International Journal of Hybrid Information Technology
Vol. 9, No.9 (2016), pp. 13-22
http://dx.doi.org/10.14257/ijhit.2016.9.9.03

Applying Modified Backpropagation Algorithm to Chemical Reaction System

Byung Joo Kim

Department of Computer Engineering Youngsan University, Korea
bjkim@ysu.ac.kr

Abstract

In this paper a dynamic variation of neural network system was presented and used to chemical reaction identification. Proposed system is consists of two parts. First one is using first principles rules, which infers a priori knowledge. The other one is a variation of backpropagation neural network algorithm, which estimates parameters of chemical reaction system. This kind of system is useful for chemical modeling processes. Merit of proposed system is once it have learned, it can be used for process control and optimization. Proposed chemical reaction inference system is a powerful and estimates well on process parameters.

Keywords*: Backpropagation algorithm, Parameter estimation, Hybrid system*

1. Introduction

Neural networks have been successfully used as "black-box" models of dynamic systems and, more specifically, as process variable estimators in bioreactor modeling applications[1]. In these efforts the process was operating in a continuous mode; however, identification of batch processes is much more difficult, since a wide range of operating regimes is involved and less data may be available. The ability of neural networks to learn nonparametric approximations to arbitrary functions is their strength, but it is also a weakness. A typical neural network involves hundreds of internal parameters, which can lead to "overfitting" and poor generalization. Furthermore, interpretation of such models are difficult[2]. As a result, there has been an increasing interest in developing modeling methods that address these problems. Since redundancy may result in poor models, one approach has been to decrease the redundancy of the neural network model by developing algorithms that "prune" the weights that have no significant effect on the network's performance[3,4,5]. These methods examine the sensitivity of the prediction error to the network's weights, and eliminate these weights that least affect the fit. However, they do not address the issue of lack of internal model structure and do not use prior knowledge about the process being modeled. One possibility is to create a structured network that combines a known linear model with a nonlinear neural network[6]. The basic idea behind this technique is that these clearly identifiable parts are the process parameter estimator (neural net) and the partial (a prior knowledge) model. The partial model provides a better starting point than "black-box" neural networks and, at the same time, allows for both structural and parametric uncertainty. Since neural networks can approximate arbitrary functions for which no a priori parameterization is known, they can ideally complement the basic model and account for the uncertainty. Our goal is to develop hybrid neural network process models which are more flexible than classical parameter estimation schemes and which generalize and extrapolate better than classical "black-box" neural networks, in addition to being more reliable and easier to interpret. We evaluate this hybrid neural network modeling scheme by comparing its prediction accuracy with standard neural networks.

ISSN: 1738-9968 IJHIT

2. Biological Reactors

Biological reactors exhibit a wide range of dynamic behaviors and offer many challenges to modeling a complex kinetic expressions. A bioreactor typically consists of a large vessel containing an aqueous solution of biomass and one or more substances. Bioreactors operating in a fed-batch manner that is, the vessel is given an initial charge of biomass and substrate, and periodic additions of substrate is made. Bioreactors operated in a fed-batch manner and are quite difficult to model, since their operation involves microbial growth under constantly changing conditions. Nevertheless, knowledge of process parameters(such as growth rate kinetics) under a wide range of operating conditions is very important in efficiently designing optimal reactor operation policies. Please refer fed-batch bioreactor differential equation in[7]. The differential equations can be viewed as a partial model of the system consisting of simple mass balances on the biomass, substrate and volume in the reactor. The dynamics of the process are contained in the kinetics parameter μ(t), known as the specific growth rate, which governs the conversion of substrate to biomass. It is the parameter that complicates the process, as it is a time-varying function of the biochemical variables of the system. In the literature, many models have been developed for this unmeasurable growth rate of which the most widely used are the Monod and Haldane function of the amount of substrate in the system

$$\mu(t) = \frac{\mu^* S(t)}{K_m + S(t) + \frac{S(t)^2}{K_i}} \tag{1}$$

Above expression will only be used to simulate the true process model; for all modeling techniques described in the remainder of the paper, the above expression describing the cell growth rate will be completely unknown. The differential equations (1)-(3) in [7] can be discretized in unit time as follows

$$B_{t+1} = B_t + (\mu_t - \frac{Q_t}{V_t})B_t \tag{2}$$

$$S_{t+1} = -\mu_t B_t + S_t(1 - \frac{Q_t}{V_t}) + \frac{Q_t}{V_t} S_i(t) \tag{3}$$

$$V_{t+1} = V_t + Q_t \tag{4}$$

If error terms ε, υ are added to reflect measurement error and model uncertainty, then the system takes the form of a nonlinear dynamical system in discrete time:

$$x_{t+1} = \int_t (x_t, \Psi_t, \mu_t(x_t)) + \varepsilon_t \tag{5}$$

$$z_{t+1} = H(x_t) + \nu_t \tag{6}$$

Note that the form of μ_t is unknown in equation (5) and it is unmeasurable. Because not all the state variables may be observable, the function H in equation (6) is used to show the transformation from the full state x to the observable state z .

3. Improved MLP Algorithm

3.1. MLP Algorithm

Artificial neural networks commonly referred to as "neural networks" (NN) has been motivated right from its origin by the recognition that the human brain computes in an entirely different way then the conventional computer. The brain is a highly complex, nonlinear and parallel computer (information processing system). It has the capability to organize its structural constituents, known as neurons, so as to perform certain computations (*e.g.* pattern recognition, perception, and motor control) many times faster

than the fastest digital computer in existence today. Consider for example, human vision, which is an information-processing task. It is the function of the visual system to provide a representation of the environment around us and, more important, to supply the information we need to interact with the environment. To be specific, the brain routinely accomplish perceptual recognition task (*e.g.* recognizing a familiar face embedded in an un-familiar scene) in approximately 100-200 ms, where as tasks of much lesser complexity may take days on a conventional computer. How, then, does a human brain do it? At birth, a brain has great structure and the ability to built-up its own rules through what we usually refer to as "experience". Indeed, experience is built up over time, with the most dramatic development (*i.e.* hard wiring) of the human brain taking place during the first two years from birth: but the development continues well beyond that stage. A "developing" neuron is synonymous with a plastic brain: Plasticity permits the developing nervous system to adapt to its surrounding environment. Just as plasticity appears to be essential to the functioning of neurons as information-processing units in the human brain, so it is with neural networks made up of artificial neurons. In its most general form, a neural network is a machine that is designed to model the way in which the brain performs a particular task or function of interest; the network is usually implemented by electronic components or is simulated in software on a digital computer. The interest is confined to an important class of neural networks that perform useful computations through a process of learning. To achieve good performance, neural networks employ a massive interconnection of simple computing definition of a neural network viewed as an adaptive machine. The procedure that is used to perform the learning process is called a learning algorithm, the function of which is to modify the synaptic weights of the network in an orderly fashion to attain a desired design objective. The modification of synaptic weights provides the traditional method for the design of neural networks. Such an approach is the closest to linear adaptive filter theory, which is already well established and successfully applied in many diverse fields. However, it is also possible for a neural network to modify its own topology, which is motivated by the fact that neurons in the human brain can die and then new synaptic connections can grow. The manner in which the neurons of a neural network are structured is intimately linked with the learning algorithm used to train the network. We may therefore speak of algorithms (rules) used in the design of neural networks as being structured. In general we may identify three fundamentally different classes of network architectures. In a layered neural network the neurons are organized in the form, of layers. In the simplest form of layered network, we have an input layer of source nodes that projects onto an output layer of neurons (computation nodes), but not vise versa. In other words, this network is strictly a feed-forward or acyclic type. It is illustrated in the Figure 1 for the case if four nodes in both the input and output layers. Such a network is called a single-layered network, with the name "single-layer" referring to the output layer of computation nodes (neurons). We do not count the input layer of source nodes because no computation is performed there.

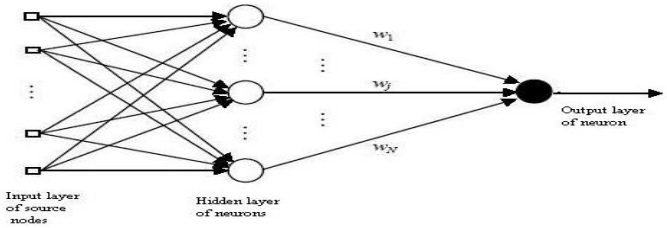

Figure 1. Single Layer Feedforward NN

Multilayer feed forward networks are an important class of neural networks. Typically, the network consists of a set of sensory units (source nodes) that constitute the input layer, one or more hidden layers of computation nodes, and an out-put layer of computation nodes. The input signal propagates through the network in a forward direction, on a layer-by-layer basis. These neural networks are commonly referred to as multilayer perceptrons (MLP's), which represent a generalization of the single-layer perceptron. The source nodes in the input layer of the network supply respective elements of the activation pattern (input vector), which constitute the input signals applied to the neurons (computation nodes) in the second layer (*i.e.*, the first hidden layer). The output signals of the second layer are used as an input to the third layer, and so on for the rest of the network. Typically the neurons at each layer of the network have as there inputs the outputs of the preceding layers only. The set of output signals of the neurons in the output (final layer) constitutes the over all response of the network to the activation pattern supplied by the source nodes in the input (first) layer. Multilayer perceptrons have been applied successfully to solve some difficult and diverse problems by training them in a supervised manner with a highly popular algorithm known as the error back-propagation algorithm. This algorithm is based on the error-correction learning rule. As such, it may be viewed as a generalization of an equally popular adaptive filtering algorithm: the least-mean-square (LMS) algorithm. Basically, error back-propagation learning consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an activity pattern (input vector) is applied to the sensory nodes of the network, and its effect propagates through the network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the networks are all fixed. During the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with an error-correction rule. Specifically, the actual response of the network is subtracted from a desired (target) response to produce an error signal. This error signal is then propagated backward through the network, against the direction of synaptic connections, hence the name "error back-propagation." The synaptic weights are adjusted to make the actual response of the network move closer to the desired response in a statistical sense. The error back-propagation algorithm is also referred to in the literature as the back-propagation algorithm. The standard architecture of backpropagation is with hidden layers and an output layer. Signal flow through the network progresses in a forward direction, from left to right and on a layer-by-layer basis.

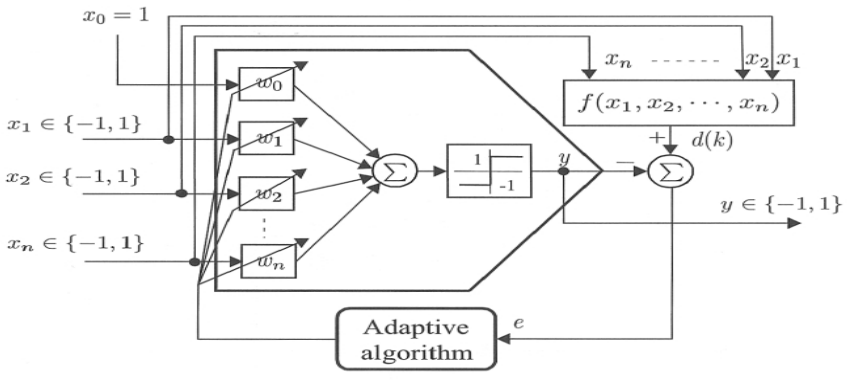

Figure 2. Weight Update Process in Neural Network

The neural network in the Figure 2 is said to be fully connected in the sense that every node in each layer of the network is connected to every other node in the adjacent forward layer. If, however, some of the communication links are missing from the network, we say that the network is partially connected. , the learning process in the neurons involves

updating of certain "weights". A number of adaptation algorithms are showed in Figure 3. The criteria/cost functions used for adaptation and the methods are usually derived from the richly developed field of adaptive filter theory. In the following we present one of the most commonly used adaptation algorithms, the LMS. Let: $w[n]$ be a time varying neuron tap weights, $v[n]$ be a input to neuron, $d[n]$ be a desired response, $\tilde{s}[n]$ be a actual output of neuron and *J* be a cost function (the mean square error) The estimation error is the difference between the desired response and the estimated output:

$$e[n] = d[n] - \tilde{s}[n] \tag{7}$$

using $\tilde{s}[n] = w^H v[n]$ gives,

$$e[n] = d[n] - w^H v[n] \tag{8}$$

The mean square error (cost function) is defined as:

$$J = E\left\{|e[n]|^2\right\} = E\left\{|d[n] - w^H v[n]|^2\right\} \tag{9}$$

Thus the cost function J is a function of vector *w*. Minimizing *J* with respect to the complex tap weights w leads to the set of equations called the Wiener-Hopf equations. If we limit the number of taps to *M* then we obtain the matrix formulation of the Wiener-Hopf equations from which the solution is obtained as

$$w_o = R^{-1} p \tag{10}$$

where

$$R = E\left\{v[n]v^H[n]\right\} \quad \text{and} \quad p = E\left\{v[n]d^*[n]\right\}$$

This method of solution requires inversion of a matrix. An alternative adaptive method of solution is the Steepest Descent algorithm. It can be shown that the cost function *J* has the shape of an M-dimensional bowl whose minimum is at the optimal solution of *w*. The steepest descent algorithm moves the tap weights towards the minimum of the *J* bowl at every iteration by moving them in the direction opposite to the gradient vector:

$$\nabla_w J = 2Rw - 2p \tag{11}$$

Thus we have an iterative definition for the tap weight updates:

$$\begin{aligned} w[n+1] &= w[n] - \frac{\mu}{2}\nabla_w J \\ &= w[n] + \mu[p - Rw[n]] \end{aligned} \tag{12}$$

μ is known as the adaptation step. (μ > 0)

In practice, although we do not know R and p, we can use their instantaneous estimates:

$$\hat{R}[n] = v[n]v^H[n] \tag{13}$$

$$\hat{p} = v[n]d^*[n] \tag{14}$$

This approach is known as the Least Mean Squares (LMS) method which is a member of a particular class of algorithms called the stochastic gradient algorithms. Thus the adaptation equation is given as:

$$\hat{w}[n+1] = \hat{w}[n] + \mu v[n]e^*[n] \tag{15}$$

As we shall see, this equation is used in a variety of adaptation algorithms each having its own definition for the error functions.

3.2. Premature Saturation Phenomenon in MLP Algorithm

The MLP algorithm is a widely used learning algorithm in artificial neural networks. However, it suffers two critical drawbacks. One is the learning process often traps into

local minimum and another is its slowness in learning speed. As a result, there are many researches on them,especially for the learning efficiency improvement by preventing the premature saturation (PS) phenomenon[8],[9] . The PS means that outputs of the artificial neural networks temporally trap into high error level during the early learning stage. In the learning issues of PS phenomenon, researchers have designed many usable modifications to solve this phenomenon. However, the proposed methods are limited to many assumptions and seem to be complex. In this section, we will propose an improved MLP algorithm that varies the number of hidden units and learning rate during learning. Let N be the number of input nodes, H be the number of hidden nodes, M be the number of output nodes, W be the connection weight, X be the value of nodes and premature decision value is 3.64T. The probability of premature at least one node during learning is expressed in equation (16) [9]

$$P = 1 - \left(\frac{1}{2W_{max}} \int_{-W_{max}}^{W_{max}} G\left(\frac{3.64T - v_{k\sigma}}{\sigma_{x_k}}\right) dv_{k\sigma} \right)^M \quad (16)$$

Where $G(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{y^2}{2}} dy$, $\sigma_{x_k}^2 \triangleq \frac{H}{3} W_{max}^2 \int_{-\infty}^{\infty} \left(\frac{2}{1+\exp(-a_j/T)} - 1 \right)^2 \frac{1}{\sigma_{a_j}\sqrt{2\pi}} \exp\left(-\frac{a_j^2}{2\sigma_{a_j}^2}\right) da_j$ and $\sigma_{a_j}^2 \triangleq \frac{N+1}{3} W_{max}^2$

Meanwhile, the premature probability in M output nodes from P training pattern is expressed as follows

$$Q = 1 - \left(\frac{1}{2W_{max}} \int_{-W_{max}}^{W_{max}} G\left(\frac{3.64T - v_{k\sigma}}{\sigma_{x_k}}\right) dv_{k\sigma}\right)^{Mp} \quad (17)$$

Equation (17) means that probability of premature saturation will increase as the value of W_{max} or number of hidden nodes increases.

3.3. Improved MLP Algorithm

To overcome the problem of premature saturation and slowness in learning, we overcome the problems in two ways. First we consider local minima problem. Our approach is as follows. According to equation (17) if we set the W_{max} or number of hidden nodes properly, then the probability of premature saturation will decrease. In our research we consider to set a proper number of hidden nodes. As learning is undergoing we add a new hidden node when MLP is trapping in a local minima. Adding a new node causes an increasing of total sum of squares(TSS). Due to the increasing of TSS. MLP may have a chance to escape from a local minima. In some cases adding a new hidden node to MLP can't guarantee of escaping from a local minima at all times. In that case a new hidden node will be added until it escapes from local minima. To overcome the slowness in learning is as follows. Adding a new hidden node has the merit that it can has a chance to escape from the local minima. But determine the optimal number of hidden nodes requires more learning time. In the standard MLP, the learning rate is held constant throughout the training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. The performance of the MLP algorithm can be improved, if we allow the learning rate to change during the training process. In our proposed system we set the learning rate as high value (0.9) when adding a new node in hidden units. This is because adding a new node makes the variation of TSS become big. On the other hand we set the learning rate as low value (0.2) when neural network is close to minima. This prevents the neural network from oscilating or overshooting phenomenon. Proposed improved MLP algorithm is as follows.


```
/ * Increasing stage    */

Set the initial nodes in hidden unit

weight_correction(){                /*    by standard MLP      */
if ( TSS < Error_criterian )

        reduction_stage();
else {
        if ( nepoch % n == 0) {
          if ( TSS decreases by less than m% ) {
                if (( TSS < Error_Criterian * s ) and ( TSS >
Error_Criterian )){
                     UNDO increasing_stage();
                     weight_correction(); }          /*    by standard MLP
*/
                     Add one hidden node;
                     Increase learning rate; }
                else
                     weight_correction();          /*    by standard MLP      */
                }
        }        /*    End of    increasing stage      */

/* Reduction_stage    */

reduction_stage(){
        Delete one hidden node
        if ( TSS is oscillate or variation is smaller than predetermined value )
                reduce learning rate;
        if ( TSS < Error_criterian ) {
                store_weight();
                remove one hidden unit; }
        else
                weight_correction();    }     /*    by standard MLP      */

where n,m,s are predetermined values
```

4. Hybrid Neural Network Models and Training

The central idea of this article is to integrate the available approximate model with a neural network which approximates the unknown kinetics, in order to form a combined model structure which can be characterized as a hybrid (or structured) neural network process model. The hybrid neural network model has internal structure which clearly determines the interaction among process variables and process parameters, and as a result it is easier to analyze than standard neural networks. A schematic representation of the hybrid neural network model is as follows. The neural network component receives as inputs the process variables and provides an estimate of the current parameter values, in this case the cell growth rate. The network's output serves as an input to the first principles component, which produces as output the values of the process variables at the end of each sampling time. The combination of these two building blocks yields a complete hybrid neural network model of the bioreaction system. For the hybrid neural network model target outputs are not directly available, as the cell growth rate is not measured. In this case, the known partial process model can be used to calculate a suitable error signal that can be used to update the network's weights. The observed error between the structured model's predictions and the actual state variable measurements can be

"back-propagated" through the known set of equations, essentially by using the partial model's Jacobian, and translated into an error signal for the neural network component. The intuition behind this is that the process parameters should be changed proportionally to their effect on the state variable predictions, multiplied by the observed error in the state predictions.

5. Experiments

We evaluate proposed system by comparing its prediction accuracy with standard MLP and support vector machine(SVM). In order to test the behavior of proposed system, we simulated data via the known first principles differential equations (1)-(3) under a realistic condition. The specific growth rate was generated using the Haldane model (equation (4)) with the parameter values listed in[10]. Future state variables were then generated using the differential equations with random noise. We take the same initial values B_0, S_0, V_0 to generate the training, validation, test data set used in [10]. Table 1 shows initial parameter value and number of data in train, validation and test data respectively.

Table 1. Initial Parameter Value of Train, Validation and Test Data Set

	B_0	S_0	V_0	Noise	Number of data
Train	0.1	0.5	10	$N(0,00.01^2)$	20
Validation	0.3	0.7	4	$N(0,00.01^2)$	80
Test	0.001	1.5	4	$N(0,00.01^2)$	80

Because learning and generalization ability of standard MLP is depends on the number of hidden nodes, we try experiment varying the number of hidden nodes for selecting the optimal number of hidden nodes in standard MLP. In this experiment optimal number of hidden nodes is 4. In proposed method the number of input, initial hidden and output node is 2,1 and 1 and weight update is done by MLP method to estimate the growth rate. The weights that produced the smallest SSE(Sum of Square Error) for the validation data were selected for making the predictions for the test set. In SVM, we take the polynomial kernel and set the degree to 2 for fast training.

Table 2. Comparison of Sum of Square Error (SSE) on B and S By Standard MLP, SVM And Proposed Method

Method	Train		Validation		Test	
	B	**S**	**B**	**S**	**B**	**S**
SVM	0.00287	0.02132	0.02101	0.01573	0.24473	0.27274
Standard MLP	0.18	0.18	17.45	17.38	6.534	8.736
Proposed Method	0.00360	0.02462	0.02462	0.01963	0.13532	0.14374

After learning is finished number of hidden units in proposed model is 3. From Table 2 experimental results shows that in estimating B and S, proposed method is about 48 and 60 times outperform than standard MLP because proposed method overcomes the overshooting problem and guarantees the global minima. Comparing proposed method to

standard SVM the former is about 2 times outperform than the later.

6. Summary and Conclusion

In this paper a hybrid neural network modeling approach was presented and used to model a fedbatch bioreactor. This hybrid model is comprised of two parts including a partial first principles model, which reflects the a priori knowledge, and a neural network component, which serves as a nonparametric approximator of difficult-to-model process parameters. This form of hybrid neural network is useful for modeling processes where a partial model can be derived from simple physical considerations (for example, mass and energy balances), but which also includes terms that are difficult (or even infeasible) to model from first principles. The hybrid model, once learned, can be used for process control and optimization. The concept of combining proposed MLP with first principle knowledge is a powerful one, and goes well beyond the example presented in this paper.

Acknowledgment

This study was supported by a grant of Youngsan University(in 2016).

References

[1] P. A Lant, M. J. Willis, Ci. A. Montague, M. T. Tham and A. J. Morris, "A Comparison of Adaptive Estimation With Neural Based Techniques for Bioprocess Application", Proc. of the AmericanControl Conf., vol. 2173, **(1990)**.
[2] , M. L. Mavrovouniotis and S. Chang, "Hierarchical Neural Networks for Process Monitoring", Comp. Chem. Eng., vol. 16, no. 4, **(1992)**, pp. 347.
[3] N. Bhat and T. McAvoy, "Use of Neural Nets for Dynamic Modeling and Control of Chemical Processes", Comp. Chem. Eng., vol. 14, no. 573, **(1990)**.
[4] E. D. Karnin, "A Simple Procedure for Pruning Back-Propagation Trained Neural Networks", IEEE Trans. on Neural Networks, vol. 1, no. 239, **(1990)**.
[5] M. C. Mozer and P. Smolensky, "Skeletonization: A Technique for Trimming The Fat From A Network via Relevance Assessment", in Advances in Neural Information Processing, D. S. Tourestzky, ed., Morgan Kaufmann, San Mateo, CA, **(1989)**.
[6] D. Haesloop and B. R. Holt, "A Neural Network Structure for System Identification", Proc. of the American Control Conf., no. 2460, **(1990)**.
[7] B.J. Kim, I.K. Kim, J.Y. Shim and C.H. Hwang, "A Study On Prediction Performance Of Hybrid Least Squares Support Vector Machine With First Principle Knowledge", Korea Information Science Society, J., vol. 30, no.7, **(2003)**.
[8] S.I. N. Baba, "A New Approach for Finding the Global Minimum of Error Function of Neural Networks", Neural Networks, vol. 2, **(1989)**, pp367-373.
[9] S. H. Oh, "Improving the Error Back-Propagation Algorithm with a Modified Error Function", IEEE Transactions on Neural Networks, vol. 8, no. 3, **(1997)**, pp799-803.
[10] P.A. Lant, M.J. Williams, G.A. Montague, M.T. Tham and A.J. Morris, "A Com-parision of Adaptive Estimation With Neural Based Techniques for Bioprocess Application", Proc. of the American Control Conf., vol. 2713, **(1990)**.

