Modeling Chemical Processes Using Prior Knowledge and Neural Networks

Michael L. Thompson and Mark A. Kramer
Dept. of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139

We present a method for synthesizing chemical process models that combines prior knowledge and artificial neural networks. The inclusion of prior knowledge is investigated as a means of improving the neural network predictions when trained on sparse and noisy process data. Prior knowledge enters the hybrid model as a simple process model and first principle equations. The simple model controls the extrapolation of the hybrid in the regions of input space that lack training data. The first principle equations, such as mass and component balances, enforce equality constraints. The neural network compensates for inaccuracy in the prior model. In addition, inequality constraints are imposed during parameter estimation. For illustration, the approach is applied in predicting cell biomass and secondary metabolite in a fed-batch penicillin fermentation. Our results show that prior knowledge enhances the generalization capabilities of a pure neural network model. The approach is shown to require less data for parameter estimation, produce more accurate and consistent predictions, and provide more reliable extrapolation.

Introduction

Artificial neural networks (ANNs), such as three-layer backpropagation networks and radial basis function networks, have been proven to be universal function approximators (Cybenko, 1989; Poggio and Girosi, 1990a; Hornik et al., 1989). This ability to approximate arbitrarily complex functions has been exploited in applying ANNs as models of chemical processes (Di Massimo et al., 1992; Bhat et al., 1990; Pollard et al., 1992). The major advantage of neural network models is that they can be synthesized without detailed knowledge of the underlying process. Only after presentation of the data does the behavior of a neural network conform to the specifics of the particular process. This property is common to the larger class of flexible functional form models known as nonparametric models, which includes ANNs, Fourier series, smoothing splines, and kernel estimators (Eubank, 1988).

The lack of a process-based internal structure is a liability for the neural network when faced with sparse, noisy data. For example, if a neural network is used to predict the composition of a reactive mixture, first principles dictate that the outputs are nonnegative and that they sum to one if expressed as mole fractions. Due to the limited amount of training data and noise in the data, the network outputs may not conform to these process constraints. This inconsistency becomes more severe as the network makes predictions beyond the limits of the training data (extrapolates). For mole fractions, the problem can be corrected by applying a normalization transform on the network outputs. In general, though, a more elaborate fixed form or parametric model must be applied.

Insufficient data hamper the accuracy of a neural network because the network relies completely on the data when inducing process behavior. However, qualitative knowledge of the function to be modeled may be helpful in overcoming data sparsity. For example, an engineer may know that a process variable increases monotonically in approaching an asymptotic limit. Both the monotonicity and the asymptote are prior knowledge that should be enforced on the neural network model. However, sparse data may prevent a neural network from capturing either of these behaviors.

Currently, researchers are investigating several design and training approaches to include prior knowledge into neural networks. These approaches exploit the knowledge available prior to receiving process data and attempt to reduce the dependence on noisy, sparse data. The approaches are summarized in Table 1. In *design approaches*, prior knowledge dictates

Correspondence concerning this article should be addressed to M. A. Kramer, who is currently at the Gensym Corporation, 125 Cambridge Park Dr., Cambridge, MA 02140.

Table 1. Approaches Combining Prior Knowledge with Neural Networks

Approach	Design Approaches			Training Approaches	
	Modular	Semiparametric: Serial	Semiparametric: Parallel	Inequality constraints	Objective function
Advantages	May improve interpretability. Easier to train.	Guaranteed output behavior.	Network compensates for discrepancies between data and inexact parametric model. May enforce different constraints over different subregions.	Consistent outputs.	Preferred functional behavior. Improved generalization.
Disadvantages	Output behavior not guaranteed. Unstructured subnetworks.	Unstructured network.	Output behavior not guaranteed.	More difficult to train.	Difficult to determine appropriate form. Must determine regularizer constant.

model structure. This reduces the dimensionality of the parameter space. In *training approaches*, prior knowledge dictates the form of the parameter estimation problem. This reduces the feasible region of the parameter space. Both approaches reduce the amount of data required to estimate the optimal parameters.

Design approaches use prior knowledge as the basis for selection of the type of network nodes, modularization of the network architecture, or inclusion of nonnetwork parametric models. *Modular design* approaches construct interconnected neural network models according to the topological and functional structure of the process. *Semiparametric design* approaches combine a neural network with a fixed form parametric model either in series or in parallel.

An example of a modular design approach is the hierarchical network proposed by Mavrovouniotis and Chang (1992). A hierarchical network decomposes the process model into groups of related variables. For example, rather than model a process train as a single large network, with every input possibly affecting every output, a modular approach constructs a subnetwork for each process unit. The subnetworks are then connected according to the structures and functions of the actual process units. The resulting modular architecture is more sparsely connected than the single network. The result is fewer parameters, easier training, reduction of infeasible input/output interactions, and easier interpretation of model behavior.

Semiparametric design approaches apply a parametric model in tandem with the neural network. A parametric model has a fixed structure derived from first principles, existing empirical correlations or known mathematical transformations. In a series approach (Figure 1a), the neural network estimates intermediate variables to be used in the parametric model (Jordan and Rumelhart, 1992). Psichogios and Ungar (1991) proposed a serial approach in modeling a fermentation. The network estimated the specific growth rate, which was input into the component mass balances. Joerding and Meador (1991) used the parametric model as a normalization post-processor to force the outputs of the network to sum to one, which is important when estimating quantities such as component mole fractions in mixtures or the probabilities of mutually exclusive events.

In a parallel semiparametric approach (Figure 1b), the outputs of the neural network and parametric model are combined to determine the total model output. The model serves as an idealized estimate of the process or a best guess at a process model. The neural network is trained on the residual between the data and the parametric model to compensate for any uncertainties that arise from the inherent process complexity (Su et al., 1992). Johansen and Foss (1992a,b) derived a parallel approach for adaptive process control that generated different model behaviors in different regions of the process input space. Their architecture weighted each model according to a Gaussian response function centered in the region of applicability of that model.

Design approaches determine model structure, but do not guarantee obedience of all process constraints. Alternatively, model training approaches use prior knowledge to impose inequality constraints on the model. Inequality constraints may

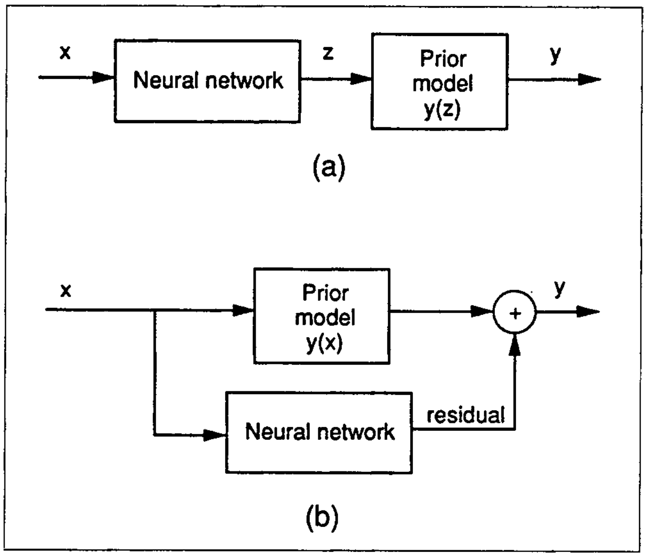

Figure 1. Semiparametric approaches to combine prior knowledge with neural networks.

(a) Serial approaches force the output to be consistent with prior model $y(z)$; (b) parallel approaches use prior model $y(x)$ as a guide to assist the network.

involve the inputs and outputs of the network as well as the model parameters. Constraints involving inputs are called infinite constraints because they must hold for all (infinitely many) possible input combinations. Joerding and Meador (1991) transformed infinite constraints into finite constraints, which involve only the network weights. In this way, they enforced monotonicity, convexity, or concavity on a network output with respect to the network inputs. However, the transformations they proposed force all of the network outputs to have the prescribed behavior and, therefore, are not generally suitable for multiple-output models.

In general, infinite constraints must be imposed during parameter estimation. Estimating the parameters becomes a semiinfinite programming (SIP) problem (Fiacco et al., 1983):

$$\begin{aligned} &\underset{w}{\text{Minimize:}} \quad F(w) \\ &\text{Subject to:} \quad g(w,x) \le 0; \quad x \epsilon D_x \subset R^N \end{aligned} \tag{1}$$

where

w = model parameters
x = N-dimensional vector over the real domain D_x
$F(w)$ = objective function
$g(w,x)$ = infinite inequality constraints

Solution of an SIP involves a two-phase minimax problem (Tanaka et al., 1988). Most SIP algorithms grid the input domain D_x, approximate the constraint surfaces using polynomial interpolation, and find the maxima of the constraint surfaces with respect to x. Constraint maximization is performed at each iteration of the search for the w that minimizes the objective function. Rigorous solution of an SIP is computationally expensive for even a single infinite constraint involving more than two or three inputs. Of course, given N_d data points, each infinite constraint could be transformed into a set of N_d finite constraints by evaluating the constraint at each point. However, this does not guarantee satisfaction of the constraints for all possible inputs (Gallant and Golub, 1984).

Another model training approach is used to impose functional preferences, such as smoothness, on the model. This approach favors a preferred model behavior by including a regularizer or penalty function in the estimation objective function. Regularizers may be terms such as $\Sigma|\partial^2 y_i/\partial x^2|$, $i=1, \ldots, N_d$, which penalize high degrees of curvature in the fitted function. Poggio and Girosi (1990b) showed that smoothed radial basis function networks (RBFNs) and regularizers derived in approximation theory have a close relationship. Bishop (1991) applied regularization theory to RBFNs to improve generalization.

Approaches to include prior knowledge as a preferred functional behavior have also been derived from a Bayesian theoretic standpoint. MacKay (1992a) and Buntine and Weigund (1991) applied Bayesian methods to backpropagation neural network training. MacKay (1992b) and Sibisi (1989) demonstrated the advantages of this approach over conventional cross validation, which is typically used in nonparametric modeling. The Bayesian approach is a consistent, formalized means of representing prior knowledge in the parameter estimation problem and in quantifying its effects on the generalization capabilities of the resultant models (Berger, 1985; Skilling and Gull, 1987; MacKay, 1992a; Gull, 1989).

This article focuses on combining prior knowledge with artificial neural networks; however, the technique is applicable to other nonparametric models. Alternative nonparametric models include kernel estimators and splines (Eubank, 1988; Härdle, 1990) and multivariate methods such as generalized additive models (Hastie and Tibshirani, 1990; McCullaugh and Nelder, 1989), projection pursuit (Friedman and Stuetzle, 1981), MARS (Friedman, 1992), and sliced inverse regression (Duan and Li, 1991). In the absence of a parametric component, these nonparametric models suffer from the same disadvantages as neural networks: they depend overly on the data, they may produce predictions that are inconsistent with process constraints, and they suffer from unreliable extrapolation. Neural networks were investigated as a representative member of the set of multivariate nonparametric models because of their ease of implementation and, in the case of RBFNs, localized node activation functions.

Despite the variety of approaches, a general methodology that combines many forms of prior knowledge with neural networks for modeling chemical processes does not exist. Our work attempts to derive a model structure that has broad capability to incorporate prior knowledge into process model synthesis. In the next section, we present our approach, including an enumeration of the types of prior knowledge it handles. Following that, we illustrate the methodology with a case study in which we predict biomass and product formation in a fed-batch penicillin fermentation. After describing the design of the study, we discuss the case study results. Finally, we conclude with comments on the performance and flexibility of the methodology.

Proposed Approach

In our work, we have developed a hybrid modeling approach. The approach attempts to maximize the value of domain-specific knowledge. It does so by combining design approach concepts to define the model structure and training approach concepts to impose inequality constraints and functional preferences. In this section, we first describe the hybrid model structure. Then, we enumerate the types of knowledge common to process modeling, describe the complementary roles of this knowledge and the neural network, and present a systematic approach to building the hybrid model structure. We conclude the section with the mathematical formulation of the parameter estimation problem.

Hybrid model

The foundational model structure for our approach is a hybrid or semiparametric model. Figure 2 shows that this structure applies a parametric "default" model in parallel with a nonparametric radial basis function network (RBFN). These components are combined in series with a parametric output model. The default model accounts for parametric model behavior that holds in the absence of data. The neural network captures unknown functional relationships between the inputs and outputs. The output model enforces the explicit functional relationships that exist between the variables. Also, if constraints among the inputs exist, the structure in Figure 2 could be augmented by an input preprocessor. This is not shown as part of the model because such a preprocessor can be regarded

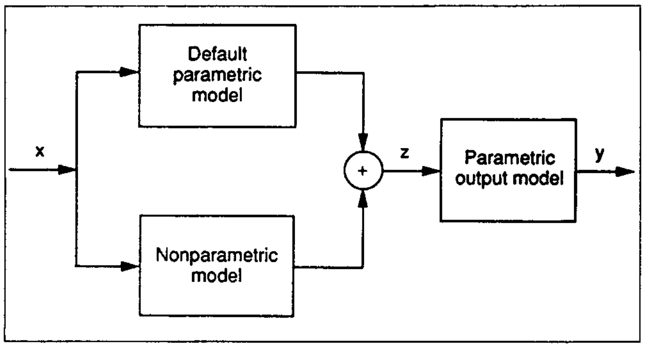

Figure 2. Hybrid model structure.

The default parametric model compensates for sparse data and improves extrapolation, the neural network compensates for uncertainty and the bias of the default, and the parametric output model enforces equality constraints upon outputs.

as a data rectification step prior to model synthesis and application.

The rationale for this architecture can be appreciated by examining the limiting cases of no prior knowledge and of full knowledge of process behavior. In the absence of prior knowledge, the hybrid model reduces to a RBF neural network. RBFNs are universal approximators (Poggio and Girosi, 1990a). In the presence of full knowledge, the hybrid reduces to a pure parametric model. The model then can be analyzed using traditional nonlinear statistical techniques. Statistical regression theory dictates that as long as the parametric output model has the same form as the true process model, in the limit of large data sample size the hybrid will converge to the true model (Gallant, 1987).

Prior Knowledge

Prior knowledge is knowledge about the process that exists prior to the synthesis of the model. It includes the intended purpose of the model, "hard" constraints imposed on the process by first principles or design considerations, and preferences in model behavior. Usually this corresponds to process knowledge known prior to the presentation of the data. However, it may include knowledge from prior iterations in the iterative model synthesis procedure. In that case, the knowledge includes information about the performance of prior models.

The model purpose implies general properties of the model structure and performance criteria. For example, an on-line control application implies a dynamic model and possibly an adaptive formulation. On the other hand, a plantwide optimization application would imply simpler, lumped models for individual process units. In the control application, accurate one-step ahead prediction may be the defining performance criterion. In optimization, accurate steady-state prediction may be most important.

In addition to the purpose of the model, equality and inequality constraints are also prior knowledge. These constraints are considered "hard" or "soft" depending on their precedence with respect to the data. Hard constraints arise from mass and energy balances and physical restrictions imposed on the plant by equipment limitations. They may be expressed in terms of equalities and inequalities among model inputs and outputs. These constraints take precedence over the process behavior implied by the data. Conversely, preferences in model behavior are "soft"constraints that influence the regression of the model, but have lower precedence than the data. These preferences may take the form of default relations, which should hold while extrapolating beyond the training data or as qualitative behavior such as smoothness.

In this work, we have focused on prior knowledge expressible as relations among the process input and output variables. A relation may be an equality or an inequality constraint. Each relation has a specific *domain of applicability* and a *precedence with respect to process data.* A relation may be categorized as quantitative or qualitative according to the degree to which the functional form of the relation can be specified. In this context, *quantitative relations* are explicit equality or inequality constraints with known functional form, while *qualitative relations* only specify that a relation exists between a subset of inputs or outputs. An example of a qualitative relation is the knowledge that several input variables always act together as a dimensionless group in their determination of an output, but how the group determines the output is unknown.

Domain of applicability

The domain of applicability of a relation specifies the region of input-output space over which the relation applies. Inherent in any function approximation problem are the definitions of the input and output variables and their ranges, which define the domain of applicability of the model. However, function behavior in specific subregions may be known, such as asymptotic limits, zeros, extrema, inflection points, and default behavior. This knowledge may be expressed by partitioning variable space into subregions, each subject to specific relations. A domain may be expressed explicitly in terms of the input variables, implicitly in terms of the inputs through bounds on the output variables, or implicitly in terms of the distribution of the available data in input space.

Relations and their roles

Enumeration of the types of prior knowledge allows us to systematically address the role of each type in the model synthesis problem. The relations and their roles in model synthesis are summarized in Table 2. The role of the relation depends on: (1) whether the relation is an equality or inequality; (2) the extent to which the functional structure of the relation is known; (3) its precedence with respect to the data; and (4) whether it involves only inputs or includes outputs.

Equality constraints with known functional form are parametric models that reduce degrees of freedom. An equality with precedence greater than the data is enforced either as an input preprocessor (*Role 1*) or an output model (*Role 2*) depending on whether inputs or outputs are involved, respectively. If the equality has precedence lower than the data, the domain of applicability defines the role. In domains beyond the data, the equality serves as a default model that provides a baseline estimate of the outputs (*Role 3a*). This estimate is refined by nonparametric submodels and controls extrapolation in the absence of data. Within the range of the data, an equality with lower precedence than the data serves as a functional preference that is imposed by a penalty function during training (*Role 3b*). Equalities with unknown form and a greater

Table 2. Role of Prior Knowledge

Relation type	Equality				
Functional form	Known			Unknown	
Precedence w.r.t. Data	Greater		Lower	Greater	Lower
Variables Involved	Inputs only	Any outputs	Any	Any	Any
Role	1. parametric preprocessor	2. parametric output model	3a. default model, if domain is region without data; 3b. penalty function	4. nonparametric submodel	provides no information
Relation type	**Inequality**				
Functional form	**Known**			**Unknown**	
Precedence w.r.t. Data	**Greater**		**Lower**	**Greater**	**Lower**
Variables Involved	**Inputs only**	**Any outputs**	**Any**	**Any**	**Any**
Role	5. input space partitioning	6a. infinite constraint; 6b. parametric output model, if transformable	7. penalty function	provides no information	provides no information

precedence than the data are the minimum knowledge necessary to propose a model. As such, they are modeled using a purely nonparametric network (*Role 4*).

Inequality constraints serve as modifiers to the training objective function or appear as constraints during training. Inequalities only impart meaning when their functional form is known. When the inequality is a hard constraint, such as nonnegativity of a mole fraction, it must be strictly adhered to. Hard inequalities involving only inputs will partition the input space into domains of applicability (*Role 5*). This serves as the basis for input data rectification or merely as a screening filter that prevents invalid inputs to the model. On the other hand, if the outputs are involved in a hard inequality, the relation is an infinite constraint (*Role 6a*). Whenever possible, infinite constraints should be imposed by applying a suitable postprocessing transformation (*Role 6b*), as is done with link functions for generalized linear and generalized additive models (McCullagh and Nelder, 1989; Hastie and Tibshirani, 1990). For example, if the output variable y must be positive, then the constraint $g(x;w) = -y(x;w) \leq 0$ must hold, and the hybrid could compute intermediate variable $z(x;w) = \ln(y)$. The output model, then, would simply be $y(x;w) = e^z$. If the inequality has a lower precedence than the data, the constraint represents merely a preference in model behavior. Such preferences can be injected into the estimation objective function as a penalty function (*Role 7*).

In sum, equality constraints define the model structure (*Roles 1,2,3a,* and *4*), and inequalities define the parameter estimation problem (*Roles 5, 6a* and *7*). The exceptions are when an equality with known form and precedence lower than the data exists, in which case it modifies the estimation problem (*Role 3b*), and when an infinite inequality can be transformed into a parametric output model (*Role 6b*). Knowledge of functional form dictates whether relations appear (if known) as parametric or (if unknown) nonparametric models. In this way, the flexibility of nonparametric models may be exploited wherever uncertainty exists about the form of any relation. A model expressible as a known relation with an unknown contribution provides the basis for decomposing it into parametric and nonparametric modules. Precedence is derived from whether the constraint is considered hard or soft. If the constraint is hard, it has higher precedence than the data and serves as a consistency constraint. If the constraint is soft, the data have higher precedence and a relation with known form is represented as a default model or a behavioral preference depending on whether the relation is an equality or inequality, respectively. Constraints with precedence lower than the data and unknown functional form provide no useful information. Involvement of inputs or outputs also help determine the role of a relation. A relation involving inputs only serves as a preprocessing filter or an input space partitioning. The roles of relations involving outputs are determined primarily by their type, form and precedence.

Example

We illustrate the role of prior knowledge through an example model synthesis problem (Table 3). The objective is to determine an equation of state for a nonideal ternary gas mixture with components A, B and C. Using experimental data, we wish to determine the relationship between volume V, pressure P, temperature T, the total moles n, and composition (mole fractions) $X = [X_A, X_B, X_C]^T$. Initially, the only knowledge that exists is that the output variable $y = V$ depends on inputs $x = [P,T,N,X_A,X_B,X_C]^T$. Table 3 shows the construction of the full hybrid as knowledge is introduced in each of six synthesis steps.

Step 1. At this point, only a pure nonparametric network can be posed. Additional knowledge comes from first principles.

Step 2. First principles dictate that a normality constraint

Table 3. Prior Knowledge in the Ternary Gas Mixture Example

Step	Relation	Attributes		Role
1.	relation we wish to model: $V=f(P,T,n,X)$	Type: Form: Precedence: Variables:	equality unknown greater inputs & output	nonparametric network
2.	mole fractions sum to one: $\Sigma X_i = 1$	Type: Form: Precendence: Variables:	equality known greater inputs only	parametric preprocessor
3.	non-negative inputs: $x \leq 0$	Type: Form: Precedence: Variables:	inequality known greater inputs only	parametric preprocessor
4.	non-negative output: $y(x) \leq 0$, for all x	Type: Form: Precedence: Variables:	infinite inequality known greater inputs only	parametric output model [such as $z(x;w) =$ $\ln(y)$ $f_{out}(z) = e^z$]
5.	default behavior at low P: $z_{def}(x) = nRT/P$	Type: Form: Precedence: Variables:	equality known lower inputs & output	parametric default model
6.	deviation from ideality: $z(x;w) = z_{def}(x) + f_{unk}(x)$ $y = f_{out}[z(x;w)]$	Type: Form: Precedence: Variables:	equality unknown lower inputs & output	nonparametric subnetwork

on the mole fractions be imposed. This constraint eliminates the need to independently gather data on one of the three mole fractions. Alternatively, it may serve as the rationale behind a preprocessing normalization (rectification) of the input data for all three mole fractions.

Step 3. The nonnegativity constraint on all inputs also arises from first principles. It is applied as a preprocessing transformation or as an input filter.

Step 4. First principles dictate that the output V must be positive. This is an infinite constraint because it must hold for all input combinations. As such, it can be transformed into an output model $f_{out}(.)$ or applied directly using semiinfinite programming. The output model will be $f_{out}(z) = e^z$, where intermediate variable $z(x;w) = \ln(y)$ is learned by the nonparametric network.

Step 5. We suspect that the gas mixture behavior can be described by a known equation of state, but that significant differences exist between reality and the known model. Also, the mixture has near ideal gas behavior at low P, though we do not have data in this region. This knowledge is the basis for subdividing the hybrid model $f_{hyb}(x)$ into a known equation of state, which is the parametric component $z_{def}(x)$, and an unknown nonparametric component $f_{unk}(x)$. The ideal gas law $z_{def}(x) = V_{def} = nRT/P$ will serve as the default model.

Step 6. The deviation from ideality at higher pressures must be compensated for, but the functional form is unknown, $f_{unk}(x)$. A nonparametric model $z_{net}(x;w)$, with parameters w, will approximate this unknown function.

The resulting hybrid equation of state $y = f_{out}[z(x;w)]$ will maintain consistent predictions when extrapolating into the low-pressure regime, where training data were unavailable.

Neural network

The neural network in the hybrid architecture is responsible for learning the difference between the default model and the target data. Although the neural network is a nonparametric estimator capable of approximating this difference, we also require that the neural network give a negligible contribution to the model output for inputs far from the training data. This property results in the hybrid having the correct default behavior in regions without training data. The transition from the data-regressed behavior to the default behavior must occur smoothly. A radial basis function network (RBFN) has this desired property.

Figure 3 depicts the RBFN architecture. It is constructed by first using k-means clustering on the x-data alone (Moody and Darken, 1989) to position the nodes in input-space (x-space) and determine the widths σ_j. Locally-tuned elliptical units can be used to account for differences in data density in different dimensions (Johnston and Kramer, 1994). After positioning and sizing the nodes, the second-layer weights, which appear linearly in the calculation of the network output, must be determined. These parameters are estimated by solving the appropriate optimization problem as dictated by the available prior knowledge.

Parameter estimation

Prior knowledge not only dictates the model structure but also specifies the form of the optimization problem used in estimating the second-layer weights of the RBFN. The parameters of the default and output models are assumed to be known, and only the RBFN weights must be estimated. In

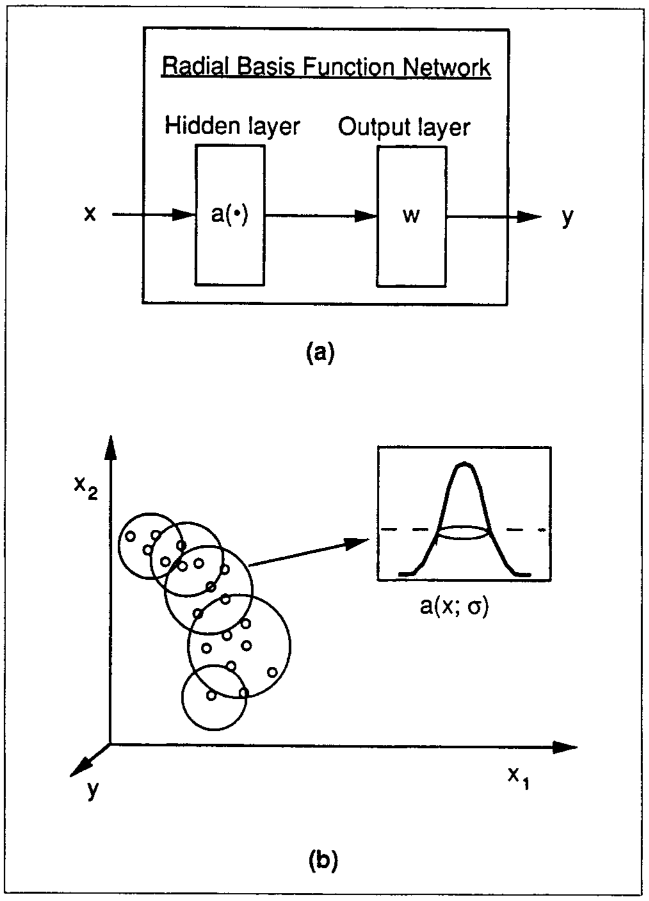

Figure 3. Architecture of the radial basis function network (RBFN).

(a) The RBFN consists of one hidden layer and one output layer. The outputs y are linear in the output weights w:$y=a(x;\sigma)w$. (b) The hidden layer nodes have multivariate Gaussian activation functions $a_i\,(x;\sigma_i)$. Each node has a data-dependent width σ_i.

general, minimization for parameter estimation is an SIP problem:

$$\underset{w}{\text{Minimize:}}\quad F(w)=(1/2)\sum_{i=1}^{N_d}\|y_i-y_{\text{hyb}}(x_i;w)\|^2$$

$$\text{Subject to:}\quad g(w,x)\le 0;\quad x\in D_x\subset R^N \tag{2}$$

where
w = RBFN weights
y_i = M-dimensional vector of outputs for the ith data point
y_{hyb} = $f_{\text{hyb}}(x_i;w)$, hybrid model estimate of y given x_i
x_i = N-dimensional vector of inputs for the ith data point
$g(w,x)$ = infinite inequality constraints

Estimation becomes a constrained nonlinear programming problem if all inequalities are finite or the infinite inequalities can be transformed into finite constraints. Finite inequality constraints are independent of the inputs x, and standard constrained optimization techniques are used to solve the associated Lagrangian, that is, the penalized objective function:

$$\underset{w,\lambda}{\text{Minimize:}}\quad L(w)=F(w)+\lambda^T g(w) \tag{3}$$

where
$F(w)$ = least-squares objective function
λ = vector of Lagrange multipliers
$g(w)$ = finite inequality constraints

Parameter estimation for the hybrid model does not include infinite equality constraints because these constraints serve as the output model. When there are no infinite inequalities and the output model is nonlinear, the RBFN weights are estimated using conventional nonlinear programming (NLP) methods. However, if the output model f_{out} is linear in x and y, the problem can be readily solved as a constrained linear regression. For example, mass-balance constraints may dictate that $By=Cx$, where matrix B is $L\times M$ and C is $L\times N$. The M outputs may be partitioned into $M-L$-independent variables y_I and L-dependent variables y_D: $y=[y_I^T|y_D^T]^T$. The y_D may be expressed in terms of x and $y_I=z(x;w)$:

$$y_D=B_D^{-1}(Cx-B_I y_I)\equiv Qx+Pz \tag{4}$$

The y_I are estimated by summing the default model z_{def} and the RBFN $z_{\text{net}}=Wa(x)$, where W is the $(M-L)\times H$ matrix of output weights and $a(x)$ is the vector of H node activations:

$$z(x;w)=z_{\text{def}}(x)+z_{\text{net}}(x;w)=z_{\text{def}}(x)+Wa(x) \tag{5}$$

The resulting output model, f_{out}, computes the outputs of the hybrid model:

$$y_{\text{hyb}}(x)=[y_I^T|y_D^T]_{\text{hyb}}^T=f_{\text{out}}[z(x;w),x]=[z^T|(Qx+Pz)^T]^T \tag{6}$$

The output model is linear in z and therefore linear in the weights W (Eq. 5). Likewise, the parameter estimation objective function $F(W)$ (for a linear model with no inequality constraints) is quadratic in z and, thus, quadratic in W:

$$\underset{W}{\text{Minimize:}}\; F(W)=(1/2)\sum_{i=1}^{N_d}\left[\sum_{j=1}^{M-L}(y_{j,i}-z_{j,i})^2 + \sum_{r=1}^{L}\left(y_{m-L+r,i}-\sum_{n=l}^{N}Q_{rn}x_{n,i}-\sum_{k=1}^{M-L}P_{rk}z_{k,i}\right)^2\right] \tag{7}$$

After transforming the weight matrix W into a vector and applying the stationarity condition for a quadratic (Eq. 8), the optimal parameters are found by either direct inversion of the Hessian H or solution using a more robust technique such as singular value decomposition (SVD):

$$Hw=-\nabla F(0) \tag{8}$$

where
w = vector of the elements of the weight matrix W: $w_{(i-1)H+j}=W_{ij}$
H = Hessian of $F(w)$: $H_{ij}=\partial^2F/\partial w_i\partial w_j$
$\nabla F(0)$ = gradient of $F(w)$ evaluated at $w=0$: $\nabla F_i=\partial F/\partial w_i$

To summarize, if there are no inequality constraints the RBFN weights are estimated using a nonlinear programming (NLP) algorithm. If in addition the output model is linear, constrained linear regression may be used to determine the weights. When all inequalities present are finite, a constrained NLP algorithm is used; if any of the inequalities are infinite, an SIP method is used.

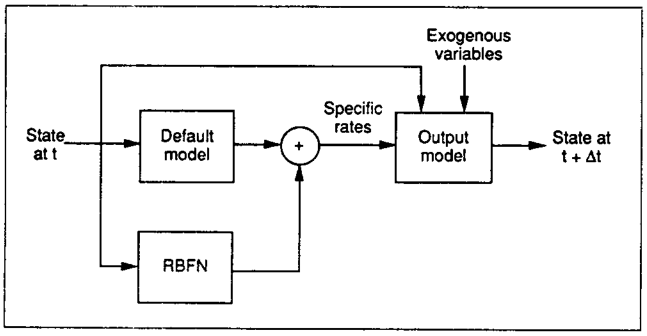

Figure 4. Hybrid model for penicillin fermentation study.
The six inputs are the three state variables (X, S, P) at time t and the three exogenous variables (D, S_f, Δt). The three outputs are state variables at time $t+\Delta t$.

Case Study: Fed-Batch Penicillin Fermentation

The hybrid modeling methodology was applied in a case study of a fed-batch penicillin fermentation. The study included prior knowledge into a neural network by synthesizing the model structure in Figure 2. The focus was to determine the performance of this methodology on a difficult parameter estimation problem posed by inducing process behavior from sparse, noisy training data. This is a common problem when modeling a process with high dimensional input space. In modeling the low-dimensionality fermentation process, we chose the training set to be large enough for all networks tested to learn the relations. However, the set was chosen to be small enough that the noise and sparsity would make pure nonparametric induction difficult.

Model synthesis

Fed-batch penicillin fermentation is a dynamic biochemical process. In this process, a fermentor is charged with a growth medium containing the penicillin microorganism in low concentration. As the fermentation progresses, substrate (a sugar such as glucose) is fed into the fermentor to control the biomass growth rate at a high level. Once sufficient biomass has accumulated, the substrate feed rate is reduced to life-sustaining levels. At this point, the culture begins to produce the penicillin product in earnest. The fermentor analyzed in this study is characterized as follows:

- Biomass concentration $X(t)$, penicillin concentration $P(t)$, and substrate concentration $S(t)$ define the process state at time t.
- Specific growth rate μ, specific product formation rate q_p, and specific substrate consumption rate σ serve as time-varying parameters relating the state variables to their time derivatives.
- Substrate concentration in the feed $S_f(t)$ and the dilution rate $D(t)$ [defined as feed rate $F(t)\equiv dV/dt$ divided by culture volume $V(t)$] drive the process.

Due to the complexity of a biological process, the true process exhibits phenomena that are not easily modeled or are unknown. For illustration, the phenomena characterizing the actual process are listed below but were assumed to be unmodeled:

- Cell biomass growth rate is limited by substrate availability.
- Growth rate decreases with biomass due to oxygen diffusion limitations.
- Cell lysis becomes appreciable at high biomass concentration.
- Maintenance energy of the culture (a component of the specific substrate consumption) depends on biomass concentration.
- Penicillin product decays as the culture ages.

The governing equations for this simulated true process appear in Appendix A.

Figure 4 depicts the 6-input/3-output hybrid model used in this study. It is assumed that prior knowledge exists, derived from either simple experiments on the current fermentor using the current strain of the penicillin microorganism, earlier experiments in lab-scale equipment, or possibly earlier experiments with a less productive strain. This prior knowledge is represented by approximate specific rate correlations serving as the default model (Appendix B) and the component mass balances serving as the output model (Appendix C). The default model differs from the unknown actual process phenomena in several ways:

- Specific growth rate is independent of biomass concentration.
- Maximum specific growth rate is higher.
- Cell lysis is negligible.
- Inhibition of product formation by substrate is greater.
- Product yield on substrate is lower.
- Maintenance energy is lower.

Taken together, these phenomena result in a default model that predicts higher biomass growth rate and lower product formation rate than the actual process.

The hybrid model $f_{\text{hyb}}(x,u)$ consists of default model $z_{\text{def}}(x)$, neural network $z_{\text{net}}(x)$, and parametric output model $f_{\text{out}}(z,u)$. The hybrid accepts as input the three state variables $x(t)$, and the exogenous inputs $u(t)$ at time t. It outputs the predictions of the process state at time $t+\Delta t$, $x(t+\Delta t)$. The specific rate correlations of the default model $z_{\text{def}}(x)$ take as input $x(t)$ and output the parameters $z(x)\equiv[\mu(x),\ q_p(x),\ \sigma(x)]^T$. These parameters are used directly in the output model:

$$x(t+\Delta t)=f_{\text{hyb}}(x,u)=f_{\text{out}}[z(x),u] \tag{9}$$

$$z(x)\equiv[\mu(x),\ q_p(x),\ \sigma(x)]^T=z_{\text{def}}(x)+z_{\text{net}}(x) \tag{10}$$

$$u(t)\equiv[D(t),\ S_f(t)]^T \tag{11}$$

The output model $f_{\text{out}}(z,u)$ is the analytical solution of the differential component balances (Appendix C). Given $x(t)$, $z(x)$, and $u(t)$, the balance equations are solved to compute $x(t+\Delta t)$:

$$x(t+\Delta t)\equiv[X(t+\Delta t),\ S(t+\Delta t),\ P(t+\Delta t)]^T \tag{12}$$

$$X(t+\Delta t)=X(t)\exp(\mu\Delta t)/[1+D(t)\Delta t] \tag{13}$$

$$S(t+\Delta t)=S_f(t)-[S_f(t)-S(t)]/[1+D(t)\Delta t] \\ -X(t+\Delta t)[1-\exp(-\mu\Delta t)]\sigma/\mu \tag{14}$$

Table 4. Training Data Operating Conditions

Run	Conditions
1	Standard run ($F=0.11$ 1/h, $S_f=525$ g/L)
2	Low feed substrate ($S_f=480$ g/L)
3	Variable feed concentration (S_f uniformly distributed from 475 to 575 g/L)
4	Ramped feed profile

$$P(t+\Delta t)=X(t+\Delta t)\{[P(t)/X(t) - q_p/(\mu+K)]\exp[-(\mu+K)\Delta t]+q_p/(\mu+K)\} \quad (15)$$

The RBFN used in the hybrid model consisted of four nodes with an overlap factor of one (used in setting the radial unit widths). This architecture was chosen as the network with the fewest nodes that was capable of accurately predicting the data. Maximum likelihood estimation, which amounts to least-squares minimization, was sufficient to determine the second-layer RBFN weights that gave a model with good generalization properties. Therefore, the full hybrid network was trained by finding the RBFN weights that minimized the least-squares objective function of Eq. 2 without inequality constraints. We used the BFGS quasi-Newton method for unconstrained NLP to solve this problem.

The process data used in training were generated by simulation using a mechanistic model consisting of nonlinear differential balance equations and algebraic specific rate correlations. Gaussian white noise with a standard deviation of 5% of the output range was added to the simulation results to represent measurement error in each state variable. Four 200-hour fermentations sampled at 3-hour intervals and subjected to various feed schedules served as experimental data. Table 4 lists the operating conditions for these runs. The input/target pairs used as training data were built by pairing state variables at time t with the state at $t+\Delta t$. A total of 268 training pairs resulted. Prior to training, all data were mean-centered and unit variance scaled. Four additional 200-hour fermentations in Table 5 were used to test the generalization and extrapolation capabilities of the model.

For comparison, the same data used to train the hybrid were used to train a conventional backpropagation network (BPN) and a stand-alone RBFN. The architectures for both networks were optimized by selecting the minimum prediction error network as indicated by cross validation, a procedure that involves a series of train/test subset partitions of the training data (Eubank, 1988; Härdle, 1990; Weiss and Kulikowski, 1991). The optimal BPN contained one hidden layer with 20 hidden sigmoidal nodes and was trained using the BFGS quasi-Newton method in least-squares minimization. The optimal RBFN had 45 hidden radial units with overlap factor of 6.

Table 5. Test Data Operating Conditions

Run	Conditions
1	Low feed rate ($F=0.02$ 1/h)
2	Low feed substrate & high biomass ($S_f=200$ g/L, $X=15gDCW/1$)
3	High-variability feed concentration (S_f uniformly distributed from 420 to 630 g/L)
4	Long sample time ($t=6$ h)

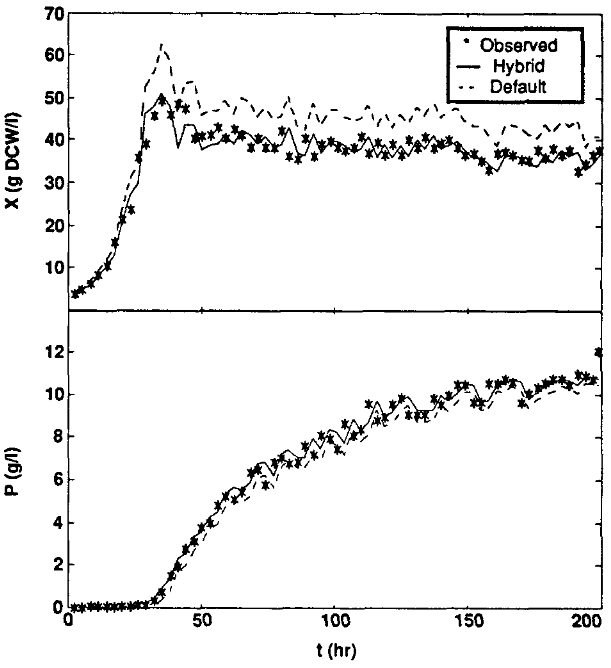

Figure 5. Comparison of hybrid to default model.

Biomass, X, and product, P, vs. time for training run 1. The hybrid compensates for the overestimation of biomass and for the underestimation of product by the default model.

Results

Results of applying the hybrid model illustrate the roles of the RBFN, default and output submodels in generating predictions that are accurate, reliable and consistent. Figure 5a plots biomass concentration vs. time for the hybrid, the default, and the actual process. It is readily seen that the RBFN component of the hybrid model compensates for the overestimation of biomass growth by the default model. Similarly, Figure 5b shows that the RBFN compensates for the underestimation of product formation by the default. In most cases, the default model predicts trajectories of the same shape as the process measurements. Therefore, learning the near-constant residuals is an easier task for the RBFN submodel than learning the complete nonlinear trajectory, which the pure RBFN must accomplish. Another indication of the benefit imparted by the default model is the relative sizes of the RBFN submodel and the pure RBFN. The submodel uses only four nodes while the pure network requires 45. Even after accounting for the difference in the number of inputs, this is a large drop in the degrees of freedom and an indication of the beneficial inductive bias of the default model.

Figures 6a and 6b compare the hybrid to the BPN and RBFN, respectively, in predicting product for a typical training run. The BPN and the hybrid perform similarly, giving nearly identical predictions throughout the fermentation with the exception of the values at early times. In some runs, the BPN predicted negative product concentrations early in the fermentation. In contrast, the output model of the hybrid ensured

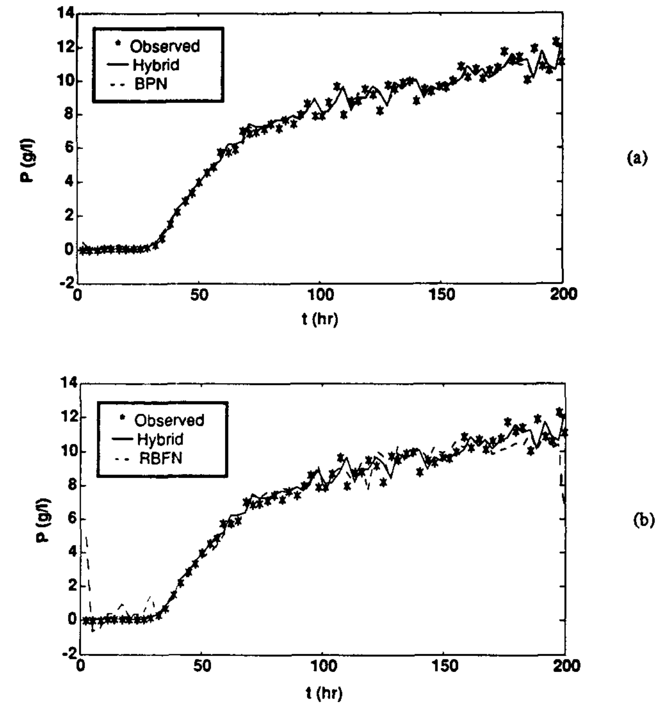

Figure 6. Product vs. time for training run 3.

(a) Comparison of hybrid to BPN; (b) comparison of hybrid to RBFN.

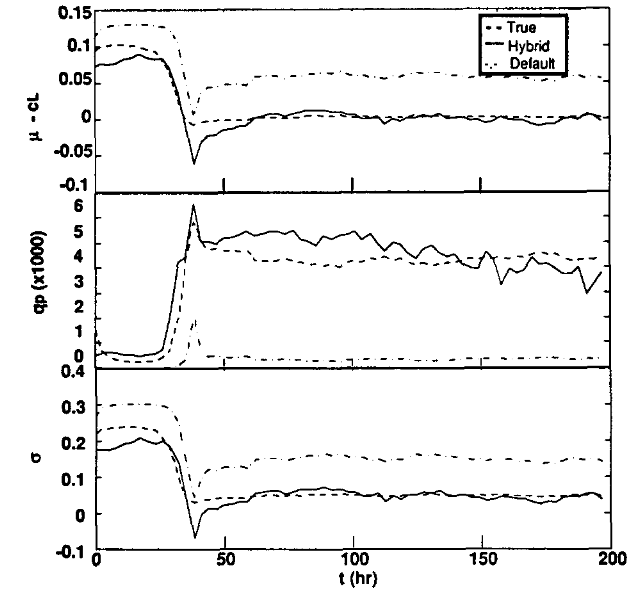

Figure 8. Specific rates (growth minus lysis, $\mu - c_L$; product formation, q_p; and substrate consumption, σ) vs. time.

The hybrid performs well at predicting the unmeasured specific rates due to the RBFN's improvement to the prediction of the default model.

that the hybrid always predicted values consistent with reality. The pure RBFN performed worse than both the BPN and the hybrid. Predictions from the RBFN showed greater variability and were inaccurate at the beginning and the end of the fermentation. These endpoints correspond to input values on the fringes of the training data. Poor prediction near the boundaries of the data is a common problem with purely nonparametic kernel estimators. This problem is overcome by the use of an output model in the hybrid. Certainly, if large data sets were gathered to train the pure networks, their performances would improve. But, noise in the data could still obscure the true process constraints, allowing inconsistent predictions.

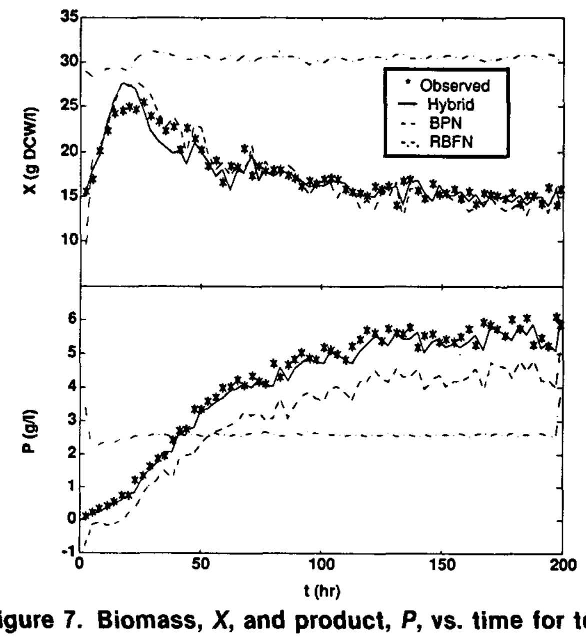

Figure 7. Biomass, *X*, and product, *P*, vs. time for test run 2.

The hybrid extrapolates reliably. The BPN extrapolates inaccurately and inconsistently. The RBFN is completely unable to extrapolate due to the locality property of its nodes.

A more dramatic improvement over the pure neural network approaches is evident when extrapolating. Figures 7a and 7b present the biomass and product predictions for the hybrid, BPN and RBFN for the low-substrate test fermentation. As expected, the RBFN cannot extrapolate reliably beyond its region of training data. The BPN performs better but also suffers when given inputs from outside of its training domain. The problem of inconsistent predictions early in the fermentation is exacerbated during extrapolation. The hybrid, on the other hand, maintains accurate and consistent predictions over a wider region of input space. The default model overcomes the liability of unreliable extrapolation common to neural networks. As long as the default is reasonably accurate over the range of extrapolation, the hybrid performs well.

The ability to learn the underlying relationship between the state variables and the specific rates is key to the hybrid model's ability to accurately predict the fermentation state. Despite the fact that the specific rates were unmeasured, the hybrid model was able to accurately learn the specific rate correlations (Figure 8). This is due to the restrictions placed on the model by the constraints of the output model and the ability of the nonparametric model to accurately approximate arbitrary nonlinear functions. The result is greater insight into process behavior than is afforded by a conventional neural network approach.

Conclusions

Our work shows that combining prior knowledge of the process domain with a radial basis function network in a hybrid semiparametric model helps to provide accurate, consistent and reliable predictions when faced with sparse, noisy data. Prior knowledge improves performance by serving as a default estimate of process behavior in the absence of training data and as equality constraints that force the model to output predictions consistent with the physical process. The resulting model is also more easily interpreted than neural networks and other nonparametric models.

The quality of the prior knowledge is important. If the output model is misspecified, the hybrid will perform as poorly as any other misspecified parametric model. However, even in the unlikely case that the default model had no relation to the true process, as long as the default is smooth and continuous, the RBFN of the hybrid will be at least as easy to train as a pure RBFN due to the universal approximation properties of the network (Poggio and Girosi, 1990a).

The hybrid modeling methodology is flexible enough to be tailored to the specific process being modeled. The model is constructed according to the prior knowledge that exists. Though applied in the fermentation model as a means of estimating hidden time-varying parameters, the default and RBFN may be combined without an output model. This approach was used successfully in modeling vinyl acetate polymerization (Kramer et al., 1992). If a default model cannot be posed *a priori* the hybrid reduces to a serial semiparametric model with the RBFN feeding a parametric output model. On the other hand, if no first principle equality constraints exist, as may be the case in modeling physical properties, the hybrid reduces to a purely nonparametric network with weights estimated subject to inequality constraints.

The hybrid structure is also amenable to evolutionary model synthesis. As more knowledge is gathered through operation and experimentation during the lifetime of a process, more relations may be incorporated into the model structure. For example, a reactor model may begin as a neural network with a mass balance imposed as an output model. As the reaction mechanism and kinetics are elucidated through experimentation, a default model may be posed. As further insight into the mechanism is gained, the hybrid model may be decomposed into a series of parametric modules representing the subreactions. The intermediates computed by these submodels would then be refined by the neural network's contribution.

Acknowledgment

Michael L. Thompson was supported by National Science Foundation Grant EID-9021047.

Notation

c_L	=	specific cell lysis rate (h^{-1})
D	=	dilution rate (h^{-1}) $= F/V$
D_x	=	input domain
$f_{hyb}(.)$	=	hybrid model
$f_{out}(.)$	=	parametric output model
F	=	feed rate (1/h) $= dV/dt$
$g(.)$	=	inequality constraints
K	=	product decay constant (h^{-1})
M	=	number of outputs y_i, $i=1, \ldots, M$
M_x	=	maintenance energy (h^{-1})
N	=	number of inputs x_i, $i=1, \ldots, N$
N_d	=	number of data points
P	=	product concentration (g/L)
q_p	=	specific product formation rate (h^{-1})
S	=	substrate concentration (g/L)
S_f	=	substrate concentration in the feed (g/L)
t	=	time
Δt	=	sampling interval
$u(.)$	=	exogenous input variables
V	=	working volume (L)
w	=	model parameters
$x(.)$	=	input (independent) variables
X	=	biomass concentration (g DCW/L)
$y(.)$	=	output (dependent) variables
$z(.)$	=	intermediate variables
$z_{def}(.)$	=	parametric default model
$z_{net}(.)$	=	nonparametric neural network

Greek letters

μ	=	net specific growth rate (h^{-1})
σ	=	specific substrate consumption rate (h^{-1})

Superscripts

T	=	matrix transpose

Parameter values

c_{Lm}	=	0.0084 h^{-1}
K	=	0.01 h^{-1}
K_I	=	1.0 g/L
K_L	=	0.05
K_p	=	0.0001 g/L
K_x	=	0.3
m_{xm}	=	0.029 h^{-1}
q_{pm}	=	0.004 h^{-1}
$Y_{p/s}$	=	1.2
$Y_{x/s}$	=	0.47
μ_m	=	0.11 h^{-1}

Literature Cited

Berger, J. O., *Statistical Decision Theory and Bayesian Analysis*, McGraw-Hill, New York (1985).

Bhat N. V., P. A. Minderman, T. J. McAvoy, and N. S. Wang, "Modeling Chemical Process Systems via Neural Computation," *IEEE Control Sys. Mag.*, **10**, 24 (1990).

Bishop, C., "Improving the Generalization Properties of Radial Basis Function Neural Networks," *Neural Comput.*, **3**, 579 (1991).

Bishop, C., "Exact Calculation of the Hessian Matrix for the Multilayer Perceptron," *Neural Comput.*, **4**, 494 (1992).

Buntine, W. L., and A. S. Weigend, "Bayesian Back-propagation," *Complex Systems*, **5**, 603 (1991).

Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Math. Control, Signals Sys.*, **2**, 303 (1989).

Di Massimo, C., G. A. Montague, M. J. Willis, M. T. Tham, and A. J. Morris, "Towards Improved Penicillin Fermentation via Artificial Neural Networks," *Computers Chem. Eng.*, **16**(4), 283 (1992).

Duan, N., and K.-C. Li, "Slicing Regression: A Link-Free Regression Method," *The Annals of Statistics*, **19**, 505 (1991).

Eubank, R., *Spline Smoothing and Non-parametric Regression*, Marcel Dekker, New York (1988).

Fiacco, A. V., and K. O. Kortanek, *Semi-Infinite Programming and Applications*, Springer-Verlag, Berlin (1983).

Friedman, J. H., "Multivariate Adaptive Regression Splines," with discussion, *Annals of Statistics*, **19**(1), 1 (1991).

Friedman, J., and W. Stuetzle, "Projection Pursuit Regression," *J. of Amer. Statis. Assoc.*, **76**, 817 (1981).

Gallant, A. R., *Nonlinear Statistical Models*, Wiley, New York (1987).

Gallant, A. R., and G. H. Golub, "Imposing Curvature Restrictions on Flexible Functional Forms," *J. of Econometrics*, **20**, 285 (1984).

Gull, S. F., "Developments in Maximum Entropy Data Analysis," *Maximum Entropy and Bayesian Methods*, J. Skilling, ed., Kluwer Academic Publishers, 53 (1989).

Härdle, W., *Applied Nonparametric Regression*, Cambridge University Press, Cambridge (1990).

Hastie, T. J., and S. Tibshirani, *Generalized Additive Models*, Chapman and Hall, London (1990).

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

Joerding, W. H., and J. L. Meador, "Encoding A Priori Information in Feedforward Networks," *Neural Networks*, **4**, 847 (1991).

Johansen, T. A., and B. A. Foss, "Representing and Learning Unmodelled Dynamics with Neural Network Memories," *Proc. Amer. Control Conf.*, 3703 (1992a).

Johansen, T. A., and B. A. Foss, "Nonlinear Local Model Representation for Adaptive Systems," *IEEE Int. Conf. on Intell. Control & Instr.*, Singapore (1992b).

Johnston, L. P. M., and M. A. Kramer, "Probability Density Estimation Using Elliptical Basis Functions," *AICHE J.*, accepted (1994).

Jordan, M. I., "Constrained Supervised Learning," *J. Math. Psych.*, **36**, 396 (1992).

Jordan, M. I., and D. E. Rumelhart, "Forward Models: Supervised Learning with a Distal Teacher," *Cognitive Sci.*, **16**, 307 (1992).

Kramer, M. A., M. L. Thompson, and P. M. Bhagat, "Embedding Theoretical Models in Neural Networks," *Proc. Amer. Control Conf.*, 475 (1992).

Leonard, J. A., and M. A. Kramer, "Improvement of the Backpropagation Algorithm for Training Neural Networks," *Comp. Chem. Eng.*, **14**, 337 (1990).

MacKay, D. J. C., "Bayesian Interpolation," *Neural Comput.*, 4, 415 (1992a).

MacKay, D. J. C., "A Practical Bayesian Framework for Backpropagation Networks," *Neural Comput.*, **4**, 448 (1992b).

MacKay, D. J. C., "The Evidence Framework Applied to Classification Networks," *Neural Comput.*, **4**, 720 (1992c).

Mavrovouniotis, M. L., and S. Chang, "Hierarchical Neural Networks for Process Monitoring," *Comp. Chem. Eng.*, **16**(4), 347 (1992).

McCullagh, P., and J. A. Nelder, *Generalized Linear Models*, Chapman and Hall, London (1989).

Moddy, J., and C. J. Darken, "Fast Learning of Locally-Tuned Processing Units," *Neural Comput.*, **1**, 281 (1989).

Poggio, T., and F. Girosi, "Networks for Approximation and Learning," *Proc. IEEE*, **78**(9), 1481 (1990a).

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

Pollard, J. F., M. R. Brousard, D. B. Garrison, and K. Y. San, "Process Identification Using Neural Networks," *Comp. Chem. Eng.*, **16**(4), 253 (1992).

Psichogios, D. C., and L. H. Ungar, "A Hybrid Neural Network-First Principles Approach to Process Modeling," *AIChE J.*, **38**(10), 1499 (1991).

Sibisi, S., "Regularization and Inverse Problems," *Maximum Entropy and Bayesian Methods*, J. Skilling, ed., Kluwer Academic Publishers, 389 (1989).

Skilling, J., "Classic Maximum Entropy," *Maximum Entropy and Bayesian Methods*, J. Skilling, ed., Kluwer Academic Publishers, 45 (1989).

Skilling, J., and S. F. Gull, "Prior Knowledge Must Be Used," *Maximum-Entropy and Bayesian Spectral Analysis and Estimation Problems*, C. R. Smith and G. J. Erickson, eds., 161 (1987).

Su, H.-T., N. Bhat, P. A. Minderman, and T. J. McAvoy, "Integrating Neural Networks with First Principles Models for Dynamic Modeling," *IFAC Symp. on Dynamics and Control of Chemical Reactors, Distillation Columns, and Batch Processes*, (DYCORD+) (1992).

Tanaka, Y., M. Fukushima, and T. Ibaraki, "A Comparative Study of Several Semi-Infinite Nonlinear Programming Algorithms," *Euro. J. Oper. Res.*, **36**, 92 (1988).

Thompson, M. L., "System Analysis, Control and Optimization of the Fed-Batch Penicillin Fermentation," MS Thesis, Chemical Engineering Dept., Massachusetts Institute of Technology (1984).

Weiss, S. M., and C. A. Kulikowski, *Computer Systems that Learn*, Morgan Kaufmann, San Mateo, CA (1991).

Appendix A: True Penicillin Fermentation Model

Governing differential equations

The balance equations on cell biomass, residual substrate in the fermentor broth, product and overall mass are the governing equations of the fermentation (Thompson, 1984). These equations were solved numerically to generate the simulated training and test data:

Cell biomass balance:

$$dX/dt = X(\mu - D - c_L) \tag{A1}$$

Substrate balance:

$$dS/dt = -\sigma X + (S_f - S)D \tag{A2}$$

Product balance:

$$dP/dt = q_p X - P(D + K) \tag{A3}$$

Overall mass balance:

$$dV/dt = F \tag{A4}$$

Specific rate correlations

The correlations and the parameters for the true model appear below. They are an adaptation of the Bajpai-Reuss model used by Thompson (1984):

Specific growth rate:

$$\mu = \mu_m S/(K_x X + 10) \tag{A5}$$

Specific cell lysis rate:

$$c_L = c_{Lm} X \exp(-S/100)/(K_L + X + 1) \tag{A6}$$

Specific substrate consumption rate:

$$\sigma = (\mu/Y_{x/s} + q_p/Y_{p/s} + m_x) \tag{A7}$$

Specific product formation rate:

$$q_p = 1.5 q_{pm} SX/\{4K_p + XS[1 + S/(3K_I)]\} \tag{A8}$$

Maintenance energy:

$$m_x = m_{xm} X/(X + 10) \tag{A9}$$

Appendix B: Default Penicillin Fermentation Model

Specific rates

To illustrate incomplete knowledge of the process, the default model specific rate correlations used different parameters than the true process. Cell lysis is completely ignored. The specific growth correlation (Eq. B1) neglects dependence of μ upon X and allows a larger maximum μ. The result is overestimation of biomass production. The specific substrate consumption correlation (Eq. B2) uses different yields on substrate and maintenance energy; and the default assumes constant maintenance energy. The result is that less substrate is consumed for the same amount of biomass production. The specific product formation correlation (Eq. B3) uses a reduced maximum product formation rate and gives substrate more of an inhibitory effect on production. The result is underestimation of product formation:

Specific growth rate:

$$\mu = (1.2\mu_m)S/(5Kx + S) \qquad \text{(B1)}$$

Specific substrate consumption rate:

$$\sigma = \mu/(1.1 Y_{x/s}) + q_p/(0.9 Y_{p/s}) + 0.9 m_{xm} \qquad \text{(B2)}$$

Specific product formation rate:

$$q_p = 0.9 q_{pm} S/\{1.1 K_p + S[1 + S/(0.5 K_I)]\} \qquad \text{(B3)}$$

Appendix C: Output Model

Governing differential equations

The output model is derived by applying simplifying assumptions to the governing differential equations for the true penicillin fermentation model (Eqs. A1-A4) and solving it analytically. The cell biomass balance (Eq. A1) drops the cell lysis term, regarding it as negligible. The other balance equations remain the same:

Cell biomass balance:

$$dX/dt = X(\mu - D) \qquad \text{(C1)}$$

Substrate balance:

$$dS/dt = -\sigma X + (S_f - S)D \qquad \text{(C2)}$$

Product balance:

$$dP/dt = q_p X - P(D - K) \qquad \text{(C3)}$$

Overall mass balance:

$$dV/dt = F \qquad \text{(C4)}$$

Analytical solution and one-step-ahead predictions

The analytical solution is evaluated at one sampling time interval into the future for one-step-ahead prediction. The solution was obtained by applying the following assumptions and solving Eqs. C1-C4.

Simplifying assumptions:

(i) Constant specific growth rate, μ
(ii) Specific product formation rate decays: $q_p(t) = q_p(t_0) + k(u)(t - t_0)$
(iii) Constant specific substrate consumption, σ
(iv) Constant substrate feed concentration, S_f
(v) Constant substrate feed rate, F.

Analytical solution:

Assumption v allows us to solve Eq. C4, giving the fermentor working volume as a simple linear function of time (Eq. C5). Substituting this into the definition of dilution rate ($D = F/V$) and solving Eq. C1 gives the expression for the biomass concentration (Eq. C6). The resulting formula for X, along with D, was substituted into Eqs. C3 and C4 to arrive at solutions for residual substrate concentration (Eq. C7) and product concentration (Eq. C8), respectively.

Fermentor working volume:

$$V(t+\Delta t) = V(t) + F\Delta t \qquad \text{(C5)}$$

Biomass concentration:

$$X(t+\Delta t) = X(t)\exp(\mu\Delta t)/[1 + D(t)\Delta t] \qquad \text{(C6)}$$

Residual substrate concentration:

$$S(t+\Delta t) = S_f(t) - [S_f(t) - S(t)]/[1 + D(t)\Delta t] - X(t+\Delta t)[1 - \exp(-\mu\Delta t)]\sigma/\mu \qquad \text{(C7)}$$

Product concentration:

$$P(t+\Delta t) = X(t+\Delta t)\{[P(t)/X(t) - q_p/(\mu + K)] \exp[-(\mu + K)\Delta t] + q_p/(\mu + K)\} \qquad \text{(C8)}$$

Manuscript received May 24, 1993, and revision received Sept. 29, 1993.