Extended input horizon generalized predictive control - a GPC algorithm with better tracking

J.A.Rossiter and B.G. Grinnell
Department of Mathematical Sciences, Loughborough University, Leicestershire LE11 3TU
email: J.A.Rossiter @ lut.ac.uk

May 1, 1995

Keywords: Predictive control, tracking

Abstract

One of the advantages of predictive control is its ability to take optimal account of information about future set point changes in the specification of the control law. However, it is demonstrated in this paper that often use of this information can lead to a deterioration rather than an improvement in the accuracy of tracking. A simple modification to GPC (Generalized predictive control) is proposed to overcome this problem.

1 Introduction

Model based predictive control (MBPC) has been a very popular research area in recent years [1]. Many variants of MBPC have been developed (e.g. [2-5]) which share similar properties. In this paper for convenience, only one of the most popular variants of MBPC, that is Generalized Predictive control (GPC), will be considered explicitly, but many of the ideas developed in this paper are also applicable to many of the other MBPC approaches.

Traditionally one of the strengths of MBPC is its ability to make systematic use of advance knowledge of set-point changes for optimum tracking. An ability to give optimum tracking should give predictive controllers a marked advantage in such areas as electrical power generation where fairly accuarate knowledge of future set-points is often available. However, rather suprisingly no detailed analysis of this claim appears in the literature and in fact the claim seems to rest solely on the presence of the set-point trajectory in the performance index used to define a predictive controller. It will be demonstrated in this paper that while predictive control should give good tracking (because it uses advance knowledge of set-point changes), this is often not the case. In fact, better tracking is often achieved when only some (not all) of the future information is given to the controller; this raises two important issues. Firstly it begs the question 'how can one systematically make a sensible choice of the amount of future information a predictive controller will use effectively' ? Secondly it suggests that MBPC is incorrectly formulated to make the best use of the set-point information available. In this paper it is the latter of these two issues which is tackled.

The reason why a predictive controller can give poor tracking is because of the formulation of the performance index from which the control law is derived. In GPC the aim is to minimise the sum of squares of predicted output errors over a 'large' output horizon (n_y), but the degrees of freedom and the control changes are over a 'small' input horizon (n_u). Typically $n_y \gg n_u$ and hence the control law is trying to minimise both predicted 'transient' and predicted 'steady-state' errors with only a few control moves that occur during the 'transient' stage of the prediction horizon. (Here 'steady-state' is used to refer to predicted values well beyond the input horizon but still within the output horizon.) As a result there is only indirect control of steady-state errors, i.e. through the degrees of freedom available during 'transients'; use of these degrees of freedom to reduce predicted 'steady-state' errors almost invariably causes severe degradation of 'transient errors'. This problem is particularly noticeable when the set-point trajectory has markedly different values in the 'transient' and 'steady-state' parts of the prediction horizon. In summary the performance index of a predictive controller is slightly lop sided as in the control changes are at one end of the horizon whereas we are interested in the outputs tracking accurately over the whole length of the horizon.

One apparent solution to this problem may be to increase the number of control changes available to the control law, however this is generally not advisable as choosing the input horizon and output horizon to be of a similar size allows the predictive controller to effectively invert the plant, (this is one of the weak-

nesses of minimimum variance controllers [6]); if the plant has any unstable zeros the resulting control law will be destabilising and even if this were not the case the controller would be too highly tuned for many industrial applications. Hence in practice the condition $n_y \gg n_u$ is an acccepted rule of thumb in predictive control[1]. However, the important idea remains, that is to to try and extend the direct influence of the control changes over a wider range of the output horizon. This can be achieved by another means, that is to select the predicted control as changing only every n-samples (n=1 in most predictive controllers). Thus the effective control (input) horizon is expanded approximately n times but while retaining the same numbers of degrees of freedom and therefore being no more computationally burdensome nor more highly tuned than GPC. A similar idea to this has been used before [7] on a boiler-turbine model, though it was used specifically to cater for a multivariable system containing both fast and slow modes. Here that idea is developed to deal explicitly with the issue of advance knowledge of the set-point and for clarity of exposition now will be restricted to single-input-single-output systems.

If the predicted control changes are to be spread over the output horizon in an efficient way, it is not clear what is the best way to do this. In this paper a simple technique is presented, but it is noted that further improvements may be possible in general. However, it is argued that the technique proposed has the virtue of being very simple and the improvements to be gained by a more advanced technique would probably be marginal, especially if one restricts oneself to the fairly small control horizons typically used.

This paper will be divided into 3 sections. In section 2 an overview of GPC and the new proposed algorithm will be given. Then, using examples, in section 3 both algorithms will be compared for their tracking performance and robustness.

2 Background to GPC

2.0.1 Notation

The system model will be taken to be

$$y(z) = \frac{z^{-1}b(z)}{a(z)}u(z) = g(z)u(z) \qquad (1)$$

where $b(z) = b_o + b_1 z^{-1} + ... + b_{n-1}z^{n-1}$, $a(z) = 1 + a_1 z^{-1} + ... + a_n z^{-n}$. The system output predictions can be derived by solving row by row the equation

$$C_A \underset{\rightarrow}{y} = C_b \Delta \underset{\rightarrow}{u} + H_b \Delta \underset{\leftarrow}{u} - H_A \underset{\leftarrow}{y} \qquad (2)$$

[1]It is noted that this is not the case in predictive controllers with guaranteed stability but they tend to be highly tuned as a result

where $\Delta = 1 - z^{-1}$, $A = a(z)\Delta(z)$, $\Delta u_t = u_t - u_{t-1}$, n_y the output horizon and

$$\underset{\rightarrow}{y} = \begin{bmatrix} y_{t+1} \\ \vdots \\ y_{t+n_y} \end{bmatrix}, \quad \Delta \underset{\rightarrow}{u} = \begin{bmatrix} \Delta u_t \\ \vdots \\ \Delta u_{t+n_y-1} \end{bmatrix}$$
$$\underset{\leftarrow}{y} = \begin{bmatrix} y_t \\ \vdots \\ y_{t-n} \end{bmatrix}, \quad \Delta \underset{\leftarrow}{u} = \begin{bmatrix} \Delta u_{t-1} \\ \vdots \\ \Delta u_{t-n+1} \end{bmatrix} \qquad (3)$$

The matrices C_m is defined as having its i,j element equal to m_{i-j} and H_m as having its i,j element equal to m_{i+j-1} where $m(z) = m_o + m_1 z^{-1} + m_2 z^{-2} + ...$

General definition of predictions in GPC

In eqn. (2) the output predictions depend upon all the future control increments which are to be chosen by the user and therefore are degrees of freedom. It is not desireable to have n_y degrees of freedom in $\Delta \underset{\rightarrow}{u}$ due to dangers of the controller inverting the plant, so $\Delta \underset{\rightarrow}{u}$ is artificially shortened. In order to have a general format define the shortened vector as

$$\Delta \underset{\rightarrow}{\hat{u}} = [\Delta u_{t+v_1}, \Delta u_{t+v_2}, \Delta u_{t+v_3}, ..., \Delta u_{t+v_{n_u}}]^T \qquad (4)$$

where it is assumed that for any $i > 0$ where Δu_{t+i} is not in $\Delta \underset{\rightarrow}{\hat{u}}$ then $\Delta u_{t+i} = 0$. n_u the control horizon.

(Note: For standard GPC [2] $\mathbf{v} = [v_1, v_2, ..., v_{n_u}] = [0, 1, 2, ..., n_u - 1]$.) Including $\Delta \underset{\rightarrow}{\hat{u}}$ and rearranging, the GPC prediction equation (2) can be written as

$$\underset{\rightarrow}{y} = H \Delta \underset{\rightarrow}{\hat{u}} + P_A \underset{\leftarrow}{y} + P_b \Delta \underset{\leftarrow}{u} \qquad (5)$$

with $P_b = C_A^{-1} H_b, P_A = C_A^{-1} H_A$ and H as the $[v_1, ..., v_{n_u}]$ columns of $C_A^{-1} C_b$.

Definition - Advance knowledge

Taking r_t to be the set point at time t, it will be convenient to have a parameter n_y^a which will be used to denote the number of r_{t+i} which are available at sample period t. Define advance knowledge as n_y^a if

$$r_{t+i} = \begin{cases} r_{t+i} & i < n_y^a \\ r_{t+n_y^a} & i \geq n_y^a \end{cases} \qquad (6)$$

and therefore the n_y vector
$\underset{\rightarrow}{r} = [r_{t+1}, ..., r_{t+n_y^a-1}, r_{t+n_y^a}, r_{t+n_y^a}, ...]^T$.

2.1 GPC

The GPC control law is defined through the minimisation of the performance index

$$J_{GPC} = \| \underset{\rightarrow}{r} - \underset{\rightarrow}{y} \|_2^2 + \lambda \| \Delta \underset{\rightarrow}{\hat{u}} \|_2^2 \qquad (7)$$

In the previous section the system predictions $\underset{\rightarrow}{y}$ (5) were given for $\Delta \underset{\rightarrow}{\hat{u}}$ containing n_u terms, the terms

being defined by the choice of $v_i, i = 1, ..., n_u$. In GPC [2] $\Delta \underset{\rightarrow}{u}$ is selected to have $v_i = i - 1$, and as a result the H matrix of eqn. (5) is truncated to only contain the first n_u columns of $C_A^{-1}C_b$.

Theorem 2.1 *The GPC control law defined through the minimisation with respect to $\Delta \underset{\rightarrow}{\hat{u}}$ of the performance index (7) is*

$$\Delta u_t = P_r \underset{\rightarrow}{r} - \tilde{D}_k \Delta \underset{\leftarrow}{u} - N_k \underset{\leftarrow}{y} \qquad (8)$$

where with e_1 the first standard basis vector

$$\begin{aligned} P_r &= e_1^T (H^T H + \lambda I)^{-1} H^T \\ \tilde{D}_k &= P_r P_b \\ N_k &= P_r P_A \end{aligned} \qquad (9)$$

Proof: Minimising the performance index (7) with respect to the vector of predicted future control increments $\Delta \underset{\rightarrow}{\hat{u}}$ gives

$$\Delta \underset{\rightarrow}{\hat{u}} = (H^T H + \lambda I)^{-1} H^T (\underset{\rightarrow}{r} - P_A \underset{\leftarrow}{y} - P_b \Delta \underset{\leftarrow}{u}) \qquad (10)$$

Of the optimal $\Delta \underset{\rightarrow}{u}$ only the first element (that is Δu_t) is implemented and the optimisation is repeated at the next sample. This gives rise to the control law of equation (8). □

2.2 Extended input horizon GPC

GPC places all of the degrees of freedom available for minimising J_{GPC} in the first n_u predicted control increments, i.e. $\mathbf{v} = [0, 1, ..., n_u - 1]$. The performance index however includes tracking errors over an output horizon n_y. It has been argued in the introduction that typically speaking $n_y \gg n_u$ and this leads to an unbalanced performance index. A better algorithm would distribute the control changes more equally over the output horizon and therefore widen the regions of influence. This is particularly important when the set-point is changing within the output horizon and the output is therefore being driven to different positions at the beginning and end of the output horizon. However, due to the dyanmic behaviour of most systems, it is also true that one would not expect, for example, control increment Δu_{t+n_y-1} to influence greatly the output predictions $\underset{\rightarrow}{y}$. Therefore while it is desireable to have control increments spread evenly over most of the output horizon, they are not as influential over the very last part of the output horizon. Accordingly an extended input horizon GPC (EIHGPC) algorithm can be defined to spread the control increments over more of the output horizon.

Algorithm 2.1 *EIHGPC is defined exactly as GPC (Theorem 2.1) but with*

$$\mathbf{v} = [0, 1, 1 + \beta, 1 + 2\beta, ..., 1 + (n_u - 2)\beta] \qquad (11)$$

where β is chosen to be any positive integer such that $1 + (n_u - 2)\beta < n_y$.

Remark: The change in $\mathbf{v}$ entails only a change in the H matrix which comprises the $\mathbf{v}$ columns from $C_A^{-1}C_b$. Thus the difference in $\mathbf{v}$ between EIHGPC and GPC is that for GPC $\beta = 1$ and for *EIHGPC* $\beta > 1$. The two algorithms are identical with the definition of eqn.(11) if $n_u \leq 2$, but of course it is possible to define $\mathbf{v} = [1, \beta]$ for $n_u = 2$ in which case the two algorithms would be different. GPC and EIHGPC are always identical for $n_u = 1$.

3 Comparison of GPC with EIHGPC

3.1 Prediction comparisons

The benefits of EIHGPC over GPC are best illustrated by considering the optimal open-loop predictions arising from the minimisation of J_{GPC}. Let the optimal $\Delta \underset{\rightarrow}{\hat{u}}$ given in eqn. (10) be $\Delta \underset{\rightarrow opt}{u}$. To find the optimal output prediction ($\underset{\rightarrow}{y}$) the optimum $\Delta \underset{\rightarrow opt}{u}$ is substituted into eqn. (5).

$$\underset{\rightarrow opt}{y} = H \Delta \underset{\rightarrow opt}{\hat{u}} + P_a \underset{\leftarrow}{y} + P_b \Delta \underset{\leftarrow}{u} \qquad (12)$$

Therefore the minimum value of the performance index at time t as

$$J_{opt}(t) = \| \underset{\rightarrow}{r} - \underset{\rightarrow opt}{y} \|_2^2 + \lambda \| \Delta \underset{\rightarrow opt}{u} \|_2^2 \qquad (13)$$

In this section a system will be taken from a given initial condition (i.e $\underset{\leftarrow}{y}, \Delta \underset{\leftarrow}{u}$ given) and the vector of future set-points $\underset{\rightarrow}{r}$ will be taken to contain a unit step at n_y^a samples into the future from the start of the simulations. To compare the efficacy of GPC and EIHGPC the plots of $\underset{\rightarrow opt}{y}$ and $\underset{\rightarrow opt}{u}$ will be plotted and compared with $\underset{\rightarrow}{r}$. Furthermore $J_{opt}(t)$ will be evaluated for each algorithm.

3.1.1 Example 1

For this example

$$a(z) = 1 - 0.9z^{-1}; \quad b(z) = 0.5 \qquad (14)$$

and $n_y = 12, n_u = 3, \lambda = 0.25, \beta = 4$. The optimal predictions $\underset{\rightarrow opt}{y}$ and $\underset{\rightarrow opt}{u}$ for both GPC and EIHGPC are computed for three cases (i) $n_y^a = n_y$, (ii) $n_y^a = n_y - 3$, $n_y^a = n_y - 9$ and plotted in Figures 1a,b,c and Figures 2a,b,c respectively. $\underset{\rightarrow opt}{y}$ and $\underset{\rightarrow opt}{u}$ are plotted in dotted lines for GPC and and solid lines for EIHGPC. $\underset{\rightarrow}{r}$ is plotted in a solid line. In each case it is assumed that the step in r is n_y^a

steps in the future and that $\underset{\leftarrow}{y} = 0$ and $\Delta\underset{\leftarrow}{u} = 0$. Furthermore $J_{opt}(t)$ is computed for each case above and also for $n_y^a = n_y - 6$ and tabulated below.

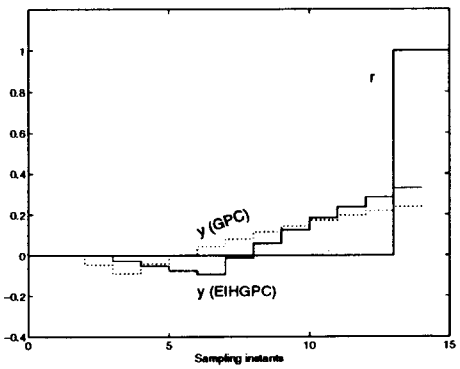

Figure 1a. Output predictions Example 1, $n_y^a = 12$

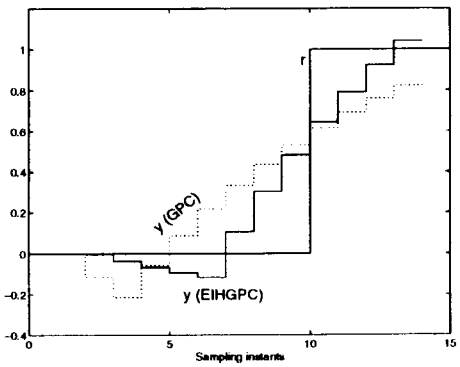

Figure 1b. Output predictions Example 1, $n_y^a = 9$

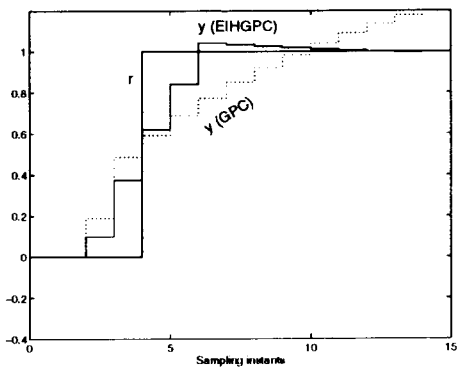

Figure 1c. Output predictions Example 1, $n_y^a = 3$

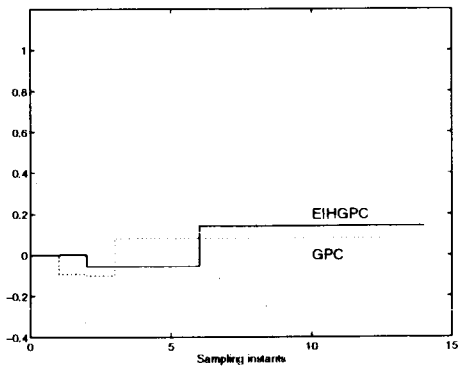

Figure 2a. Input predictions Example 1, $n_y^a = 12$

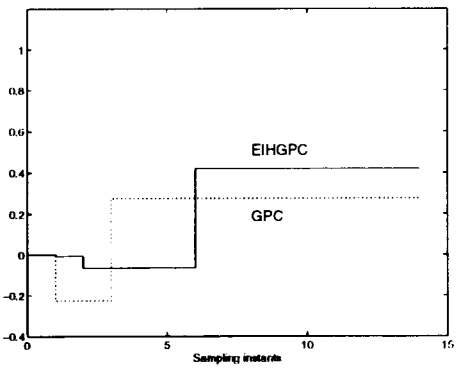

Figure 2b. Input predictions Example 1, $n_y^a = 9$

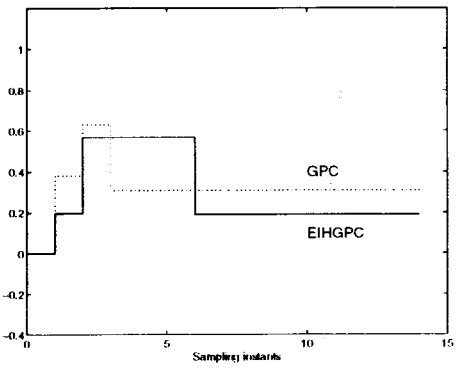

Figure 2c. Input predictions Example 1, $n_y^a = 3$

	Values of $J_{opt}(t)$			
	$n_y^a = 12$	$n_y^a = 9$	$n_y^a = 6$	$n_y^a = 3$
GPC	0.7629	1.1078	0.6844	0.7534
EIHGPC	0.6707	0.6029	0.7197	0.4021

Table 1

Figures 1a,b,c illustrate that the GPC algorithm gives optimal output predictions $\underset{\rightarrow}{y}_{opt}$ which have poor predicted tracking, especially if n_y^a is similar in magnitude to n_y. This is because in the prediction stage $\underset{\rightarrow}{u}$ (Figures 2a,b,c) stops changing after only 3 samples. EIHGPC on the other hand gives markedly better tracking because the predicted changes of $\underset{\rightarrow}{u}$ are more evenly spread over the output horizon. The improvement in tracking given by EIHGPC is further confirmed by the comparison of the values for $J_{opt}(t)$ in table 1.

3.1.2 Example 2

Let

$$\begin{aligned} a(z) &= 1 - 1.85z^{-1} + 1.11z^{-2} - 0.221z^{-3} \\ b(z) &= 1 - 0.5z^{-1} - 0.14z^{-2} \end{aligned} \tag{15}$$

with $n_y = 10, n_u = 3, \beta = 3, \lambda = 2$. A representative set of optimal output and input predictions are plotted in Figure 3a,b respectively for $n_y^a = 8$ and $\underset{\leftarrow}{y} = 0, \Delta\underset{\leftarrow}{u} = 0$ with the $\underset{\rightarrow}{r}$ containing a unit step n_y^a samples into the future.

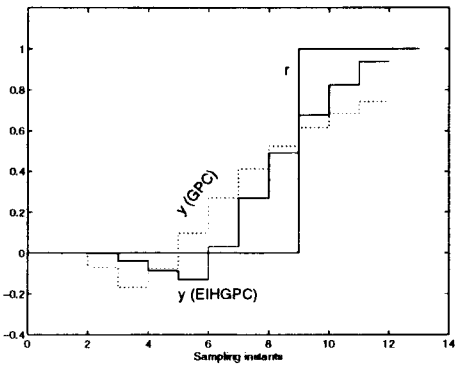

Figure 3a. Output predictions Example 2, $n_y^a = 8$

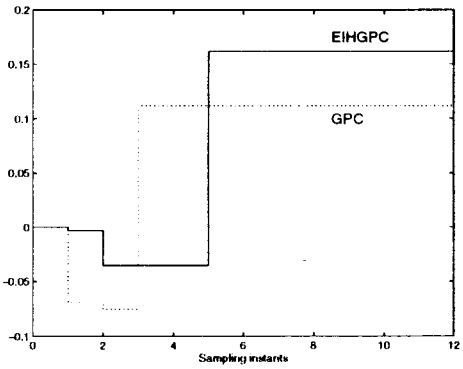

Figure 3b. Input predictions Example 2, $n_y^a = 8$

It is seen that the optimal predictions from EIHGPC have far better predicted tracking than those from GPC. The values of $J_{opt}(t)$ for a selection of n_y^a are given in in table 2. The corresponding plots for $n_y^a = 10, 6, 4$ are very similar to those already in the paper and hence are not included for reasons of space.

	Values of $J_{opt}(t)$			
	$n_y^a = 10$	$n_y^a = 8$	$n_y^a = 6$	$n_y^a = 4$
GPC	0.7290	0.9600	0.5684	0.5171
EIHGPC	0.6313	0.5599	0.4737	0.4591

Table 2

3.2 Closed-loop simulation comparisons

It is clear from the example of section (3.1) that EIHGPC affords better optimal predictions than GPC and therefore one would expect that the closed-tracking arising from an EIHGPC control law would also be better than that arising from GPC. In this section using the same two examples and also using the same initial conditions as in the earlier section, the full closed-loop simulations are presented. For convenience the simulations results for GPC and EIHGPC are presented on the same graph; the output and input simulations for GPC are plotted in a dotted line and for EIHGPC in a solid line. The set point is plotted with a solid line. As another means of comparing the accuracy of tracking the following measure of achieved performance is computed over the runtime

$$J_{run} = \sum_{i=1}^{\text{runtime}} (r_{t+i} - y_{t+i})^2 + \lambda \Delta u_{t+i-1}^2 \qquad (16)$$

3.2.1 Example 1

The closed-loop simulations for example 1 are plotted in Figures 4a,b,c for $n_y^a = 12, 9, 3$ respectively and the values of J_{run} (eqn. 16) are tabulated in table 3 below.

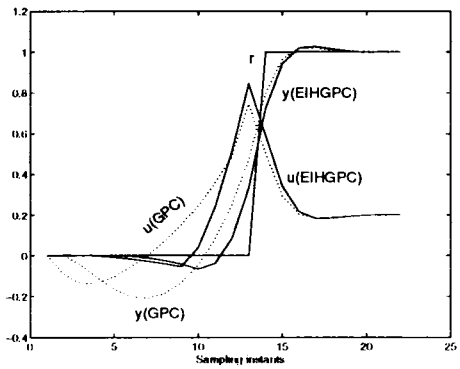

Figure 4a. Closed-loop Simulations with $n_y^a = 12$

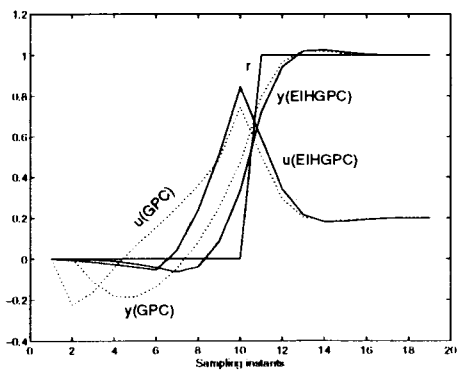

Figure 4b. Closed-loop Simulations with $n_y^a = 9$

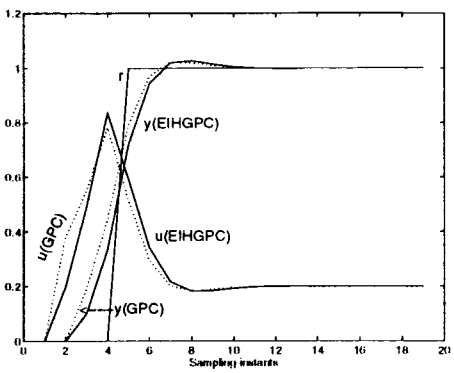

Figure 4c. Closed-loop Simulations with $n_y^a = 3$

	Values of J_{run}			
	$n_y^a = 12$	$n_y^a = 9$	$n_y^a = 6$	$n_y^a = 3$
GPC	0.5784	0.5151	0.4023	0.3688
EIHGPC	0.3026	0.3026	0.3019	0.3008

Table 3

It is clear that in Figs. 4a,b GPC has significant transient errors and in particular a very large non-minimum phase characteristic. EIHGPC on the other hand gives good tracking with minimal non-minimum phase behaviour. This is confirmed by the values of J_{run} (Table 3) which show that EIHGPC has done significantly better than GPC. What is also significant however is that the performance of GPC actually improves as n_y^a gets smaller which suggests that in this case it would be better not to use all the advance knowledge of the set point even when it is available. In fact for this case with GPC the smallest value of $J_{run} = 0.3404$ and occurs for $n_y^a = 2$. EIHGPC on the other hand is relatively unaffected by the choice of n_y^a and achieves its smallest value for $n_y^a = 3$. However, for all choices of n_y^a EIHGPC's largest value of J_{run} is over 10% better than GPC's smallest value. It is noted in passing that the change in control activity as n_y^a changes is minimal.

3.2.2 Example 2

The full closed-loop simulations with example 2 show a similar pattern to that of example 1, i.e. GPC has an undesireable non-minimum phase behaviour caused by n_y^a being too high whereas EIHGPC does not. The closed-loop simulation for $n_y^a = 8$ is given in Figure 5, and the values of J_{run} are in table 4.

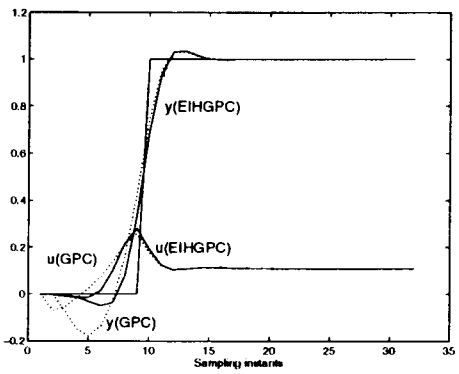

Figure 5. Closed-loop Simulations with $n_y^a = 8$

	Values of J_{run}			
	$n_y^a = 10$	$n_y^a = 8$	$n_y^a = 6$	$n_y^a = 4$
GPC	0.4532	0.4233	0.3543	0.3436
EIHGPC	0.3066	0.3066	0.3061	0.3106

Table 4

Once again it is noted that the performance of GPC has improved if $n_y^a < n_y$ and for this example it achieves its minimum at $J_{run} = 0.3408$ for $n_y^a = 3$. Again EIHGPC is relatively insensitive to n_y^a and reaches its minimum for $n_y^a = 6$ with all its values for $n_y^a > 1$ being lower than the minimum afforded by GPC.

3.3 Simulations with no advance knowledge

For completeness we also present simulations where there is no advance knowledge ($n_y^a = 1$). In this case GPC does not experience any difficulties associated to poor predictions at each sample instant as the set point is taken to be constant over the prediction horizon. The simulations with no advance knowledge are given in Fig. 6a,b for examples 1,2 respectively and the values of J_{run} are given in Table 5.

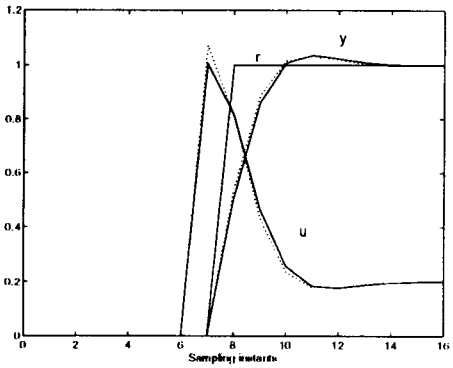

Figure 6a. Closed-loop Simulations with $n_y^a = 1$

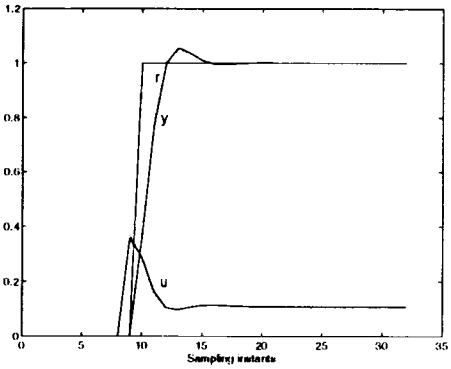

Figure 6b. Closed-loop Simulations with $n_y^a = 1$

	Values of J_{run}	
	Example 1	Example 2
GPC	0.7744	0.5800
EIHGPC	0.7730	0.5733

Table 5

In Figure 6a it is seen that the GPC repsonse is marginally faster than the EIHGPC response, but the difference is insignificant. In Figure 6b the responses are so similar that is not possible to distinguish them on the graph. This is further illustrated by a comparison of the respective values of J_{run} in Table 5. It is noted that the values of J_{run} are higher than those in Tables 3,4 which is as expected with no anticipation at all. The similarity seen in examples

1,2 is not generic and for different examples and different horizons the simulation resposnes of EIHGPC and GPC can vary considerably, as in Fig 4a-c. However, it has been observed that neither algorithm is generally better than the other when $n_y^a = 1$.

3.4 Robustness

It is important to consider the robustness of control strategies so here we present a simple comparison of the robustness of the closed-loops arising from GPC and EIHGPC on examples 1 and 2. Intuititively one might expect EIHGPC to be more robust than GPC as it uses smoother input predictions, however conversely one might argue that this is tantamount to employing a slower control sample period which would reduce stability margins. Analytical work on the expected robustness of the nominal GPC controller or indeed the achievable robustness through the use of a T-filter [11-12] or a Q polynomial [13] is non-simple. Hence it is difficult to judge in practice whether any robustness improvements might be gained through the use of EIHGPC. So here we satisfy ourselves with a simple numerical comparison of the nominal robustnesses, that is without Q or T filters.

The robustness of a closed-loop with a predictive controller $K(z) = N_k(z)/D_k(z)\Delta(z)$ to additive model uncertainty is measured by

$$S = \frac{a(z)N_k(z)}{a(z)\Delta(z)D_k(z) + z^{-1}b(z)N_k(z)} \qquad (17)$$

The smaller $|S|$ is, the more robust the closed-loop is. Bode gain plots of $|S|$ are ploted for $0 < \omega T < \pi$ where $z = e^{-j\omega T}$ in Figures 7a,b for examples 1,2 respectively. (It is noted that n_y^a has no affect on the loop robustness properties.)

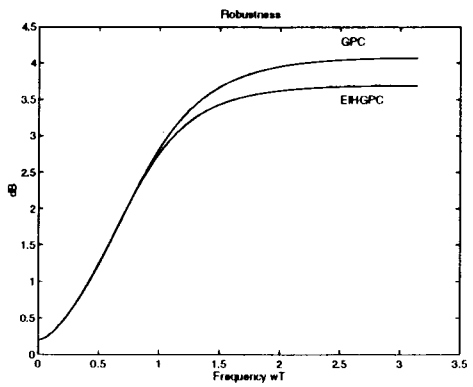

Figure 7a. Robustness of example 1

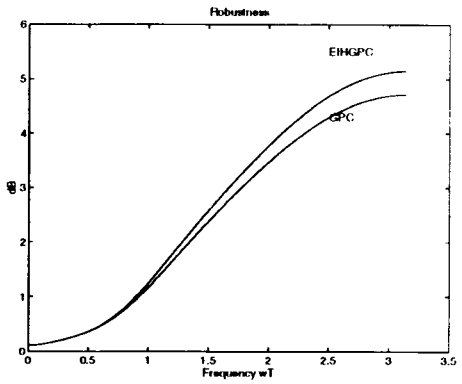

Figure 7b. Robustness of example 2

It is observed that EIHGPC is more robust than GPC for example 1 and the reverse is true for example 2 though in each case the differences are small. This illustrates that the robustness of the two controllers is often very similar and for any particular example which control strategy is more robust depends upon the selection of control parameters.

3.5 Conclusions

GPC is expected to take optimal account of future set-point changes and hence give good tracking. However it has been illustrated that this expectation is often not borne out in practice and in fact including advance knowledge of the set point can often lead to worse tracking than if much of that information was ignored altogether, e.g. compare figures 4a,b,c and Table 3. Reducing the amount of advance knowledge available has actually improved the tracking achieved by the GPC algorithm. It is noted however that using no advance knowledge at all gives poorer performance, (compare Table 5 with Tables 3,4). Hence it is observed that for GPC performance is optimised by choosing $1 \leq n_y^a \leq n_y$, ($n_y^a \approx 3$ for the examples in this a paper) but it is not clear in general, except by trial and error, what value n_y^a for advance knowledge will give the best performance for any particular example. A remedy to the poor use of advance knowledge of set-point changes by GPC is proposed in the form of a new algorithm, EIHGPC. Here the degrees of freedom in the GPC algorithm, that is the vector of predicted future control increments is altered to spread the predicted control changes over a wider part of the output horizon without increasing the numbers of degrees of freedom. EIHGPC is seen to have much improved tracking performance without increasing input activity and to be relatively insensitive to n_a^y. The EIHGPC controller appears to have similar robustness to GPC and similar nominal performance if $n_y^a = 1$ so that no penalty is paid for this improvement.

This paper has therefore illustrated a weakness of

GPC with regard to its use of advance knowledge of the set point and proposed a simple solution. However, a more analytical approach is now required to find an algorithm which makes 'best' use of advance knowledge. Moreover it is necessary to look more closely at the implications of the presence of hard input constraints on this issue.

References

[1] D.W .Clarke (Editor) *Advances in Model based predictive control*, Oxford Science Publications. Conference on Model based Predictive Control, Oxford 1993

[2] D.W.Clarke, C.Mohtahdi and P.S.Tuffs *Generalized predictive control, Parts 1 and 2* Automatica, Vol.23, pp137-160, 1987

[3] C.R.Cutler and B.C. Ramaker *Dynamic marix control - A computer control algorithm* paper WP5B, JACC, San Fransico, 1980

[4] R. Rouhani and R.K. Mehra *Model Algorithmic Control(MAC); Basic theoretical properties* Automatica, 18, 4, pp401-414, 1982

[5] V. Perterka *Predictor based self-tuning control* Automatica, 20, pp39-50, 1984

[6] D.W. Clarke and P.J. Gawthrop *Self tuning controller*, Proceedings IEEE, Pt.D, 122, pp929-934, 1975

[7] J.A. Rossiter, B.Kouvaritakis and R.M. Dunnett *Application of generalised Predictive control to a boiler-turbine unit for electricty generation*, Proceedings IEE Pt.D,138, 1, pp59-67 , 1989

[8] T.T.C. Tsang and D.W. Clarke *Generalized predictive control with input constraints*, Proceedings IEE, Pt.D, 135, pp451-460, 1988

[9] J.A.Rossiter and B.Kouvaritakis *Constrained Stable Generalized predictive control* Proc. IEE, Vol.139, No.4, pp349-262, 1993

[10] M. Sznaier and M.J. Damborg *Heuristically enhanced feedabck control of constrained discrete linear systems* Automatica, 26, 3, pp521-532, 1990

[11] B.D. Robinson and D.W. Clarke *Robustness effects of a prefilter in generalised predictive control*, Proceedings IEE Pt.D, 138,1,pp2-8, 1991

[12] T-W. Yoon and D.W. Clarke *Prefiltering in receding horizon predictive control* , Internal Report OUEL 1995/93, Department Engineering Science Oxford University

[13] B.Kouvaritakis, J.A. Rossiter and A.O.T.Chang *Stable generalised predictive control: An algorithm with guaranteed stability*, Proc. IEE Pt.D, 139,4, pp349-362 , 1992