Artificial Neural-Network-Assisted Stochastic Process Optimization Strategies

Somnath Nandi, Soumitra Ghosh, Sanjeev S. Tambe, and Bhaskar D. Kulkarni
Chemical Engineering Div., National Chemical Laboratory, Pune-411 008, India

This article presents two hybrid robust process optimization approaches integrating artificial neural networks (ANN) and stochastic optimization formalisms — genetic algorithms (GAs) and simultaneous perturbation stochastic approximation (SPSA). An ANN-based process model was developed solely from process input–output data and then its input space comprising design and operating variables was optimized by employing either the GA or the SPSA methodology. These methods possess certain advantages over widely used deterministic gradient-based techniques. The efficacy of ANN-GA and ANN-SPSA formalisms in the presence of noise-free as well as noisy process data was demonstrated for a representative system involving a nonisothermal CSTR. The case study considered a nontrivial optimization objective, which, in addition to the conventional parameter design, also addresses the issue of optimal tolerance design. Comparison of the results with those from a robust deterministic modeling/optimization strategy suggests that the hybrid methodologies can be gainfully employed for process optimization.

Introduction

Conventionally, chemical plant design consists of choosing and sizing appropriate process equipment, as well as fixing the nominal operating points. In this endeavor, deterministic gradient-based optimization techniques that mostly use steady-state process models are utilized. Here, the objective function to be optimized is a suitably chosen cost function (to be minimized) or a profit function (to be maximized). Traditionally, issues such as the choice and design of the process control system are addressed once the nominal operating point is known consequent to the process design activity.

Availability of a process model assumes considerable importance in the process design activity. For a given process, a "first principles (phenomenological)" model can be constructed from the knowledge of mass, momentum, and energy balances, as well as from other chemical engineering principles. Owing to the lack of a good understanding of the underlying physicochemical phenomena, development of phenomenological process models poses considerable difficulties. Moreover, nonlinear behavior being a common feature of chemical processes, it leads to complex nonlinear models, which in most cases are not amenable to analytical solutions; thus, computationally intensive numerical methods must be utilized for obtaining solutions. The difficulties associated with the construction and solution of the phenomenological models necessitate exploration of alternative modeling formalisms. Process identification via empirical models is one such alternative. They are mostly discrete-time dynamic models comprising, for instance, Hammerstein and Wiener models, Volterra models and polynomial autoregressive moving-average models with exogenous inputs (ARMAX) (Henson, 1998). These linear models and their nonlinear counterparts are constructed exclusively from the input–output process data. A fundamental deficiency of the empirical modeling approach is that the model structure (form) must be specified *a priori*. Satisfying this requirement, especially for nonlinearly behaving processes is a cumbersome task, since it involves selecting heuristically an appropriate model structure from numerous alternatives.

In recent years, artificial neural networks (ANNs) have been found to be an attractive tool for steady-state/dynamic process modeling, and model-based control in situations where the development of phenomenological or the empirical models just given either becomes impractical or cumber-

Correspondence concerning this article should be addressed to S. S. Tambe.
Present address of S. Ghosh: Chemical Engineering Department, Indian Institute of Technology (IIT), Kharagpur, West Bengal 721 302, India.

some (such as Bhat and McAvoy, 1990; Hernandez and Arkun, 1992; Nahas et al., 1992; Ramasamy et al., 1995; Tendulkar et al., 1998; and reviews by Narendra and Parthasarathy, 1990; Hunt et al., 1992; Agarwal, 1997). ANNs are based on the concept that a highly interconnected system of simple processing elements (called *neurons* or *nodes*) can approximate complex nonlinear relationships existing between independent (ANN input) and dependent (ANN output) variables to an arbitrary degree of accuracy (Hornik et al., 1989; Poggio and Girosi, 1990). The advantages of a neural-network-based process model are (1) it can be developed solely from the process input–output data (that is, without invoking process phenomenology); (2) even multi-input multioutput relationships can be approximated easily; and (3) it possesses generalization ability owing to which the model can accurately predict the outputs corresponding to a new set of inputs that were not part of the data used for constructing the ANN model. For design purposes, it is not adequate that a generalization-capable ANN model is available. What is important is that the ANN model should be amenable to optimization. Specifically, it should be possible to optimize the input space of the ANN model, representing process variables, such that the model output (product concentration, reactor temperature, etc.) is maximized or minimized. This objective differs from that involving ANN model development where given an input–output example data set, a suitably chosen optimization algorithm finds a set of network parameter (weights) that minimizes a prespecified error function.

In commonly used deterministic optimization techniques, the solution to an optimization problem is represented in the form of a vector consisting of values of decision variables at which the gradient of the objective function with respect to the decision variables becomes zero. Thus, gradient computation is an integral feature of such optimization paradigms. Additionally, most gradient-based techniques require the objective function to be smooth, continuous, and differentiable.

In the case of an ANN, it is possible to express the nonlinear mapping that it executes in terms of a generic closed-form function. It can be noted that the nonlinear mapping ability of ANNs is due to the nonlinear activation function used for computing the node-specific outputs. For computing an output, the nonlinear activation function makes use of the arguments comprising a number of network parameters (weights) and node-specific inputs. Consequently, the mapping executed by an ANN attains a complex nonlinear character that cannot be guaranteed to simultaneously fulfill the smoothness, continuity, and differentiability criteria for the objective function. This feature of ANN models poses difficulties in using the conventional deterministic techniques for optimizing their input space. Hence, formalisms that do not impose stringent conditions on the form of the objective function need to be explored. The stochastic optimization formalisms, namely, genetic algorithms (GAs) and simultaneous perturbation stochastic approximation (SPSA), among others, are not heavily constrained by the properties of the objective function, and thus they are potential candidates for employment in the optimization of an ANN model. An important characteristic of the GA and SPSA methodologies is that they need measurements of the objective function only, and not the measurements (or direct calculation) of the gradient (or higher-order derivatives) of the objective function (Spall, 1998a,b). This characteristic of the GA and SPSA methods can be fruitfully exploited for optimizing the ANN-based models whose functional forms do not assuredly satisfy the requirements of the gradient-based optimization methods. An additional benefit of the GA and SPSA methods is that they can be used in situations where input information into the optimization method (such as objective function evaluations) may be noisy. The objective of this article, therefore, is to present two hybrid "modeling-optimization" techniques, namely, ANN-GA and ANN-SPSA, for the purpose of robust chemical process design and optimization. In this approach, an ANN-based process model is first developed and its input space is optimized next using either the GA or the SPSA formalism. The principal advantage of the hybrid methods is that process design and optimization can be conducted solely from the steady-state process input–output data.

For validating the ANN-GA and ANN-SPSA methods, we have considered a nontrivial process optimization objective, which not only aims at obtaining the optimal values of process variables, but also the optimal values of tolerances (operating windows) for the process variables. Fixing the values of the tolerances becomes important owing to the fact that chemical processes involve a number of variables and/or parameters that are always subjected to some degree of uncertainty (stochastic variability). For instance, irrespective of how good a control system is, process variables such as concentration, temperature, and pressure do vary randomly, albeit within a narrowly bounded window. Depending upon their origin, uncertainties can be classified into the following four categories (Pistikopoulos, 1995; Pistikopoulos and Ierapetritou, 1995; Diwekar and Kalagnanam, 1996; 1997a,b).

- *Process-inherent uncertainty.* Due to random variations in process parameters/variables, such as flow rate, temperature, and pressure.
- *Model-inherent uncertainty.* Accounts for variations in the phenomenological model parameters representing, for instance, kinetic constants, heat-/mass-transfer coefficients, and physical properties.
- *External uncertainty.* Considers variations in parameters that are external to the process, but influencing the process cost (feed stream availability, product demand, pollution/economic indices, etc.).
- *Discrete uncertainty.* Accounts for the equipment availability and other random discrete events.

The conventional deterministic process optimization approach ignores uncertainties, thereby resulting in suboptimal solutions. Uncertainties are capable of influencing, for instance, the product quality and control cost and, therefore, they need to be considered during process design and optimization activity. Accounting for uncertainties leads to tolerance design, which aims at obtaining the optimal size of the window for each uncertainty-affected process variable/parameter. The best average process performance can be achieved consequent to optimal tolerance design so long as the process operates within the optimized operating zones.

In a recent article by Bernardo and Saraiva (1998), the authors have introduced a novel robust optimization (RO) framework that deals with the optimization objective alluded to earlier. An advantage of the optimal solution given by the RO framework is that it provides the best operating regions for designing a control system. One of the optimization prob-

lems considered by Bernardo and Saraiva was minimization of the continuous stirred-tank reactor's (CSTR) annual plant cost that comprises four components, namely, equipment, operating, control, and quality costs. The RO formalism, which accounts for the control costs right at the process design stage, was shown to yield qualitatively improved results as compared to those obtained using a fully deterministic optimization approach. Specifically, it was shown for the case of CSTR that simultaneous optimization of process operating variables and the respective tolerances could reduce the total annual plant cost by nearly one order of magnitude, that is, from \$101,872/yr to \$14,716/yr. For the sake of affording a direct comparison, we adopt the RO framework with necessary modifications to account for the ANN-based process model. The principal differences between the RO methodology and the hybrid formalisms presented here are as follows.

- While the RO framework assumes knowledge of a phenomenological process model, the ANN-GA and ANN-SPSA formalisms utilize a neural-network-based process model.
- In the RO approach, the phenomenological process model is optimized using a deterministic successive quadratic programming algorithm (NPSOL package; Gill et al., 1986), whereas hybrid methodologies optimize the ANN-based model using inherently stochastic optimization techniques, namely, GA and SPSA. It may be noted, however, that evaluation of the objective function accounting for the stochastic behavior of the uncertainty-affected process variables, remains the same in the RO- and ANN-based hybrid formalisms.
- In the present study it is shown that the hybrid optimization methodologies are capable of yielding comparable solutions when noise-free and noisy process data are utilized for constructing the ANN-based process model.

This article is structured as follows. First, the mathematical formulation of the ANN model-based robust optimization is presented. A detailed discussion of the development of the ANN-based process model and the stepwise implementation of the ANN-GA and ANN-SPSA hybrid optimization methodologies are provided next. Finally, results pertaining to the CSTR optimization case study are presented and discussed.

ANN-Assisted Robust Optimization Framework

While developing the framework, it is not necessary to take the model-inherent uncertainty into account, since the ANN-GA and ANN-SPSA strategies do not utilize a phenomenological model. Among the remaining three uncertainty categories, only process-inherent uncertainty has been considered although the optimization framework presented below is sufficiently general toward inclusion of the remaining two (external and discrete) uncertainties. The origin of the commonly encountered process-inherent uncertainty lies in the small but significant random (uncontrolled) fluctuations present in the process operating variables.

We define the optimization problem under consideration as: given the process input data comprising the steady-state values of equipment (design) and operating variables, and the corresponding values of process output variables, obtain, in a unified manner, the optimal values of (1) design variables, (2) operating variables, and (3) tolerances defining bounds on the operating variables. The optimal solutions so obtained should ensure minimization of the annual plant cost while maintaining the desired product quality.

The ANN-assisted RO framework assumes that a steady-state process model defined as

$$y = f(\Psi, \Phi, W) \tag{1}$$

is available, where y represents the process output variable that also determines product quality; Ψ and Φ, respectively, refer to the M- and N-dimensional vectors of design and operating process variables ($\Psi = [\psi_1, \psi_2, \ldots, \psi_m, \ldots, \psi_M]^T$; $\Phi = [\phi_1, \phi_2, \ldots, \phi_n, \ldots, \phi_N]^T$); W denotes the weight matrix of the ANN model; and f represents the ANN-approximated nonlinear function.

The total annual plant cost (C_{yr}) to be minimized is assumed to consist of four components, namely, equipment cost (C_{eqp}), operating cost (C_{op}), control cost (C_c), and quality cost (C_q):

$$C_{yr} = C_{eqp} + C_{op} + C_c + C_q. \tag{2}$$

Among the four cost components, the operating, control, and quality costs have uncertainties associated with them that emanate from random fluctuations in the process operating variables. However, the equipment cost, which is usually a deterministic quantity, has no uncertainty attached to it. The extent of uncertainty in an operating variable can be characterized in terms of a probability density function (PDF) (Diwekar and Rubin, 1991, 1994) where mean value of the PDF represents the nominal value of that operating variable. Accordingly, defining $J(\Phi)$ to be the set of PDFs associated with the operating variables, Φ, the corresponding set ($\hat{\Phi}$) describing the operating space regions can be represented as

$$\hat{\Phi} = \{\Phi : \Phi \in J(\Phi)\}. \tag{3}$$

The physical operating regions, $\hat{\Phi}$, comprise a set of operating windows $\{\hat{\Phi}_{\phi_n}\}$, where Φ_{ϕ_n}, denoting the window for the nth operating variable, is defined as

$$\hat{\Phi}_{\phi_n} = [\phi_n^l, \phi_n^u]; \qquad n = 1, 2, \ldots, N. \tag{4}$$

Here, ϕ_n^l and ϕ_n^u, respectively, representing the lower and upper bounds on the nth operating variable, are expressed as

$$\phi_n^l = \mu_n(1 - \epsilon_n); \qquad \phi_n^u = \mu_n(1 + \epsilon_n), \tag{5}$$

where μ_n refers to the mean value of the nth operating variable and ϵ_n is the associated tolerance. Commonly, variations in ϕ_n obey Gaussian (normal) probability distribution, and therefore the respective tolerance (ϵ_n) can be approximated as

$$\epsilon_n = 3.09(\sigma_n/\mu_n), \tag{6}$$

where σ_n refers to the standard deviation of the Gaussian PDF.

Owing to the random fluctuations in process operating variables, the steady-state value of the process output (quality) variable, y, also deviates from its desired (nominal) setpoint. Thus, it becomes essential to define a PDF, $L(y)$, pertaining to the quality variable y as well. Accordingly, an expression similar to Eq. 3, but involving $L(y)$, can be written for defining the corresponding space region ($\hat{y}$):

$$\hat{y} = \{y : y \in L(y)\}. \tag{7}$$

Using Eqs. 2–7, it is possible to write the complete mathematical formulation of the robust optimization problem under consideration as

$$\min_{\Psi, \hat{\Phi}} C_{yr}(\Psi, \hat{\Phi}, \hat{y}) = C_{eqp}(\Psi) + C_{op}(\hat{\Phi}) + C_c(\hat{\Phi}) + C_q(\hat{y}) \tag{8}$$

subject to,

$$\text{(I)} \quad f(\Psi, \Phi, W) = y \quad \text{for all} \quad \Phi \in \hat{\Phi}; \quad \Phi = [\phi_1, \phi_2, \ldots, \phi_N]^T; \hat{\Phi} = \{\Phi : \Phi \in J(\Phi)\} \tag{9}$$

$$\text{(II)} \quad \hat{y} = \{y : y \in L(y)\}. \tag{10}$$

This formulation makes use of an ANN model (Eq. 1) possessing the following properties: (1) the input space of the ANN model comprises design variables, Ψ, and uncertainty-involving process operating variables, Φ; (2) the output space of the ANN model represents the quality variable y; (3) the weight matrix (W) of the ANN model is available; and (4) the model is valid over operating variable regions, $\hat{\Phi}$, and the output region, $\hat{y}$; consequently, the equality constraint defined in Eq. 9 always holds.

In the objective function defined by Eq. 8, the elements of vectors Ψ and $\hat{\Phi}$ signify the decision variables; $\hat{y}$, representing the space region of the output variable, depends on Ψ and $\hat{\Phi}$, and therefore is not a decision variable. However, the optimization objective, which involves the simultaneous determination of the operating variables (nominal operating points) and associated tolerances, necessitates: (1) optimization of the mean values (μ_n; $n = 1, 2, \ldots, N$) of the Gaussian PDFs characterizing uncertainty-involving operating variables Φ, and (2) optimization of the tolerances, ϵ_n ($n = 1, 2, \ldots, N$). Simultaneous determination of the mean $\{\mu_n\}$ and tolerance $\{\epsilon_n\}$ values also fixes the corresponding standard deviations, $\{\sigma_n\}$ (see Eq. 6), which can be used to characterize the PDF set, $J(\Phi)$. It is thus clear that optimization of the mean and associated tolerance values in turn leads to the optimization of the PDF set, $J(\Phi)$.

Following the prescription of Bernardo and Saraiva (1998), the objective function defined in Eq. 8 can be evaluated as

$$C_{yr} = E(C_q) + E(C_{op}) + C_{eqp}(\Psi) + C_c(\hat{\Phi}), \tag{11}$$

where $E(C_q)$ and $E(C_{op})$ refer to the expected values of the quality cost and the operating cost, respectively. For computing these expected costs, an efficient sampling technique known as *Hammersley sequence sampling* (HSS) (Diwekar and Kalgnanam, 1996, 1997a,b) could be utilized. In this technique, a statistically adequate number (N_{obs}) of observations is sampled from the Gaussian PDFs associated with the uncertainty-involving operating variables. Next, each of the N_{obs} sampled sets, Φ_j ($j = 1, 2, \ldots, N_{obs}$), along with the design variable vector, Ψ, is applied to the ANN model for computing the magnitude of the process output, y. The estimate of the quality cost can then be computed using the Taguchi loss function (Taguchi, 1986), as given below:

$$E(C_q) = k_l\left[(\mu_y - y^*)^2 + \sigma_y^2\right], \tag{12}$$

where k_l refers to the *quality loss coefficient*; y^* is the desired value of y; and σ_y denotes the standard deviation of N_{obs} number of y values. The mean value of the quality variable, μ_y, is calculated as

$$\mu_y = \frac{\sum_{j=1}^{N_{obs}} f(\Psi, \Phi_j, W)}{N_{obs}}. \tag{13}$$

For computing the estimate of the operating cost, $E(C_{op})$, the following expression is used:

$$E(C_{op}) = \frac{\sum_{j=1}^{N_{obs}} C_{op}(\Phi_j)}{N_{obs}}. \tag{14}$$

Since the design variables are not associated with any uncertainty, the Ψ-dependent equipment cost, $C_{eqp}(\Psi)$, can be calculated deterministically. The last of the four cost components representing the control cost, C_c, can be determined using the mean (μ_n) and standard deviation (σ_n) of the PDFs describing operating variables:

$$C_c = \sum_{n=1}^{N} \left(a + b\frac{\mu_n}{\sigma_n}\right), \tag{15}$$

where a, b are constants and n refers to the operating variable index.

Construction of ANN-Based Process Model

A prerequisite to the implementation of the ANN-GA and ANN-SPSA methodologies is the development of a suitable ANN-based process model. For this purpose, a class of ANNs known as *multilayered feedforward networks* (MFFNs) can be used. An MFFN (see Figure 1) is a nonlinear mapping device between an input set (I) and an output set (O). It represents a function f that maps I into O, that is, $f: I \rightarrow O$, or $y = f(x)$, where $y \in O$ and $x \in I$. The widely used MFFN paradigm is *multilayered perceptron* (MLP), mostly comprising three sequentially arranged layers of processing units. The three successive layers, namely, *input*, *hidden*, and *output* layers, house N_I, N_H, and N_O number of nodes, respectively. Usually, the input and hidden layers also contain a bias node possessing a

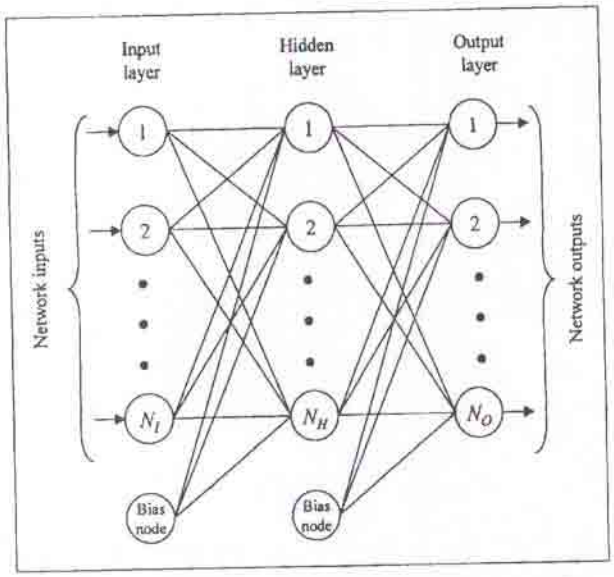

Figure 1. Three-layered feed-forward neural network.

constant output of +1. All the nodes in the input layer are connected using weighted links to the hidden layer nodes; similar links exist between the hidden and output-layer nodes. Nodes in the input layer do not perform any numerical processing, and thus act as "fan-out" units; all numerical processing is done by the hidden and output-layer nodes, and thus they are termed "active" nodes.

The problem of neural network modeling is to obtain a set of weights such that the prediction error (difference between network-predicted outputs and their desired values) measured in terms of a suitable error function, for instance, the *root-mean-squared error* (RMSE), is minimized. The RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{l=1}^{N_{\text{pat}}} 2E_l}{N_{\text{pat}} \times N_o}}, \tag{16}$$

where l refers to the input pattern index ($l = 1, 2, \ldots, N_{\text{pat}}$); N_o denotes the number of output layer nodes, and E_l is a measure of the *sum-of-squares error* (SSE), defined as

$$E_l = \frac{1}{2} \sum_{i=1}^{N_o} \left(y_i^l - o_i^l\right)^2, \tag{17}$$

where, y_i^l denotes the desired output of the ith output node when the lth input pattern is presented to the network, and o_i^l refers to the corresponding desired output. The task of RMSE minimization is accomplished by "training" the network wherein a gradient descent technique, such as the *generalized delta rule* (GDR) (Rumelhart et al., 1986), is utilized for the updation of connection weights.

Network training is an iterative procedure that begins with initializing the weight matrix randomly. A training iteration consists of two types of passes, namely, *forward* and *reverse*, through the network layers. In the forward pass, an input pattern from the example data set is applied to the input nodes and outputs of the active nodes are evaluated. For computing the output, the weighted-sum of the inputs to an active node is calculated first, which is then transformed using a nonlinear activation function, such as the sigmoid function. The outputs of the hidden nodes computed in this manner form the inputs to the output-layer nodes whose outputs are evaluated in a similar fashion. In the reverse pass, the pattern-specific SSE defined in Eq. 17 is computed and used for updating the network weights in accordance with the GDR strategy. The weight-updation procedure when repeated for all the patterns in the training set completes one training iteration.

The RMSE minimization procedure described earlier does not ensure that the trained network possesses satisfactory generalization ability. To possess good generalization ability, it is essential that the network captures the underlying trends existing in the example input–output data. The phenomenon, which affects the network's generalization ability, is known as "overfitting." When it occurs, the network attempts to fit even the noise in the example data set at the cost of learning the trends therein. As a result, the network makes poor predictions in the case of new inputs. Overfitting occurs due to two factors: (1) when the network is trained excessively (overtrained), that is, over a large number of training iterations, and (2) when the network's hidden layer contains more units than necessary. To prevent the occurrence of overfitting, the network's generalization performance is monitored at the end of every training iteration on a set different than the one used for updating the weights. Specifically, the available example input–output data are partitioned into two sets, namely, the training and test sets. While the former set is used for adjusting the network's weights, the latter set is utilized for monitoring the network's generalization performance. In essence, the RMSE magnitude with respect to the training set (E_{trn}) indicates the data-fitting ability of the network undergoing training, and the test set RMSE (E_{tst}) measures how well the network is generalizing. Upon training the network over a large number of iterations, the weight matrix resulting in the smallest E_{tst} magnitude for the test set data, is taken to be an optimal weight set. It may, however, be noted that this weight set pertains to the specific number of hidden units (N_H) considered in the network architecture.

For a given ANN-based modeling problem, the number of nodes in the network's input layer (N_I) and output layer (N_O) are dictated by the input–output dimensionality of the system being modeled. However, the number of hidden units (N_H) is an adjustable structural parameter. If the network architecture contains more hidden units than necessary, they lead to an oversized network and, consequently, an overparameterized network model. Such a model, like an overtrained one, gives poor representation of the trends in the example data. For excluding the possibility of an oversized network, it becomes essential to study the effect of the number of hidden units on the network's function approximation and generalization capabilities. Accordingly, multiple network training simulations are conducted by systematically varying the number of hidden units. Theses simulations essentially aim at obtaining an optimal network architecture

(that is, housing only adequate number of hidden units), leading to the smallest possible RMSE magnitude for the test set data.

The entire procedure for selecting an optimal MLP architecture and associated weight matrix using the GDR strategy is summarized in the following steps (Bishop, 1994):

1. Fix a small value (such as one or two) for the number of hidden units, N_H, and initialize the network weight matrix randomly. Also select values of the GDR parameters, namely, learning rate (η_l; $0 < \eta_l \leq 1.0$) and momentum coefficient (α_m; $0 < \alpha_m \leq 1.0$): addition of the momentum term in the GDR-based weight-updation expression helps accelerate the weight convergence and to avoid the local minima on the error surface.

2. Minimize the test set RMSE (E_{tst}) using the GDR-based error-back-propagation (EBP) algorithm. Repeat the training procedure a number of times using different random number sequences for initializing the network weights. This procedure is performed for exploring the weight–space rigorously and, consequently, locating the deepest local minimum (or the global one) on the error surface. Store the network weight matrix that has resulted in the smallest E_{tst}.

3. Repeat steps 1 and 2 by systematically increasing the number of hidden units until E_{tst} attains its smallest possible magnitude.

Implementation of these steps optimizes the network architecture and the associated weight matrix, thereby creating an optimal network model possessing the much desired data-fitting and generalization capabilities. For details of the GDR-based EBP training algorithm, the reader may refer to, for example, Hecht-Nielsen (1990), Freeman and Skapura (1991), and Tambe et al. (1996).

ANN-Model-Assisted Stochastic Process Optimization Methodologies

The principal difference between the widely used deterministic gradient-based optimization schemes and the stochastic ones, such as GA and SPSA, is that the latter class of methodologies involves a random component at some stage in their implementation. For instance, GAs manipulate a set of candidate solutions at random with the objective of sampling the search (solution) space as widely as possible, while at the same time trying to locate promising regions for further exploration (Venkatasubramanian and Sundaram, 1998). In the present work, GA and SPSA methodologies have been used, in conjunction with an ANN-based process model, for optimizing (1) process design variables, Ψ; (2) process operating variables, Φ; and (3) tolerances, $E = [\epsilon_1, \epsilon_2, \ldots, \epsilon_n, \ldots, \epsilon_N]^T$, associated with the process operating variables. In what follows, the salient features and implementation details of the ANN-GA and ANN-SPSA methodologies are provided.

ANN-GA optimization methodology

Genetic algorithms (Holland, 1975; Goldberg, 1989) combine the "survival of the fittest" principle of natural evolution with the genetic propagation of characteristics, to arrive at a robust search and optimization technique. Principal features possessed by the GAs are: (1) they are zeroth-order optimization methods requiring only the scalar values of the objective function; (2) capability to handle nonlinear, complex, and noisy objective functions; (3) they perform global search, and thus are more likely to arrive at or near the global optimum; and (4) their search procedure being stochastic, GAs do not impose preconditions, such as smoothness, differentiability, and continuity on the objective function form. Owing to these attractive features, GAs are being used for solving diverse optimization problems in chemical engineering (such as Cartwright and Long, 1993; Hanagandi et al., 1996; Garcia and Scott, 1998; Garcia et al., 1998; Garrard and Fraga, 1998; Polifke et al., 1998). An application of the ANN-GA formalism for the purpose of parameter design of an industrial dryer using noise-free data has been shown recently by Hugget et al. (1999). To illustrate the working principles of GAs, we recast the cost-minimization problem (Eq. 8) as

$$\text{Minimize } C_{yr}(x);\ x_k^L \leq x_k \leq x_k^U;\ k = 1, 2, \ldots, K; \qquad x = \Psi \cup \Phi \cup E, \quad (18)$$

where K-dimensional vector, $x = [x_1, x_2, \ldots, x_k, \ldots, x_k]^T$, represents the set of decision variables, and x_k^L and x_k^U are the lower and upper bounds on x_k.

In a typical GA procedure, search for the optimal solution is conducted from a randomly initialized population of candidate solutions, wherein each solution is usually coded as a string (chromosome) of binary digits. A coded string is composed of as many segments as the number (K) of decision variables. The resultant population of candidate solutions is then iteratively refined in a manner imitating selection and adaptation in biological evolution, until convergence is achieved. Within an iteration, GA evaluates the goodness of a candidate solution by employing a fitness function, whose magnitude is indicative of the objective function value. For a function maximization (minimization) problem, the fitness function value should scale up (scale down) with the increasing value of the objective function. In each GA-iteration, a new population (generation) of candidate solutions is formed using the following GA operators:

- *Selection:* This operator chooses chromosome strings to form a mating pool of parent strings that are subsequently used for producing offspring. Selection of parent strings is conducted in a manner such that fitter strings enter the mating pool on a priority basis. For selection purposes, the *roulette wheel* (RW) or less noisy *stochastic remainder* (SR) methodologies (Goldberg, 1989) may be used.
- *Crossover:* The action of this critical GA operator produces an offspring population wherein randomly selected parts of the parent strings are exchanged mutually to form two offspring strings per parent pair. Whether a pair (also selected randomly) undergoes crossover or not is governed by the prespecified value of the crossover probability (P_{cross}). Action of the crossover operator tends to improve the combinatorial diversity of the offspring population by utilizing the building blocks of the parent population (Venkatasubramanian and Sundaram, 1998). A high P_{cross} magnitude (such as $0.5 \leq P_{cross} \leq 1.0$) ensures more crossover between parent pairs, and thereby greater diversity in the offspring population.
- *Mutation:* This operator introduces new characteristics in the offspring population by randomly flipping bits of the

offspring strings from zero to one and vice versa. The bit-flipping operation performed with a small (0.01–0.05) probability of mutation (P_{mut}) helps in conducting a local search around point solutions represented by the unmutated offspring strings.

The stepwise procedure for implementing the ANN-model-assisted GA strategy is now in order (also see flow chart in Figure 2).

Step 1. Initialize generation counter, N_{gen}, to zero.

Step 2. Create the initial population of N_{pop} number of candidate solution strings randomly using binary digits; each string of length equal to l_{chr} digits comprises K segments.

Step 3. Using the HSS technique on the ith ($i = 1, 2, \ldots, N_{pop}$) string, sample N_{obs} number of operating variable sets from the Gaussian PDFs associated with the operating variables, Φ.

Step 4. Apply the jth ($j = 1, 2, \ldots, N_{obj}$) sampled set ($\Phi_j$) along with the corresponding design variable vector (Ψ_i) to the ANN model and obtain the model output, y_j. Next, compute mean (μ_y) and standard deviation (σ_y) of the ANN output set, $\{y_j\}$.

Step 5. Evaluate the objective function (Eq. 11) and fitness score of the ith population string.

Step 6. Repeat steps 3–5 for all population strings (that is, $i = 1, 2, \ldots, N_{pop}$) and rank the strings in the decreasing order of their fitness scores.

Step 7. Create a mating pool of parent strings using the SR selection scheme.

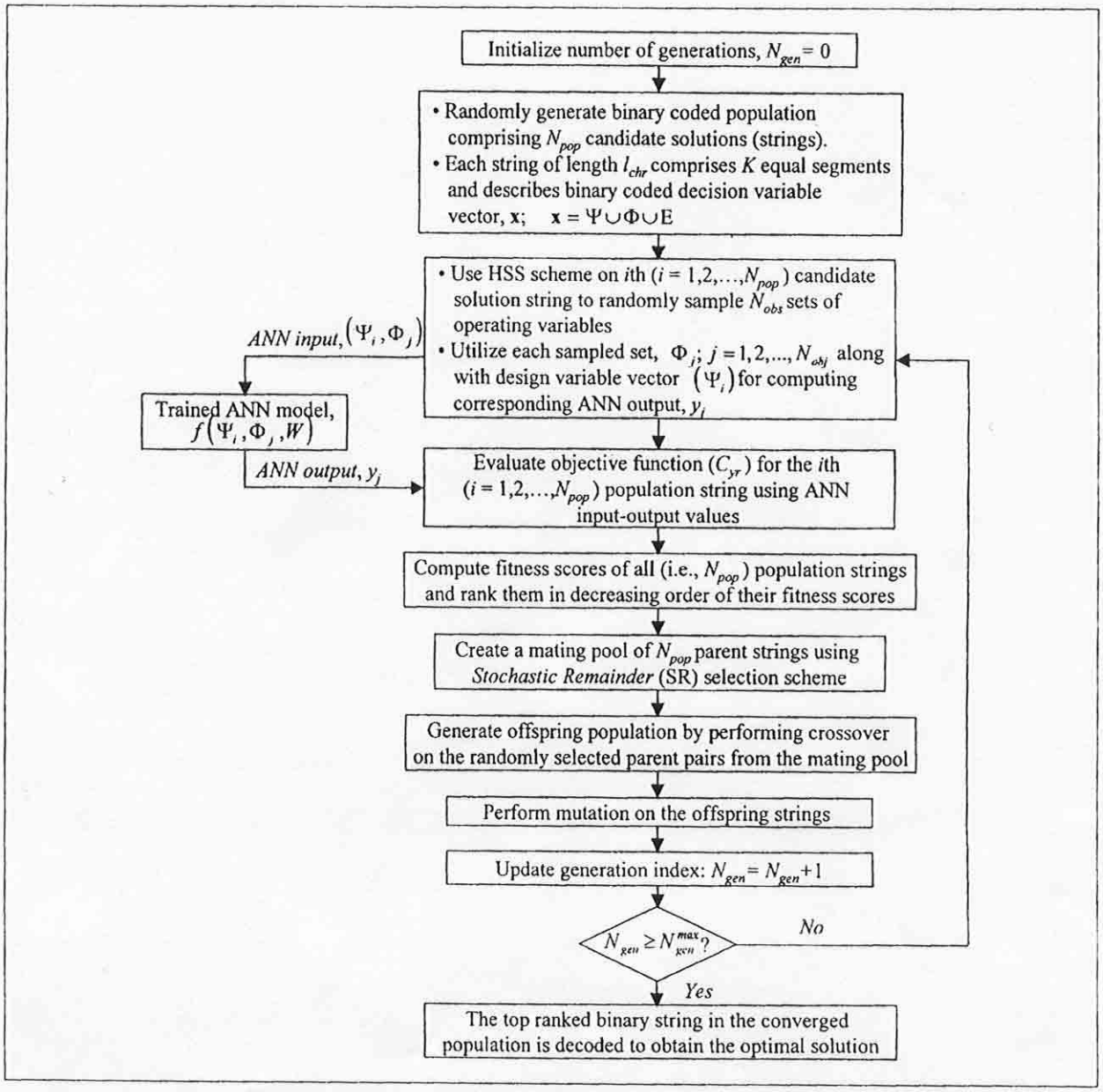

Figure 2. Implementation of ANN-GA hybrid methodology.

Step 8. From the mating pool, choose pairs of parent strings randomly and perform crossover operation on each one of them to obtain the offspring population.

Step 9. Perform mutation on the offspring population.

Step 10. Update generation index by one: $N_{gen} = N_{gen} + 1$.

Step 11. Repeat steps 3–10 on the new generation strings until a convergence criterion, such as (1) N_{gen} exceeds the prespecified maximum generations limit (N_{gen}^{max}), or (2) the fitness score of the best string in a population no longer increases, is satisfied.

ANN-SPSA optimization methodology

The SPSA optimization methodology (Spall, 1987, 1998a,b) differs from the commonly employed deterministic gradient-based techniques in the following respects. Instead of directly evaluating the gradient with respect to each decision variable by perturbing it separately (as done in the standard two-sided finite difference approximation), the SPSA methodology approximates the gradient by perturbing all the decision variables simultaneously. Thus, irrespective of the number (K) of decision variables, only two objective function measurements are necessary for gradient approximation; this is in contrast to the finite-difference approximation, where $2K$ function measurements are necessary for the gradient evaluation. The implementation procedure of ANN-SPSA formalism is an iterative that begins with a randomly initialized (guess) solution vector, $\hat{x}$. The SPSA technique stipulates the cost function, $C_{yr}(x)$, to be differentiable, since it searches for the minimum point, x^*, at which the gradient of the objective function, $g(x^*)$, attains zero magnitude. That is,

$$g(x^*) = \left| \frac{\partial C_{yr}(x)}{\partial x} \right|_{x = x^*} = 0. \tag{19}$$

In each SPSA iteration, the gradient is approximated by utilizing the numerically efficient simultaneous perturbation technique alluded to earlier. With these preliminaries, the stepwise procedure for ANN-SPSA implementation can be given as (also see flow chart in Figure 3):

Step 1. Set the iteration index, t, to zero and choose randomly a K-dimensional guess solution vector, $\hat{x}_t|_{t=0}$.

Step 2. Compute the t-dependent values, A_t and Z_t, termed "gain sequences" using,

$$A_t = \frac{A}{(r+t+1)^{\eta}}; \qquad Z_t = \frac{Z}{(t+1)^{\beta}}, \tag{20}$$

where constants, A, Z, r, η, and β assume nonnegative values. The optimal values of η and β are either 0.602 and 0.101 or 1.0 and 0.1667, respectively (Spall, 1998a).

Step 3. Generate a K-dimensional perturbation vector, Δ_t, using Bernoulli ± 1 distribution, where probability of occurrence of either $+1$ or -1 is 0.5; next, perturb all the K elements of the vector $\hat{x}_t$ simultaneously, as given by

$$\hat{x}_t^+ = \hat{x}_t + Z_t\Delta_t; \qquad \hat{x}_t^- = \hat{x}_t - Z_t\Delta_t. \tag{21}$$

Step 4. Using $\hat{x}_t^+$ and $\hat{x}_t^-$ as arguments, compute two measurements, that is, $C_{yr}(\hat{x}_t^+)$ and $C_{yr}(\hat{x}_t^-)$, of the objective function defined in Eq. 11. This step involves usage of (1) the HSS technique for sampling N_{obs} sets of operating variables, and (2) the ANN model for computing the expected costs, $E(C_q)$ and $E(C_{op})$, as well as the control cost, $C_c(\hat{\Phi})$.

Step 5. Generate the simultaneous perturbation approximation of the unknown gradient, $\hat{g}_t(\hat{x}_t)$, using

$$\hat{g}_t(\hat{x}_t) = \left[\frac{C_{yr}(\hat{x}_t^+) - C_{yr}(\hat{x}_t^-)}{2Z_t} \right] \times \left[\Delta_{t_1}^{-1}, \Delta_{t_2}^{-1}, \ldots, \Delta_{t_K}^{-1} \right]^T, \tag{22}$$

where $\hat{g}_t(\hat{x}_t)$ is K-dimensional and Δ_{t_k} refers to the kth element ($+1$ or -1) of the perturbation vector, Δ_t.

Step 6. Update estimate of the decision vector according to

$$\hat{x}_{t+1} = \hat{x}_t - A_t\hat{g}_t(\hat{x}_t). \tag{23}$$

Step 7. Increment the iteration counter t to $t+1$ ($1 \le t \le t_{max}$) and repeat steps 2–6 until convergence; the criterion for convergence could be that in successive iterations the decision variable values exhibit very little or no change.

Optimization of CSTR Using ANN-GA and ANN-SPSA Strategies

Consider a steady-state process involving a jacketed non-isothermal CSTR wherein two first-order reactions in series, $A \rightarrow B \rightarrow C$, take place. For implementing the ANN-GA and ANN-SPSA schemes, an MLP-based model approximating the functional relationship between the CSTR's design and operating variables, and the corresponding steady-state values of the output variable, was first developed. For convenience, the steady-state CSTR data required for developing the MLP model were generated using the CSTR's phenomenological model. In actual practice, the MLP model can be developed from a representative steady-state database that is already at hand or generated by conducting specially designed experiments. The phenomenological equations of CSTR used for simulating its steady-state behavior are given in Appendix A, where volume (V, m^3) represents the design variable and, flow rate (F, m^3/min), heat removal rate (Q, kJ/min), inlet concentration of reactant A (C_A^0, mol/m^3), inlet concentration of B (C_B^0, mol/m^3), and inlet temperature (T^0, K), collectively denote the five operating variables. The CSTR output variable, namely, the rate of production (mol/min) of B, has been chosen as the quality variable, and its steady-state value (y) has been obtained as

$$y = \left(\hat{C}_B - C_B^0\right) \times F. \tag{24}$$

where, $\hat{C}_B$ represents the steady-state concentration of B. The desired value of y defined as y^* is 600 mol/min.

For operating a process, ranges of design and operating variables are usually specified. Accordingly, the following ranges were considered for the steady-state CSTR simulation: $V = [0.3–0.4]$ m^3, $F = [0.5–1.1]$ m^3/min, $Q = [100–1{,}100]$

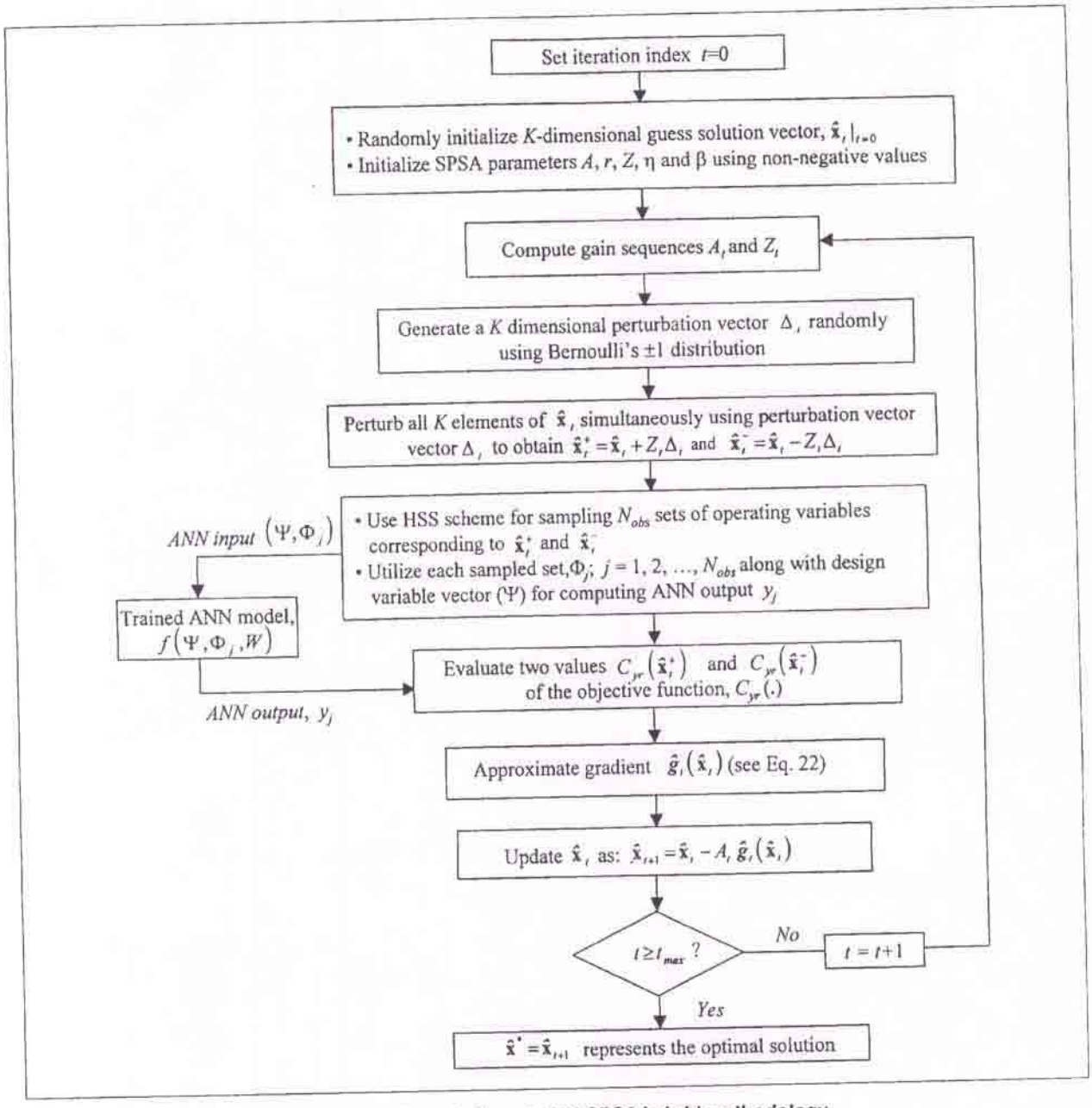

Figure 3. Implementation of ANN-SPSA hybrid methodology.

kJ/min, C_A^0 = [3,000–4,000] mol/m^3, C_B^0 = [30–600] mol/m^3, and T^0 = [300–320] K. Using these ranges, 50 random combinations of the CSTR's design and operating variables were generated, and using each combination, the corresponding steady-state value of the quality variable, y, was computed. The data set comprising design and operating variables forms the network's input space, and the corresponding y values represent the network's desired (target) output space. After normalizing and partitioning these data into the training set (40 patterns) and the test set (10 patterns), an optimal MLP network model was developed in accordance with the three-step network training procedure described earlier. In the MLP training simulations, use of sigmoid transfer function was made for computing the outputs of the hidden and output-layer nodes. The optimal MLP architecture obtained thereby has six input nodes, two hidden nodes, and one output node (N_I = 6, N_H = 2, N_O = 1); the corresponding values of the learning rate (η_l) and momentum coefficient (α_m) were 0.7 and 0.01, respectively. An MLP network with good function approximation and generalization abilities results in small but comparable RMSE values for both the training set (E_{trn}) and the test set (E_{tst}). In the case of the MLP-based CSTR model, the E_{trn} and E_{tst} magnitudes were 0.0061 and 0.0063, respectively. Additionally, values of the coefficient of

correlation (CC) between the MLP-predicted and target y values were calculated. The CC values for the training and test sets were 0.999 and 0.998, respectively. It can be inferred from the very small magnitudes of the training and test set RMSES and the corresponding high (≈ 1) CC magnitudes, that the MLP network has excellently approximated and generalized the nonlinear relationship existing between its six inputs and the single output.

The generalized framework of the ANN-based robust optimization aims at not only obtaining the optimal values of the design and operating variables, but also the optimal values of tolerances (E) associated with the process operating variables. Thus, for the CSTR case study, the overall decision space denoted by vector $\boldsymbol{x}$ (see Eq. 18) becomes eleven-dimensional (one design variable + five operating variables + five tolerances); for clarity, the correspondence between $\boldsymbol{x}$-vector elements and the CSTR variables has been tabulated in Table 1. Upon appropriate substitution from the notation given in Table 1 and the definition of the Taguchi loss function (Eq. 12), the CSTR-specific form of the objective function representing the total annual cost (Eq. 8) can be written as

$$C_{yr} = C_{eqp}(V) + C_{op}(\mu, E) + C_c(\mu, E) + C_q(\mu_y, y^*, \sigma_y), \tag{25}$$

where, μ_y and σ_y, respectively, represent the mean and standard deviation of the quality variable, y, and y^* denotes the desired value of y (= 600 mol/min). The five-dimensional vectors μ and E, respectively, representing the mean and tolerance values of the operating variables are defined as:

$$\mu = [\,\mu_F, \mu_Q, \mu_{C_A^0}, \mu_{C_B^0}, \mu_{T^0}\,]^T \quad \text{and}$$

$$E = [\,\epsilon_F, \epsilon_Q, \epsilon_{C_A^0}, \epsilon_{C_B^0}, \epsilon_{T^0}\,]^T.$$

GA-based optimization of CSTR

For performing the GA-based optimization of the CSTR model, the following values of the GA-specific parameters were used: $K = 11$, $N_{pop} = 30$, $l_{chr} = 55$, $P_{cross} = 0.95$, $P_{mut} = 0.01$, and $N_{gen}^{max} = 250$. The values of parameters l_{chr} and K indicate that within a solution string each decision variable is represented with five-bit precision. This precision can be enhanced further by choosing the l_{chr} and K values such that the ratio, l_{chr}/K, attains a higher (> 5) integer magnitude. For sampling values of the five operating variables from their respective PDFs, a statistically adequate sample size of 400 measurements (Bernardo and Saraiva, 1998) was utilized (that is, $N_{obs} = 400$). The function used for computing the fitness score of a candidate solution (ξ_i) was

$$\xi_i = \frac{15{,}000}{15{,}000 + C_{yr}^i}; \qquad i = 1, 2, \ldots, N_{pop}, \tag{26}$$

where C_{yr}^i refers to the overall annual plant cost corresponding to the ith candidate solution string. The functions used for computing various components of the annual plant cost are described in Appendix B.

In a typical plant, the controller action constraints an operating variable from deviating beyond a specific limit. Accordingly, the tolerances ($\epsilon_F, \epsilon_Q, \epsilon_{C_A^0}, \epsilon_{C_B^0}, \epsilon_{T^0}$) for five operating variables were made to satisfy the following constraints: (1) $0.0001 \le \epsilon_F \le 0.15$, (2) $0.0001 \le \epsilon_Q \le 0.15$, (3) $0.0001 \le \epsilon_{C_A^0} \le 0.1$, (4) $0.0001 \le \epsilon_{C_B^0} \le 0.1$, and (5) $0.0001 \le \epsilon_{T^0} \le 0.02$; these constraints can also be expressed as inequality constraints. Any candidate solution violating the constraints was penalized during the fitness evaluation by resetting its fitness score to zero magnitude (Goldberg, 1989). A more rigorous, that is, penalty function (PF), approach can also be used for constraint handling. In the PF approach, the objective function $f(\boldsymbol{x})$ to be minimized is replaced by the penalty function, $P(\boldsymbol{x})$ (Goldberg, 1989; Deb, 1995):

$$P(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{j_0} \gamma_{j_0} \kappa_h \left[h_{j_0}(\boldsymbol{x})\right] + \sum_{k_e} \gamma_{k_0} \kappa_g \left[g_{k_0}(\boldsymbol{x})\right], \tag{27}$$

where j_0 and k_0 denote the index of inequality and equality constraints, respectively; γ_{j_0} and γ_{k_0} refer to the penalty coefficients (these are usually kept constant throughout the GA simulation); $h_{j_0}(\boldsymbol{x})$ and $g_{k_0}(\boldsymbol{x})$, respectively, represent the inequality and equality constraints, and κ_h and κ_g describe the penalty terms associated with $h_{j_0}(\boldsymbol{x})$ and $g_{k_0}(\boldsymbol{x})$. The penalty terms can assume different forms, which are discussed in greater detail in Deb (1995).

The GA-optimized values of eleven decision variables and corresponding costs are listed in Table 2 (column 1). Additionally, in Figure 4 the Gaussian PDFs defining the optimal space regions of the five operating variables (F, Q, C_A^0, C_B^0, and T^0) are depicted in panels 4a to 4e, respectively.

Table 1. Equivalence Between Decision Vector Elements and CSTR Variables

Decision Variable Index (k)	Decision Vector Variables	Decision Variable Type*	Corresponding CSTR Variable
1	x_1	DES	Volume, V (m^3)
2	$x_2 (= \mu_F)$	OPR	Flow rate, F (m^3/min)
3	$x_3 (= \mu_Q)$	OPR	Heat removal rate, Q (kJ/min)
4	$x_4 (= \mu_{C_A^0})$	OPR	Inlet concentration of A, C_A^0 (mol/m^3)
5	$x_5 (= \mu_{C_B^0})$	OPR	Inlet concentration of B, C_B^0 (mol/m^3)
6	$x_6 (= \mu_{T^0})$	OPR	Inlet temperature, T^0 (K)
7	$x_7 (= \epsilon_F)$	TOL	Tolerance for F (m^3/min)
8	$x_8 (= \epsilon_Q)$	TOL	Tolerance for Q (kJ/min)
9	$x_9 (= \epsilon_{C_A^0})$	TOL	Tolerance for C_A^0 (mol/min)
10	$x_{10} (= \epsilon_{C_B^0})$	TOL	Tolerance for C_B^0 (mol/min)
11	$x_{11} (= \epsilon_{T^0})$	TOL	Tolerance for T^0 (K)

*Abbreviations: DES: design variable; OPR: operating variable; TOL: tolerance of an operating variable.

SPSA-based optimization of CSTR

Implementation of the ANN-SPSA formalism was performed using the following SPSA-specific parameter values: $A = 0.1$, $r = 20.0$, $Z = 0.02$, $\eta = 0.602$, $\beta = 0.101$, and $t_{max} = 32{,}000$. In the SPSA procedure, it was ensured before objective function evaluation that those elements of the perturbed vector $\hat{\boldsymbol{x}}_i$ describing tolerances do fall within the specific limits; if a tolerance value failed this test, then it was reset to its nearest limiting value (Spall, 1998b). Alternatively, the more

Table 2. Comparison of Solutions Obtained Using RO Framework and Hybrid Methodologies

	ANN Model Based on Noise-Free Process Data		ANN Model Based on Noisy Process Data		
	GA-Based Solution 1	SPSA-Based Solution 2	GA-Based Solution 3	SPSA-Based Solution 4	RO Framework Solution* 5
V (m^3)	0.3428	0.3458	0.3726	0.3617	0.3463
F (m^3/min)	$0.5437(1 \pm 0.0936)$	$0.5506(1 \pm 0.0929)$	$0.5270(1 \pm 0.0649)$	$0.5096(1 \pm 0.0573)$	$0.5023(1 \pm 0.099)$
Q (kJ/min)	$262.253(1 \pm 0.0585)$	$208.878(1 \pm 0.0492)$	$684.137(1 \pm 0.0617)$	$611.213(1 \pm 0.0482)$	$146.7(1 \pm 0.080)$
C_A^0 (mol/m^3)	$3{,}312.74(1 \pm 0.0314)$	$3{,}260.577(1 \pm 0.0200)$	$3{,}801.78(1 \pm 0.0219)$	$3{,}775.73(1 \pm 0.0122)$	$3{,}140\,(1 \pm 0.050)$
C_B^0 (mol/m^3)	$422.062(1 \pm 0.0505)$	$399.18(1 \pm 0.0507)$	$542.44(1 \pm 0.0543)$	$592.957(1 \pm 0.0380)$	$510.7(1 \pm 0.050)$
T^0 (K)	$310.444(1 \pm 0.0051)$	$310.82(1 \pm 0.0040)$	$304.42(1 \pm 0.0051)$	$305.414(1 \pm 0.0030)$	$313.8(1 \pm 0.005)$
μ_y (mol/min)**	599.12	601.39	599.85	600.18	600
σ_y (mol/min)	11.48	8.54	11.26	7.0	16.8
C_{rqp} ($/yr)	1,057.03	1,062.79	1,113.31	1,092.99	1,064
C_{op} ($/yr)	9,729.4	9,750.16	9,623.98	9,617.18	9,712
C_c ($/yr)	866.02	2,597.86	2,288.2	3,288.43	2,105
C_q ($/yr)	2,201.02	489.77	828.32	320.61	1,835
C_{yr}	13,853.47	13,900.58	13,853.81	14,319.21	14,716
CPU time†	80.3	47.0	85.5	54.0	—

*Obtained by Bernardo and Saraiva (1998). Solutions with respect to the operating variables are listed using $\mu_n(1 \pm \epsilon_n)$ format.
**Desired μ_y value ($= y^*$): 600 mol/min.
†Seconds taken by 366-MHz Pentium-II CPU to arrive at the optimal solution.

rigorous penalty function approach described earlier (Eq. 27) can be used for handling constraints (Wang and Spall, 1999). It was observed during implementation of the SPSA methodology that the proper choice of the SPSA parameters, namely A, r, and Z, is a prerequisite to successful convergence. For a judicious selection of the stated parameters, the reader may refer to several guidelines provided in Spall (1998a). The results of the SPSA-based CSTR optimization are presented in column 2 of Table 2, where it is seen that the SPSA-minimized annual total cost ($13,900.58/yr) is nearly equal to that given by GA-based optimization ($13,853.47/yr). However, the control and quality cost values corresponding to the GA- and SPSA-based solutions differ significantly. A high value for the quality cost results when (1) the mean of the quality variable (μ_y) deviates significantly from its desired magnitude, and/or (2) the corresponding standard deviation value (σ_y) is high (see Eq. 12). It is noticed in the GA-based optimization results that the σ_y value (11.48) is higher than the corresponding SPSA-based value (8.54). As a result, the product quality will exhibit greater variability, eventually leading to higher quality cost. In the case of the SPSA-based solution, it is observed that the control cost has a higher magnitude ($2,597.86/yr) as compared to the GA-based solution ($866.02/yr). By definition, the control cost is inversely proportional to the tolerance values (refer to Appendix B, Eq. B4), since smaller tolerances necessitate stricter process control, thereby increasing the cost of control. This can be verified from the tolerance values corresponding to the SPSA-based solution. It is observed that the optimized tolerances, 0.049, 0.02, and 0.004, in respect of the process variables Q, C_A^0, and T^0, are smaller as compared to those optimized by the GA (0.059, 0.031, and 0.005). Consequently, the control cost has assumed a higher value ($2,597.86/yr compared to $866.02/yr).

CSTR optimization in the presence of noisy process data

Sensors monitoring process variables and parameters often generate noisy measurements. Consequently, the mean recorded value of the noise-corrupted steady-state measurements may show a positive or negative deviation from its true mean (nominal set point). The deviation magnitude, which is variable/parameter-specific, is likely to vary from one run to another. This situation is different from the process uncertainties that are caused by the random physical variations in the process variables/parameters.

For the present case study, we consider a scenario wherein steady-state values of all the monitored process variables are corrupted, with noise obeying the Gaussian PDF. Accordingly, the steady-state values of the CSTR's design, operating, and quality variables obtained earlier by solving the phenomenological model, were corrupted using the Gaussian noise. The extent of measurement noise in each variable was assumed to lie within $\pm 5\%$ tolerance limit. Letting μ_l be the true steady-state value (nominal setpoint) of the lth process input/output variable, the corresponding standard deviation σ_l, required for generating the noisy measurements, was computed as

$$\sigma_l = \frac{0.05 \times \mu_l}{3.09}. \tag{28}$$

All seven elements of the 50 patterns representing the CSTR's noise-free steady-state input–output data set were randomly corrupted using variable-specific Gaussian mean (μ_l) and standard deviation (σ_l) values. Specifically, a time series sequence comprising one thousand noisy measurements was generated for each pattern element. The sequence obtained thereby was denoised using a nonlinear noise-reduction algorithm (Kantz and Schreiber, 1997), and the resulting sequence was averaged out. The database obtained thereby consists of 50 patterns representing noise-filtered steady-state values of the CSTR's seven input–output variables. It is worth pointing out here that even after noise-filtration, the resultant steady-state values do contain a small amount of residual noise. For creating training and test sets the noise-filtered steady-state database was normalized and partitioned in a 4:1

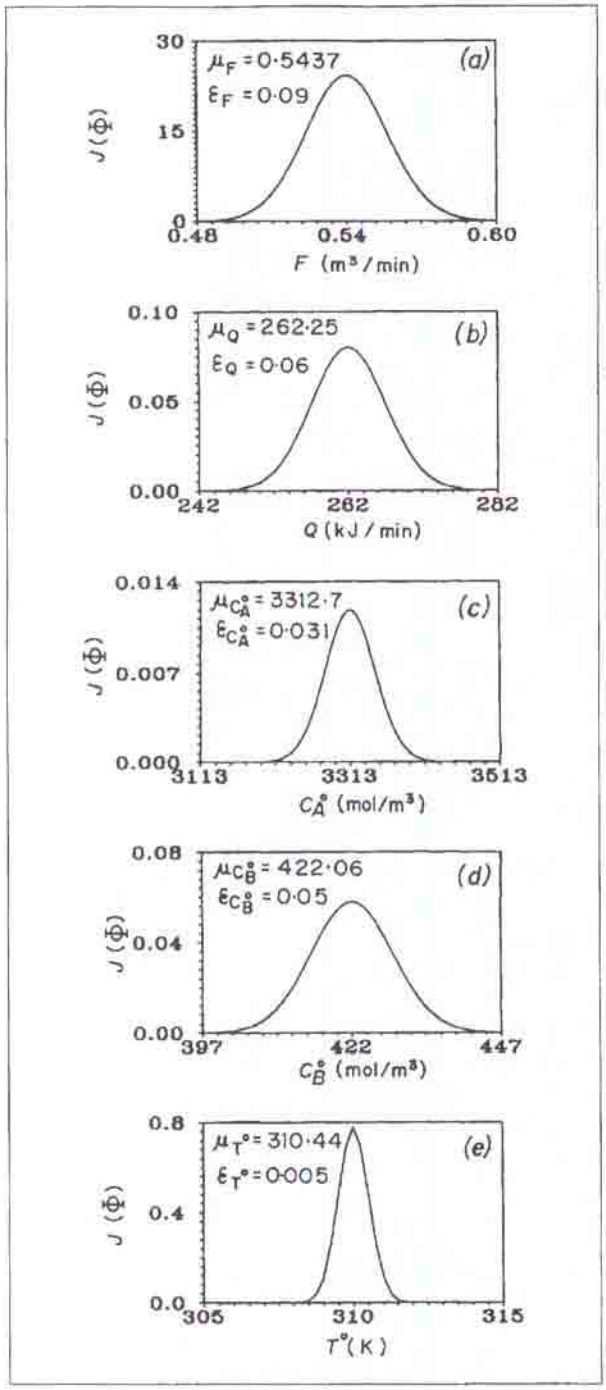

Figure 4. GA-optimized probability density functions (PDFs) corresponding to five operating variables.

ratio. Using these sets, an MLP-based optimal steady-state model was developed following the three-step training procedure elaborated earlier. The optimal network model comprising 6, 2, and 1 nodes in its input, hidden, and output layers, respectively, was trained using η_l and α_m values of 0.7 and 0.01. For this network model, the RMSE magnitude for the training set was 0.0134, and for the test set, the magnitude was 0.0123. The corresponding CC magnitudes were 0.998 (training set) and 0.996 (test set). The RMSE (CC) values are sufficiently low (high) to infer that the MLP-network has captured well the inherent relationship between the CSTR's noise-filtered input and output variables. A comparison of the E_{trn} and E_{tst} values (0.0134 and 0.0123) pertaining to the noise-reduced steady-state data with the corresponding ones (0.0061 and 0.0063) when noise-free data were used for constructing an ANN model, reveals that the former set of RMSE values is marginally higher. This indicates that the network model has fitted the noise-filtered data with marginally lower accuracy. It can be noted that the MLP-training procedure utilized in this study ensures that the network is not an overfitted one. Due to avoidance of overfitting, the network has not fitted the small amount of residual noise contained in the noise-filtered data, but instead has approximated the underlying trends (physicochemical phenomena) therein. This in turn has resulted in marginally higher RMSE values in respect of the ANN model trained on the noise-filtered steady-state data. A similar inference can also be drawn from the lower CC values in respect of the predictions made by the network trained on the noise-filtered data.

The MLP-network model just described was optimized using GA and SPSA formalisms; values of the various GA and SPSA parameters used in the respective optimization simulations were

- GA: $N_{pop} = 30$, $l_{chr} = 55$, $P_{cross} = 0.95$, $P_{mut} = 0.01$, and $N_{gen}^{max} = 250$.
- SPSA: $A = 0.08$, $r = 20$, $Z = 0.05$, $\eta = 0.602$, $\beta = 0.101$, and $t_{max} = 32{,}000$.

The optimal solutions searched by the GA and SPSA methods are presented in columns 3 and 4 of Table 2, respectively.

In the case of nonlinear objective functions, the decision surface can comprise several local minima with varying shapes and sizes. Thus, for a problem involving function minimization, it becomes important to obtain a solution that corresponds to the deepest local or global minimum on the objective function surface. Stochasticity in the implementation procedures of the GA and SPSA methodologies to some extent helps in achieving the stated goal. Nevertheless, it was ensured during GA/SPSA-implementation that the search space is thoroughly explored. This was done by using different pseudorandom number sequences (generated by changing the random number generator seed) for initializing the candidate solution population (in the GA-based optimization), and the guess solution vector (in the SPSA-based optimization). Usage of different random initializations in essence helps in exploring different subsets of the decision space, thereby locating the deepest local minimum on the decision surface. By mapping the decision surface, it is possible to verify whether the optimization algorithm has indeed captured a solution corresponding to the deepest local minimum. In the present case study, it is not possible to view the surface formed by the objective function, since the decision space is eleven-dimensional. We therefore resort to mapping the objective function in single dimension only. For such a mapping, the GA-optimized solution listed in column 3 of Table 2 has been considered. Accordingly, values of the objective function defined in Eq. 25 were evaluated by systematically

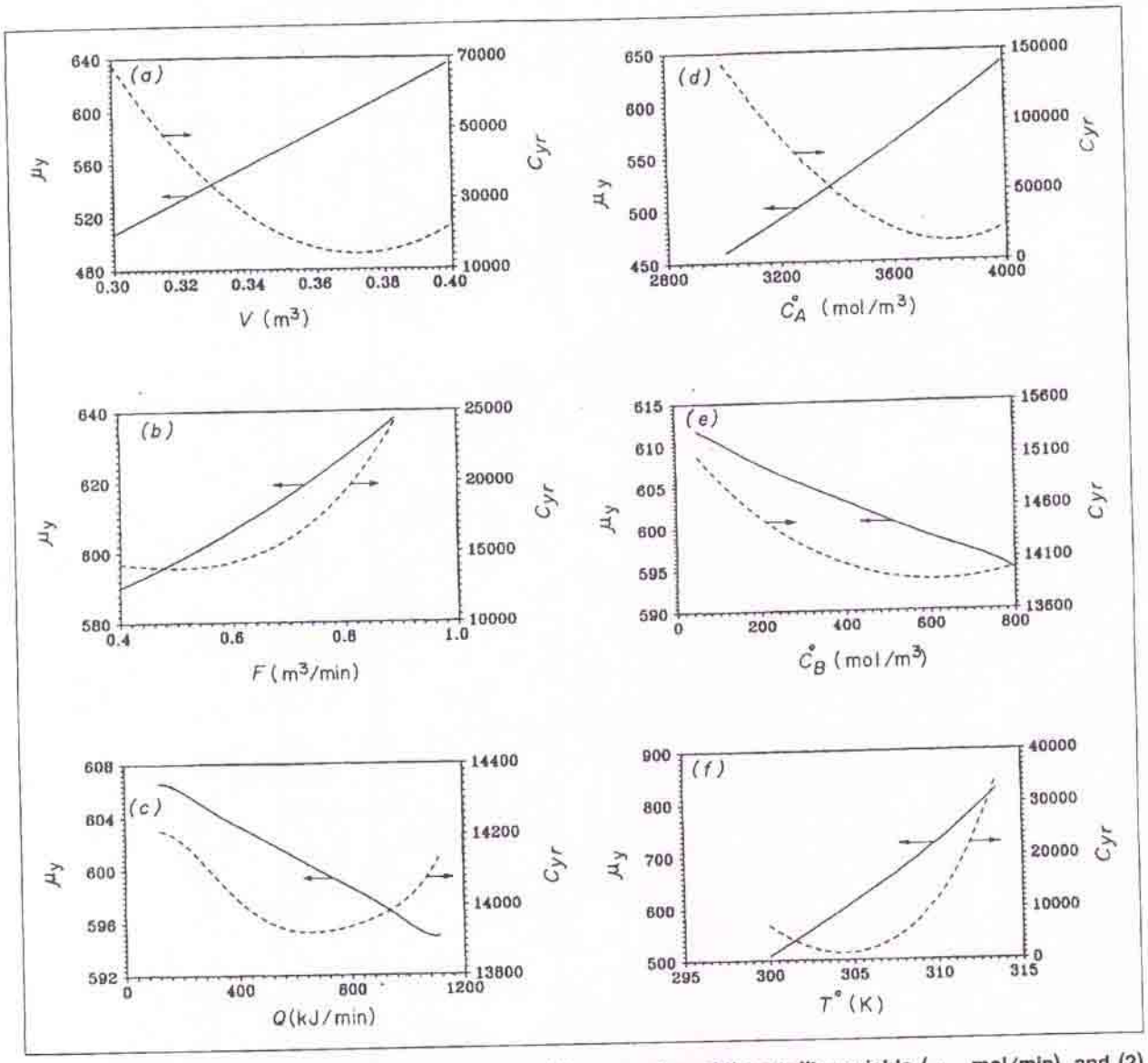

Figure 5. Effect of variation in a process variable on (1) mean value of the quality variable (μ_y, mol/min), and (2) annual plant cost (C_{yr}, \$/yr).

Panels (a)–(f) depict results corresponding to variations in V, F, Q, C_A^0, C_B^0, and T^0, respectively.

varying the magnitude of a design or an operating variable while maintaining values of the remaining ten decision variables at their optimum. The six panels in Figure 5 depict the effect of variations in V, F, Q, C_A^0, C_B^0, and T^0 on the values of C_{yr} and μ_y. It is seen in all six C_{yr} profiles that a single minimum exists and that the GA-searched solution always lies at the valley bottom. In view of the efforts made toward locating a deepest local or global minimum, it can thus be inferred that the GA was successful in fulfilling the objective.

Discussion

Upon examining the solutions given by the hybrid methodologies (listed in Table 2), it is observed that the mean values of the quality variable (599.12, 601.39, 599.85, and 600.18) are very close to their desired magnitude (600 mol/min). Following comparison with the RO solution (Table 2, column 5) obtained by Bernardo and Saraiva (1998), the annual plant costs in respect to the GA-based (\$13,853.47/yr, \$13,853.81/yr) and SPSA-based (\$13,900.58/yr, \$14,319.21/yr) solutions are a few percent lower than the corresponding RO solution value (\$14,716/yr). Such a reduction of C_{yr} was brought about either by the reduction in the control cost, C_c (see column 1 of Table 2), or by the reduction in the quality cost, C_q (see columns 2, 3, 4 of Table 2). It is noticed from the standard deviations (σ_y) of the quality variable that their magnitudes 11.48, 8.54, 11.26, and 7.0, pertaining to the solutions given by the GA and SPSA methodologies, are smaller than the corresponding RO solution value of 16.8, although the respective μ_y values deviate marginally from their desired mag-

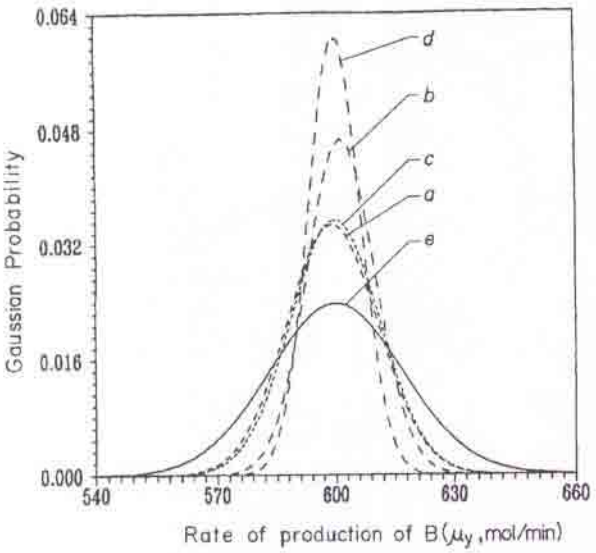

Figure 6. Comparison of PDFs pertaining to the quality variable, y.

(a) GA-optimized solution using noise-free data (μ_y = 599.12, σ_y = 11.48); (b) SPSA-optimized solution using noise-free data (μ_y = 601.39, σ_y = 8.54); (c) GA-optimized solution corresponding to the noisy process data (μ_y = 599.85, σ_y = 11.26); (d) SPSA-optimized solution corresponding to the noisy process data (μ_y = 600.18, σ_y = 7.0); and (e) RO framework solution (Bernardo and Saraiva, 1998) (μ_y = 600.0, σ_y = 16.8).

nitude of 600 mol/min. These results suggest that for the case of the CSTR, there exists a trade-off between the mean and standard deviation values of the quality variable. The nature of this trade-off can be understood from Figure 6, wherein the PDFs pertaining to the quality variable y are plotted. In the figure, the PDFs formed by the dashed lines correspond to the solutions given by the GA and SPSA methods, whereas the PDF formed by the continuous line refers to the RO solution. It is noticed in the figure that implementation of GA/SPSA-based solutions will result in the μ_y values, which are marginally different from their desired value of 600 mol/min. On the other hand, implementation of the RO solution will result in a μ_y value exactly equal to 600 mol/min. This, however, will be achieved at the cost of more widely spread steady-state values of the quality variable.

A peculiar feature of the GA and SPSA techniques, which is shared by most stochastic methods, is that the obtained solution is influenced by the random number sequence used during their implementation. As a result, multiple optimization runs, each time taking a different random number sequence (by changing the random number generator seed), were performed to obtain an overall optimal solution. It is seen from the CPU times consumed by the GA/SPSA methodologies (last row of Table 2) that the SPSA procedure consumes less time (47 and 54 s) as compared to the time taken by the GAs (80.3 and 85.5 s). These values also suggest that implementation of hybrid formalisms is not computationally burdensome, even if multiple runs need to be performed. In the case of GA-based optimization, it took 10–15 runs to arrive at the overall optimal solutions reported in Table 2, although it was noticed that the converged solution in each case was not very different. This is in contrast to the SPSA implementation, where 15–20 simulations—each time resulting in a different solution—were needed to arrive at the overall optimal solution.

Conclusions

To summarize, this article presents two process optimization strategies combining an ANN-based process model with stochastic optimization formalisms, namely, GA and SPSA. The principal advantage of using neural networks for process modeling is that the model can be developed exclusively from process input–output data without invoking process phenomenology. Having built an ANN model, its input space comprising process input variables is optimized using the GA and SPSA techniques. These optimization paradigms possess positive characteristics, such as: (1) only objective function measurements (and not the measurements of objective function derivatives) are needed in their optimization procedures, and (2) the paradigms can tolerate noisy objective functions. It is necessary to point out at this juncture that the magnitudes of various algorithmic parameters utilized in the development of the ANN models and implementation of the GA/SPSA methodologies are problem-specific and, except for a few (for instance, η and β values of the SPSA algorithm), must be selected heuristically. Notwithstanding this fact, development of ANN-based process models is still an easier and more cost-effective task compared to the development of phenomenological models. The efficacy of ANN-GA and ANN-SPSA formalisms has been demonstrated by considering a nontrivial optimization objective, which in addition to the parameter design, also addresses the issue of tolerance design. Thus, the ANN-model-based mathematical framework required for fulfilling the stated optimization objective has been formulated. A case study involving CSTR has been conducted for validating the optimization performance of the ANN-GA and ANN-SPSA strategies; the optimization objective considered was minimization of the CSTR's total annual cost. In the case study, two ANN models were developed using noise-free and noisy steady-state process data. It was observed that both the ANN models possess closely comparable data-fitting and generalization abilities. Input space of the ANN models consisting of the CSTR's design and operating variables was then optimized using the GA and SPSA methods; the tolerances associated with the operating variables were simultaneously optimized. The solutions obtained thereby have been found to compare excellently with that given by a robust deterministic optimization formalism. The ANN-GA and ANN-SPSA approaches presented here are sufficiently general, and therefore can be employed for all kinds of process design and optimization problems. These strategies become considerably simple to implement when the optimization objective involves only parameter design. In that case, tolerances defining operating windows need not be determined, thereby avoiding the usage of a sampling technique and associated numerical computations.

Acknowledgment

One of the authors (S.N.) thanks the Council of Scientific and Industrial Research (CSIR), the Government of India, New Delhi, for a Junior Research Fellowship.

Literature Cited

Agarwal, M., "A Systematic Classification of Neural-Network-Based Control," *IEEE Control Syst.*, **26**(2), 75 (1997).

Bernardo, F. P., and P. M. Saraiva, "Robust Optimization Framework for Process Parameter and Tolerance Design," *AIChE J.*, **44**, 2007 (1998).

Bhat, N., and T. McAvoy, "Use of Neural Nets for Modeling and Control of Chemical Process Systems," *Comput. Chem. Eng.*, **14**, 573 (1990).

Bishop, C. M., "Neural Networks and Their Applications," *Rev. Sci. Instr.*, **65**, 1803 (1994).

Cartwright, H. M., and R. A. Long, "Simultaneous Optimization of Chemical Flowshop Sequencing and Topology Using Genetic Algorithms," *Ind. Eng. Chem. Res.*, **32**, 2706 (1993).

Deb, K., *Optimization for Engineering Design: Algorithms and Examples*, Prentice Hall, New Delhi (1995).

Diwekar, U. M., and J. R. Kalagnanam, "Robust Design Using an Efficient Sampling Technique," *Comput. Chem. Eng.*, **20**, S389 (1996).

Diwekar, U. M., and J. R. Kalagnanam, "An Efficient Sampling Technique for Optimization Under Uncertainty," *AIChE J.*, **43**, 440 (1997a).

Diwekar, U. M., and J. R. Kalagnanam, "An Efficient Sampling Technique for Off-Line Quality Control," *Technometrics*, **39**, 308 (1997b).

Diwekar, U. M., and E. S. Rubin, "Stochastic Modeling of Chemical Processes," *Comput. Chem. Eng.*, **15**, 105 (1991).

Diwekar, U. M., and E. S. Rubin, "Parameter Design Methodology for Chemical Processes Using a Simulator," *Ind. Eng. Chem. Res.*, **33**, 292 (1994).

Freeman, J. A., and D. M. Skapura, *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley, Reading, MA (1991).

Garcia, S., and E. P. Scott, "Use of Genetic Algorithms in Thermal Property Estimation: I. Experimental Design Optimization," *Numer. Heat Transfer* (Part A), **33**, 135 (1998).

Garcia, S., J. Guynn, and E. P. Scott, "Use of Genetic Algorithms in Thermal Property Estimation: II. Simultaneous Estimation of Thermal Properties," *Numer. Heat Transfer* (Part A), **33**, 149 (1998).

Garrard, A., and E. S. Fraga, "Mass Exchange Network Synthesis Using Genetic Algorithms," *Comput. Chem. Eng.*, **22**, 1837 (1998).

Gill, P. E., W. Murray, M. A. Saunders, and M. H. Wright, "User's Guide for NPSOL: A Fortran Package for Nonlinear Programming," Tech. Rep. SOL 86-2, Systems Optimization Laboratory, Stanford University, Stanford, CA (1986).

Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA (1989).

Hanagandi, V., H. Ploehn, and M. Nikolaou, "Solution of the Self-Consistent Field Model for Polymer Adsorption by Genetic Algorithms," *Chem. Eng. Sci.*, **51**, 1071 (1996).

Hecht-Nielsen, R., *Neurocomputing*, Addison-Wesley, Reading, MA (1990).

Henson, M. A., "Nonlinear Model Predictive Control: Current Status and Future Directions," *Comput. Chem. Eng.*, **23**, 187 (1998).

Hernandez, E., and Y. Arkun, "Study of the Control-Relevant Properties of Back-Propagation Neural Network Models of Nonlinear Dynamical Systems," *Comput. Chem. Eng.*, **4**, 227 (1992).

Holland, J. H., *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor (1975).

Hornik, K., M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, **2**, 359 (1989).

Hugget, A., P. Sebastin, and J. P. Nadeau, "Global Optimization of a Dryer by Using Neural Networks and Genetic Algorithms," *AIChE J.*, **6**, 1227 (1999).

Hunt, K., D. Sbarbaro, R. Zbikowski, and P. Gawthrop, "Neural Networks for Control Systems—A Survey," *Automatica*, **28**, 1083 (1992).

Kantz, H., and T. Schreiber, *Nonlinear Time Series Analysis*, Cambridge Univ. Press, Cambridge, U.K. (1997).

Nahas, E., M. Henson, and D. Seborg, "Nonlinear Internal Model Strategy for Neural Network Models," *Comput. Chem. Eng.*, **16**, 1039 (1992).

Narendra, K., and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. Neural Networks*, **1**, 4 (1990).

Pistikopoulos, E. N., "Uncertainty in Process Design and Operations," *Comput. Chem. Eng.*, **19**, S553 (1995).

Pistikopoulos, E. N., and M. G. Ierapetritou, "Novel Approach for Optimal Process Design Under Uncertainty," *Comput. Chem. Eng.*, **19**, 1089 (1995).

Poggio, T., and F. Girosi, "Regularization Algorithms for Learning that are Equivalent to Multilayer Networks," *Science*, **247**, 978 (1990).

Polifke, W., W. Geng, and K. Dobbeling, "Optimization of Rate Coefficients for Simplified Reaction Mechanisms with Genetic Algorithms," *Combust. Flame*, **113**, 119 (1998).

Ramasamy, S., S. S. Tambe, B. D. Kulkarni, and P. B. Deshpande, "Robust Nonlinear Control with Neural Networks," *Proc. R. Soc. Lond. A.*, **449**, 655 (1995).

Rumelhart, D., G. Hinton, and R. Williams, "Learning Representations by Backpropagating Errors," *Nature*, **323**, 533 (1986).

Spall, J. C., "A Stochastic Approximation Technique for Generating Maximum Likelihood Parameter Estimates," *Proc. of the Amer. Cont. Conf.*, AACC, Evanston, IL, p. 1161 (1987).

Spall, J. C., "Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization," *IEEE Trans. Aerosp. Electron. Syst.*, **AES-34**, 817 (1998a).

Spall, J. C., "An Overview of the Simultaneous Perturbation Method for Efficient Optimization," *Johns Hopkins APL Tech. Dig.*, **19**, 482 (1998b).

Taguchi, G., *Introduction to Quality Engineering*, Amer. Supplier Inst., Dearborn, MI (1986).

Tambe, S. S., B. D. Kulkarni, and P. B. Deshpande, *Elements of Artificial Neural Networks with Selected Applications in Chemical Engineering, and Chemical & Biological Sciences*, Simulation & Advanced Controls, Louisville, KY (1996).

Tendulkar, S. B., S. S. Tambe, I. Chandra, P. V. Rao, R. V. Naik, and B. D. Kulkarni, "Hydroxylation of Phenol to Dihydroxybenzenes: Development of Artificial Neural-Network-Based Process Identification and Model Predictive Control Strategies for a Pilot Plant Scale Reactor," *Ind. Eng. Chem. Res.*, **37**, 2081 (1998).

Venkatasubramanian, V., and A. Sundaram, "Genetic Algorithms: Introduction and Applications," *Encyclopedia of Computational Chemistry*, Wiley, Chichester, U.K. (1998).

Wang, I. J., and J. C. Spall, "A Constrained Simultaneous Perturbation Stochastic Approximation Algorithm Based on Penalty Functions," *Proc. of the Amer. Cont. Conf.*, AACC, Evanston, IL, p. 393 (1999).

Appendix A: CSTR Model Equations and Parameter Values

The steady-state model for the nonisothermal CSTR, wherein two first-order reactions, $A \rightarrow B \rightarrow C$, take place, is given below. While Eq. A1 refers to the energy balance, the remaining two equations account for the material balances of components A (Eq. A2) and B (Eq. A3), respectively:

$$\rho c_p F(T^0 - T) + k_{A_0}\exp(-E_A/RT)\hat{C}_A(-H_1)V + k_{B_0}\exp(-E_B/RT)\hat{C}_B(-H_2)V - Q = 0 \quad \text{(A1)}$$

$$F\left(C_A^0 - \hat{C}_A\right) - k_{A_0}\exp(-E_A/RT)\hat{C}_A V = 0 \quad \text{(A2)}$$

$$F\left(C_B^0 - \hat{C}_B\right) + k_{A_0}\exp(-E_A/RT)\hat{C}_A V - k_{B_0}\exp(-E_A/RT)\hat{C}_B V = 0. \quad \text{(A3)}$$

For generating representative steady-state data comprising 50 input-output patterns, the following parameter values were considered: $E_A = 3.64 \times 10^4$ J/mol, $E_B = 3.46 \times 10^4$ J/mol, $k_{A_0} = 8.4 \times 10^5$ min^{-1}, $k_{B_0} = 7.6 \times 10^4$ min^{-1}, $H_1 = -2.12 \times 10^4$ J/mol, $H_2 = -6.36 \times 10^4$ J/mol, $\rho = 1.180$ kg/m^3,

$c_p = 3.2\times10^3$ J/(kg·K), and $R = 8.314$ J/(mol·K). The values of other model parameters, namely, V, F, Q, C_A^0, C_B^0, and T^0, describing design and operating variables were randomly chosen within the ranges specified in the main text.

Appendix B: Functions Used for Evaluating Components of the Annual Plant Cost (C_{yr})

Here, only the final expressions used for computing various CSTR costs have been given. For more details, the reader may refer to Bernardo and Saraiva (1998).

1. The equipment cost C_{eqp} is computed by using volume (V, m^3), as given by:

$$C_{eqp}(\$/yr) = 4{,}199.55\left(\frac{V}{\pi}\right)^{0.6227}. \qquad \text{(B1)}$$

2. The operating cost (C_{op}) includes the utility cost (C_{util}) and the pumping cost (C_{pump}). The C_{util} value is calculated using heat recovery rate, Q (J/min) and Q_N ($=2.54\times10^7$ J/min), according to

$$C_{util}(\$/yr) = 1.145\left[7{,}896 - 6{,}327(Q/Q_N) + 4.764\times10^4(Q/Q_N)^2 - 1.022\times10^4(Q/Q_N)^4\right], \qquad \text{(B2)}$$

and C_{pump} is evaluated using the flow rate (F, m^3/min), as given by

$$C_{pump}(\$/yr) = 13.8831(264.2F)^{0.8050}. \qquad \text{(B3)}$$

3. The overall control cost (C_c) is obtained by summing the control cost contributions of five operating variables (see Eq. 15). Using $a = 143.2$ and $b = 1.736$, C_c has been evaluated as

$$C_c(\$/yr) = 5a + b\left(\frac{\mu_F}{\sigma_Q} + \frac{\mu_Q}{\sigma_F} + \frac{\mu_{C_A^0}}{\sigma_{C_A^0}} + \frac{\mu_{C_B^0}}{\sigma_{C_B^0}} + \frac{\mu_{T^0}}{\sigma_{T^0}}\right)$$

$$= 716 + \left[1.736\times3.09\left(\frac{x_2}{x_7} + \frac{x_3}{x_8} + \frac{x_4}{x_9} + \frac{x_5}{x_{10}} + \frac{x_6}{x_{11}}\right)\right], \qquad \text{(B4)}$$

where μ_n and σ_n refer to the mean and standard deviation of the PDF pertaining to the nth operating variable.

4. The quality cost has been computed using Taguchi loss function (Taguchi, 1986) given as

$$C_q(\$/yr) = k_l\left[(\mu_y - y^*)^2 + \sigma_y^2\right], \qquad \text{(B5)}$$

where μ_y and σ_y denote the mean and standard deviation, respectively, of N_{obs} number of quality variable values, $\{y\}$, obtained using the ANN-based CSTR model; y^* refers to the desired value (600 mol/min) of the quality variable, y; and k_l ($=6.536$) is the loss coefficient.

Manuscript received Jan. 5, 2000, and revision received June 7, 2000.