Genetic Programming Assisted Stochastic Optimization Strategies for Optimization of Glucose to Gluconic Acid Fermentation

Jitender Jit Singh Cheema, Narendra V. Sankpal, Sanjeev S. Tambe, and Bhaskar D. Kulkarni*

Chemical Engineering Division, National Chemical Laboratory, Pune 411008, India

This article presents two hybrid strategies for the modeling and optimization of the glucose to gluconic acid batch bioprocess. In the hybrid approaches, first a novel artificial intelligence formalism, namely, genetic programming (GP), is used to develop a process model solely from the historic process input-output data. In the next step, the input space of the GP-based model, representing process operating conditions, is optimized using two stochastic optimization (SO) formalisms, viz., genetic algorithms (GAs) and simultaneous perturbation stochastic approximation (SPSA). These SO formalisms possess certain unique advantages over the commonly used gradient-based optimization techniques. The principal advantage of the GP-GA and GP-SPSA hybrid techniques is that process modeling and optimization can be performed exclusively from the process input-output data without invoking the detailed knowledge of the process phenomenology. The GP-GA and GP-SPSA techniques have been employed for modeling and optimization of the glucose to gluconic acid bioprocess, and the optimized process operating conditions obtained thereby have been compared with those obtained using two other hybrid modeling-optimization paradigms integrating artificial neural networks (ANNs) and GA/SPSA formalisms. Finally, the overall optimized operating conditions given by the GP-GA method, when verified experimentally resulted in a significant improvement in the gluconic acid yield. The hybrid strategies presented here are generic in nature and can be employed for modeling and optimization of a wide variety of batch and continuous bioprocesses.

1. Introduction

Gluconic acid is used as the metal supplement of calcium, iron, etc. in pharmaceuticals and as an acidulent in the food industry. It also finds applications as a biodegradable chelating agent, filler, metal cleaner, dye stabilizer, and in the textile industry for removing instructions. Commercially, gluconic acid is produced mainly via two biochemical routes, although more expensive chemical and electrochemical routes are also available. The biochemical routes primarily involve free-cell fermentation and immobilized enzyme (glucose oxidase, GOD, of *Aspergillus niger* and *Gluconobacter*)-based bioconversion of glucose (Roehr et al., 1996). Here, the GOD oxidizes glucose to glucono-*d*-lactone, which is hydrolyzed spontaneously by the enzyme lactonase to the gluconic acid. Producing gluconic acid by immobilized enzymes is a costly and cumbersome process owing to the difficulties in the immobilization and separation steps; additional difficulties are encountered as a result of denaturization of the enzyme (Rao et al., 1994). In the free-cell fermentation, the mycelia are subjected to various mass and heat-transfer stresses (Karel and Robertson, 1989). Although mechanical agitation helps in alleviating these limitations, it induces a turbulent flow, which may lead to cell disintegration (Wittler et al., 1986), cell breakage or surface erosion (Taguchi et al., 1968), and pellet breakage (Metz and Kossen, 1994). Consequently, a spontaneous or gradual loss in the cellular activity may be witnessed. In comparison, fermentation by cells immobilized on a support matrix under submerged conditions is a cost-effective and an efficient method of gluconic acid production (Sankpal and Sahasrabuddhe, 2001).

Recently, a new batch fermentation technique has been proposed for the production of gluconic acid from glucose wherein *A. niger* immobilized on a support matrix consisting of a cellulosic fabric has resulted in higher yields (Sankpal et al., 1999). The improved overall productivity from this technique is primarily due to the enhanced interaction between the dissolved oxygen and the fungal mycelia. Enhancement in the said interaction is effected via a continuous substrate dripping mechanism (see Figure 1) and not by the mechanical agitation as used in the free-cell fermentation. Our objective in this paper is to develop a mathematical model of the new glucose to gluconic acid batch fermentation process and also to obtain the optimized process conditions leading to the enhanced gluconic acid yield. For developing the fermenter model, experimental data incorporating the effects of the substrate (glucose) and biomass concentrations, and the dissolved oxygen content, have been used.

It is seen in Figure 1 that the glucose to gluconic acid bioconversion using *A. niger* immobilized on the cellulosic microfibrils involves complicated reaction and mass-transfer phenomena. Development of a phenomenological ("first principles") process model has therefore become a difficult task since the physicochemical phenomena underlying the bioconversion and the associated kinetic and transport mechanisms are not well-understood. Also, it has been observed that the process dynamics is nonlinear,

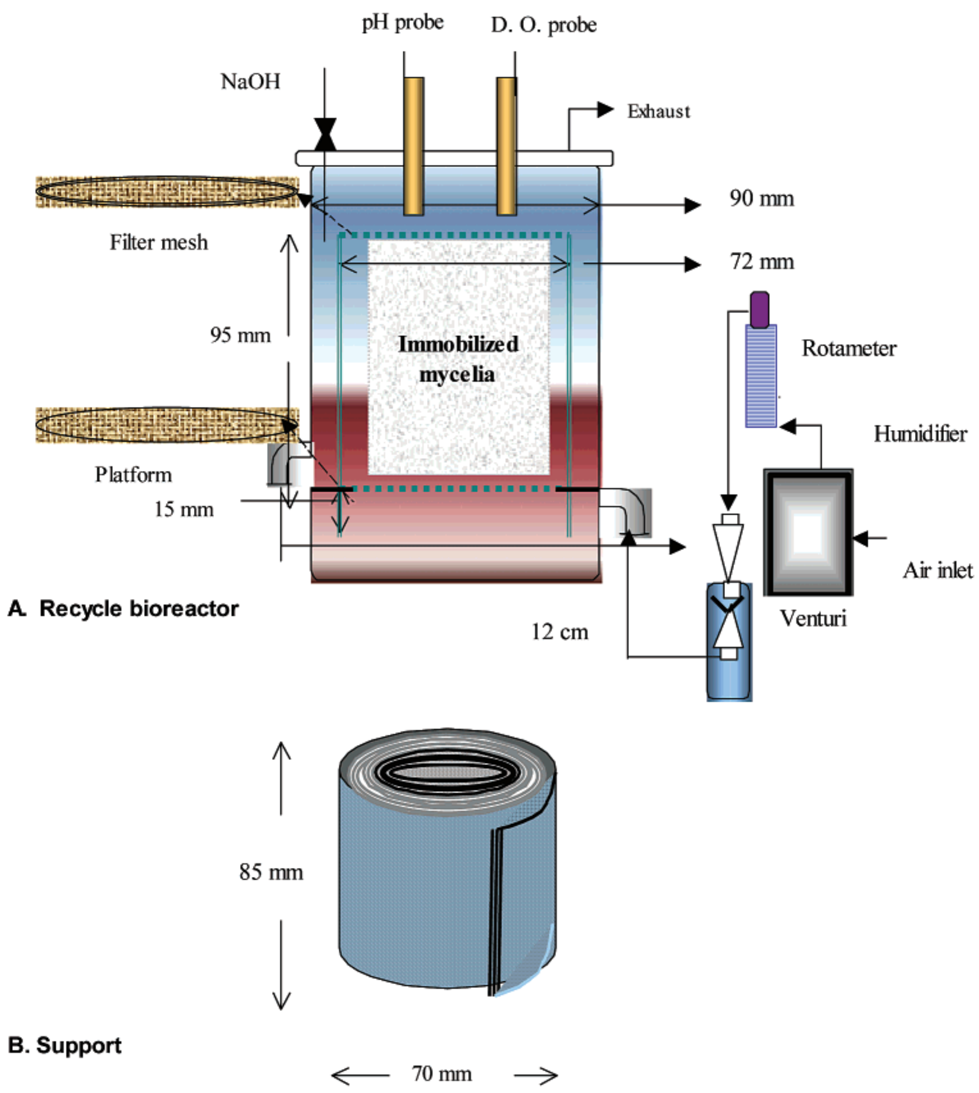

Figure 1. Schematic diagram of batch fermentation setup.

which has made the modeling task even more complex. In view of these difficulties, a novel artificial intelligence based paradigm, namely, genetic programming (GP) (Koza, 1992; Szpiro, 1997; Nandi et al., 2000; Yadavalli et al., 1999; Sankpal et al., 2001a) has been employed here for modeling the fermenter. The principal advantage of the GP-formalism is that it automatically arrives at an empirical closed-form mathematical model relating process inputs and outputs exclusively from the historic process input-output data. Consequently, the detailed knowledge of the process phenomenology (reaction kinetics and mass transfer mechanisms) is not necessary in the GP-based process modeling. Toward improving the fermenter performance, in the next step, the input space of the GP-based model has been optimized using two stochastic optimization (SO) paradigms, namely, genetic algorithms (GAs) and simultaneous perturbation stochastic approximation (SPSA) (Spall, 1987). Specifically, three fermenter operating variables, viz., initial glucose concentration, dissolved oxygen concentration, and biomass concentration, have been optimized with a view to increase the gluconic acid yield. The sketch summarizing the GP-based modeling and the GA-/SPSA-based optimization of the GP model is portrayed in Figure 2. The choice of GA and SPSA methods for optimizing the GP-based model stems from their ability in solving complex optimization problems involving multimodal and noisy objective functions. The SO methods differ from the widely used deterministic gradient-based optimization methods in that they involve a random component at some stage(s) in their implementation procedure. This randomness helps in escaping from a locally optimum solution and thereby reaching the globally optimum solution. 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 the higher order derivatives) of the objective function (Spall, 1998a,b). The GA formalism possesses an added advantage that, unlike most deterministic gradient-based methods, it can operate successfully even when the objective function to be maximized or minimized is not smooth, continuous, or differentiable.

Hereafter, the two hybrid strategies integrating GP-based process modeling and GA-/SPSA-based optimization will be referred to as GP-GA and GP-SPSA, respectively. The optimized fermenter operating conditions obtained using the GP-GA and GP-SPSA formalisms have been compared with those obtained using two other hybrid modeling-optimization methodologies, namely, ANN-GA (Nandi et al., 2001; 2002) and ANN-SPSA (Nandi et al., 2001). In these methods, a feed-forward artificial neural network (ANN) is first used for developing a black-box model for correlating process inputs and

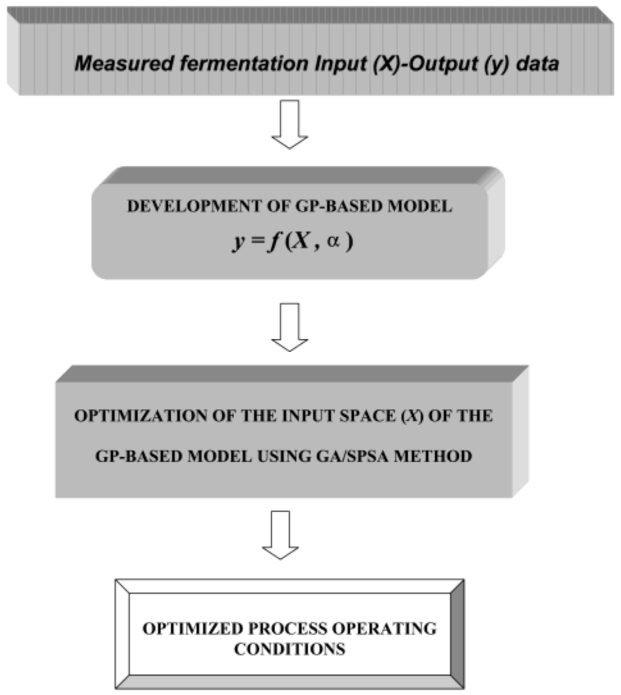
Measured fermentation Input (X)-Output (y) data
DEVELOPMENT OF GP-BASED MODEL
$y = f(X, \alpha)$
OPTIMIZATION OF THE INPUT SPACE (X) OF THE GP-BASED MODEL USING GA/SPSA METHOD
OPTIMIZED PROCESS OPERATING CONDITIONS

Figure 2. Summary sketch of the hybrid modeling-optimization strategy.

outputs. Next, the input space of the ANN model is optimized, as in the GP-GA and GP-SPSA strategies, using GA and SPSA methods, respectively. Replacing the GP-based model with an ANN-based model in Figure 2 leads to the ANN-GA and ANN-SPSA hybrid formalisms. The optimal conditions obtained using the GP-GA, GP-SPSA, ANN-GA, and ANN-SPSA hybrid methodologies have been compared, and the best solution (as given by the GP-GA method) was subjected to experimental verification. The results of verification experiment suggest that the optimized fermenter operating conditions have indeed succeeded in improving the gluconic acid yield.

This paper is organized as follows. In section 2, the conceptual framework of the GP formalism and its implementation details are provided. Next, in section 3 the GA and SPSA optimization formalisms have been described. A brief description of the ANN-based modeling and optimization of the ANN model using GA and SPSA methods is provided in section 4. Finally, section 5 presents the numerical results of the optimization of the GP-based model using GA and SPSA methods, respectively. This section also compares the optimized conditions obtained using the ANN-GA and ANN-SPSA hybrid formalisms and describes the results of the experimental validation of the overall optimum operating conditions.

2. Genetic Programming Based Process Modeling

The GP formalism (Koza, 1992; Kinnear, 1994) is closely related to the genetic algorithms, which are stochastic function optimization techniques. Both approaches are based on the principles of natural selection and genetics followed by the biologically evolving species. Given an objective function, the GAs can efficiently and robustly obtain the optimal values of the decision variables that would maximize or minimize the function. The GP technique, though it uses the same principles as employed by the GAs, performs symbolic regression. It is a process of discovering both the functional form of a data-fitting function and all of its necessary coefficients (parameters) or at least an approximation to these. Thus, it is seen that the GP technique is capable of automatically obtaining the mathematical equation that fits a given set of process input-output data. In contrast, the GAs search and optimize values of the decision variables appearing in the prespecified objective function. In earlier applications of GP, the technique has been used for process identification (Kulkarni et al., 1999), dynamic modeling (McKay et al., 1997), time series prediction (Szpiro, 1997), and steady-state modeling of the FCC reactor (Nandi et al., 2000). It may be noted that the GP-based models are good at interpolation but not so good at extrapolation. This weakness, which is shared by all nonlinear models obtained exclusively from the process input-output data, can be overcome by gathering more data in the regions where extrapolation is desired. In the present study, the GP formalism has been used to obtain a model predicting the fermentation product concentration as a function of the fermenter operating conditions. The general form of the model to be obtained is given as

$$y = f(X,\alpha) \qquad (1)$$

where y denotes the process output variable (product concentration); X refers to the N-dimensional vector of process operating conditions ($X = [x_1, x_2, ..., x_n, ..., x_N]^T$), and f represents a nonlinear function whose parameters are defined in terms of a K-dimensional vector, α ($= [\alpha_1, \alpha_2, ..., \alpha_k, ..., \alpha_K]^T$). Given process data comprising values of the operating variables $\{X\}$ and the corresponding values of the process output $\{y\}$, the task of GP is to obtain the best fitting functional form, f, and its parameter vector, α.

The GP procedure begins by creating a random population of strings (chromosomes) representing candidate solutions to the data fitting problem defined in eq 1; each string in the population represents a different mathematical data fitting function. A string consists of multiple symbolically coded building blocks that, when combined together, form a complete mathematical expression. A block includes operands and mathematical operators representing addition, subtraction, multiplication, and division. Additional operators such as logarithm, sine, and exponentiation, although not used in this study, may also be considered. The methodology for coding a mathematical expression in the form of a symbolic string is as follows. Two arguments, i.e., operands, are selected randomly, and a randomly chosen arithmetic operator is placed between them; the resulting expression is enclosed in parentheses. The operands can be either real numbers (elements of the parameter vector α) or the process operating variables, $\{x_n\}$, represented using suitable symbols. An argument can also be a small (sub)expression in itself comprising function parameters, operating variables, and operators. Such arguments are also bracketed within parentheses. The complete string that emerges is composed of multiple building blocks; for example, a five-block string may be obtained as

$$[(B_1 \oplus B_2) \oplus (B_3 \oplus B_4 \oplus B_5)] \qquad (2)$$

where the contents of blocks B_1 to B_5 either represent a parameter, a process operating variable, or a subexpression comprising both; the symbol $\oplus$ denotes an arithmetic operator. In this equation coding strategy, handling of numerous parentheses that enclose various blocks becomes cumbersome. This difficulty can be overcome by using an expression coding scheme known as “post-fix” representation (Kulkarni et al., 1999; Yadavalli et al., 1999). In this scheme, the operator follows operands. For instance, an expression $[(B_1 + B_2) \div B_3]$, when written using the post-fix scheme, becomes $[B_1\ B_2\ B_3 \div +]$. It can thus be noted that parentheses are implied but need not be specified explicitly, which greatly simplifies the computer programming effort. The detailed stepwise GP procedure is given in (Nandi et al., 2000).

Step 1 (*Initialization*): Set the generation index N^P to zero and randomly create an initial population of N_p number of strings. Each string can have up to l^P_{chr} number of elements representing operators and operands. The number of blocks within a string and the respective block sizes are also chosen randomly. Since a block of u operands require $(u - 1)$ operators between them, a total of $(2u - 1)$ elements are present in the block. Similarly, u_B number of blocks require $(u_B - 1)$ number of operators between them to define an expression string. The initial population of strings describing candidate equations is formed by filling the blocks randomly with operands (process operating variables and parameters) and mathematical operators using the post-fix representation scheme; the numerical values of the parameters appearing in an expression are chosen randomly from the uniformly distributed interval, $[-Z, +Z]$. The mathematical operators and process operating variables are represented using large integer numbers as symbols. Here, care must be exercised that the chosen integers do not fall in the same range, $[-Z, +Z]$, as used to select the parameter values, $\{\alpha_k\}$.

Step 2 (*Ranking*): Evaluate the fitness value of the jth population string ($j = 1, 2, ..., N_p$) using a prespecified fitness function, $\hat{R}$. This function measures the data fitting ability of the expressions represented by the population strings. String fitness is evaluated as follows: given the training set, $\{X_i, y_i\}$, $i = 1, 2, ..., N_{trn}$, comprising N_{trn} number of pairs defining the operating condition vectors, $\{X_i\}$, and the corresponding process outputs, $\{y_i\}$, the mathematical expression (model) represented by each string in the current population is used to compute the model predicted value of the process output variable, y. The procedure can be described as

$$y^{prd}_{i,j} = f_j(X_i, \alpha); \quad j = 1, 2, ..., N_p; i = 1, 2, ..., N_{trn} \quad (3)$$

where $y^{prd}_{i,j}$ refers to the output value evaluated using the mathematical expression, f_j, represented by the jth string. A sum-squared-error (SSE) dependent fitness function of the following form may then be used to compute the fitness, $\hat{R}^2_j$, of the jth string:

$$\hat{R}_j^2 = \frac{1}{1 + \Delta_j^2}; \quad j = 1, 2, ..., N_p \quad (4)$$

where Δ^2_j refers to the SSE, which is computed as

$$\Delta_j^2 = \sum_{i=1}^{N_{trn}} (y_i - y^{prd}_{i,j})^2 \quad (5)$$

where y_i refers to the desired (target) process output value from the training set. After evaluating fitness values of all the N_p strings in the current population, the strings are ranked in the decreasing order of their fitness scores.

Step 3 (*Selection*): From the ranked population, select pairs of the parent strings for creating a mating pool; an efficient scheme termed "elitist mating" (Yadavalli et al., 1999) may be utilized for selecting the parent pairs. According to this scheme, $(N_p/2) + 1$ number of top ranking strings are paired in the following manner to obtain the mating pool: [*string*-1 and *string*-2], [*string*-2 and *string*-3], ..., [*string*-$(N_p/2)$ and *string*-$(N_p/2 + 1)$]. The purpose of allowing mating only among the fitter strings is that they are most likely to contain parts of an optimally fitting expression.

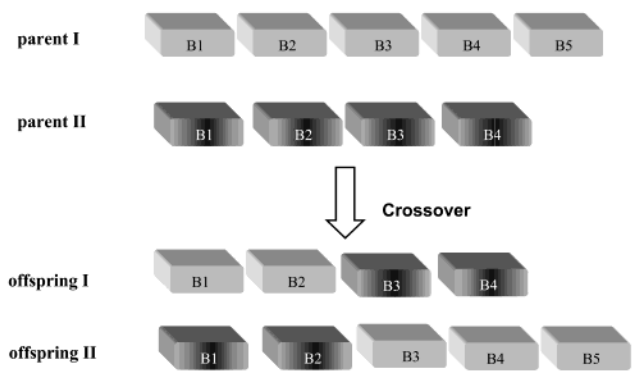

Figure 3. Genetic programming crossover mechanism illustrated using string blocks.

Step 4 (*Reproduction and Crossover*): This is a crucial GP step wherein four offspring strings are formed from each parent pair. The first two offspring are copies of the parent strings, and the other two offspring are generated by crossing over the contents of the parent strings. In the crossover operation, a crossover point is selected randomly along the block sequence of each parent string followed by cutting the respective sequences at that point. The second (first) sliced portions of the parent strings are then mutually exchanged and combined together with the respective first (second) portions to produce two new offspring strings (see Figure 3). Usually, the crossover operation is carried out using a high crossover probability (P^P_{cr}) value (range 0.95–1.0). The reproduction and crossover operations when repeated on the $N_p/2$ number of parent pairs generate $2 \times N_p$ number of offspring strings.

Step 5 (*Mutation*): This operation maintains population diversity and broadens the search for good data-fitting models. Mutation is conducted using a small probability value (P^P_{mut}) for its occurrence (range 0.01–0.05), and it involves replacement of a randomly chosen string element by another of the same type, i.e., an operator (operand) replaces another operator (operand) (see Figure 4). The population of strings obtained following mutation operation represents the new generation of candidate solutions to the data-fitting problem and thus $N^P = N^P + 1$.

Step 6 (*Termination*): Repeat steps 2–5 iteratively till one of the two stopping criteria is satisfied: (i) the GP has evolved over a prespecified number of generations ($N^P = N^P_{max}$), and (ii) the fitness of the best string does not increase over sufficiently large (say 500) number of generations.

The essence of the above-described GP procedure is that it progressively alters the structure of mathematical equations contained in a population with the objective of improving their fitness. It may be noted that since creation of initial population, crossover, and mutation are performed stochastically (randomly), the final solution given by the GP method depends on the sequence of uniformly distributed random numbers used in performing the stated operations. Thus to arrive at an overall optimal function possessing highest fitness value, it

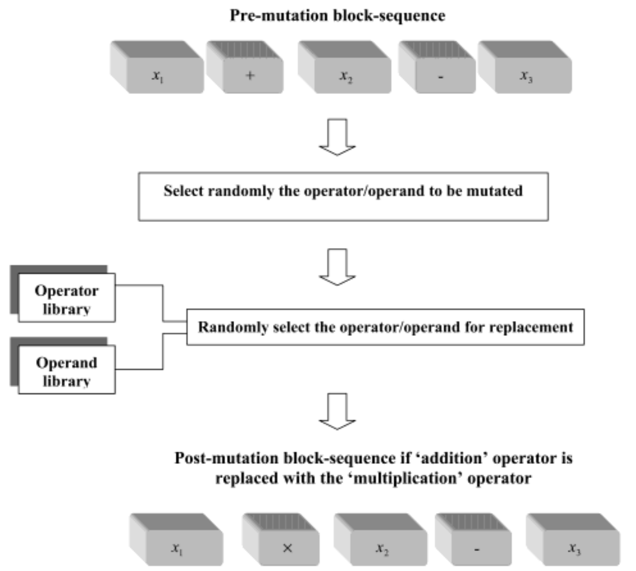
Pre-mutation block-sequence
x_1
+
x_2
-
x_3
Select randomly the operator/operand to be mutated
Operator library
Operand library
Randomly select the operator/operand for replacement
Post-mutation block-sequence if 'addition' operator is replaced with the 'multiplication' operator
x_1
×
x_2
-
x_3

Figure 4. Mutation mechanism illustrated using a block sequence.

becomes necessary to repeat the GP runs a number of times by using each time a different seed value for the random number generator. Usage of different seed values generate nonsimilar random number sequences that help in locating the globally optimal solution.

Since the fitness of a population string is evaluated using the training data set, it becomes essential to test how well the model represented by that string performs on a data set different from the training set. Such a testing known as cross-validation is performed using a validation data set. Specifically, the entire data available for the model building is partitioned into two sets, namely, training and test sets, and in each GP iteration the fitness of a string is evaluated using these sets separately. The model that produces highest fitness ($\hat{R}^2$) value for the test set is taken as an overall optimal data fitting model (f^*). Once a properly cross-validated optimal process model is secured, its parameters, $\{\alpha_k\}$, can be refined further by using a standard nonlinear regression technique such as Marquardt's algorithm (Marquardt, 1963). Such a refinement, if possible, improves the prediction accuracy of the GP-based process model.

3. Optimization of GP-Based Model Using Stochastic Optimization Formalisms

Having developed an optimal GP-based model, f^*, in the next phase the model's input space representing process operating variables (X) is optimized separately using the GA and SPSA formalisms, wherein the optimization objective is maximization of the fermentation yield. Accordingly, the optimization objective under consideration is defined as

$$\text{Maximize } y_{gl} = H(y) = H[f^*(X,\alpha)]; \quad x_n^L < x_n < x_n^M,\ n = 1, 2, ..., N \tag{6}$$

where y_{gl} denotes the gluconic acid yield to be maximized, H denotes a function of y, and, x_n^L and x_n^M refer to, respectively, the lower and upper bounds for the nth process operating (decision) variable. The maximization problem defined above assumes that the nonlinear function, f^*, and its parameter vector, α, are known. In the following, implementation details of the GA and SPSA methods for maximizing y_{gl} are described.

3.1. Genetic Algorithms. Genetic algorithms developed by Holland (1975) (also see Goldberg, 1989; Davis, 1991; Deb, 1995) is a nonlinear search and optimization technique based on the mechanisms of natural selection and genetics. Analogous to the GP technique, GAs also use selection, reproduction, crossover, and mutation steps. However, GAs employ these steps with an objective of obtaining a solution that maximizes the prespecified objective function, $H(y)$. Details of GAs can be found at numerous places (e.g., Deb, 1995; Venkatasubramanian and Sundaram, 1998; Ramanathan et al., 2001; Mukherjee et al., 2001) and therefore only a brief description of their implementation is provided below.

In the GA procedure, a random population of N_A number of strings is created. The strings may be coded either using binary digits or real numbers. In binary coding (as used in this study), each string containing l_{chr}^A number of bits is divided into N segments where an nth ($n = 1, 2, ..., N$) segment of length l_n^A represents the binary representation of the nth decision variable. The decimal equivalent, x_n, of the nth binary segment is evaluated as

$$x_n = x_n^L + \frac{(x_n^U - x_n^L)S_n}{2^{l_n{}^A} - 1}; \quad n = 1, 2, ..., N; \quad \sum_{n=1}^{N} l_{chr}^A = l_n^A \tag{7}$$

where S_n represents the decimal value of the nth binary segment comprising l_n^A bits. Upon decoding all the N_A strings in this manner, their fitness values, $\{R^A\}$, are evaluated using a prespecified fitness function. Next, the string population is subjected to the actions of four genetic operators, namely, selection, reproduction, crossover, and mutation, to obtain a new generation of candidate solutions. The actions of these GA operators are repeated with successive generations of solutions till convergence is achieved. The entire GA-implementation can now be summarized as follows:

Step 1 (*Initialization*): Set the generation index N_{gen}^A to zero and randomly generate N_A number of binary strings. Each string contains l_{chr}^A bits and is divided into N segments.

Step 2 (*Fitness computation*): Decode jth ($j = 1, 2, ..., N_A$) binary string to obtain the corresponding decimal values of the decision variables, x_{jn}, $n = 1, 2, ..., N$ (see eq 7), and evaluate the fitness (R_j^A) of the jth string as given by

$$R_j^A = H(y_j) = H[f^*(X_j,\alpha)] \tag{8}$$

where X_j refers to the real-valued decision variable vector, $X_j = [x_{j1}, x_{j2}, ..., x_{jN}]^T$. After computing fitness values of all the N_A strings in the current population, the strings are ranked in the decreasing order of their fitness values.

Step 3 (*Selection of parents*): From the current population, form a mating pool comprising N_A number of parent strings using a suitable parent selection scheme such as the Roulette-Wheel (RW) method, tournament selection, and the stochastic remainder selection (Goldberg, 1989). According to these schemes, strings with higher fitness values have a better chance of entering the mating pool.

Step 4 (*Crossover*): From the mating pool, form (N_A/2) number of parent pairs randomly; the crossover operation is performed on each pair using a high value for the crossover probability, P_{cr}^A (range 0.9−1.0). In a simple crossover operation known as the "single point" crossover, a point is selected randomly along the length of each binary parent string and both strings are cut at that point. Next, the sliced portions are exchanged mutually between the parent strings and combined to obtain two offspring strings. This crossover operation, when repeated on the ($N_A/2$) number of parent pairs, produces N_A number of offspring strings.

Step 5 (*Mutation*): Mutate the bits of offspring strings wherein the probability that a randomly selected bit undergoes mutation is P^A_{mut} (range 0.01−0.05). In mutation, a randomly selected bit is flipped from zero to one and vice versa. The population emerging after the mutation operation represents a new generation of solutions and thus, $N^A_{gen} = N^A_{gen} + 1$.

Step 6 (*Termination*): Repeat steps 2 to 5 on the new generation of strings till it is observed that the fitness of the best solution shows no increase over a large number (say 500) of successive generations, or the GA has evolved over the specified number (N^A_{max}) of generations. Finally, the N binary segments in the string possessing maximum fitness score are decoded (see eq 7), and the optimal values of the decision variables obtained thereby represent the optimized solution, $X^* = [x_1^*, x_2^*, ..., x_N^*]^T$.

Analogous to the GP procedure, it is necessary in the GA procedure also that the entire GA implementation is repeated several times using different seed values for the random number generator. The optimal solutions obtained thereby are compared, and the one leading to maximum gluconic acid yield is selected as an overall optimal solution.

3.2. Simultaneous Perturbation Stochastic Approximation Algorithm (SPSA). The SPSA (Spall, 1987; Spall, 1998a,b) is a numerically efficient stochastic optimization algorithm. Unlike the commonly utilized gradient-based optimization methods, which evaluate the gradient of the objective function by perturbing each decision variable separately (as in the standard two-sided finite difference approximation), the SPSA formalism approximates the gradient by perturbing all the variables simultaneously. Thus, irrespective of the number (N) of decision variables to be optimized, only two objective function measurements are sufficient for the gradient approximation. In contrast, the finite difference approximation needs $2N$ function evaluations.

The SPSA implementation is iterative that begins with a randomly initialized (guess) solution vector, $\hat{X}_0$. The SPSA formalism stipulates that the objective function to be maximized should be differentiable since it searches for the maximum point, X^*, at which the gradient of the objective function, $G(X^*)$, attains zero magnitude. That is

$$G(X^*) = \left|\frac{\partial f^*(X)}{\partial X}\right|_{X=X^*} = 0 \qquad (9)$$

In the SPSA-based maximization of y_{gl}, an estimate of the decision variable vector (X) is updated iteratively as given by (Renotte et al., 2000])

$$\hat{X}_{t+1} = \hat{X}_t - a_t\hat{G}_t(\hat{X}_t) \qquad (10)$$

where t denotes the iteration index; a_t is a non-negative scalar gain coefficient, and the N-dimensional vector, $\hat{G}_t(\hat{X}_t)$, is an approximation of the unknown gradient, $G(X^*)$. The gradient approximation is carried out by varying all the elements of $\hat{X}_t$ simultaneously, and using

$$\hat{G}_t = \begin{bmatrix} \dfrac{f^*(\hat{X}_t + c_t\Delta_t) - f^*(\hat{X}_t - c_t\Delta_t)}{2c_t\Delta_{t_1}} \\ \dfrac{f^*(\hat{X}_t + c_t\Delta_t) - f^*(\hat{X}_t - c_t\Delta_t)}{2c_t\Delta_{t_2}} \\ \cdot \\ \cdot \\ \cdot \\ \dfrac{f^*(\hat{X}_t + c_t\Delta_t) - f^*(\hat{X}_t - c_t\Delta_t)}{2c_t\Delta_{t_N}} \end{bmatrix} \qquad (11)$$

where c_t is a positive scalar, and $\Delta_t = [\Delta_{t_1}, \Delta_{t_2}, ..., \Delta_{t_N}]^T$ is a vector of random variables drawn from the symmetric Bernoulli $\pm$ 1 distribution. The decaying gain sequences, $\{a_t\}$ and $\{c_t\}$, are defined as

$$a_t = \frac{a}{(A + t + 1)^{\phi}}; \quad c_t = \frac{c}{(t + 1)^{\gamma}} \qquad (12)$$

where γ, ϕ, c, A, and a are the SPSA-specific parameters. Several guidelines for the judicious selection of these parameters are provided in Spall (1998a,b). In each SPSA iteration, the decision variable estimate is updated using eqs 10 and 11, respectively. After a sufficiently large number of iterations, i.e., $t \geq t_{max}$, the SPSA procedure converges to an optimal solution. As with GA and GP procedures, the SPSA procedure also needs to be repeated several times by changing the value of the random number generator seed for obtaining an overall optimal solution vector, X^*.

4. ANN-GA and ANN-SPSA Hybrid Modeling-Optimization Formalisms

Artificial neural networks (ANNs) are an information processing paradigm based on the mechanisms followed by the highly interconnected parallel structure of the human brain. ANNs are a set of mathematical models that mimic some of the observed properties, such as pattern recognition and classification, possessed by biological nervous systems; they also exhibit similarity with the “learning-by-experience” principle followed by biological species. ANNs possess a structure comprising a highly interconnected network of processing elements (also termed “nodes”). The processing elements (PEs) serve as the simplified artificial analogues of the biological neurons, and the weighted connections linking the PEs are analogues to the biological synapses. Given a data set containing measurements of the input-output variables, an ANN learns the nonlinear interrelationships existing between them by appropriately adjusting the connection weights. This process, termed network training, is similar to the learning in a biological system involving adjustment of the synaptic connections. There exist a wide variety of ANN architectures and algorithms to train them. The most commonly used ANN architecture is known as multilayered perceptron (MLP) and the widely used method to train an MLP network is the error-back-propagation (EBP) algorithm (Rumelhardt et al., 1986). The MLP network comprising three layers (input, hidden, and output) of nodes is popular for approximating the nonlinear relationships existing between an input set of data (casual variables) and the corresponding output (dependent variables) data set. It has been established that an MLP network with a single hidden layer housing a sufficient number of PEs can approximate any nonlinear function to an arbitrary degree of accuracy (Hornik et al., 1989; Poggio et al., 1990). In the EBP-based training of an MLP network, the weights are adjusted iteratively such that the network, in response to the input patterns in the example set, accurately predicts the corresponding output values. The details of the EBP algorithm and the various issues involved in obtaining an optimal MLP model can be found, for example, in Freeman and Skapura (1991), Bishop (1994), and Tambe et al. (1996). In the second phase of implementing the ANN-GA and ANN-SPSA hybrid formalisms, the input space of the ANN model is optimized, in a manner similar to the optimization of the GP-based model, using GA and SPSA strategies (Nandi et al., 2001, 2002).

5. Results and Discussion

5.1. Experimental Details. For developing the GP-based model for the glucose to gluconic acid bioprocess, experimental input-output data from the fermenter were used. In these experiments, the gluconic acid producing strain *Aspergillus niger* NCIM 545 was utilized. The details of the spore germination medium, the growth medium, and the cellulosic fiber support have been described earlier by Sankpal et al. (1999).

Fermentation Medium for Immobilized Mycelia. Anhydrous purified glucose (100 g), $MgSO_4 \cdot 7H_2O$ (0.035 g), KH_2PO_4 (0.05 g), and 0.1 g of $(NH_4)_2HPO_4$ were dissolved in 1 L of water. The pH of this medium was adjusted to 6.0 using 1 M H_2SO_4. A woven cellulosic fabric support (69 × 8.5 × 0.6 cm) with void volume of approximately 140 mL was sterilized at 15 psi for 60 min.

Submerged Fermentation. Submerged fermentation utilizing the immobilized culture was carried out in a modified locally fabricated batch fermenter (Figure 1). In the fermenter, the matrix with fully grown *A. niger* was folded in a spiral shape. For preventing mycelial recirculation, the upper end of the fixed bed was closed by the filter mesh. The batch reactor was drained after the substrate reached its lowest concentration.

Maintaining Oxygen Partial Pressure. A constant flow of air was used to maintain the oxygen partial pressure and a DO probe (Ingold, 170-ppm type DO amplifier) was used for measuring the dissolved oxygen concentration.

Glucose and Gluconic Acid Analyses. Feed and the unconverted glucose were analyzed by the dinitrosalicyclic acid method (Miller, 1959), and the gluconic acid concentration in the bioreactor was measured by titrating against 6 N NaOH.

5.2, GP-Based Fermenter Modeling. For obtaining the GP-based model, process data from 46 runs were used. The data set (see Table 1) comprises values of the three operating variables, namely, glucose concentration (g/L) (x_1), biomass concentration (g/L) (x_2), and dissolved oxygen (DO) concentration (mg/L) (x_3), and the corresponding values of the process output variable, i.e., gluconic acid concentration (y). This data set was partitioned into the training set (batches 1−23) and the test set (batches 24−46). While the training set was used for computing the fitness of the GP-searched expressions, the test set was used to cross-validate the expressions. The objective of cross-validation is to test the prediction (generalization) ability of the GP-searched expressions on a data set different from the set used for obtaining the expression. To secure an overall optimal data-fitting expression, the GP procedure was repeated 100 times by employing different seed values for the pseudo-random number generator. In each such repeated run, a different mathematical expression was searched by the GP. The expression with the highest fitness value for the training set ($\hat{R}^2 = 0.987$) and the test set ($\hat{R}^2 = 0.986$) was

Table 1. Experimental Data Utilized for Building GP- and ANN-Based Models

batch	glucose concn (x_1) (g/L)	biomass concn (x_2) (g/L)	DO (x_3) (mg/L)	gluconic acid concn (y) (g/L)	gluconic acid yield (y_{gl}) (%)[a]
1	100.0	1.00	10.0	6.416	5.90
2	150.0	2.00	10.0	48.015	29.42
3	120.0	2.00	15.0	27.100	20.76
4	150.0	2.50	15.0	57.946	35.51
5	150.0	3.00	15.0	57.389	35.16
6	120.0	2.00	25.0	36.262	27.77
7	120.0	2.00	30.0	45.020	34.48
8	150.0	2.00	30.0	94.424	57.86
9	150.0	3.00	25.0	80.486	49.32
10	150.0	2.00	40.0	128.907	78.99
11	150.0	2.00	45.0	146.036	89.48
12	150.0	2.00	50.0	154.230	94.50
13	180.0	2.00	50.0	175.525	89.63
14	150.0	3.00	40.0	129.006	79.05
15	150.0	2.50	50.0	154.360	94.58
16	150.0	2.50	55.0	152.440	93.41
17	150.0	2.50	60.0	148.940	91.26
18	160.0	2.50	60.0	163.067	93.67
19	175.0	3.00	55.0	176.490	92.69
20	160.0	3.00	60.0	162.420	93.30
21	180.0	3.00	60.0	172.598	88.13
22	150.0	3.00	60.0	151.280	92.70
23	100.0	3.00	60.0	21.803	20.04
24	100.0	2.00	10.0	6.670	6.13
25	120.0	2.50	10.0	22.952	17.58
26	100.0	2.00	15.0	7.829	7.20
27	150.0	2.00	15.0	57.261	35.09
28	120.0	2.00	20.0	31.486	24.12
29	150.0	2.00	20.0	66.900	40.99
30	150.0	2.50	20.0	67.449	41.33
31	150.0	3.00	20.0	67.328	41.25
32	150.0	2.00	35.0	111.328	68.22
33	150.0	2.50	30.0	95.988	58.82
34	150.0	3.00	30.0	94.707	58.03
35	150.0	2.50	40.0	129.930	79.61
36	150.0	3.00	35.0	111.604	68.38
37	150.0	2.00	60.0	152.430	93.40
38	120.0	2.00	60.0	73.502	56.30
39	150.0	3.00	45.0	144.651	88.63
40	180.0	2.50	55.0	179.064	91.94
41	150.0	3.00	50.0	152.890	93.68
42	180.0	2.50	60.0	174.483	89.09
43	150.0	3.00	55.0	154.230	94.50
44	166.0	3.00	60.0	169.450	93.82
45	165.0	3.00	60.0	167.910	93.53
46	162.0	3.00	60.0	164.870	93.54

[a] Gluconic acid percentage yield, $y_{gl} = 100y/1.088x_1$.

$$y = \left[\frac{\alpha_1 x_1}{(x_1 - \alpha_2)^4 + \alpha_3}\right]\left[\frac{1}{x_2^{\,2} - \alpha_4 x_2 + \alpha_5}\right]\left[\frac{1}{\alpha_6 x_3^{\,2} - \alpha_7 x_3 + \alpha_8}\right] \tag{13}$$

The GP-specific parameters and the mathematical operators used to obtain eq 13 are listed in Table 2. The values of the function parameters $\{\alpha_k\}$ searched by the GP are $\alpha_1 = 3.1911 \times 10^{10}$, $\alpha_2 = 158.219$, $\alpha_3 = 2.974 \times 10^6$, $\alpha_4 = 5.421$, $\alpha_5 = 107.15$, $\alpha_6 = 0.116$, $\alpha_7 = 12.752$, and $\alpha_8 = 448.112$. Having obtained the best fitting functional form, it was subjected to the nonlinear regression using Marquardt's algorithm with a view to bring about an improvement in the model's prediction accuracy. The refined $\{\alpha_k\}$ values obtained thereby are $\alpha_1 = 3.206 \times 10^{10}$, $\alpha_2 =$

Table 2. Functional Set, Operators, and GP-Specific Parameters

terminals (operands) set (variables and constants)	x_1, x_2, x_3, α
operator set	$+, -, \div, \times$
relative abundance of operators	equal
fitness function	sum-squared-error (SSE) based
selection method	elitist selection
population size	40
number of generations	550
probability of crossover	0.95
probability of mutation	0.02
number of replicates	100

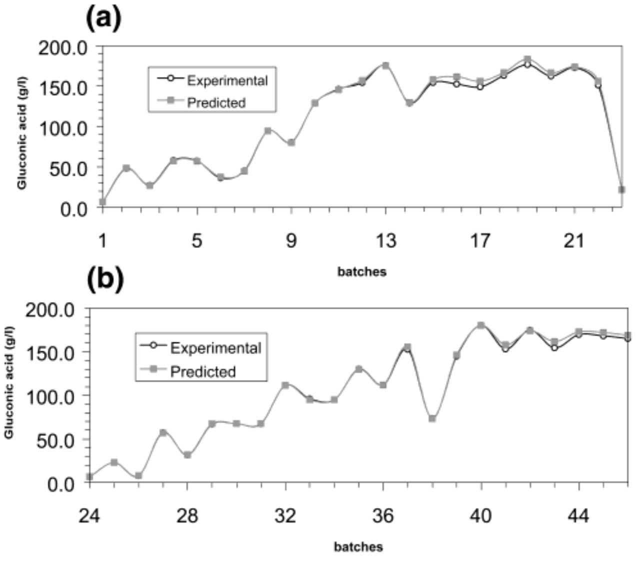

Figure 5. (a)Gluconic acid concentration as predicted by the GP-based model (eq 13) and the corresponding actual output values from the training set. (b) Gluconic acid concentration as predicted by the GP-based model (eq 13) and the corresponding actual output values from the test set.

158.201, $\alpha_3 = 2.9795 \times 10^6$, $\alpha_4 = 5.413$, $\alpha_5 = 107.16$, $\alpha_6 = 0.117$, $\alpha_7 = 12.753$, and $\alpha_8 = 447.389$; these parameter values brought about a significant enhancement in the $\hat{R}^2$ values corresponding to the training set ($\hat{R}^2 = 0.9984$) and the test set ($\hat{R}^2 = 0.9979$). The high $\hat{R}^2$ values for both the training and test sets are indicative of the good prediction accuracy and generalization ability of the GP-based fermenter model. A comparison of the model predicted and actual process output values for the training and test set data is presented in Figure 5a and b, respectively.

5.3. GA and SPSA-Based Optimization of the GP Model. The input space of the GP-based fermenter model was optimized by employing GA and SPSA formalisms separately. The objective of optimization was to obtain the optimal values of the three fermenter operating variables ($x_1 - x_3$) that would maximize the gluconic acid yield, y_{gl}. While performing the optimization, the following values of the GA- and SPSA-specific parameters were employed.

- *GA parameters*: population size (N_A) = 55, crossover probability (P^A_{cr}) = 0.92, mutation probability (P^A_{mut}) = 0.05, string length (l^A_{chr}) = 45, and maximum number of generations (N^A_{gen}) = 500.
- *SPSA parameters*: $a = 0.1$, $c = 0.02$, $A = 250$, $\phi = 0.602$, $\gamma = 0.101$, maximum number of iterations (t_{max}) = 32,000.

The fitness function used in the GA-based optimization was

Table 3. Optimized Fermenter Conditions Given by GP-GA and GA-SPSA Methods

method	glucose concn (x_1) (g/L)	biomass concn (x_2) (g/L)	DO (x_3) (mg/L)	gluconic acid concn (y) (g/L)	maximized gluconic acid yield (y^*_{gl}) (%)
GA					
	157.551	2.699	54.49	170.465	99.327
	157.432	2.598	54.36	170.061	99.211
	157.813	2.701	54.49	170.259	99.087
SPSA					
	157.816	2.598	54.38	170.239	99.074
	158.061	2.699	54.49	170.464	99.051
	157.726	2.599	55.12	170.067	99.031

$$R^A = y_{gl} = \frac{100y}{1.088x_1} \tag{14}$$

Using the above-listed parameter values several (100) GA and SPSA replicates were run using different random number generator seeds. From the 100 sets of optimized operating variables obtained thereby, Table 3 lists the top three solutions as given by the GA and the SPSA methods, respectively.

5.4. Process Modeling and Optimization using ANN-GA and ANN-SPSA Hybrids. For an ANN-based fermenter modeling, the most popular MLP network has been used. The MLP architecture contained a single hidden layer and was trained using the EBP algorithm. To develop an optimal MLP-based model, the network's structural parameter, namely, the number of hidden layer nodes (N_H) and the EBP algorithm specific parameters, viz., the learning rate (η_{ebp}) and momentum coefficient (μ_{ebp}), were varied systematically. The same training and test sets as used in the GP-based modeling were utilized for training the MLP network. Here, the training set was used to adjust the network weights iteratively while the test set was used to monitor the network's generalization performance. The criterion used for selecting an optimal MLP model was minimum root-mean-squared-error (RMSE) with respect to the test set. The values of MLP's structural parameters and the EBP algorithm specific parameters that led to the minimum RMSE with respect to the test set ($RMSE_{tst}$) are: the number of input nodes representing three input variables = 3, N_H = 2, number of output nodes (representing fermenter output variable, y) = 1, $\eta_{ebp} = 0.15$, and μ_{ebp}= 0.001. The training and test set RMSEs corresponding to the optimal model were 0.0377 and 0.0489, respectively. Once an optimal MLP-based model was obtained, in the next phase its input space comprising the three operating variables was optimized using the GA and SPSA methods (Nandi et al., 2001, 2002). The GA/SPSA procedure for this optimization remained same as used in optimizing the input space of the GP-based model. The optimized fermenter conditions leading to the maximized y_{gl} value as given by the ANN-GA and ANN-SPSA methods are listed in Table 4 for which following GA-/SPSA-specific parameter values were utilized.

- *GA parameters:* population size (N_A) = 35, mutation probability (P^A_{mut}) = 0.05, crossover probability (P^A_{cross}) = 0.95, string length (l^A_{chr}) = 45, and maximum number of generations (N^A_{max}) = 500.
- *SPSA parameters*: $a = 0.1$, $c = 0.02$, $A = 200.0$, $\phi = 0.602$, $\gamma = 0.101$, maximum number of iterations (t_{max}) = 32,000.

5.5. Experimental Validation. It can be observed from the 12 solutions given by the GP-GA, GP-SPSA, ANN-GA, and ANN-SPSA hybrid methods (see Tables 3 and 4) that the optimized values of the three fermenter

Table 4. Optimized Fermenter Conditions Given by ANN-GA and ANN-SPSA Methods

method	glucose concn (x_1) (g/L)	biomass concn (x_2) (g/L)	DO (x_3) (mg/L)	gluconic acid concn (y) (g/L)	maximized gluconic acid yield (y^*_{gl}) (%)
GA					
	158.265	2.622	54.42	170.524	98.958
	157.985	2.671	54.48	170.217	98.955
	158.132	2.625	54.51	170.212	98.861
SPSA					
	158.415	2.593	54.49	170.248	98.704
	158.523	2.577	54.38	170.311	98.673
	158.723	2.642	55.13	170.472	98.642

operating conditions differ only marginally. The maximized yield (y^*_{gl}) values listed in Tables 3 and 4 indicate that the first solution given by the GP-GA method is an overall optimal solution ($x_1 = 157.5$, $x_2 = 2.699$, $x_3 = 54.49$, $y = 170.465$, and $y^*_{gl} = 99.3$). Thus, this set of the operating variables was chosen for the experimental validation. The bioreactor operating under the stated conditions resulted in the gluconic acid concentration of 170.237 (g/L). This value, which is in close agreement with the GA maximized value of 170.465 (g/L), corresponds to 99.08% gluconic acid yield. From the yield data listed in Table 1, it is observed that the best yield value obtained using the unoptimized operating conditions was 94.58% (batch 15). It can thus be seen that the optimized solution has improved the yield by 4.5%. In the absence of a nonlinear process model, an unaided manual inspection of the process data would give no clue to the precise values of the optimized operating conditions necessary for the maximization of gluconic acid yield. However, the usage of the GP-GA and GP-SPSA hybrid modeling-optimization techniques has allowed us to obtain the optimized fermenter operating conditions that have imparted a significant improvement in the gluconic acid yield.

6. Conclusions

To summarize, this paper presents two process modeling and optimization strategies integrating genetic programming (modeling formalism) with the two stochastic optimization formalisms, namely, GA and SPSA, respectively. The principal advantage of using the GP formalism for process modeling is that the model can be developed exclusively from the historic process input-output data without invoking the detailed process phenomenology. Also, the GP technique automatically obtains a closed-form equation relating process inputs and outputs; this is in contrast to the ANNs using a generic universally applicable nonlinear functional form for the data fitting. Upon constructing a GP-based process model, its input space comprising process input variables is optimized using GA and SPSA techniques. These optimization paradigms possess certain unique advantages over the commonly utilized gradient-based techniques. Moreover, the GA methodology can be used even when the objective function is not smooth, differentiable, and continuous. In the present study, the GP-GA and GP-SPSA formalisms have been utilized for the modeling and optimization of glucose to the gluconic acid batch fermentation process. The overall optimized process conditions obtained thereby, when verified experimentally, have brought about a significant improvement in the gluconic acid yield. The GP-GA and GP-SPSA formalisms proposed in this paper are sufficiently general and thus can be utilized for modeling and optimization of other batch and continuous bioprocesses.

Notation

a, A	SPSA specific parameters
f	nonlinear function for input-output data fitting
f^*	optimally fitting nonlinear model
$G(\cdot)$	gradient function in SPSA-based optimization
l^A_{chr}	length of a chromosome or a string in GA simulation
l^A_n	number of bits to represent nth decision variable
l^P_{chr}	number of elements in a symbolically coded string
N_H	number of nodes in MLP network's hidden layer
N_A	number of binary strings (population size) in GA simulation
N_{trn}	number of training patterns
N^A_{gen}	generation index in GA simulation
N^A_{max}	maximum number of generations for GA evolution
N^P_{max}	maximum number of generations for GP evolution
P^A_{cr}	crossover probability in GA procedure
P^A_{mut}	mutation probability in GA procedure
P^P_{cr}	crossover probability in GP procedure
P^P_{mut}	mutation probability in GP procedure
$\hat{R}^A_j$	squared fitness of jth string in GP procedure
R^A	string fitness in GA procedure
S_n	decoded decimal value of nth binary segment
t_{max}	maximum number of iterations for the SPSA procedure
x_i	ith decision variable
X	N-dimensional vector of decision variables
X^*	optimal decision variable vector
$\hat{X}_0$	randomly initialized (guess) solution vector in SPSA procedure
y	process output variable defining gluconic acid concentration
y_{gl}	gluconic acid yield
y^*_{gl}	maximized gluconic acid yield
y_{prd}	GP model predicted value of the output variable
Z	limit for the parameter values searched by the GP

Greek Symbols

α	vector of parameters appearing in the GP-based model
γ	SPSA specific parameter
ϕ	SPSA specific parameter
μ_{ebp}	learning rate in the EBP-based training procedure
η_{ebp}	momentum coefficient in the EBP-based training procedure

Acknowledgment

J.J.S.C. thanks Council of Scientific and Industrial Research (CSIR), New Delhi, for a Senior Research Fellowship.

References and Notes

Bishop, C. Neural Networks and their Applications. *Rev. Sci. Instrum.* **1994**, *65*, 1803.

Davis, L. *Handbook of Genetic Algorithms*; Von Nostrand Reinhold: New York, 1991.

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

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

Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley: New York, 1989.

Holland, J. H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, 1975.

Hornik, K.; Stinchcombe, M.; White, H. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks* **1989**, *2*, 359.

Karel, S. F.; Robertson, C. R. Autoradiographic Determination of Mass Transfer Limitations in Immobilized Cell Reactors. *Biotech. Bioeng.* **1989**, *34*, 320−336.

Kinnear, K. E., Jr., Ed.; *Advances in Genetic Programming*; The MIT Press: Cambridge, MA, 1994.

Koza, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*; The MIT Press: Cambridge, MA, 1992.

Kulkarni, B. D.; Tambe, S. S.; Dahule, R. K.; Yadavalli, V. K. Consider Genetic Programming for Process Identification. *Hydrocarbon Process.* **1999**, *78*, 89−97.

Marquardt, D. W. An Algorithm for Least Squares Estimation of Nonlinear Parameter. *J. Soc. Ind. Appl. Math.* **1963**, *11*, 431−441.

McKay, B.; Willis, M.; Bartob, G. W. Steady-State Modeling of Chemical Process Systems Using Genetic Programming. *Comput. Chem. Eng.* **1997**, *29*, 981−996.

Metz, B.; Kossen, N. F. W. The Growth of Molds in the Form of Pellets: Literature Review. *Biotechnol. Bioeng.* **1977**, *19*, 781−799.

Miller, G. L. Use of Dinitrosalicyclic Acid Reagent for Determination of Reducing Sugar. *Anal. Chem.* **1959**, *31*, 426−429.

Mukherjee, S.; Dahule, R. K.; Tambe, S. S.; Ravetkar, D. D.; Kulkarni, B. D. Consider Genetic Algorithms To Optimize Batch Distillation. *Hydrocarbon Process.* **2001**, *80*, 121−127.

Nandi, S.; Ghosh, S.; Tambe, S. S.; Kulkarni, B. D. Artificial Neural-Network-Assisted Stochastic Process Optimization Strategies. *AIChE J.* **2001**, *47*, 126−135.

Nandi, S.; Mukherjee, P.; Tambe, S. S.; Kumar, R.; Kulkarni, B. D. Reaction Modeling and Optimization Using Neural Networks and Genetic Algorithms: Case Study Involving TS-1 Catalyzed Hydroxylation of Benzene. *Ind. Eng. Chem. Res.* **2002**, *41*, 2159−2169.

Nandi, S.; Rahman, I.; Tambe, S. S.; Sonalikar, R. L.; Kulkarni, B. D. Process identification using genetic programming: a case study involving fluidized catalytic cracking (FCC) unit. In *Petroleum Refining and Petrochemical Based Industries In Eastern India*; Saha, R. K., Ed.; Allied Publisher Ltd.: Mumbai, India, 2000; pp 195−201.

Ramanathan, S. P.; Mukherjee, S.; Dahule, R. K.; Ghosh, S.; Rahman, I.; Tambe, S. S.; Ravetkar, D. D.; Kulkarni, B. D. Optimization of Continuous Distillation Columns Using Stochastic Optimization Approaches. *Trans. Inst. Chem. Eng.* **2001**, *79*, 310−321.

Rao, D.; Subba, A.; Panda, T. Comparative Analysis of Different Whole Cell Immobilized *Aspergillus niger* Catalysts for Gluconic Acid Fermentation Using Pretreated Cane Molasses. *Bioprocess Eng.* **1994**, *11*, 209−212.

Roehr, M.; Kubicek, C. P.; Kominek, J. In *Biotechnology*; Rehm, H. J., Reed, G., Eds.; VCH: New York, 1996; Vol. 6, pp 348−362.

Ronnette, C.; Vande, W. A.; Remy, M. Neural modeling and control of a heat exchanger based on SPSA techniques. *Proc. Am. Control Conf.* **2000**, 3299−3303.

Rumelhart, D.; Hinton, G.; Williams, R. Learning Representations by Back-Propagating Errors. *Nature* **1986**, *323*, 533.

Sankpal, N. V.; Cheema, J. J. S.; Tambe, S. S.; Kulkarni, B. D. An Artificial Intelligence Tool for Bioprocess Monitoring: Application to Continuous Production of Gluconic Acid by Immobilized *Aspergillus niger*. *Biotechnol. Lett.* **2001a**, *23*, 911−916.

Sankpal, N. V.; Joshi, A. P.; Sutar, I. I.; Kulkarni, B. D. Continuous Production of Gluconic Acid by *Aspergillus niger* Immobilized on a Cellulosic Support: Study of Low pH Fermentative Behavior of *A. niger*. *Process Biochem.* **1999**, *35*, 317−325.

Sankpal, N. V.; Sahasrabuddhe, N. A. Production of organic acids and metabolites on fungi and applications in food industry. In *Applied Mycology and Biotechnology*; Khachatourians, G. G., Arora, D. K., Eds.; Elsevier Science: Netherlands, 2001b.

Spall, J. C. A Stochastic Approximation Technique for Generating Maximum Likelihood Parameter Estimates. *Proc. Am. Control Conf.* **1987**, *12*, 1161−1167.

Spall, J. C., An Overview of the Simultaneous Perturbation Method for Efficient Optimization. *Johns Hopkins APL Tech. Dig.* **1998a**, *19*, 482−491.

Spall, J. C. Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization. *IEEE Trans. Aerosp. Electron. Syst.* **1998b**, *34*, 817−822.

Szpiro, G. G. Forecasting chaotic time series with genetic algorithms. *Phys. Rev. E* **1997**, *55*, 2557−2568.

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

Taguchi, H.; Yoshida, T.; Tomita, Y.; Teramoto, S. The Effects of Agitation on Disruption of the Mycelial Pellets in Stirred Fermenters. *J. Ferment. Technol.* **1968**, *46*, 814−822.

Venkatasubramanian, V.; Sundaram, A. Genetic Algorithms: Introduction and Applications. In *Encyclopedia of Computational Chemistry*; John Wiley: Chichester, 1998; pp 1115−1136.

Wittler, R.; Baumgartl, H.; Lubbers, D. W.; Schugerl, K. Investigations of Oxygen Transfer into *Penicillium chrysogenum* Pellets by Microprobe Measurements. *Biotechnol. Bioeng.* **1986**, *28*, 1024−1036.

Yadavalli, V. K.; Dahule, R. K.; Tambe, S. S.; Kulkarni, B. D. Obtaining Functional Form for Chaotic Time Series Evolution Using Genetic Algorithm. *Chaos* **1999**, *9*, 789−794.

Accepted for publication August 30, 2001.

BP015509S