Acta Agraria Kaposváriensis (2004) Vol 8 No 3, 205-218
Kaposvári Egyetem, Állattudományi Kar, Kaposvár
University of Kaposvár, Faculty of Animal Science, Kaposvár

New Approaches to the Identification of Semi-mechanistic Process Models

J. Madár, J. Abonyi, F. Szeifert
University of Veszprém, Department of Process Engineering, P.O. Box 158, H-8201, Hungary

ABSTRACT

In process engineering, first-principles models derived from dynamic mass, energy and momentum balances are mostly used. When the process is not perfectly known, the unknown parts of the first principles model should be represented by black-box models, e.g. by neural networks. This paper is devoted to the identification and application of such hybrid models. For the identification of the neural network elements of a hybrid model two methods are investigated in this article: back-propagation algorithm and direct optimization. We study three optimization algorithms: Sequential Quadratic Programming, Evolutionary Strategy and Particle Swarm Optimization. The different algorithms are compared in a case study, the baker's yeast production process.
(Keywords: Hybrid models, Artificial Neural Networks, Baker's yeast fermentation)

ÖSSZEFOGLALÁS

Új módszerek a szemi-mechanisztikus modellek identifikációjára
Madár J., Abonyi J., Szeifert F.
Veszprémi Egyetem, Folyamatmérnöki Tanszék, Veszprém, 8201 Pf. 158.

A folyamatmérnöki gyakorlatban a tömeg-, energia- és impulzusmérlegen alapuló fehér-doboz modellek terjedtek el. Ha azonban a folyamat teljes egészében nem ismert, akkor az ismeretlen részeket fekete-doboz modellekkel kell leírni. Erre különösen alkalmas a mesterséges neurális hálózat. Ennek a munkának a célja olyan módszerek bemutatása, amik alkalmasak az ilyen hibrid nemlineáris modellek identifikációjára. Ebben a cikkben két identifikációs módszert vizsgáltunk meg: a back-propagation algoritmust és a direkt optimalizációt. Megvizsgáltuk a legelterjedtebb nemlineáris optimalizáló technikák használhatóságát, mint a klasszikus Sequential Quadratic Programming, Evolutionary Strategy és a Particle Swarm Optimization. Ezeket a technikákat egy alkalmazási példán keresztül mutatjuk be, amiben a kenyérélesztő fermentációs eljárást modelleztük.
(Kulcsszavak: hibrid modellek, neurális hálózatok, kenyérélesztő fermentációs eljárás)

INTRODUCTION

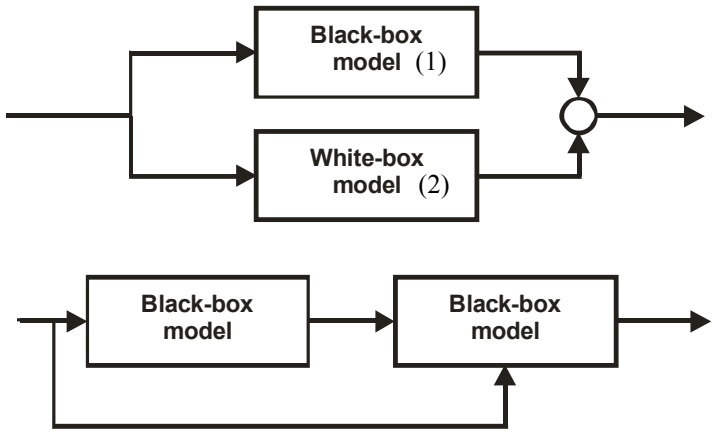
In process engineering two main classes of modeling methods exist: 1. *white-box* or *first principle* modeling (*Hangos*, 2001), 2. *black-box* modeling (*Sjöberg*, 1995). In chemical engineering, first-principles models derived from dynamic mass, energy and momentum balances are mostly used. But the application of first principle models can fail because sometimes the model parameters are not known, or the state variables of the process cannot be measured or we do not know the process well. In these cases the data-driven

black-box modeling strategy should be used instead of the white box strategy. Unfortunately, completely data-driven black-box identification techniques often yield unrealistic models (*Abonyi*, 1999). This is typically due to an insufficient information content of the identification data and due to overparameterization of the models. Another disadvantage is the non-scalability of black box models, i.e., one has to collect new training-data when the process is modified.

Due to the given drawbacks, combinations of *a priori* knowledge with black-box modeling techniques are gaining considerable interest. Two different approaches can be distinguished: *grey box* modeling and *semi-mechanistic* modeling. In a *grey box* model, a priori knowledge or information enters the black-box model as constraints on the model parameters or variables, smoothness of the system behavior, or open-loop stability (*Tulleken*, 1993). A major drawback of this approach is that it may suffer from the same drawbacks as the black-box model, i.e. no extrapolation is possible and time-varying processes remain a problem. One may also start by deriving a model based on first-principles and then include black-box elements as parts of the white-box model frame (*Thompson*, 1994 and *Psichogios*, 1992). This modeling approach is usually denoted as *hybrid-modeling* or *semi-mechanistic modeling*. The latter term is used in the sequel because the first one is rather confusing with other methods. Johansen (*Johansen*, 1994) describes various techniques to incorporate knowledge into black-box models. Thompson and Kramer (*Thompson*, 1994) used the so-called parallel approach of semi-mechanistic modeling, see *Figure 1*. Here, the error between the data and the white-box model is represented by a neural network. They also describe the serial approach where a neural network is used to model unknown parameters. Several authors applied hybrid models for the modeling of biotechnological systems (*Simutis*, 1997 and *Psichogios*, 1992).

Figure 1

Parallel and serial combinations of white and black box models



1. ábra: Párhuzamos és soros kombinációja a fehér és fekete doboz modelleknek

Fekete doboz modell(1), Fehér doboz modell(2)

In this paper, we focus on *serial semi-mechanistic* models where an *artificial neural network* is used to represent, in a mechanistic sense, difficult-to-model parts of our

system. The application of neural networks is also a well studied idea, as data based modeling offers an alternative to rigorous modeling based on physical principles. This hybrid modeling approach includes the following steps:

- Design the structure of the hybrid model.
- Design the structure of the black-box element (neural network) of the hybrid model.
- Make experiments on the modeled plant; collect input-output process data.
- Train the black-box element for the dynamics of the plant based on the measurements (identification of black-box element).

The aim of this work is to show how the neural networks can be efficiently incorporated into the semi-mechanistic modeling environment and to propose new approaches for identification of neural networks. We propose two methods for identification of hybrid models: the back-propagation algorithm and the direct optimization technique. As the classical identification of the neural network should be based on input-output process data and the neural network is incorporated into a differential equation system, the identification of neural network is not an easy task. In this paper, it will be shown how the proposed methods can be applied for identifying a hybrid-model. Three optimization algorithms will be presented and compared in the paper: the classical Sequential Quadratic Programming (SQP), the Evolutionary Strategy (ES) and the Particle Swarm Optimization. The hybrid modeling and the identification techniques are illustrated on an application example where we model the baker's yeast production process.

HYBRID MODELS BASED ON NEURAL NETWORKS

The advantage of white box models is that they have the capability to explain the underlying mechanistic relationships of the process. These models are, applicable independently of the process scale to a certain extent. Furthermore, these models are easy to modify or extend by changing parts of the model. Generally, models of lumped process systems in chemical engineering are formulated in a form of differential equations:

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, u) \\ y &= h(\mathbf{x}) \end{aligned} \tag{1}$$

where $\mathbf{x}$ is the state-variable vector, y is the output, u is the input (we assume single input - single output systems). These differential equations are formulated by macroscopic balance equations, for instance, mass or energy balances. These balances are based on conservation principle that leads to differential equations written as

$$\begin{bmatrix}\text{accumlation}\\ \text{of x}\end{bmatrix} = \begin{bmatrix}\text{inflow}\\ \text{of x}\end{bmatrix} - \begin{bmatrix}\text{outflow}\\ \text{of x}\end{bmatrix} + \begin{bmatrix}\text{amount of}\\ \text{x generated}\end{bmatrix} - \begin{bmatrix}\text{amount of}\\ \text{x consumed}\end{bmatrix} \tag{2}$$

where x is a certain conserved extensive quantity, for example mass or energy.

In a practical situation, the construction of white-box models can be difficult because the underlying mechanisms may not be completely clear, experimental results obtained in the laboratory do not carry over to practice, or parts of the white-box models are in fact not known. This means that sometimes not all of the terms in (2) are exactly or even partially known. In semi-mechanistic modeling, black-box models, like neural networks, are used to represent the otherwise difficult-to-obtain parts of the model. One of the advantages of the semi-mechanistic modeling strategy to the black-box modeling is that it seems to be more promising with respect to extrapolation properties (*Can*, 1996). Usually, the reaction rates (the amount of generated and consumed materials in

chemical reactions) and the thermal effects (e.g. reaction heat) are especially difficult to model, but the first two transport terms (inlet and outlet flows) in (2) can be obtained easily and accurately. Hence, in the modeling phase it reveals which parts of the first principles model are easier and which are more laborious to obtain and often we can get the following hybrid model structure:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}_{HYB}\left(\mathbf{x}, u, \mathbf{f}_{NN}\left(\mathbf{z}, \boldsymbol{\theta}\right)\right) \\ y &= h\left(\mathbf{x}\right) \end{aligned} \tag{3}$$

where **x**, u and y denote state vector, manipulated input and output of the system; $\mathbf{f}_{NN}$ is the black-box element of the model which models the "difficult-to-model" part of the system; the **z** and $\boldsymbol{\theta}$ are input vector and parameter set of $\mathbf{f}_{NN}$.

In this paper, the black box element is represented by a feed-forward multi-input multi-output neural network with one hidden layer. The neurons of hidden layer have *tanh* activation functions, the neurons of output layer have linear functions:

$$\mathbf{f}_{NN}\left(\mathbf{z}, \boldsymbol{\theta}\right) = \mathbf{W}_2 \tanh\left(\mathbf{W}_1 z + \mathbf{b}_1\right) + \mathbf{b}_2 = \sum_{j=1}^{nn}\left(\mathbf{w}_{2j} \tanh\left(\mathbf{w}_{1j} z + b_{1j}\right)\right) + \mathbf{b}_2 \tag{4}$$

where *nn* is the number of hidden neurons, $\mathbf{z}=[z_1,\ldots,z_{ni}]^{\mathrm{T}}$ is the input of network, $\mathbf{W}_1$ is the weight matrix of hidden layer, $\mathbf{b}_1$ is the bias vector of hidden layer, $\mathbf{W}_2$ is the weight matrix of output layer, $\boldsymbol{b}_2$ is the bias vector of output layer. $\boldsymbol{\theta}$ contains all of the parameters: $\boldsymbol{\theta} = \{\mathbf{W}_1, \mathbf{w}_2, \mathbf{b}_1, b_2\}$. The number of output neurons of the neural network is determined by the structure of hybrid model and by the modeler.

IDENTIFICATION OF NEURAL NETWORKS OF HYBRID MODELS

The identification of neural networks, so-called training of neural networks, means determining the parameters of networks. The classical algorithm for training is the back propagation algorithm which uses a training input-output data. We propose another method, the direct optimization technique. For direct optimization several optimization algorithms can be applied; in this article the Evolutionary Strategy or Particle Swarm Optimization will be presented.

Back-propagation algorithm

The classical back-propagation algorithm uses a given input-output data set (training data set) in such a way that the sum of the squared deviations, $\mathbf{V}_N = \frac{1}{2N}\sum_{k=1}^{N}\left(y(k) - \hat{y}(k)\right)^2$ between the predicted output of the network (from the training input data) and the corresponding training output data becomes minimal. The usual way to minimize $\mathbf{V}_N$ is to use gradient procedures, like Gauss-Newton algorithm. Weights in the *i*-th step of this iterative process are changed in the direction of gradient (for single output neural network):

$$\begin{aligned} \boldsymbol{\theta}^{i+1} &= \boldsymbol{\theta}^{i} - \mu \mathbf{R}_i^{-1} \mathbf{V}'_N \\ \mathbf{R}_i &= -\frac{1}{N}\sum_{k=1}^{N} \mathbf{j}(k, \boldsymbol{\theta}) \mathbf{j}^{\mathrm{T}}(k, \boldsymbol{\theta}) \\ \mathbf{V}'_N &= -\frac{1}{N}\sum_{k=1}^{N} \mathbf{j}(k, \boldsymbol{\theta})\left(f_{NN}(k) - \hat{f}_{NN}(k)\right) \end{aligned} \tag{5}$$

$$\mathbf{j}(k,\boldsymbol{\theta}) = \frac{\partial f_{NN}(k)}{\partial \boldsymbol{\theta}} \tag{6}$$

where $f_{NN}(k)$ is the output of network for the *k*-th data point, $\hat{f}_{NN}(k)$ is the corresponding desired (measured) output, *N* is the number of data points, $\boldsymbol{\theta}$ is the parameter vector of network.

The disadvantage of this algorithm is that the neural network is incorporated into the hybrid model (3), so the **x** state variables of the system can be measured directly instead of the desired network output. Hence to apply this algorithm, one must estimate the corresponding desired outputs from the measurement:

$$\hat{f}_{NN}(k) = f_{HYB}^{-1}(\mathbf{x}(k), u(k), \dot{\mathbf{x}}(k)) \tag{7}$$

where $\mathbf{x}(k)$ and $u(k)$ are the *k*-th measured state variable vector and manipulated input. The $\dot{\mathbf{x}}(k)$ is the derivative $\mathbf{x}(k)$, and it can be calculated from the measurement, e.g. using Euler discretization: $\dot{\mathbf{x}}(k) = (x(k+1) - x(k))/T_0$. Certainly, if the T_0 sampling time is large and/or there is relatively high measurement noise, the performance of the whole algorithm will the decrease because the neural network will be trained to estimate the average reaction rates over sampling periods. The accuracy of the algorithm can be improved by generating new data points by an interpolation algorithm, e.g. linear interpolation or cubic-spline interpolation.

The advantage of this algorithm is that it is fast, so this algorithm can be used for generating initial points for slower direct optimization algorithms, which will be presented in the following subsections.

Direct optimization of neural network parameters

The identification of the neural network parameters can be formulated as a classical optimization problem, where the optimization goal is to minimize the square error between measured output and the hybrid model output:

$$\min_{\boldsymbol{\theta}} \sum_{k=1}^{N} \left\| \hat{\mathbf{y}}(k) - \mathbf{y}(k) \right\| \tag{8}$$

where $\mathbf{y}(k)$ is the output of (3) hybrid model at the *k*-th time instant, $\hat{\mathbf{y}}(k)$ is the measured output of system in the *k*-th time instant, *N* is the number of data points. The hybrid model output can be calculated in *n-step ahead* mode. Certainly, if one chooses *n=N*, the *n-step ahead* mode will be equal to *free-run mode*; and if one chooses *n=1*, it will be equal to *one-step ahead* mode. If one has few measurement point, the one-step ahead mode is recommended, but if there is a large measurement noise, bigger *n* values may be better. In the application example, the free-run mode was used. There are several optimization algorithms which can be used in this method. In this article we investigated three algorithms: Sequential Quadratic Programming (SQP), Evolutionary Algorithms (ES) and Particle Swarm Optimization (PSO). In the following the ES and PSO algorithms will be presented.

Evolutionary Strategy (ES)

The Evolutionary Strategy (*Rechenberg*, 1973, *Spears*, 1993) is an Evolutionary Algorithm (EA). EAs work with a population of potential solutions to a problem, where each individual within the population represents a particular solution. Every individual has a fitness value which expresses how good the solution is at solving the problem. *Table 1* outlines this optimization algorithm.

Table 1

Evolutionary algorithm

```
procedure EA (1);          {
        Initialize population (2);
        Evaluate all individuals (3);
        while (not terminate) do (4)          {
                Select individuals (5);
                Create offspring from selected individuals (6)
                        using Recombination and Mutation (7);
                Evaluate offspring (8);
                Replace some old individuals by some offspring (9);
        }
}
```

1. táblázat: Evolúciós algoritmus

Eljárás EA(1), Kezdeti populáció generálása(2), Összes egyed Fitness-értékének kiszámítása(3), ciklus amíg(nem leállási-feltétel)(4), Egyedek kiválasztása(5), Új egyedek generálása a kiválasztottakból(6), Rekombináció és mutáció operátorral(7), Összes egyed Fitness-értékének kiszámítása(8), Régi egyedek kicserélése az új egyedekkel(9)

A population of individuals is randomly initialized and then evolved from generation to generation by repeated applications of evaluation, selection, mutation and recombination. In the selection step, the algorithm selects the parents of the next generation. After the selection of the individuals, the new individuals of the next generation (also called offspring) are created by recombination and mutation. The recombination (also called crossover) exchanges information between two selected individuals to create one or two new offspring. The mutation operator makes small, random changes to the individual. The final step of the evolutionary procedure is the replacement, when the new individuals are inserted into the new population. Once the new generation has been constructed, the processes that result in the subsequent generation of the population are begun once more.

The individuals in ES are represented by n-dimensional vectors, often referred to as object variables. To allow for a better adaptation to the particular optimization problem, the object variables are accompanied by a set of the so-called strategy variables. Hence, an individual $\mathbf{a}_j$ consists of two components, the object variables $\mathbf{x}_j$, and strategy variables $\boldsymbol{\sigma}_j$.

As mutation operator normally distributed random numbers are added to the individuals:

$$x_{j,i} = x_{j,i} + N\left(0, \sigma_{j,i}\right) \tag{9}$$

Before the update of the object variables, the strategy variables are also mutated using a multiplicative normally distributed process:

$$\sigma_{j,i}^{(t)} = \sigma_{j,i}^{(t-1)} \exp\left(\tau' N(0,1) + \tau N_i(0,1)\right) \tag{10}$$

with as a global factor which allows an overall change of the mutability and allowing for individual changes of the mean step sizes. The parameters can be interpreted in the sense of global learning rates. *Schwefel* (1995) suggests setting them as

$$\tau' = \frac{1}{\sqrt{2n}} \quad \tau = \frac{1}{\sqrt{2\sqrt{n}}} \tag{11}$$

Recombination in ES can be either sexual (local), where only two parents are involved in the creation of an offspring, or global, where up to the whole population contributes to a new offspring. Traditional recombination operators are discrete recombination, intermediate recombination, and geometric recombination, all existing in a sexual and global form. When F and M denote two randomly selected individuals from the μ parent population, the following operators can be defined:

$$x_i' = \begin{cases} x_{F,i} & \text{no recombination} \\ x_{F,i} \text{ or } x_{M,i} & \text{discrete} \\ \left(x_{F,i} + x_{M,i}\right)/2 & \text{intermediate} \\ \sum_{k=1}^{\mu} x_{k,i} / \mu & \text{global avarge} \end{cases}$$

$$\sigma_i' = \begin{cases} \sigma_{F,i} & \text{no recombination} \\ \sigma_{F,i} \text{ or } \sigma_{M,i} & \text{discrete} \\ \left(\sigma_{F,i} + \sigma_{M,i}\right)/2 & \text{intermediate} \\ \sum_{k=1}^{\mu} \sigma_{k,i} / \mu & \text{global avarge} \end{cases} \tag{12}$$

The selection is stochastic in the ES. First we chose the best μ individuals to be parents, and then we select the parent-pairs uniformly randomly from these individuals.

Particle Swarm Optimization (PSO)

The particle swarm concept originated as a simulation of a simplified social system. The original intent was to graphically simulate the choreography of a bird block or a fish school. However, it was found that a particle swarm model can be used as an optimizer. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is. So what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food. Particle swarm optimization (PSO) is based on this scheme. This stochastic optimization technique has been developed by *Kennedy* (1995) and *Eberhart* (1998). In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. All of the particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called pbest. Another "best" value

that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gbest.

When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

$$\mathbf{v}_j(k+1) = w \cdot \mathbf{v}_j(k) + c_1 \cdot rand() \cdot \left(\mathbf{x}_{pbest} - \mathbf{x}_j(k)\right) + c_2 \cdot rand() \cdot \left(\mathbf{x}_{gbest} - \mathbf{x}_j(k)\right)$$
$$\mathbf{x}_j(k+1) = \mathbf{x}_j(k) + \mathbf{v}_j(k+1) \cdot dt \quad (13)$$

where v is the particle velocity, present is the current particle (solution), pbest and gbest are defined as stated before, rand() is a random number between [0,1), c_1, c_2 are learning factors usually $c_1 = c_2 = 2$. *Table 2* shows the pseudo code of the PSO algorithm.

Table 2

PSO algorithm

```
procedure PSO (1); {
        Initialize particles (2);
        while (not terminate) do (3){
                for each particle (4){
                        Calculate fitness value (5);
                        if fitness < pBest than  pBest = fitness (6);
                }
                Choose the best particle as the gBest (7);
                for each particle (8){
                        Calculate particle velocity (9);
                        Update particle position (10);
                }
        }
}
```

2. táblázat: PSO algoritmusa

Eljárás PSO(1), Kezdetei részecskék generálása(2), Ciklus amíg(nem leállási-feltétel)(3), Mindegyik részecskére(4), Fitness-érték számítása(5), Ha fitness < pBest Akkor pBest = fitness(6), A legjobb részecske kiválasztása, mint gBest(7), Mindegyik részecskére(8), Részecske sebesség-vektorának számítása(9), Részecske helyének módosítása(10)

The role of the, w, inertia weight in (13), is considered critical for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current one. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration, i.e. fine–tuning the current search area. In this work, we used a compromise solution: the weight parameter linearly decreased during the optimization from a maximum value to a minimum value.

The ES and PSO have several tuning parameters. The *Table 3* shows the parameters which we used in this work.

Table 3

ES and PSO parameters

	Parameter	Value/Setting(10)
ES:	Population size(1)	40
	Number of parents(2)	10
	Selection / Replacement(3)	(μ + λ)
	Recombination of design variables(4)	Discrete sexual(5)
	Recombination of strategy variables(6)	Intermediate sexual(7)
	τ'	$\frac{1}{\sqrt{2n}}$
	τ	$\frac{1}{\sqrt{2n}}$
PSO:	Population size(1)	30
	c1	2
	c2	2
	w at the start(8)	0.95
	w at the end(9)	0.4

3. táblázat: ES és PSO paraméterei

Populáció mérete(1), Szülők száma(2), Kiválasztás/kicserélés(3), Tervezési változók rekombinációja(4), Diszkrét szexuális(5), Stratégiai változók rekombinációja(6), Köztes (átlagoló) szexuális(7), w az első iterációban(8), w az utolsó iterációban(9), Érték/Típus(10)

APPLICATION EXAMPLE

This example is based on the baker's yeast production process. The model (*Oliveria*, 2004) considers three pathways for carbon source utilization that may be lumped together into three macroscopic reactions. The model of the system is the following:

$$\frac{d}{dt}\begin{bmatrix} X \\ S \\ E \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -k_1 & -k_2 & 0 \\ 0 & k_3 & -k_4 \end{bmatrix} \begin{bmatrix} \mu_S^O \\ \mu_S^R \\ \mu_E^O \end{bmatrix} - D\begin{bmatrix} X \\ S \\ E \end{bmatrix} + \begin{bmatrix} 0 \\ DS_{in} \\ 0 \end{bmatrix} \qquad (14)$$

where *X*, *S*, *E* are the biomass, sugar and ethanol concentration, and the μ_S^O, μ_S^R and μ_E^O are the reaction kinetics (see *Oliveria,* 2004). The bioreactor operates in fed-batch, the dilution rate $D = F/V$, and the volume balance is: $dV/dt=F$. The X and S initial concentrations were generated uniformly randomly: $X(0)$=0-1 g/l, $S(0)$=0-2 g/l, the initial ethanol concentration and volume were $E(0)$=1 g/l and $V(0)$=1 l. The feed rate was generated from the uniform distribution in the range of 0-1 l/h with 2 h frequency. The feed sugar concentration was S_{in}=50 g/l, the oxygen concentration is controlled 0,1 g/l.

This reaction kinetic of baker's yeast fermentation process is difficult, so we assumed that it is not known. Hence in this example, the first principle model was used as a 'real' plant, but for modeling of the process we consider the application of a hybrid

model. Because the critical part of the model is the chemical reaction, we constructed a hybrid model, where the chemical reaction is modeled by neural network:

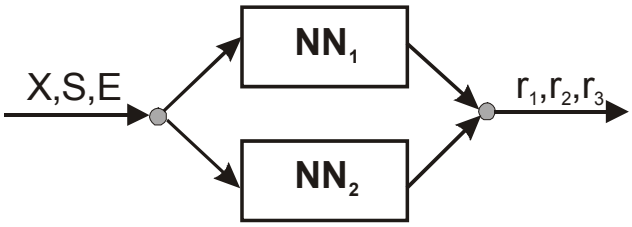
$$\frac{d}{dt}\begin{bmatrix} X \\ S \\ E \end{bmatrix} = \mathbf{f}_{NN}(\mathbf{z}) - D\begin{bmatrix} X \\ S \\ E \end{bmatrix} + \begin{bmatrix} 0 \\ DS_{in} \\ 0 \end{bmatrix} \tag{15}$$

where the output neural network is a vector of calculated reaction rates. The input of neural network is the current biomass, sugar and ethanol concentrations $z = [X(k), S(k), E(k)]$. The neural network (NN) was a feed-forward network and it consists of five hidden layer nodes with tangent sigmoid transfer functions, and three output layer node with linear transfer function.

It is well known that during the fermentation process, the reaction kinetic can switch between aerobic and fermentative metabolic states. Hence we applied two NNs, one for fermentative metabolic state and one for aerobic. In the first-principle model, the fermentative state changes to the aerobic state when the sugar uptake rate becomes smaller than the oxygen uptake rate. But this relationship can not be used in the hybrid model, because we assume that we do not know the reaction kinetics. But we know that the oxygen concentration is controlled during the process, so the type of metabolic state depends on sugar concentration. Hence we choose the simple rule for the hybrid model: if the sugar concentration is bigger than 0.05 g/l, then the first neural network is used, and if it is smaller, the other neural network is used, see *Figure 2*. (The 0.05 g/l is an approximately value, which can be estimated from the experiments.)

Figure 2

Structure of neural networks for the application example
If: S>0.05 g/l than NN_1 else NN_2



2. ábra: A neurális hálózat modell struktúrája az alkalmazási példában (Ha S>0,05 g/l, akkor NN_1, különben NN_2)

For the training of NNs, four different data-sets were used. These data-sets contained measurement data of 20 experiments of simulated plant. Every experiment took 18 hours long, and these experiments were different in initial concentrations and the inlet flow-rate profiles. The four data-sets:

- Data-set 1: sample time is 1 hour, the relative measurement noise level is 0%
- Data-set 2: sample time is 1 hour, the relative measurement noise level is 5%
- Data-set 3: sample time is 2 hour, the relative measurement noise level is 0%
- Data-set 4: sample time is 2 hour, the relative measurement noise level is 5%

The measurement noise was generated from normally distributed pseudo random numbers. The performances of identified hybrid models were measured by mean-absolute-error between 'real' X, S and E concentration trajectories and the estimated trajectories by hybrid models (in free-run mode). The performance values were calculated based on two independent experiments, which were different from the 20 training experiments.

Table 4

Estimation error by identified hybrid models (Back-propagation method)

	Data set(1)			
	1	**2**	**3**	**4**
No interpolation(2)	0.41	0.29	1.12	2.85
Linear interpolation(3)	0.22	0.28	0.74	0.84
Spline interpolation(4)	0.14	0.27	2.73	2.63

Remark (*megjegyzés*)*:* In free-run mode. (*Szabad futású szimulációval.*) Mean absolute error of concentrations (g/l). (*A koncentrációk átlagos abszolút hibája (g/l).*)

4. táblázat: Az identifikált hibrid-modell becslési hibája (direkt optimalizálással)

Adat halmaz(1), Nincs interpoláció(2), Lineáris interpoláció(3), Spline interpoláció(4)

Table 5

Estimation error by identified hybrid models (Direct optimization method)

	Data set(1)			
	1	**2**	**3**	**4**
SQP	0.10	0.10	0.31	0.26
ES	0.14	0.27	0.74	0.84
PSO	0.14	0.27	0.74	0.84

Remarks (*megjegyzés*): In free-run mode. (*Szabad futású szimulációval.*) Mean absolute error of concentrations (g/l). (*A koncentrációk átlagos abszolút hibája (g/l).*) The result of the ES and PSO are the same as the initial points of them (from back-propagation). (*Az ES és a PSO algoritmusok nem konvergáltak, így nem volt eredményük.*)

5. táblázat: Az identifikált hibrid-modell becslési hibája (direkt optimalizálással)

Adat halmaz(1)

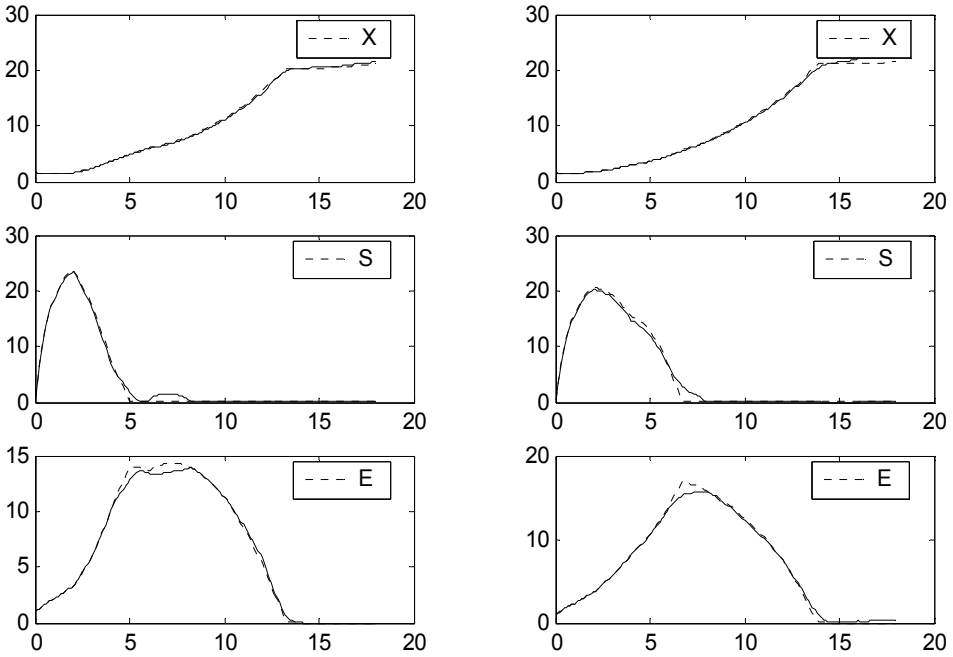
Table 4 shows some comparisons of the numerical results of back-propagation training algorithm without interpolation, with linear interpolation and with cubic spline interpolation. In the first case, there was not interpolation, so the training data only contained the measurement data (with 1 or 2 hour sample time). In the second and third case, we applied interpolation algorithms to estimate the X, S, and E concentration values at every 0.1 hour, so the training data contained not only the measurement data, but the interpolated values too. The initial point (initial weights of neural networks) was

generated randomly, and we maximized the number of iterations to 100, which was enough for the algorithm to converge into a local minima.

The numerical results show that when the sample time is not too large (1 h), the spline interpolation is the best technique, but when the sample time is large (2 h) the linear interpolation is the best. The reason for this is that when the sample time is large, there are few data points and the spline approximation became too inaccurate even compared to linear approximation. An example for the hybrid modeling result can be seen on *Figure 3*.

Figure 3

Free-run simulation result of a hybrid model



Dotted line (s*zaggatott vonal*): real conc. trajectories (*koncentrációk*), Solid line (*sima vonal*): hybrid model output (*hibrid modell eredménye*), X: biomass (*biomassza*), S: sugar (*cukor*), E: ethanol conc. (g/l), (Back propagation – linear interpolation, Data set 1) (*Hiba-visszaterjesztés – Lineáris interpoláció, 1. Adat halmaz*)

3. ábra: Hibrid modell szabad futású szimulációs eredménye

Table 5 shows the numerical results of direct optimization methods. The initial point for the direct optimization algorithms were the resulted hybrid model by back-propagation algorithms with spline interpolation (Data set 1 and 2) and linear interpolation (Data set 3 and 4). Because the direct optimization is slow, we maximized the number of function evaluations in 4000. Unfortunately, the ES and PSO algorithm was not able to find a better solution than the initial point. The reason for this is that the problem space is very large; the number decision variables is 76 (summary the two neural network have 76

parameters). The SQP algorithm was able to optimize the parameters of neural networks, and finally the resulted hybrid models had the ability to model the process very well.

CONCLUSIONS

In chemical and biochemical engineering the white-box modeling method is widely used. But sometimes the process is not perfectly known. To avoid this difficulty, the unknown parts of the first principles model can be represented by black-box models, e.g. by neural networks. This paper is devoted to the identification and application of such hybrid models. For the identification of the neural network elements of the hybrid model two main methods are investigated: back-propagation algorithm and direct optimization by SQP, ES or PSO. These methods are illustrated in a case-study: the baker's yeast production process. We found that the efficiency of the back-propagation algorithm can be improved by interpolating measured data points using linear or cubic spline interpolation methods. On the other hand, we found that the training of the hybrid model can be improved by using direct optimization algorithm which uses the free-run simulation of the hybrid model, however only SQP was able to found better solution than the classical back-propagation algorithm, while ES and PSO was not. The reason why ES and PSO was ineffective is that the problem space was very large for them. The results of the application example show that the proposed hybrid modeling approach works well even if the state variables cannot be measured and the full first-principle model of the control process cannot be obtained.

REFERENCES

Abonyi, J., Babuska, R., Verbruggen, H.B., Szeifert, F. (1999). Constraint parameter Estimation in fuzzy modeling. FUZZ-IEEE'99 Conference, Seoul, Korea, 951-956.

Can, H. van, Braake, H. te, Hellinga, C., Luyben, K., Heijnen, J. (1996). Strategy for dynamic process modeling based on neural networks and macroscopic balances. AIChE Journal, vol. 42. 3403-3418.

Eberhart, R.C., Shi, Y. (1998). Comparison between genetic algorithms and particle swarm optimization., Proceedings of 7th annual conference on evolutionary computation, 611-616.

Hangos, K.M., Cameron, I.T. (2001). Process Modeling and Model Analysis. Academic Press, ISBN: 0121569314

Johansen, T. (1994). Operating regime based process modeling and identification. Ph.D thesis, Department of Engineering Cybernetics, Norwegian Institute of Technology, University of Trondheim, Norway.

Kennedy, J., Eberhart, R.C. (1995). Particle swarm optimization. Proceedings of IEEE International Conference on neural networks, Perth, Australia, 1942-1948.

Oliveria, R. (2004). Combining first principle modelling and artificial neural networks: a general framework., Comp. and Chem. Eng., 755-766.

Psichogios, D.C., Ungar, L.H. (1992). A hybrid neural network - first principles approach to process modeling. AIChE Journal, 1499-1511.

Rechenberg, I. (1973). Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. Stuttgart: Frommann-Holzboog.

Schwefel, H.P, (1995). Numerical Optimization of Computer Models. publisher: Wiley, Chichester.

Simutis, R., Lubbert, A. (1997) Exploratory analysis of bioprocesses using artificial neural network-based methods. Biotechnology Progress, 479-487.

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P.Y., Hjalmarsson, H., Juditsky, A. (1995) Nonlinear black-box modeling in system identification: an unified overview. Automatica, 12. 1691-1724.

Spears, W.M., De Jong, K.A., Back, T., Fogel, D.B., Garis, H. (1993) An Overview of Evolutionary Computation. European Conference on Machine Learning.

Thompson, M.L., Kramer, M.A. (1994). Modeling chemical processes using prior knowledge and neural networks. AIChE Journal, 1328-1340.

Tulleken, H.J.A.F. (1993). Gray-box modelling and identification using physical knowledge and {bayesian} techniques. Automatica, 285-308.

Corresponding author (*levelezési cím*):

Abonyi János
University of Veszprém, Department of Process Engineering
H-8200 Veszprém, P.O.Box 158.
Veszprémi Egyetem, Folyamatmérnöki Tanszék
8200 Veszprém, Pf. 158.
Tel.: +36-88-624 209, fax: +36-88-624 171
E-mail: abonyij@fmt.vein.hu