A PC-BASED REAL-TIME GENERALIZED PREDICTIVE CONTROLLER

A.P. de Madrid, F. Morilla and S. Dormido

Dpto de Informática, UNED, Avda Senda del Rey s/n, 28040 Madrid, Spain

Abstract. The Generalized Predictive Control (GPC) appears to be suitable as a general purpose algorithm for the stable control of the majority of real processes. Recent applications confirm the effort to take predictive control out of the laboratories and into industry. This work presents examples of simulation and real time control of a dc motor using the tool, for Personal Computers and based on the GPC basic algorithm, developed in our Department. These examples show that: it is possible to obtain good performance with prediction horizons less conservative than the 'default setting'; the developed tool is really useful to test different types of recursive parameter estimation; a new module should be added to GPC in order to get really adaptive control.

Keywords. Predictive control; adaptive control; process control; computer control, software tools.

INTRODUCTION

Predictive methods in adaptive control refer to a collection of control design formulations that pose control criteria at a given time explicitly in terms of predictions of future plant outputs and sometimes of future plant inputs, Bitmead and co-workers. Because such predictions become more difficult as they become more distant in time, these criteria are typically finite horizon optimal control criteria.

All these methods have certain features in common which distinguish them from previous design philosophies: the solution of a finite horizon optimization problem at each time instant implemented in a receding horizon way, the incorporation of plant output predictions, the provision of a small number of design parameters connected to various degrees with the closed loop dynamics.

The version which appears to have had the most acceptability is that derived by Clarke, Mohtadi and Tuffs (1987) and called GPC for Generalized Predictive Control. Because of its control parameters, it is possible to obtain several previous control strategies from it. This method is effective with a plant which is simultaneously nonminimum-phase and open-loop unstable with variable or unknown dead-time and whose model is overparameterized by the estimation scheme without special precautions being taken.

Predictive controllers are not usually employed in the industry; self-tuning PID controllers are the most frequently used. Our Department has been working in the development of self-tuning PID controllers for industrial application during the last ten years, Cruz (1984), Morilla (1987) . Nowadays, we are interested in the development of predictive controllers, for industrial application, and in the comparative study of GPC vs PID controllers. In this way, we have developed a simulation and real-time control tool using the GPC basic algorithm.

This work presents the GPC algorithm used, the tool developed and examples of simulation and real time control of a dc motor.

GPC: THE BASIC ALGORITHM

GPC is based on an ARIMAX process model. For a SISO system, it is given by the following form:

$$A(q^{-1})\, y_t = B(q^{-1})\, u_{t-k} + \frac{C(q^{-1})}{\Delta}\, \xi_t$$

where: u_t is the control input, y_t is the measured variable or output, ξ_t is an uncorrelated random sequence and Δ is the differencing operator. The control law is obtained by minimizing the cost function

$$J(u,t) = E\left\{ \sum_{j=N_1}^{N_2} (y_{t+j} - w_{t+j})^2 + \lambda \sum_{j=1}^{N_u} (\Delta u_{t+j-1})^2 \right\}$$

where: w_t is the set-point, N_1 is the minimum output horizon, N_2 is the maximum output horizon, N_u is the control horizon, λ is the control-weighting and $\Delta u_{t+j-1} = 0 \quad j \geq N_u$. The equation that yields the future control increments for times t to $t+N_u-1$ is

$$\hat{u} = (G^T G + \lambda I)^{-1} G^T (w - f)$$

where (assuming that $N_1=1$) :

$$\hat{u} = [\Delta u_t, \ \Delta u_{t+1}, \ ..., \Delta u_{t+Nu-1}]^T$$

$$w = [w_{t+1}, \ w_{t+2}, \ ..., \ w_{t+N_2}]^T$$

$$f = [y_{t+1|t}, \ y_{t+2|t}, \ ..., \ y_{t+N_2|t}]^T$$

and the matrix G is composed of the impulse response parameters, g_i, of the plant model $B/A\Delta$,

$$G = \begin{bmatrix} g_0 & 0 & ... & 0 \\ g_1 & g_0 & ... & 0 \\ ... & ... & ... & ... \\ g_{N_2-1} & g_{N_2-2} & ... & g_{N_2-N_u} \end{bmatrix}$$

The controller parameters are chosen based on the type of process and the desired output. Different control strategies are obtained from different settings of the GPC parameters, but one of them the 'default settings', Lambert (1987), has proved to be adequate for a wide variety of applications:

$$N_1 = 1, \quad N_u=1,$$

$N_2=10$ (equal to the plant rise-time), $\lambda=10^{-6}$

A recursive parameter estimator may be included, to obtain an adaptive control strategy. In this case, the control law has to be recalculated at each time instant.

PROGRAM DESCRIPTION

We have developed a simulation and real-time control tool for Personal Computers, based on the GPC basic algorithm. It was designed to allow the user to experiment with GPC in an interactive way and become more skilled in this control strategy, studying how the control parameters influence in the system response. It is configured for a Metrabyte DAS-16 board, but it can be easily modified for any other kind of data acquisition board. The user can access to all of its features by menus, and gets all the results in graphic format.

Next, we describe the four main menus, how the program works and how it has been implemented.

PROCESS: For the simulation case, the user defines the process to be controlled. As this process can change its dynamics in time, it is necessary to introduce the coefficients of polynomials A and B for every time instant. They are written in an ASCII file, and the user indicates the file name in this menu. For real time control, there is a real process, so this file (if it exists) is ignored.

The process model used to calculate the control law can be different (order and/or parameters) to the real process. If the parameters of this model are fixed, they are introduced in this menu, for both simulation and real time control.

GPC: This menu is used to introduce the GPC controller parameters: N_1, N_u, N_2, λ.

PARAMETERS: From this menu, the user can access to other menus, to set the sampling time, the set-point, the experiment duration and introduce white noise in the simulations (optional). For the adaptive controller, it is necessary to indicate the type of parameter estimation which will be used. A recursive least squares algorithm is selected, such as fixed exponential forgetting, variable exponential forgetting or fixed trace, and its parameters (initial forgetting factor and trace) are initialized. It is also necessary to select the process model order, and give the initial values of the estimated A and B polynomials.

For simulation, the user can decide if the control signal sent to the process will be just that one calculated by GPC, or if a control saturation must be included. This is explained in detail in the next section of this work.

RUN: From this menu, the user selects the type of experiment to do: simulation or real-time control, both in a fixed-parameters or adaptive way, or exit .

Once all the options and parameters are set up, the experiment starts. All the results are displayed in graphic mode, and some ASCII data files are generated.

If a fixed-parameter controller has been selected, a graphic screen with the closed-loop zeros and poles positions is shown. It is useful to decide whether the selected GPC parameters are correct or not, and discover if the system response is going to be unstable.

After this, in the simulation case, the set-point and the system output are displayed in a window, and the control signal applied to the process in another one. For real-time control, the set-point is shown just at the beginning, and the control and output signals are shown as they are obtained.

Finally, for the adaptive controllers, there is a screen for the estimated parameters, forgetting factor and trace evolution. This values are also written in an ASCII file.

Implementation

The main program was developed in Turbo Pascal. In order to get a flexible and easy-to-program graphic output, we first tried to use the Turbo Pascal Graphix Toolbox, but this tool was not able to display real-time data as they are being obtained. So, to solve this problem, we developed a graphic library in Turbo Pascal. It can handle 'real-time graphics', and it is quite similar to the graphic functions of Matlab: the screen can be split up in several windows, it is possible to display just a single point or some data vectors and, given a transfer function, it calculates and displays the poles and zeros in the z plane.

The library file TP4D16.PAS from Quinn-Curtis Metrabyte Turbo Pascal Data Adquisition and Control Tools (IPC-TP-018) has been used to communicate with the DAS-16 board but, in order to better handle the sampling time, we have modified some of its functions.

The 'pull-down' menu system has been developed in C, using the CSCAPE functions and the screen designer Look And Feel, both by Oakland Group, Inc. It allows the user to modify all the parameters and select the experiment to do in a friendly and interactive way.

APPLICATION EXAMPLES

We have chosen the Ball and Hoop equipment by TecQuipment to show how the program works. It is a quite easy system to control, but the influence of the GPC parameters can be studied. The speed of the dc electric motor was modelled as a first order linear system, unit gain and time constant of 1 second; a sampling time of 0.1 seconds is adequate. Thus, $A = 1 - 0.9048q^{-1}$ and $B = 0.0952q^{-1}$.

Next we introduce some representative experiments done with our program, and comment some of the results we have got. A sequence of set-point changes between 3 and 5 volts was provided with switching every 100 samples. The graphs display results over 500 samples of simulation or real-time control, using solid line for the set-point and dotted line for the process output and the control signal.

1) Effect of the horizons

Before starting the experiments, it is necessary to decide what horizons and λ are going to be used. We have studied the system response for prediction horizons smaller than 10. As we can see from the root-locus for N_2 (see Fig. 1), the position of the poles indicates that the output can be good for such an easy system with values of N_2 smaller than 10.

We have found the 'default settings' too 'conservative' for this process (see Fig. 2). Its is possible to obtain a good performance with a design less conservative than the 'default one', (see Fig. 3), at least for easy systems. And the smaller the horizons, the faster the computer can calculate the control signal. With a cheap - and slow - computer, the adaptive control of a fast process with a prediction horizon of 10 could be impossible.

We are studing the possibility of adding a new 'module' to GPC, to real-time modify the values of the horizons and λ, based on the system dynamics and the kind of output desired (see Fig. 4). This module would use both analytical and heuristic tools to modify the control parameters in two possible cases:
- The actual control parameters are too high, and it is possible to obtain the desired output with smaller values (faster computing).
- The system output is bad, because the horizons are no high enough (improving the performance).

2) Control signal saturation

GPC can obtain a really fast system response to changes in the set-point. But the cost is the high value of the control signal applied to it.

In general, the tension that can be sent from the board to the system has a minimum and a maximun value (in our case, we have configured the board for a range from 0 to 10 Volts). So, real processes

cannot be controlled with different control values. If the control program tries to send a higher value to the system, it is saturated by the board, but in this case the algorithm is working with wrong tension values. To avoid this error, our program saturates the control signal before it is sent to the board, and includes this real value in all its future computations. What is more, this can be done in simulation too: the user indicates the control range he wants, which can be different to the board output range.

3) Adaptive control

Finally, the program has proved to be very useful to test different types of recursive least squares algorithms, allowing the user to study how they work and become more skilled in system identification and adaptive control.

The computer used for these experiments is a PC/AT (8 MHz) with math coprocessor, but it results too slow to real-time estimate the model parameters with a sampling time of 0.1 seconds. Thus, a sampling time of 0.3 seconds was used. In this case, $A = 1 - 0.7408q^{-1}$ and $B = 0.2592q^{-1}$. The plant rise-time is 3.3 sampling times, so we have chosen $N_2 = 4$.

In the example that we present, we have used a recursive least squares algorithm with fixed trace of 10. The initial estimated parameters are -0.7 and 0.2 (a bit different to those of the model).

The adaptation mechanism worked well and the system response improved in time (see Fig. 5) as the estimator converged to a model different to the initial one. In Fig. 6, the estimated parameters are displayed at the top left corner, the forgetting factor evolution at the bottom, and the trace (constant) at the top right corner.

CONCLUSIONS AND FURTHER WORK

The program we have developed has proved to be a good tool to study on a PC how GPC performs not only under ideal simulation conditions but when it is applied to real processes. GPC has proved to be a control algorithm as robust and versatile as it was expected.

It is our personal belief that a new feature should be added to GPC: the possibility of 'auto-tune' the horizons in real-time, based on the system observed dynamics. This could be done by using both analytical and heuristic criteria together. Nowadays, we are working in this way.

REFERENCES

Bitmead, R. B., M. Gevers and V. Wertz. Adaptive Optimal Control: The Thinking Man's GPC. Prentice Hall International. Series in Systems and Control Engineering.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987a). Generalized Predictive Control. Part I. The Basic Algorithm. Automatica, Vol. 23, No. 2, pp. 137-148.

Clarke, D. W., C. Mohtadi and P. S. Tuffs (1987b). Generalized Predictive Control. Part II. Extensions and Interpretations. Automatica, Vol. 23, No. 2, pp. 149-160.

Cruz, J. M. de la (1984). Contribución al estudio y síntesis de reguladores autosintonizados. Doctoral Dissertation, Universidad Complutense.

Lamber, E. P. (1987). Process Control Applications of Long-Range Prediction. Report No OUEL 1715/87. Department of Engineering Science. University of Oxford.

Morilla, F. (1987). Contribución a los métodos de autosintonía de reguladores PID. Doctoral Dissertation, UNED.

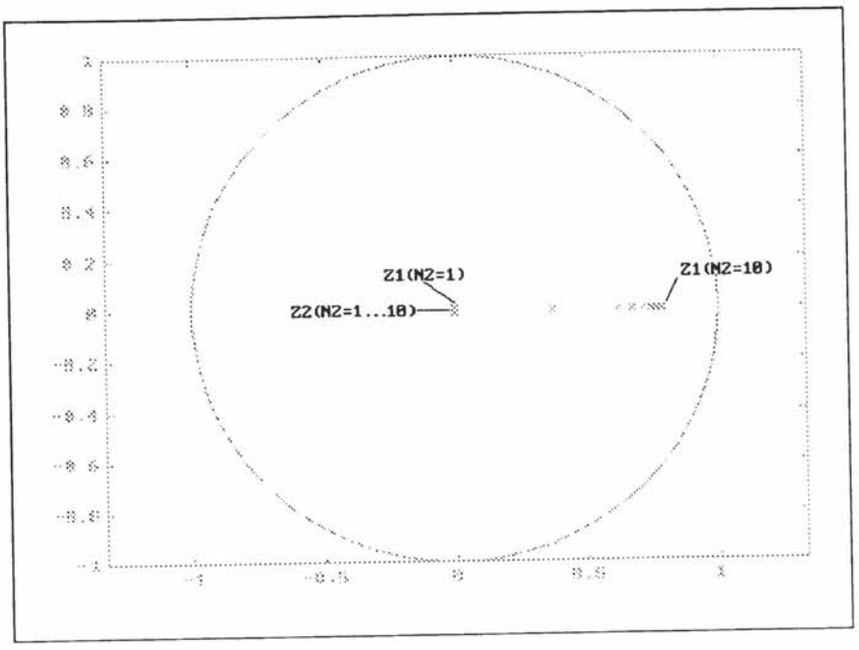

Fig.1 Root-locus for the dc motor with $N_1=N_u=1$, $\lambda=10^{-6}$ and N_2 variable from 1 to 10.

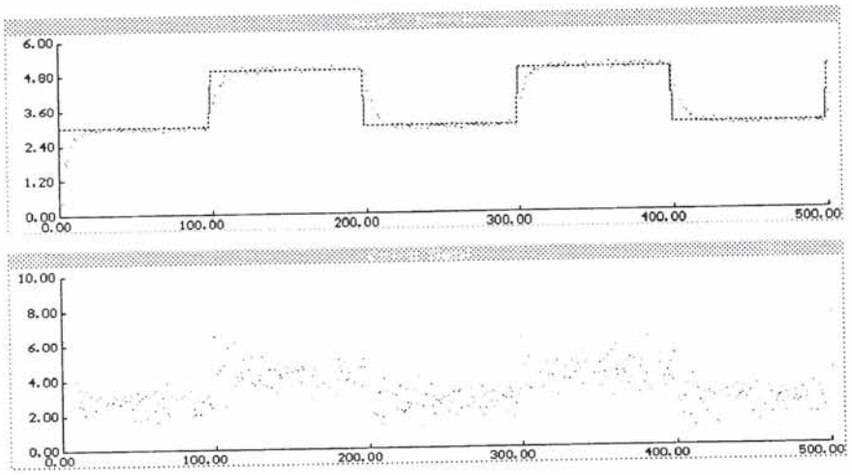

Fig.2 Speed control of the motor with $N_1=N_u=1$, $\lambda=10^{-6}$, $N_2=10$ and the fixed process model.

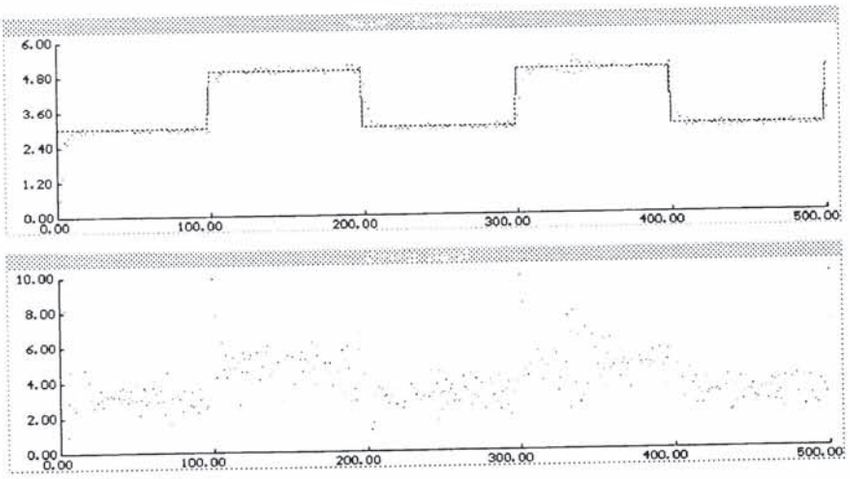

Fig.3 Speed control of the motor with $N_1=N_u=1$, $\lambda=10^{-6}$, $N_2=5$ and the fixed process model.

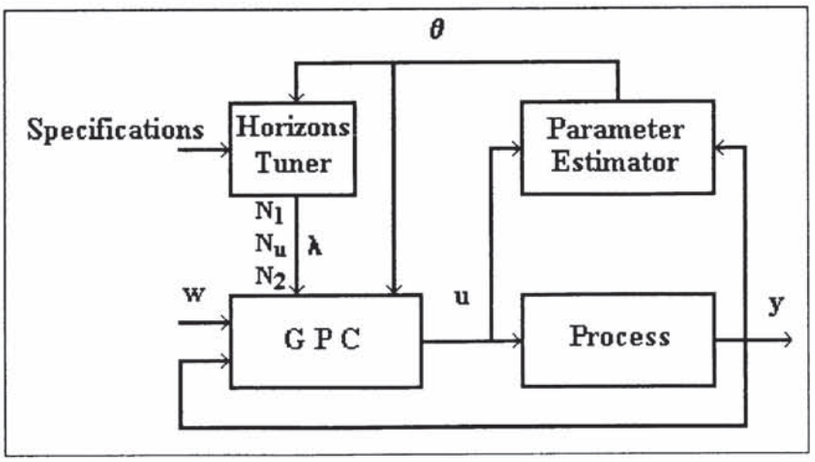

Fig.4 Scheme of a really Adaptive GPC.

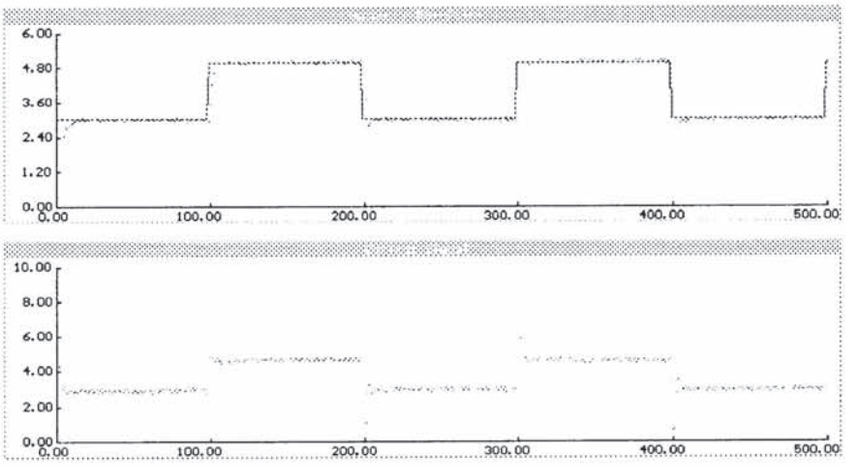

Fig.5 Speed control of the motor with $N_1=N_u=1$, $\lambda=10^{-6}$, $N_2=4$ and adaptive process model (sampling time 0.3 sec).

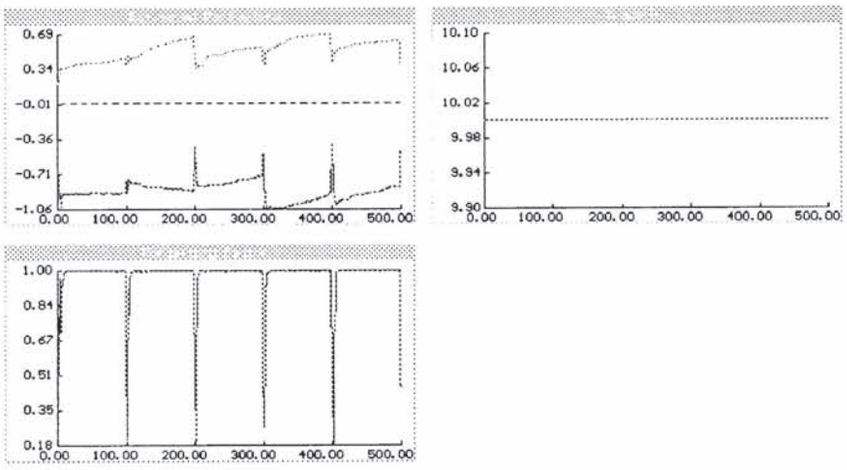

Fig.6 Speed control of the motor with $N_1=N_u=1$, $\lambda=10^{-6}$, $N_2=4$ and adaptive process model.