Computational Comparisons of GPC and NGPC Schemes

D.N.RAO, Dr. M.R.K. MURTHY, D.N.HARSHAL and Dr. S.R.M.RAO

***Abstract*—In this paper the GPC and NGPC model predictive control techniques are applied to a typical plant and compared in terms of computational efficiency. The system identification and predictive control computational performance are examined. The future control inputs using GPC can be found by the direct analysis and thus reducing the computational time when compared with NGPC scheme with variable weight factor. The NGPC shows the true dynamics of the plant. The weight factor of the GPC scheme is explored in the implementation of the scheme. The results show the GPC with variable weight factor is on par with NGPC.**

Index Terms—**GPC, Computational time, NGPC, Typical plant**.

I. INTRODUCTION

Generalized Predictive Control (GPC) belongs to the class of Model-based predictive control (MBPC) techniques and was first introduced by Clarke and his Co-workers in 1987[1, 2]. Linear model predictive control has a long reputation as a powerful control tool in industrial control processes. The applications of the neural network to nonlinear control system can basically be divided into two classes [3]: control systems where the controller in itself is a neural network and model based control systems where the model of the nonlinear dynamic system is a neural network. The Predictive control is characterized by accomplishing the prediction of future values through a model [7]. Then, based on the predicted values, on an optimization function and on a control law, the controller generates a future control action. The objective of this work is to present in a brief way the implementation of predictive controllers based on the ANN [6]. It uses some optimization indexes and control laws usually applied to the conventional predictive control. In Figure.1 a general structure is presented to implement **Neural Generalized Predictive Controllers (NGPC)** cases, the ANN are properly trained and represent the dynamics of the plant in a satisfactory way.

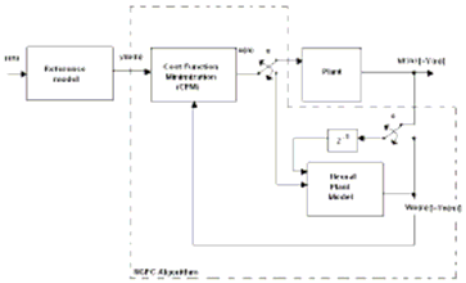

Figure 1: Blocks Diagram of Neural Predictive Controllers

The predictive controller, in summary, is characterized by computing future control actions based on output values predicted by a model, with vast literature and academic and industrial interest (Clarke, 1987; Astrom and Wittenmark 1995; Garcia et all, 1989; Menezes, 1993; Arnaldo, 1998). Currently there are two techniques used to model nonlinear plants. One is to linearize the plant about a set of operating points. If the plant is highly nonlinear the set of operating points can be very large. The second technique involves developing a nonlinear model which depends on making assumptions about the dynamics of the nonlinear plant. If these assumptions are incorrect the accuracy of the model will be reduced. Models using neural networks have been shown to have the capability to capture nonlinear dynamics [3].For nonlinear plants, the ability of the GPC to make accurate predictions can be enhanced if a neural network is used to learn the dynamics of the plant instead of standard modeling techniques. Improved predictions affect rise time, over-shoot, and the energy content of the control signal. This next section presents the concepts of Generalized Predictive Control predictive techniques, using the usual optimization functions and control laws. The computational performance of a GPC implementation is largely based on the minimization algorithm chosen for the CFM block. The selection of a minimization method can be based on several criteria such as; number of iterations to a solution,

Manuscript received July 27th , 2006.

First Author: D.N.Rao ,graduate with a specialization in Electronics and Communication Engineering He is working as faulty in T.R.R.College of Engg.J.N.T.U.email:dhyapulai@yahoo.com,ph No:+91-040-27552909IAENG member

Second Author: Dr.M.R.K.Murthy, obtained his degree in (1977), Osmania University (O.U) and P.G. in 1980, from O.U. He received his PhD degree from IIT, Chennai in 1993.e.mail:dhyapulai@gmail.com, ph No:+91-040-27552909

S Third Author: Dr.S.R.M.Rao obtained B.E. degree from IISC, Bangalore, and M.E. from NIT, Warangal. He obtained his PhD from IISC, Bangalore. Email.com;nag_rai8761@yahoo.com

Fourth Author: D.N.Harshal. He is research Associate at Pentagram Ltd, Hyderabad. Graduation in specialization 'Information and technology'. e..mail:harshal_nag20@yahoo.com,IAENG member.

computational costs and accuracy of the solution. The quality of the plant's model affects the accuracy of a prediction.

II.GENERALIZED PREDICTIVE CONTROL SCHEME

The generalized predictive control (GPC) is discussed in [1] [2]. It is a receding horizon method. For a series of projected controls the plant output is predicted over a given number of samples, and the control strategy is based upon the projected control series and the predicted plant output. We will summaries some of the features of the GPC but refer to [1] [2] for a more detailed discussion.

The GPC algorithm is derived according to the following. We start by calculating the predictor. The optimal predictor can be derived according to (starting from the assumed CARIMA model structure)

$$A(q^{-1})y(k) = B(k^{-1})q^{-1}u(k-1) + C(q^{-1})e(k)/\Delta \mid \Delta E_j(q^{-1})q^j \quad (1)$$
$$\tilde{A}E_j y(k+j) = E_j B\Delta u(k+j-d-1) + E_j e(k+1)$$

In the second equation, we have omitted the index (q^{-1}) for the sake of brevity and denoted by $\tilde{A} = \Delta A$. (2)

Defining $1 = E_j\tilde{A} + q^{-1}F_j$

Gives

$$(1-q^{-1}F_j)y(k+1) = E_j B\Delta u(k+j-d-1) + E_j e(k+1) \quad (3)$$

$$y(k+1) = F_j y(k) + E_j B\Delta u(k+j-d-1) + E_j e(k+j) \quad (4)$$

Since $\deg(E_j) = j-1$ (thus the notice term completely in the future) the best prediction becomes

$$y(k+j \mid k) = E_j B\Delta u(k+j-d-1) + F_j y(k) \quad (5)$$

And where $E_j B = G_j$

The polynomials *j and j can* easily be calculated recursively using equation (4). The Polynomials are given by

$$E_j(q^{-1}) = e_{j,0} + e_{j,1}q^{-1} + \ldots e_{j,j-1}q^{-(j-1)}$$
$$F_j(q^{-1}) = f_{j,0} + f_{j,1} + \ldots + f_{j,n_0}q^{-n_0} \quad (6)$$
$$E_{J+1}(q^{-1}) = E_j(q^{-1}) + e_{j+1}q^{-j}$$

with $e_{j+1,j} = f_{j,0}$, the coefficients of the polynomial F_{j+1} can then be expressed as

$$f_{j+1,j} = f_{j\mid i+1} - f_{j,0}\tilde{a}_{i+1} \quad (7)$$

and $i = 0\ldots(n_a-1)$ and the polynomial G_{j+1} can be obtained recursively as follows;

$$G_{j+1} = E_{j+1}B = (E_j + f_{j,0}q^{-1})B = G_j + f_{j,0}q^{-1}B \quad (8)$$

that is, the first *j* coefficients of G_{j+1} will be identical to those of G_j and the remaining coefficients are will be given by

$$g_{j+1,j+1} = g_{j,j+1} + f_{j,0}b_i$$

A set of optimal predictors as given by equation (3.6) can be rewritten in the form

$$y = Gu + F(q^{-1})y(k) + G'(q^{-1})\Delta u(k-1) \quad (9)$$

where

$$\begin{bmatrix} \hat{y}(k+d+1 \mid k) \\ \hat{y}(k+d+2 \mid k) \\ \hat{y}(k+d+N \mid k) \end{bmatrix} = y\,', \quad \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \Delta u(k+N-1) \end{bmatrix} = u$$

$$\begin{bmatrix} g_0 & 0 & \ldots & 0 \\ g_1 & g_0 & \ldots & 0 \\ . & . & . & . \\ g_{N-1} & g_{N-2} & \ldots & g_0 \end{bmatrix} = G\,', \quad \begin{bmatrix} F_{d+1}(q^{-1}) \\ F_{d+2}(q^{-1}) \\ . \\ . \\ F_{d+N}(q^{-1}) \end{bmatrix} = F(q^{-1}) \quad \&$$

$$\begin{bmatrix} (G_{d+1}(q^{-1}) - g_0)q \\ (G_{d+1}(q^{-1}) - g_0 - g_1q^{-1})q^{-2} \\ . \\ . \\ (G_{d+1}(q^{-1}) - g_0 - g_1q^{-(N-1)})q^N \end{bmatrix} = G'(q^{-1})$$

the predictor becomes $y = F(q^{-1})y(k) + G(q^{-1})\Delta u(k-1)$ (10)

Since the last two parts of equation (13) only depend on the past, these can be grouped as the free response and denoted *f*.

$$y = Gu + f \quad (11)$$

The free response *f* can be calculated recursively as

$$f_{i+1} = q(1-\tilde{A}(q))f_j + B(q^{-1})\Delta u(k-d+j) \quad (12)$$

The calculation of the controller in turn is performed as to minimize with respect to the control moves the objective function. The objective is to find a control time series that minimizes the cost Function

$$J_{NGPC} = E\left\{\sum_{j=1}^{N_2}(y(k+j) - r(k+j))^2 + \lambda\sum_{j=1}^{N_2}\Delta u(k+j-1))^2\right\} \quad (13)$$

Subject to the constraint that the projected controls are a function of available data.N2 is the costing horizon. λ is the control weight. The expectation of (1) is conditioned on data up to time kith plant output is denoted y, the reference r, and the control signal u. ⊗is the shifting operator (1 − q−1), such that

$$\Delta u(k+j-1) = 0, j > N_u \quad (14)$$

The first summation in the cost function penalizes deviation of the plant output y (k) from the reference r (t) in the time interval $k+1 \le k \le k+N_2$. The second summation penalizes control signal change Δu (t) of the projected control series. We will furthermore introduce the so called control horizon Nu ≤ N2.Beyond this horizon the projected control increments Δu is fixed at zero: $\Delta u(k+j-1) = 0, j > N_u$

The projected control signal will remain constant after this time. This is equivalent to placing an effectively infinite weight on control changes for k > Nu. Small values of Nu generally result in smooth and sluggish actuations, while larger values provide more active controls. The use of a control horizon Nu < N2 reduces the computation burden. In the nonlinear cast the reduction is dramatic [1].
The quadratic loss function, can be written as

$$J = (Gu + f - w)^T (Gu + f - w) + \lambda u^T u \qquad (15)$$

where w is the future set-point trajectory.

$$w = \left[w(k+d+1) w(k+d+2) ... w(k+d+N) \right]^T \qquad (16)$$

The loss function can be rewritten as

$$J = \frac{1}{2} u^T H u + b^T u + f_0 \qquad (17)$$

$$H = 2(G^T G + \lambda I), \qquad b^T = 2(f - w)^T G$$

$$f_0 = (f - w)^T (f - w) \qquad (18)$$

The minimum of the loss function can be calculated analytically (in this linear example and assuming no constraints) by making the gradient of the loss-function equal to zero. This gives the controller as: $u = -H^{-1} b = (G^T G + \lambda I)^{-1} G^T (w - f)$ (19)

A: Simple GPC design example:

An example that designs a GPC controller for a linear, first order system without time delay is chosen. In discrete-time representation the process can, for example, be expressed by the use of the backward shift operator as a CARIMA-model

$$(1 + aq^{-1}) y(k) = (b_0 + b_1 q^{-1}) u(k-1) + e(k)/\Delta$$

where $\Delta = 1 - q^{-1}$, we choose the numerical values of the system parameters as

$$(1 - 0.8) y(k) = (0.4 + 0.6 q^{-1}) u(k-1) + \frac{1}{(1 - q^{-1})} e(k)$$

This gives a system-behavior similar to the one illustrated in figure 2

B: The control law for GPC systems (summary):

Let us summarize the control law for a GPC system that can be modeled by the CARIMA model for completeness here:

$$A(q^{-1}) y(k) = q^{-1} B(q^{-1}) u(k) + C(k)/\Delta \qquad (1)$$

where e (k) is an uncorrelated random sequence.
We introduce

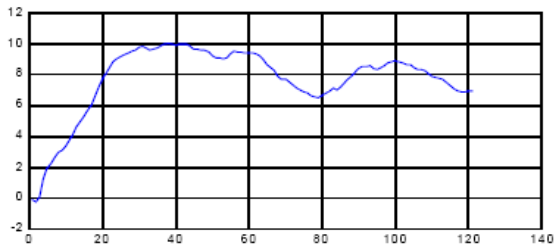

Figure 2. A step-response when the input changes from zero to one at time k = 0. The effect of the noise is significant, note that the system in the absence of noise has a low frequency gain of five (and thus the output would move from zero to five in response to the input change).

In our example we get the controller algorithm, where only the first row is used, since only the first control move is implemented:

$$u(k) = 0.396u(k-1) + 0.604u(k-2) - 0.133w(k+1) + 0.147w(k+3)$$

This controller is implemented in figure 2 and some simulation results are as seen in figure 3.

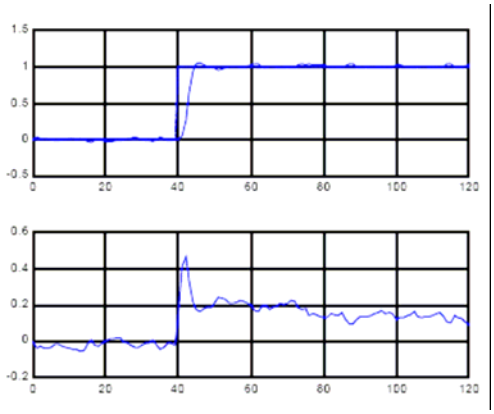

Figure 3. The control response to a set point change with small amplitude of white noise present. Upper graph shows output and set point, the lower graph shows the manipulated variable. We assume no model/plant mismatch. This example did not include any constraints, and therefore the optimization could be solved analytically.

$$YN = [y(k+1), y(k+2), ... y(k+N_2)]^T \qquad (19)$$

$$\tilde{U} = [\Delta u(k), \Delta u(k+1), ... \Delta(k+N_2-1) \qquad (20)$$

$$W = [r(k+1), r(k+2), ... r(k+N_2)]^T \qquad (21)$$

and let Y_k represent data up to time k. We now note, that

$$J = E\left\{ (Y_N - W)^2 + \lambda (\Delta u)^2 \mid Y_k \right\}$$
$$= (\tilde{Y}_N - W)^2 + \lambda (\tilde{u})^2 + \sigma \qquad (22)$$

where in turn

$$\hat{Y} = E\{Y_N \mid \tilde{u}_{,k}\, Y\} \quad (23)$$

In GPC case σ is independent of $\tilde{u}$ and yN, and when minimizing the cost function, σ constitutes a constant value that can be ignored. We can separate $\tilde{y}_N$ into two terms: [1].

$$\tilde{Y}N = G\tilde{u} + f \quad (11)$$

where *f* is the free response, and $G\tilde{u}$ is the forced response. The free response is the response of the plant output as a result of the current state of the plant with no change in input. The forced response is the plant output due to the control sequence $\tilde{u}$. The linearity allows for this separation of the two responses. The important fact is that *f* depends on the plant parameters and Y_N (i.e. past values of u and y) while the matrix G only depends on the plant parameters. G does not change over time, and *f* can be computed very efficiently. Using (22) and (23)

$$J = (YN - W)^2 + \lambda(\tilde{u})^2$$

$$= (G\tilde{u} + f - w)^2 + \lambda(\tilde{u})^2$$

$$= (G\tilde{u} + f - w)^T (G\tilde{u} + f - w) + \lambda\tilde{u}^T\tilde{u}$$

The minimization of the cost function on future controls results in the following control Increment vector: [1]

$$\min J = \tilde{u}^* = (G^T G + \lambda I)^{-1} G^T (w - f)$$

and in the following current control signal

$u(k) = u(k-1) + g^{-T}(w - f)$ .Where g^{-T} is the first row. of $(G^T G + \lambda I)^{-1} G^T$

III. *The Neural generalized predictive control*

A: Nonlinear Case:

If the plant is nonlinear, the cost function (1) & (19) to (23) remains the same, but implementation of the algorithm is done in a different way. Especially the prediction of the plant outputs $\tilde{Y}N$ based on the projected control series $\tilde{u}$ can not be done in the same way. For a nonlinear plant the prediction of future output signals can not be found in a way similar to (25): The relation of the control signal series to the plant output series is nonlinear, i.e. the multivariable function *f,* that fulfils $YN = f(y_k, \tilde{u})$ is nonlinear. Therefore we can not separate the free response from the forced response. To solve this problem we need to implement the prediction of yN, i.e. *f* in (12) in a different way. We need nonlinear predictor $\tilde{y}N$ for future plant outputs.

$$\tilde{Y}N = f(y_k, \tilde{u}) \quad (11)$$

This is where the neural network comes into play [4] [5] [6]: We will facilitate the neural network to the workings of the function *f* in (11) to obtain the predictions $\tilde{Y}N$ .We will train a neural network to do a time series prediction of plant outputs $\tilde{Y}N$ for a given control signal time series $\tilde{u}$. This will let us evaluate the GPC cost function (1) for a nonlinear plant. Using a suitable optimization algorithm on the projected control signal series with respect to the cost function, we find a control series that minimizes the cost function**:** min $J=\tilde{u}^*$**.** *(Note that σ in equation (22) is not independent of $\tilde{u}$ nor yN for a nonlinear plant. We will however assume that the effects of σ are small and can be ignored.)*

B: The complete control algorithm iterates through these three steps:

1. Choose (a new) control change time series $\tilde{u}$.
2. Predict plant output time's series $\hat{Y}$.
3. Calculate cost function *J*

That is, first we choose a control time series $\tilde{u}$. Using a model we look at what would happen if we choose this series of control signals: we predict the plant output $\hat{Y}$. Then we use a cost function (1) to tell us how good or bad the outcome is. Then we let the optimization algorithm choose a new control time series $\tilde{u}$ that is better by means of the cost function (1). We iterate these steps in order to find a series of control signals that is
optimal with respect to the cost function (1).

There are 3 distinctive components in this controller:
- Time series predictor of plant outputs
- cost function
- Optimization algorithm.

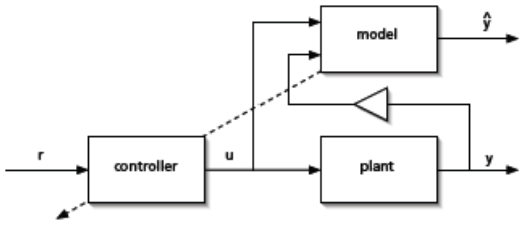

Figure 4 The Neural Networks based Predictive Control

The cost function is given by (1). For the optimization algorithm we can use a simplex method which is implemented as a Matlab built-in function, such as fminsearch. The time series predictor will use a neural network model of the plant. The complete system diagram is sketched in figure 4. The neural network model runs alongside the plant and is used by the controller as discussed above to predict plant outputs to be used by the controller's optimization algorithm There are four tuning parameters in the cost function, N1, N2, Nu, and λ. The predictions of the plant will run from N1 to N2 future time steps. The bound on the control horizon is Nu. The only constraint on the values of Nu and N1 is that these bounds must be less than or equal to N2. The second summation contains a weighting factor, λ that is introduced to control the balance

between the first two summations. The weighting factor acts as a damper on the predicted *u* (k+1). As noted earlier in section 2.1, the control horizon Nu≤ N2 limits the number of projected control signal changes such that Δu (k + j − 1) = 0, j > Nu.

If we set the control horizon to some value Nu < N2, the optimizer will only work on a series of Nu projected control signal changes, while evaluating a predicted plant output series that has N2 samples. Reducing Nu will dramatically reduce the computational burden, as it effectively reduces the dimension of the multidimensional variable that the optimizer works on.

C: Neural model

Feed forward networks with a single hidden layer using threshold or sigmoid activation functions are universally consistent estimators of binary classifications (Farago and Lugosi, 1993). Also cited in Tan and Cauwenberghe (1996) and Hecht-Nielsen (1990), it is proven that a neural net, with only one hidden layer, **is** able to represent any function of $R^n \rightarrow R^m$ Rm, just limited by the number of neurons in the hidden layer.

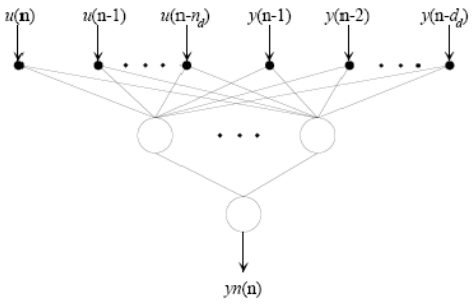

Figure 5. Multi-layer feed forward neural network with a time delayed structure

The purpose of our neural network model is to do time series prediction of the plant output. Given a series of control signals $\tilde{u}$ and past data *Yk* we want to predict the plant output series *YN* . We will train the network to do multi-step ahead prediction, i.e. to predict the plant output Y_{K+1} given the current control signal u_k and plant output Y_K (see figure3). The neural network will implement the function

$$\tilde{Y}_{K+1} = f(u_k, y_k) \qquad (28)$$

As will be discussed below, yk has to contain sufficient information for this prediction to be possible. To achieve multi step ahead prediction of all the plant outputs in (28) we will cycle the one step ahead prediction back to the model input. In this manner, we get predicted signals step by step for time k+1,k+2, . . .k+n.

D: NGPC– T Steps ahead:

The preceding algorithm can be improved using predictive control techniques. Thus, scalar values can now be expressed as predicted vectors consider a real nonlinear model expressed by:

$$y(k+1) = f[y(k), y(k-1), .., y(k-n), u(k), u(k-1), ., u(k-m)] \qquad (29)$$

where f is a nonlinear function of the system output [y (k), y (k-1), y (k-n)] and input values [u (k), u (k-l), u (k-m)]. m is the number of exogenous regressors

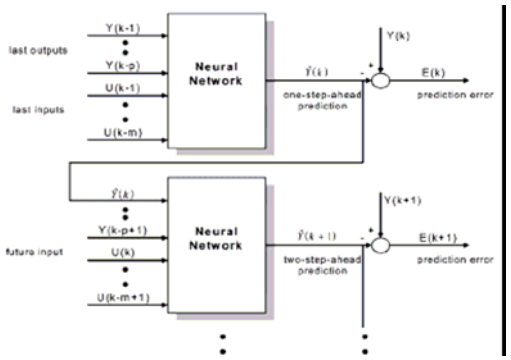

Figure 6. Multi-step predictor

The variable n is the number of auto regressors and But, to represent dynamic systems, it is necessary to introduce the feedback effect in the ANN, characterizing the application of the Recursive ANN (RANN). As in the ARX models (Autoregressive with exogenous inputs), the input signals of the net are associated to its own input and output
past values. Then, let use a net with three layers, and base the study on the neural structure illustrated in the Figure 5. where N represents the number of the hidden layer neurons, and the blocks represents a single-valued monotonic nonlinear sigmoid function to be applied at the output of each hidden layer neuron. bs represents the bias of the output linear neuron. The matrixes of weights W1 and W2 have dimensions N x (n+m) and 1 x N respectively and represent the weights of the connections among layers 1-2 and 2-3.

The NGPC [13] algorithm has the following important steps. (see figure. 1**):**

1) Generate a reference trajectory. If the future trajectory of *ym* (*n*) is unknown, keep *ym* (*n*) constant for the future trajectory.
2) Start with the previous calculated control input vector, and predict the performance of the plant using the model.
3) Calculate a new control input that minimizes the cost function,
4) Repeat steps 2 and 3 until desired minimization is achieved,
5) Send the first control input, to the plant,
6) Repeat entire process for each time step.

E: Topology of a multi-layer perception:

The neural network was trained to model the plant using the same pulse train. The network architecture contains a single hidden layer with four hidden nodes and a single output node. The input layer is composed of two inputs, one that is externally fed with four time delayed nodes and the other is fed back from the output of the plant with four time delayed nodes. The hidden layer's nodes use the hyperbolic tangent as an activation function and the output is scaled linearly. Normalized Root Mean Squared error (NRMS) and Max error were used to measure training of the network. The NRMS error measure, where T is the desired target output, O is the output of the network, p is the training pattern, and n is the output node. In this example, n is 1 and p is 1000 (100 time steps sampled at 0.2 seconds).A cycle is defined as all of the input/output relations that form the entire pulse train. The NRMS error and the Max error for 10,000 cycles of network training are shown in Figure 7 and Figure 8.

$$NRMS = \sqrt{\frac{\sum_n \sum_p (T_{np} - O_{np})^2}{\sum_n \sum_p (T_{np})^2}}$$

The principal importance of a neural network is not only the way a neuron is implemented but also how their interconnections (more commonly called topology) are made. The topology of a human brain is too complicated to be used as a model because a brain is made of hundreds of billions of connections which can't be effectively described using such a low-level (and highly simplified) model.

The topology we will study is therefore not the topology of a human brain but actually a simple topology designed for easy implementation on a digital computer. One of the easiest forms of this topology at the moment is made of three layers:

- one input layer (the inputs of our network)
- one hidden layer
- one output layer (the outputs of our network)

All neurons from one layer are connected to all neurons in the next layer This forms a whole network with full interconnection, please note also that the weight (and therefore the importance) of each connection is not represented (for practical reason) here but must exist in the reality.

F: Training and memory of a neural network

The neural network presented above is not of any interest because it has not been trained (so it is not able to solve any particular problem). This situation can be compared to a little baby, whose brain is fully developed and ready for work but who is not able to do anything because it has not experienced any stimulus. So a neural network without learning is analogous to a human without education. Therefore, a neural network must be trained to solve some particular problem. The methodology of the training is analogous to the way you would teach a child to read or to count, that is by presenting some number or letter and by assigning the letters and numbers some values. For example you could show a child of 5 years an image of an A and you could then tell him that it's an "A'ye" with the best pronunciation you could manage.

You will teach a neural network in exactly the same way, namely you will feed our network with a set of numbers and the network will give you a result in its output layer. Since the weights of the connections in the network are initially in a random state, this result will surely at the beginning not satisfy you, so you will change the weight of some connections in order to obtain a better result. You will change the weight of the connections in fact until you get the desired result (this is the training stage). Next you will feed the input layer of the network with other examples and continue adjusting weights, until eventually you obtain the desired output for each example.

The first step in the control design process is the development of the plant model. The performances of neural network controllers are highly dependent on the accuracy of the plant identification. In this section we will begin with a discussion of
some of the procedures that can use to facilitate accurate plant identification. As with linear system identification, we need insure that the plant input is sufficiently exciting. For nonlinear black box identification, we also need to be sure that the system inputs and outputs cover the operating range for which the controller will be applied. For our applications, we typically collect training data while applying random inputs which consists of a series of pulses of random amplitude and duration.

The duration and amplitude of the pulses must be chosen carefully to produce accurate identification. If the identification is poor, then the resulting control system may fail. Controller performance tends to fail in either steady state operation, or transient operation, or both. When steady state performance is poor, it is useful to increase the duration of the input pulses. Unfortunately, within a training data set, if we have too much data in steady state conditions, the training data may not representative of typical plant behavior. This is due to the fact that the input and output signals do not adequately cover the region that is going to be controlled. This will result in poor
transient performance. The following example will illustrate the performance of the predictive controller when we use different ranges for the pulse widths of the input signal to generate the training data.

After training the network with the data set as shown in figure 7a the resulting neural network predictive control system if unstable and based on the poor response of the initial controller,

we determined data did not provide significant coverage. Therefore, we changed the range of the input pulse widths as

shown in figure7b one can see that the training data set, the resulting predictive system was stable, although it resulted in

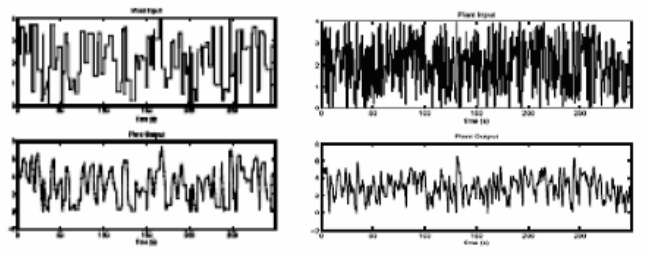

Figure 7. (a&b) Training data with long and short pulse width.

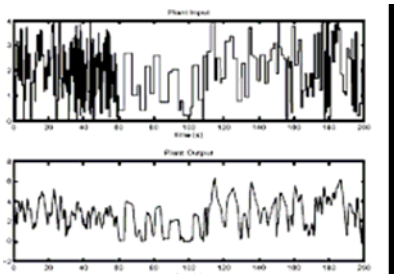

Figure 8Training data with mixed pulse width**.**

large steady-state errors. In the third test, we combined short pulse width and long pulse width (steady state) data. The long pulses were concentrated only on some ranges of plantoutputs. We chose to have steady-state data over the ranges where the tracking errors from previous case were large, shown in the figure8.The resulting controller performed well in both transient and study state conditions.

The entire set of training examples must be shown to the network many times in order to get a satisfactory result. You would not expect a child to learn to read having
seen each letter or word only once, similarly the network requires many examples. After all of this training, the network is hopefully able to solve the problem - we say that it has learned, and its `knowledge' is stored by all the different connection weights. Care must be taken, when training perception networks, to ensure that they do not over fit the training data and then fail to generalize well in new situations. Several techniques for improving generalization are discussed. Consider a network with two hidden nodes (had=2), one output node, and input consisting of u (n) and two previous inputs (nod=2), and of three previous outputs (did=3). Suppose that a 2-step prediction needs to be found, that is, the network needs to predict the output at times n+1 and n+2. The information required by the neural network at each time instant is conveyed in Figure 4.
The example plant below depicts a prediction of the plant for k=2. To produce the output y (n+2), inputs u (n+1) and u (n+2) are needed. The prediction process is started at time n, with the initial conditions of [u (n) u (n-1)] and [y (n) y (n-1) y (n-2)] and the estimated input u (n+1). The output of this process is yn (n+1), which is fed back to the network and the process is repeated to produce the predicted plant's output yn (n+2).

IV: SIMULATION RESULTS

For a large number of systems a linearising approach is acceptable and any relatively small non-linearities, being effectively linearised by the GPC. In order to obtain improved control over systems with small non-linearities or to deal with systems with stronger nonlinearities, it is necessary to take into account of the nonlinearities in an appropriate way. By using instantaneous linearization of a nonlinear model incorporating the GPC and the NGPC techniques can be applied to NARMAX and neural networks model. The control performance using both methods is examined and compared.
Consider the typical plant;
Y (k) = 0706 y (k-1) + 0.2885 y (k-2) + 0.066 u (k-1) - 0.03 u (k-2) + 0.7202 y (k-1) y (k-2) (30)

Proposed constant and variable λ :

It is observed that the plant nonlinearity and no symmetry make the plant output behaves differently for rising edge and falling edge of the reference signal. The idea is try to found a perfect adaptability to , to compensate the plant characteristics. Some empiric rules were used with satisfactory practical results. Considering the plant linearization around the operating point it is observed that it is required to adopt the variable λ , for each step, and the practical results become excellent as showed below.

According to Section 3, the neural net will be characterized by the number of auto regressors, the number of exogenous regressors and the number of hidden layer neurons. In the proposed application, we select the neural network model with only 6 hidden layer neurons (N=6), 4 auto regressors (n=4), 4 exogenous regressors (m=4) and use the hyperbolic tangent as sigmoid function. After appropriate training, the validity of the model is verified in a large operation strip. The obtained results are presented in the Figure 4; it shows the level changing from 2 to 9 units that practically represents the whole plant. So, the proposed neural network model properly trained can satisfactorily represent the model in a quite satisfactory way. Hence the net is ready to NGPC implementation [12, 13]

B: With constant λ :

Figure 9. shows the plant output behavior with respect to reference signal and Figure10. clearly shows the control action stability. It also illustrates de satisfactory performance of the NGPC. Note in Figure 9 the value of λ is large enough to generate an output overshoot when the reference signal raises. However, it is too small to guarantee an effective time to stabilize the output plant when the reference signal falls.

C: With variable λ:

To enhance the system control action, with variable λ . It compensates the overshoot at the reference signal rises and the

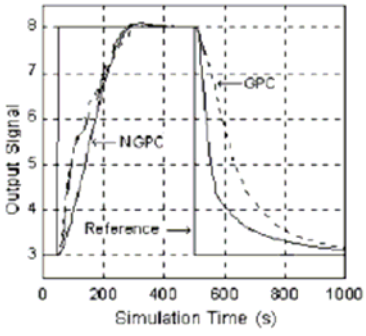

Figure9: Reference signal and plant output with constant

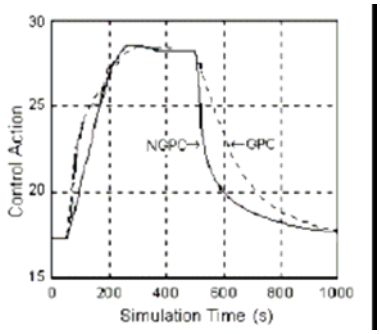

Figure 10: Control action with constant λ

stabilization time at the falls.Then λ , is maximum at the low limit and decreases when the output level rises. As in to the proposed system with variable λ and the obtained results is showed in Figure 11&12. Figure.13 illustrates the respective control action behavior.

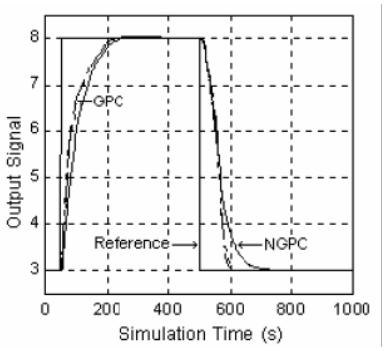

Figure 11: Reference signal and plant output with adaptable λ

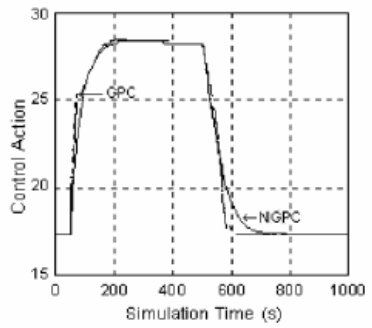

Figure 12: Control action with adaptable λ

CONSTANT WEIGHT FACTOR [λ]: In the figure when constant weight factor it performs very poor as shown in the figures9&10 and, figure13&14. The table 1 shows the poor performance of the GPC when the λ is constant, and the NGPC shows good performance .The poor performance could be corrected by adaptable weight factor λ. which is very close to NGPC scheme. This is applicable when the order of the plant is lower

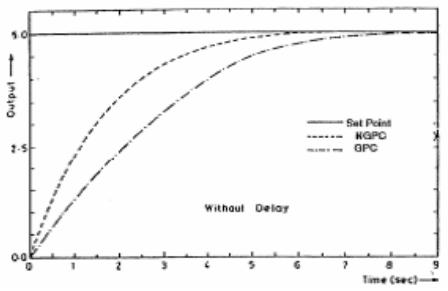

Figure 13 GPC and NGPC with constant set value.

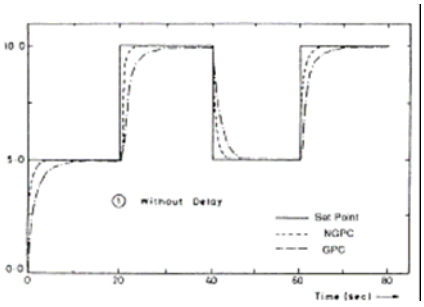

Figure 14 GPC and NGPC with variable set value.

TABLE 1 SUMMARY COM. OF GPC AND NGPC

		GPC	NGPC
1.CPU time used in m sec.	II order without delay	179	114
	II order with delay	179	115
2. No. of predictions	III order without delay	180	119
	III order with delay	182	120
3. Time taken to reach set value	II order without delay	419	191
	II order with delay	428	197
	II order without delay	400	80

V: CONCLUSIONS:

The typical plant control using instantaneous GPC and NGPC based techniques was discussed in this paper. Their features and main properties were presented and their control performance was illustrated and compared. It was shown that both of them
provided an optimal control performance. The future control inputs using GPC can be found by the direct analysis, thus reducing the computational time and so it is more suitable for the real industrial applications. The results shows that the GPC with the variable weight factor 'λ' approach performs on par with NGPC. But the NGPC is enable us to address the plant" true" dynamics. The CPU time, no. of predictions, the time to reach the set value with constant weight factor the GPC performs on par with NGPC. But the NGPC though with more computational efforts the performance accuracy is good because of system identification capabilities.

REFERENCES

1. Clarke, D.W.; Mohtadi, C. and Tuffs, P.S.: "Generalized predictive control. Pan I: the basic algorithm". Automatic, Vol. 23 n°2, pp. 137-148. 1987.
2. Clarke, D.W.; Mohtadi, C. and Tuffs, P.S.: "Generalized predictive control. Part II: extensions and interpretations". Automatica, Vol. 23 n° 2, pp. 149-160. 1987.
3. Ahmed, M.S. and Anjum, M.F.: "Neural-net-based direct self-tuning control of nonlinear plants". International Journal of Control. Vol. 66, n° 1, pp. 85-104. 1997.
4. Brown, M. and Harris, C.: "Neurofuzzy adaptive modeling and control". Prentice Hall. 1994.
5. Chauvim, Y. and Rumelhart, D.E.: "Backpropagation theory: architectures and applications". Laurence Erlbaum Associates Inc. 1995.
6. Cichocki A. and Unbehauen R.: "Neural networks for optimization and signal processing". John Wiley & Sons Ltd. & E.G. Teubner. 1995.
7. Garcia, C.E.; Prett, D.M. and Morari, M..:"Model predictive control: theory and practice - a survey". IFAC Workshop on Model Based Control, Atlanta-USA, pg 335- 348. June, 1989.
8. Haykin, S.: "Neural Networks: a comprehensive foundation". Macmillian College Publishing Company Inc. 1994.
9. Hunt, K.J.; Sbarbaro, D.; Zbikowski, R. and Gawthrop, P.J.: "Neural networks for control systems - a survey". Automatica. Vol. 28 n° 6. pp. 1083-1112. 1992.
10. Hyland, D.C.; Collins, E.G.Jr.; Haddad, W.M. and Hunter, D.L.: "Neural network system identification for improved noise rejection". International Journal of Control. Vol. 68, n° 2, pp. 233-258. 1997.
11. Miller, W.T.; Sutton, R.S. and Werbos, P.J.: Neural networks for control'. MIT Press, Cambridge, MA. 1990.
12. Narendra, K.S.: "Neural networks for control: theory and practice". Proceedings of the IEEE. Vol.84, n° 10, pp. 1385-1406. 1996.
13. Soloway, D. and Haley, P.: "Neural generalized predictive control: a Newton- Raphson implementation". NASA Technical Memorandum 110244. February, 1997.
14 YANG Can , ZHU Shan-an, KONG Wan-zeng LU Li-ming "Application of neuralized predictive controlling networked control system" Journal of Hewing University SCIENCE A, pp 225-233, ISSN 1009-3095, Jan.2006
15.N.Groschwitz, G.Polyzos, A. "Time series model of long term traffic on the NSFnet Backbone", in proceedings of the IEEE International Conference on Communicatios.2004

First Author:D.N.Rao ,graduate with a specialization in Electronics and Communication Engineering from Government college of engineering, Anantapur, J.N.T.University, Hyderabad(1981),India and.M.E. Degree in control systems engineerg, Walchand College of Engineering, India. He is presently pursuing Ph.D. from Osmania University, Hyd. India. He has published several research papers. International journals, has many conferences to his credit. His areas of interest include neuro computing, system modeling, and control engineering. He is working as faulty in T.R.R.College of Engg.J.N.T.U.email:dhyapulai@yahoo.com

Second Author: Dr.M.R.K.Murthy, obtained his degree in (1977), Osmania University (O.U) and P.G. in 1980, from O.U. He received his PhD degree from IIT, Chennai in 1993. He worked as professor and H.O.D. in the department, O.U., Hyderabad, India. He published several national, international journals and has numerous conferences to his credit. His areas of interest include communications, control systems, neural networks. At Present he is working as Head of Department in M.J.I.College of Engineering, Hyderabad.

S Third Author: Dr.S.R.M.Rao obtained B.E. degree from IISC, Bangalore, and M.E. from NIT, Warangal. He obtained his PhD from IISC, Bangalore. He worked as a Scientist-F in ISRO, INDIA for more than 25 years; His areas of interest are Artificial Neural Networks, signal and systems. At present he is working as professor in E.C.E., Departent.He pulished several papers in reputed journals.

Fourth Author: D.N.Harshal. He is research Associate at Pentagram Ltd, Hyderabad. Graduation in specialization 'Information and technology'. C.B.I.T., Osmania University, Hyderabad, India. His areas of interest are Robotics, Neural Networks and Image processing