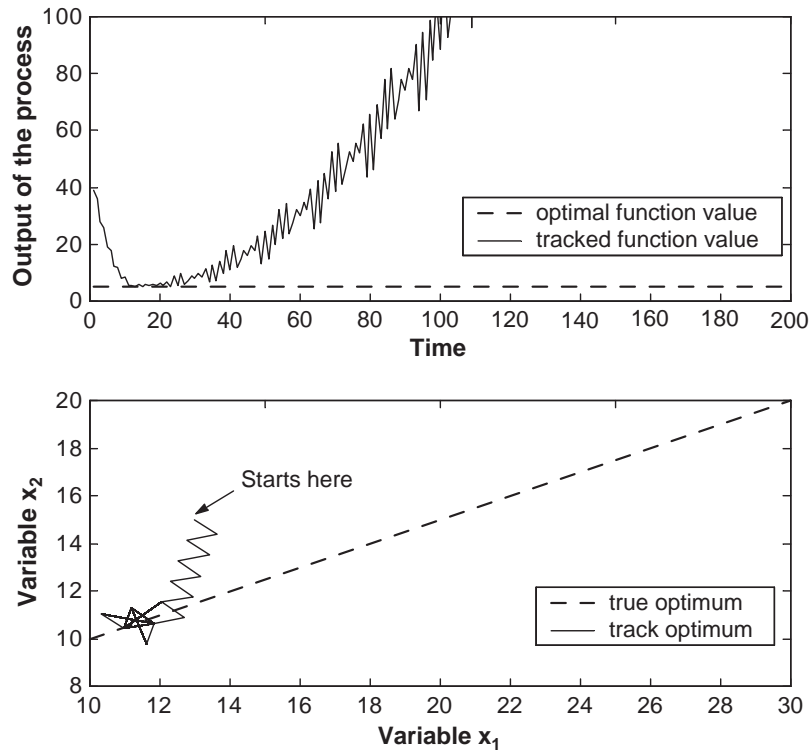


Fig. 4. Tracking 2-D linear drifting optimum using Nelder–Mead simplex algorithm.

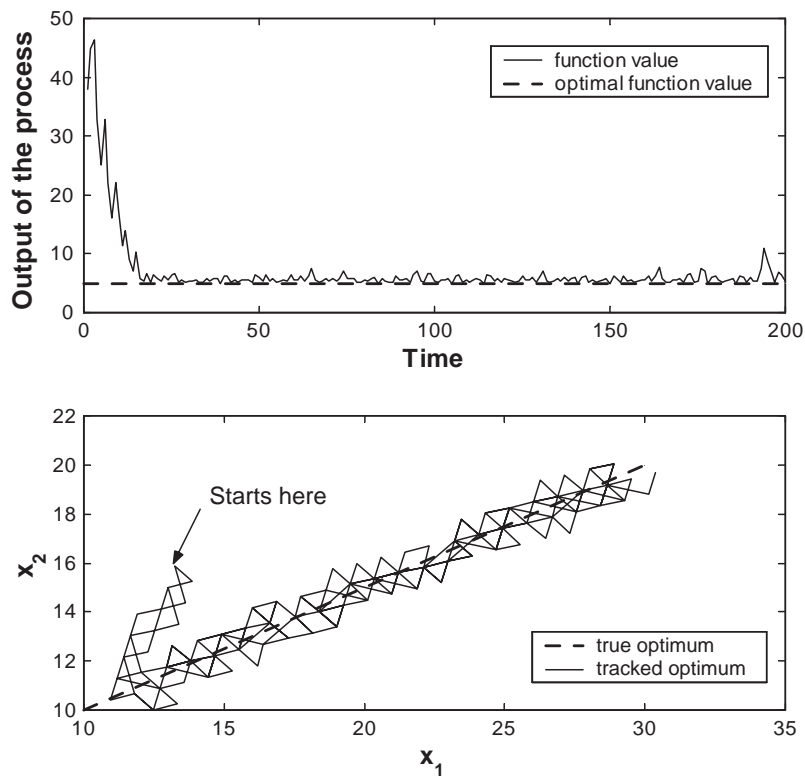


Fig. 5. 2-D tracking of a linear drifting optimum using dynamic simplex algorithm.

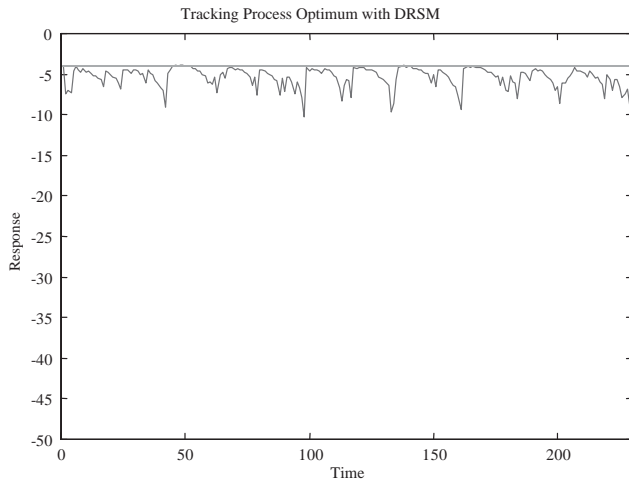


Fig. 6. Tracking result of a 2-D linear drifting optimum using DRSM.

Using the dynamic simplex method, we have the result shown in Fig. 5 which clearly shows how the dynamic simplex algorithm first catches the optimum and follows its movement continuously.

Edwards and Jutan (1997) applied the DRSM algorithm to a problem similar to that in (10) which tracks a process given by

$$J(t) = -(x_1 - 10)^2 - (x_2 - 10 - 0.1t)^2 - 4 \quad (11)$$

which has a maximum function value of -4 with the result shown in Fig. 6. The DRSM is able to track the moving optimum, but the performance is extremely sensitive to two DRSM parameters μ and λ which are chosen very carefully (for details, see Edwards & Jutan, 1997); even a slight variation can lead to significant deterioration in the performance. Also, importantly, the number of measurements required at each iteration is significantly higher for RSM. In particular, for a 2-D system, the RSM would require a minimum of 4 measurements compared to 2 for the dynamic simplex.

The dynamic simplex algorithm can track a more complex moving process which has a varying optimal output. The following objective function, has an optimum point which drifts linearly, but the optimum function value changes with time in a sinusoidal manner:

$$J(t) = (x_1 - 10 - \alpha t)^2 + (x_2 - 10 - \beta t)^2 + 5 + A \sin(\omega t) + e(t), \quad (12)$$

where $A = 5$, $\omega = 0.05$ and the other parameters remains the same. The tracking result for the dynamic simplex method is shown in Fig. 7. Again good tracking is achieved.

When choosing the new base simplex for the next iteration in the dynamic simplex algorithm, the average function value of the newly generated simplex is calculated and compared at each iteration. However with the Nelder–Mead

method only the new function value of the reflected point is compared with the previous simplex and a decrease in the function value is required in each iteration until final convergence. In treating a static problem, a decrease in function value in each iteration is required to allow the algorithm to converge. However for a dynamic problem, which has a moving optimum and a varying optimal function value, the objective is to track the optimum as closely as possible since there is no static point to converge to.

4.2. 3-D tracking

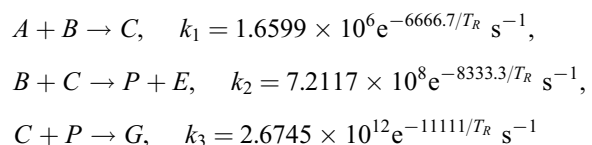
The dynamic simplex algorithm is now applied to a three-dimensional function given by

$$J(t) = (x_1 - 10 - r \sin \omega t)^2 + (x_2 - 10 - r \cos \omega t)^2 + (x_3 - \alpha t)^2 + 5 + e(t), \quad (13)$$

where $r = 5$, $\omega = 0.02$, $\alpha = 0.02$ and $\sigma = 0.1$. Again the sampling time T is 1. The starting point is chosen at $x_0 = (11, 11, 3)$ and the step size $s_1 = s_2 = s_3 = 1.2$. The tracked function value is shown in Fig. 8 and the tracked optimum is shown in Fig. 8. This example demonstrates the ability of the dynamic simplex to track three-dimensional functions. Some drop in tracking skill is observed and this is to be expected as the optimal algorithm must work significantly harder as it tries to track in multiple dimensions dynamically. Nevertheless, it is clearly able to follow the spiral path of the objective as it moves around in three-dimensions in the presence of noise.

4.3. Williams-Otto CSTR

The diagram of the modified (by Roberts (1979)) Williams–Otto CSTR (Williams & Otto, 1960) process is shown in Figs. 9 and 10. It is an ideal CSTR with a three generic reaction sequence



and the following dynamic mass balances represent the true plant behavior:

$$\begin{aligned} W \frac{dX_A}{dt} &= F_A - (F_A + F_B)X_A - k_1 X_A X_B W, \\ W \frac{dX_B}{dt} &= F_B - (F_A + F_B)X_B - (k_1 X_A X_B + k_2 X_B X_C) W, \\ W \frac{dX_C}{dt} &= -(F_A + F_B)X_C \\ &\quad + (2k_1 X_A X_B - 2k_2 X_B X_C - k_3 X_C X_P) W, \\ W \frac{dX_E}{dt} &= -(F_A + F_B)X_E + 2k_2 X_B X_C W, \end{aligned}$$

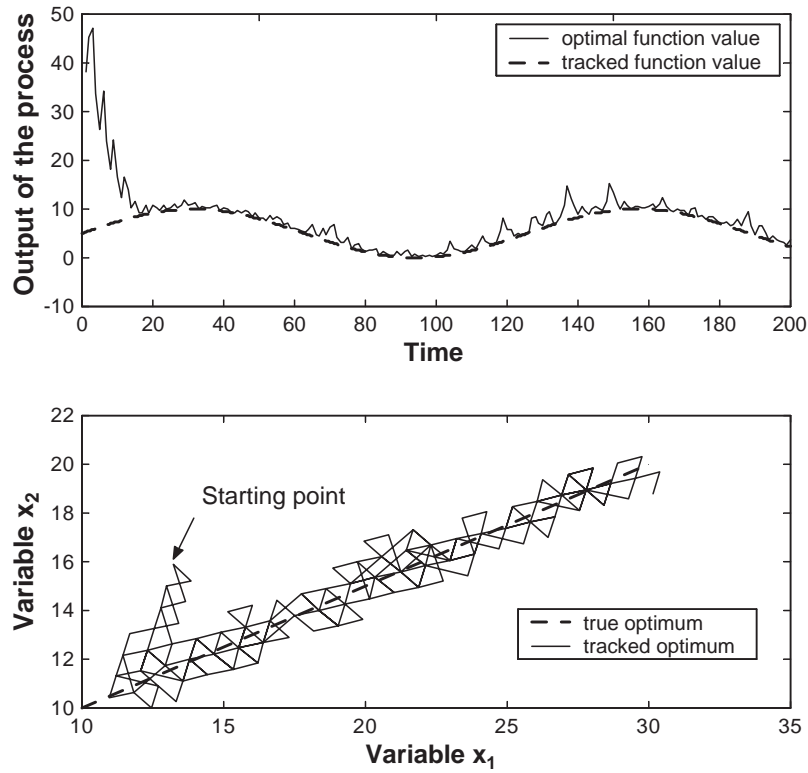


Fig. 7. Tracking a 2-D function which has a linear drifting optimum as well as a sinusoidal optimum value.

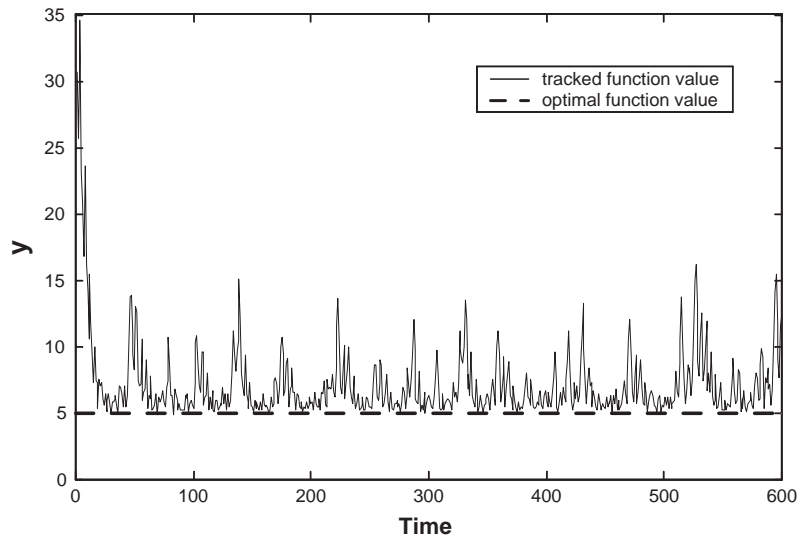


Fig. 8. Tracking result of the 3-D function.

$$\begin{aligned}
 W \frac{dX_G}{dt} &= -(F_A + F_B)X_G + 1.5k_3X_CX_PW, \\
 W \frac{dX_P}{dt} &= -(F_A + F_B)X_P \\
 &\quad + (k_2X_BX_C - 0.5k_3X_CX_P)W,
 \end{aligned}
 \tag{14}$$

where F_A and F_B are the mass flowrates of the raw materials A and B , W is the mass content inside the reactor, X_A , X_B , X_C , X_E , X_G and X_P are the mass fractions of the six chemical components A , B , C , E , G , and P , respectively. The objective for optimization is to maximize the profit which is the difference between the sales of the products E and P

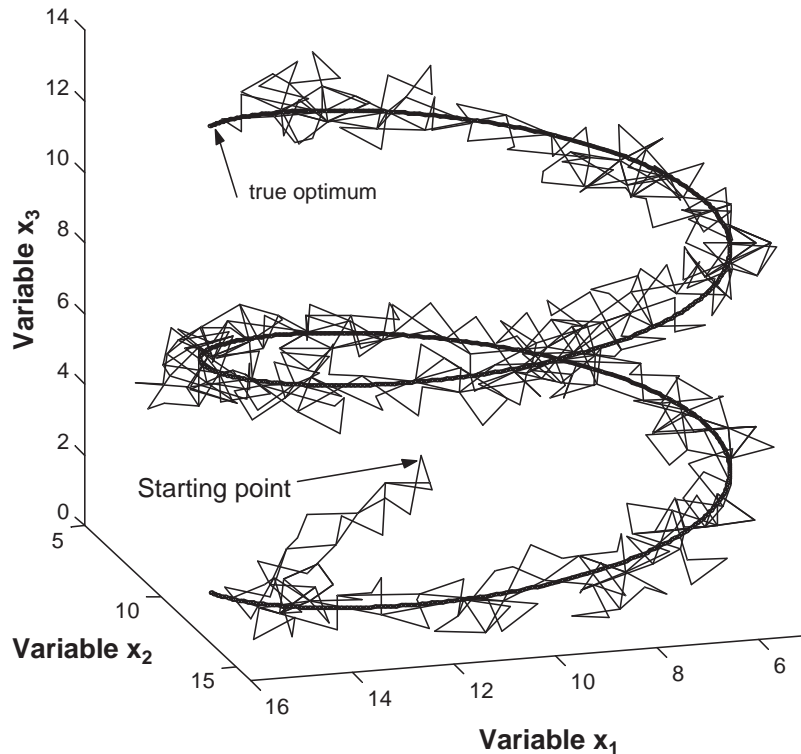


Fig. 9. Tracking optimum of a 3-D function.

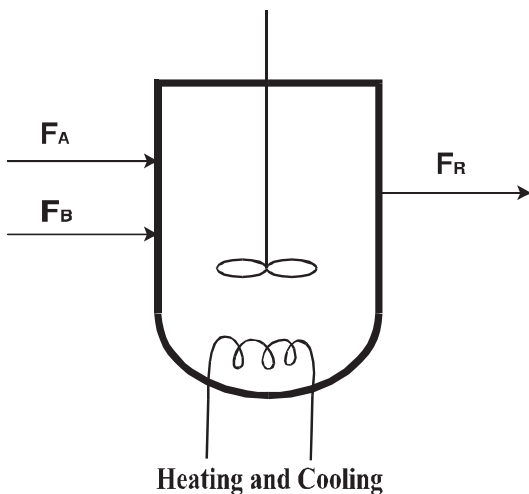


Fig. 10. Williams–Otto CSTR.

and the costs of raw materials A and B

$$J = 5554.1(F_A + F_B)X_P + 125.91(F_A + F_B)X_E - 370.3F_A - 555.42F_B. \quad (15)$$

During simulation, the mass content of the reactor is fixed at $W = 2104.7$ kg and the fluctuation of the flowrate of the reactant A is considered as the disturbance which gives the process a time-varying optimum. The two manipulated

variables for optimization are F_B , the flowrate of reactant B and T_R , reactor temperature, which are denoted by $x = (F_B, T_R)$. The variables to be measured on-line are

$$\bar{x} = (F_B, T_R, X_A, X_B, X_C, X_E, X_G, X_P, F_R) \quad (16)$$

with the following covariance matrix of measurement noise:

$$U = \text{diag}([0.004 \quad 0.05 \quad 0.0026 \quad 0.0117 \quad 0.0005 \quad 0.002 \quad 0.0033 \quad 0.0012 \quad 0.004]). \quad (17)$$

The profit has a single maximum point at fixed flowrate of reactant A , e.g., Fig. 11 shows the surface as well as the contour plot of the profit when $F_A = 1.8275$ kg/s, where the optimum lies at $x^* = (4.784$ kg/s, $89.65^\circ\text{C})$.

To simulate a moving optimum, we let F_A change with time. Fig. 12 shows the trace of the true optimum when the feed flowrate of component A changes from 1.2 to 2.4 kg/s. If we let F_A follow a profile as shown in Fig. 13(a), the resulting optimized profit and tracked optimum using dynamic simplex method are shown in Figs. 13(b) and (c). The parameters used in the dynamic simplex are chosen as: start point $x_0 = (F_{B0}, T_{R0}) = (3.5$ kg/s, $90^\circ\text{C})$, simplex size $s = (\Delta F_B, \Delta T_R) = (0.1$ kg/s, $1^\circ\text{C})$, sampling time $T_s = 2000$ s and number of successive reflections $M = 2$. The sampling time T_s is chosen to be large enough to let the process approach its steady state. Here it is two times the time constant of the process $\tau = 1000$ s.

It can be seen from Fig. 13 that the dynamic simplex method tracks the optimum very well and the resulting profit

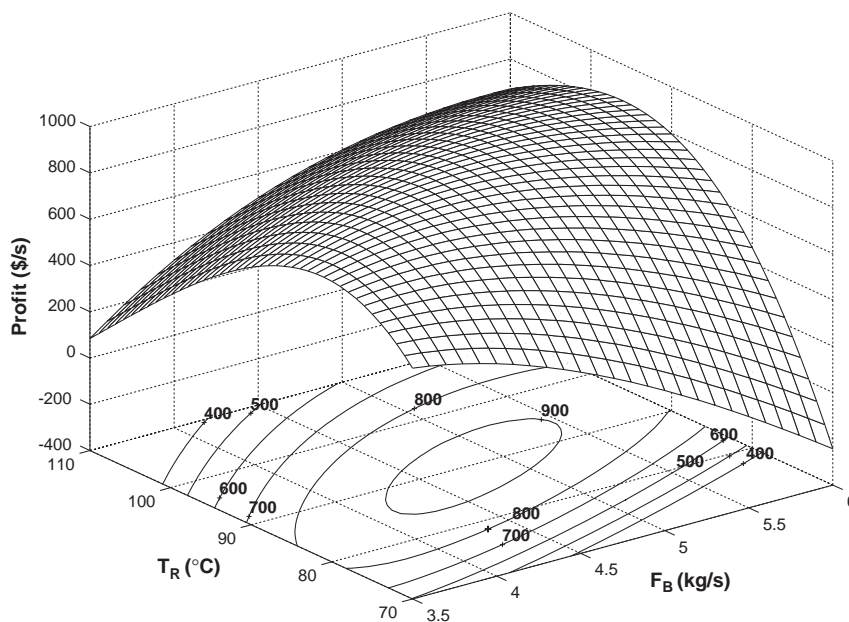
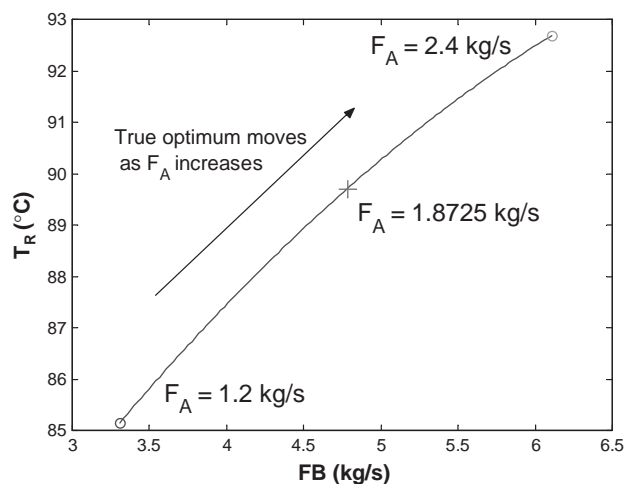


Fig. 11. Mesh and contour plots of the profit surface of the Williams–Otto CSTR.

Fig. 12. Trace of the optimum on time-varying inlet flowrate of component *A*.

follows the true optimal value closely. When the true optimum is moving between time 0.5×10^6 and 1.3×10^6 s, the optimizer tries to force the process towards its true optimum, but there is always a lag between the true profit and the optimized profit. When the process does not move so quickly the tracking is able to follow the optimum more closely.

5. Conclusions

Real-time optimization is required to instantaneously maximize the profit of plants which have time-varying operating conditions and product schedules. Direct search methods are good candidates for real-time optimization

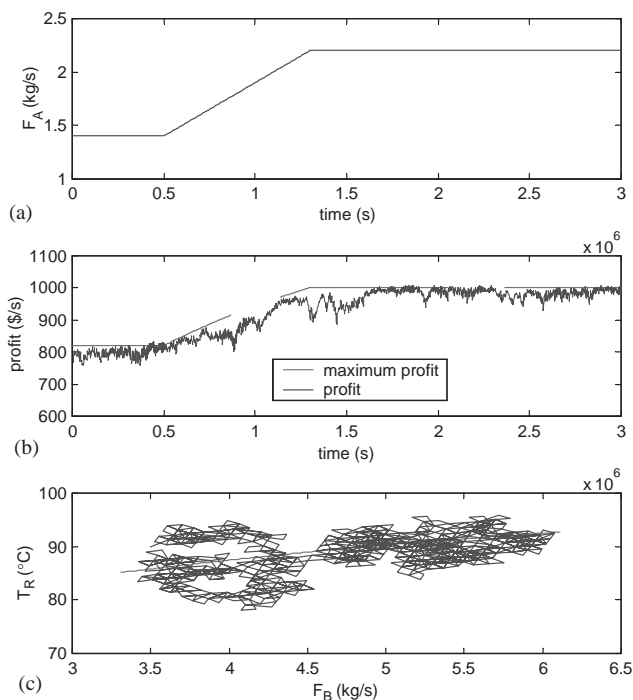


Fig. 13. Optimization result of the Williams–Otto CSTR using dynamic simplex method.

in situations where process models are difficult to obtain. Some of these, in particular the simplex method, are also very parsimonious in function evaluations (on-line measurements) and simple to implement. Here, the dynamic simplex algorithm is developed by extending the Nelder–Mead simplex method to allow tracking of moving optima.

A simple example on 2-D tracking problem was shown to exhibit much better performance and more robustness compared to that of the dynamic response surface method. Both 2-D and 3-D tracking examples with linear and nonlinear drifting optima were used to demonstrate the flexibility of the dynamic simplex method.

The simulation study on the Williams–Otto CSTR demonstrates the application of the dynamic simplex method to a practical chemical process. No model was required and only a few variables, which are directly related to profit calculation, need to be measured *once* at each iteration. Several future extensions (such as constraint handling and disturbance rejection) need to be considered, however the dynamic simplex method already shows great promise for continuous dynamic optimization in processes for which it is difficult or expensive to obtain a good model and where measurements are costly or time consuming.

References

- Bamberger, W., & Isermann, R. (1978). Adaptive on-line steady-state optimization of slow dynamic processes. *Automatica*, *14*(2), 223–230.
- Box, G. E. P. (1954). The exploration and exploitation of response surfaces: Some general considerations and examples. *Biometrics*, *10*, 16–60.
- Box, G. E. P., & Draper, N. R. (1969). *Evolutionary operation: A statistical method for process improvement*. New York: Wiley.
- Cutler, C. R., & Perry, R. T. (1983). Real time optimization with multivariable control is required to maximize profits. *Computers and Chemical Engineering*, *7*, 663–667.
- Darby, M. L., & White, D. C. (1988). On-line optimization of complex processes. *Chemical Engineering Progress*, *84*(8), 51–59.
- Dennis, J. E., & Torczon, V. (1991). Direct search methods on parallel machines. *SIAM Journal on Optimization*, *1*, 448–474.
- Edwards, I. M., & Jutan, A. (1997). Optimization and control using response surface methods. *Computers and Chemical Engineering*, *21*(4), 441–453.
- Garcia, C. E., & Morari, M. (1981). Optimal operation of integrated processing systems. *A.I.Ch.E. Journal*, *27*(6), 960.
- Golden, M. P., & Ydstie, B. E. (1989). Adaptive extremum control using approximate process models. *A.I.Ch.E. Journal*, *35*(7), 1157–1169.
- Hammer, J. W., & Richenberg, C. B. (1988). On-line optimizing control of a packed-bed immobilized-cell reactor. *A.I.Ch.E. Journal*, *34*(4), 626–632.
- Kuznetsov, A. G., & Rykov, A. S. (1990). Convergence of adaptive algorithms of minimization under a drift of the minimum of an objective function. Translated from *Avtomatika i Telemekhanika*, *9*, 92–100.
- Lagarias, J. C., Reeds, J. A., Wright, M. H., & Wright, P. E. (1998). Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal of Optimization*, *9*, 112–147.
- Lee, K. S., & Lee, W. -K. (1985). On-line optimizing control of a nonadiabatic fixed bed reactor. *A.I.Ch.E. Journal*, *31*(4), 667–675.
- Marlin, T. E., & Hrymak, A. N. (1997). Real-time optimization of continuous processes. *Fifth international conference on chemical process control. A.I.Ch.E. Symposium Series*, Vol. 316 (pp. 156–164).
- McFarlane, R. C., & Bacon, D. W. (1989). Adaptive optimizing control of multivariable constrained chemical processes. 1. Theoretical development; 2. Application studies. *Industrial and Engineering Chemistry Research*, *28*, 1828–1834.
- Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, *7*, 308–313.
- Roberts, P. D. (1979). An algorithm for steady-state optimization and parameter estimation. *International Journal of System Science*, *10*, 719–734.
- Rykov, A. (1983). Simplex algorithms for unconstrained optimization. *Problems of Control and Information Theory*, *12*, 195–208.
- Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, *7*, 1–25.
- Walters, F. H., Parker, L. R. Jr., Morgan, S. L., & Deming, S. N. (1991). *Sequential simplex optimization*. Boca Raton, FL: CRC Press.
- Williams, T. J., & Otto, R. E. (1960). A generalized chemical processing model for the investigation of computer control. *AIEE Transactions*, *79*, 458.
- Wright, M. H. (1995). Direct search methods: Once scorned, now respectable. In D. F. Griffiths, & G. A. Watson (Eds.), *Numerical analysis 1995 (Proceedings of the 1995 Dundee biennial conference in numerical analysis)* (pp. 191–208). Harlow, UK: Addison-Wesley, Longman.