Cautious stable predictive control: a guaranteed stable predictive control algorithm with low input activity and good robustness

J.R. Gossner B.Kouvaritakis
Department of Engineering Sciences, Parks Road, Oxford. OX1 3PJ
email: basil.kouvaritakis@eng.ox.ac.uk

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

Keywords: Predictive control, guaranteed stability, constraints, robustness

Abstract

Current predictive control strategies with guaranteed stability can be very highly tuned due to the use of a sufficient but not necessary end-point constraint in the formulation of the system input/output predictions. Hence they are likely to demand overactive input trajectories and can also have poor robustness properties. Here a necessary end-point constraint is developed for stable predictions and then used to define a predictive control algorithm with guaranteed stability. The use of this end-point constraint results in a control law with less active input signals and therefore it is less likely to cause difficulties in the presence of system constraints. As the controller is less highly tuned it is also expected to be more robust.

1 Introduction

Generalised predictive control (GPC) [1] has been used as a basis for predictive controllers with guaranteed stability, e.g. stabilising I/O receding horizon control (SIORHC) [2], constrained receding horizon control (CRHPC) [3] and stable generalised predictive control (SGPC) [4]. As it transpires all three approaches [2-4] give identical control laws [5] (hence they will all be referred to simply as SC). All three approaches employ the same end-point constraint, that is the system output predictions are forced to assume a fixed value equal to the set point beyond the output horizon. However, this end-point constraint is tantamount to requiring an explicit cancellation of all the system poles from the output predictions and therefore can also require highly active predicted inputs. SC controllers have three undesireable properties: (i) they can give highly tuned and hence overactive responses ; (ii) they may encounter difficulties meeting hard input/output constraints and hence give rise to instability; (iii) they often have poor robustness properties.

The objective of designing a receding horizon predictive control algorithm with guaranteed stability however can be achieved without the stringent requirement of SC [2-4]. The proof of the stability of SC algorithms can be established by showing that the appropriate performance index is a stable Lyapunov function. It will be shown in this paper that a simple modification of the standard GPC performance index slightly and by putting some suitable constraints on the system input/output predictions, it is possible to specify GPC performance indices which are stable Lypanuov functions and hence give rise to stable control laws. More importantly, the end-point constraints on the system predictions need not be nearly as exacting as those employed in SC; SC conditions are sufficient for stability, but not necessary. Here we derive necessary conditions. This means that the resulting control laws are less highly tuned and this will have implications when dealing with hard input constraints (e.g. actuator rate and absolute limits); a detuned algorithm is less likely to drive a system hard and therefore input constraints will be easier to deal with. As a by product the control laws also have improved robustness properties.

In this paper we give in section 2 a brief description of SC (we use SGPC as it has good numeric properties and is computationally efficient [5] and provides the natural framework for the new work presented in this paper). Using the SGPC framework, in section 3 we introduce further predictive control algo-

rithms with guaranteed stability: mean level (ML), cautious mean-level (CaML) and cautious stable predictive control (CaSC). These algorithms have progressively less stringent end-point constraints with none as stringent as that used in SGPC and the last having only that which is necesssary for stability, hence we expect the algorithms to use progressively less active input signals and hence to behave progressively better in the presence of constraints. Similarly the algorithms become progressively more robust. A comparison of the performance with and without constraints and the robustness of the resulting control laws is given in section 4.

2 Background to SC

With z the Z-transform variable and z^{-1} the delay operator, the system model is taken to be

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

$a(z) = 1 + ... + a_n z^{-n}$, $b(z) = b_0 + ... + b_{n-1}z^{-n+1}$. The difference equation $a(z)\Delta(z)y(z) = z^{-1}b(z)(\Delta u(z))$; $A(z) = a(z)\Delta(z)$, $\Delta = 1 - z^{-1}$, is simulated forward in time n_y samples with n_u changes in control to give the prediction equation

$$\underset{\rightarrow}{y} = C_A^{-1}[\Gamma_b \Delta \underset{\rightarrow}{u} + H_b \Delta \underset{\leftarrow}{u} - H_A \underset{\leftarrow}{y}] \tag{2}$$

where the vectors are defined as

$$\underset{\rightarrow}{y} = \begin{bmatrix} y_{t+1} \\ \vdots \\ y_{t+n_y} \end{bmatrix}; \quad \Delta\underset{\rightarrow}{u} = \begin{bmatrix} \Delta u_t \\ \vdots \\ \Delta u_{t+n_u-1} \end{bmatrix}$$
$$\underset{\leftarrow}{y} = \begin{bmatrix} y_t \\ \vdots \\ y_{t-n} \end{bmatrix}; \quad \Delta\underset{\leftarrow}{u} = \begin{bmatrix} \Delta u_{t-1} \\ \vdots \\ \Delta u_{t-n+1} \end{bmatrix} \tag{3}$$

C_m, H_m are $(n_y \times n_y)$ Toeplitz and $(n_y \times q)$ Hankel matrices respectively where if $m(z) = m_o + ... + m_q z^{-q}$ then C_m is defined as having its i,j element equal to m_{i-j}. H_m has its i,j element given as m_{i+j-1}. Γ_m is the first n_u columns of C_m and M_m is defined from $C_m = [\Gamma_m, M_m]$.

2.1 Stable Generalised Predictive Control [4]

In GPC, all control changes Δu_{t+i} beyond a horizon n_u are set to zero ($\Delta u_{t+i} = 0, i \geq n_u$) and the the aim is to minimise over $\Delta u_{t+i}, 0 \leq i < n_u$ the aggregate of the sum of the squares of the next n_y predicted output errors and the sum of the squares of the next n_u future control changes. This is summarised as minimise J_{GPC} over $\Delta\underset{\rightarrow}{u}$

$$J_{GPC} = \|\underset{\rightarrow}{r} - \underset{\rightarrow}{y}\|_2^2 + \lambda\|\Delta\underset{\rightarrow}{u}\|_2^2 \tag{4}$$

In SGPC the future control moves are further constrained so that the predicted output reaches a fixed value after n_y samples, i.e.

$$\min_{\Delta\underset{\rightarrow}{u}} \; J_{GPC} \tag{5}$$

subject to

$$y_{t+i} = r_{t+n_y} \; (i > n_y) \text{ and } \Delta u_{t+i} = 0, \; (i \geq n_u) \tag{6}$$

Theorem 2.1 *SGPC gives rise to a stable control law.*

Proof: It can be shown that when r is constant there exists a control trajectory $\Delta\underset{\rightarrow}{u}$ at sample $t+1$ such that

$$J_{GPC_{t+1}} = J_{GPC_t} - (r_{t+1} - y_{t+1})^2 - \lambda(\Delta u_t)^2 = J_o \tag{7}$$

This is because at time t it is assumed that $r_{t+i} - y_{t+i} = 0, i > n_y$ and $\Delta u_{t+i-1} = 0, i > n_u$. Furthermore as J_{SGPC} is minimised at each sampling instant, $J_{GPC_{t+1}} \leq J_o$ and therefore J_{GPC} is monotonically decreasing (non-increasing) with time until it reaches zero. □

One way of realising constraint (6) is via a closed-loop as in Fig.1 where $X(z), Y(z)$ satisfy the bezout identity

$$a(z)\Delta(z)Y(z) + z^{-1}b(z)X(z) = 1 \tag{8}$$

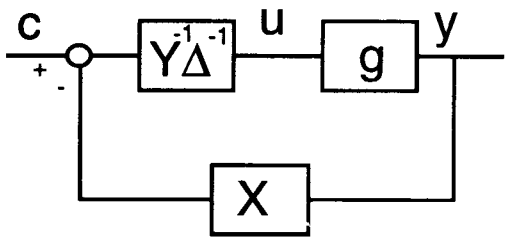

Figure 1

If the input to the loop is $c(z)$ the input/outputs are given by

$$y(z) = z^{-1}b(z)c(z); \qquad \Delta u(z) = A(z)c(z) \tag{9}$$

The end-point constraint (6) is enforced by choosing the predicted future $c(z)$ according to

$$c_{t+i} = c_\infty, \quad i \geq n_c \quad \text{with} \quad c_\infty = \frac{r_{t+n_y}}{b(1)} \tag{10}$$

where $n_y = n_c + n - 1$ and $n_u = n_c + n + 1$.

The system input/output predictions are then

$$\begin{aligned} \underset{\rightarrow}{y} &= \Gamma_b \underset{\rightarrow}{c} + M_b \mathbf{1} c_\infty + H_b \underset{\leftarrow}{c}; \\ \Delta\underset{\rightarrow}{u} &= \Gamma_A \underset{\rightarrow}{c} + M_A \mathbf{1} c_\infty + H_A \underset{\leftarrow}{c} \end{aligned} \tag{11}$$

with Γ_A, Γ_b $(n_y \times n_c)$ matrices and

$$\underrightarrow{c} = \begin{bmatrix} c_t \\ c_{t+1} \\ \vdots \\ c_{t+n_c-1} \end{bmatrix}; \quad \mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{12}$$

$\underrightarrow{c}$ now constitutes the degrees of freedom that can be used to minimise J_{GPC}.

The control law is derived by substituting the system predictions (11-12) into the performance index (4) and solving for the optimum $\underrightarrow{c}$. Of the optimum $\underrightarrow{c}$ only the first element is used and then the whole $\underrightarrow{c}$ vector is recomputed at the next sampling instant. Hence the control law implied is

$$\begin{aligned} c_t &= -S\underleftarrow{c} + P_r\underrightarrow{r} \\ P &= e^T(\Gamma_b^T\Gamma_b + \lambda\Gamma_A^T\Gamma_A)^{-1} \\ S &= P[\Gamma_b^T H_b + \lambda\Gamma_A^T H_A] \\ P_r &= P\Gamma_b^T - [O, P(\Gamma_b^T M_b\mathbf{1} + \lambda\Gamma_A^T M_A\mathbf{1})/b(1)] \end{aligned} \tag{13}$$

where e is the first standard basis vector and O is an $n_y - 1$ vector of zeros. Further if $S = [S_o, S_1, ...]$ and $P_r = [P_{r1}, P_{r2}, ..., P_{rn_y}]$ define

$$\begin{aligned} S(z) &= S_o + S_1 z^{-1} + S_2 z^{-2} + ... \\ P_r(z) &= P_{r1} z + P_{r2} z^2 + ... + P_{rn_y} z^{n_y} \end{aligned} \tag{14}$$

Theorem 2.2 *The SGPC control law of equations (13) can be implemented by the controller* $K(z) = N_k(z)/(D_k(z)\Delta(z))$ *with the configuration of Figure 2. where* $N_k(z), D_k(z)$ *are defined from*

$$a(z)\Delta(z)D_k(z) + z^{-1}b(z)N_k(z) = 1 + z^{-1}S(z) = P_c(z) \tag{15}$$

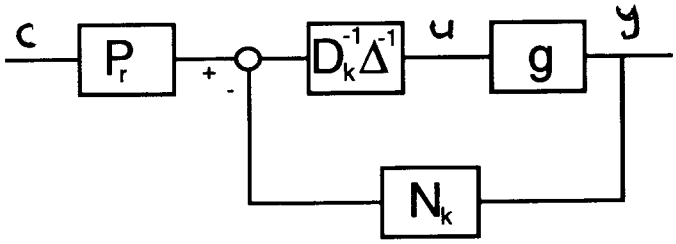

Figure 2.

Proof: The SGPC control law is summarised by eqns. (9) in conjunction with eqn. (13) which gives

$$c(z) = \frac{P_r(z)}{P_c(z)}r(z) \Rightarrow \begin{cases} y(z) &= \frac{z^{-1}b(z)P_r(z)}{P_c(z)}r(z) \\ \Delta u(z) &= \frac{A(z)P_r(z)}{P_c(z)}r(z) \end{cases} \tag{16}$$

However eqns. (16) also arise from the controller of eqn. (15) in the configuration of Figure 2. (A more comprehensive proof of the equivalence is in [4]). □

Remark 2.1 It was shown in [4] that the implementation of Theorem (2.2) has advantages over the implementation of eqns. (13) in conjunction with the configuration of Figure 1 and therefore is to be preferred. It is further noted therefore that the $X(z), Y(z)$ of eq. (8) and the loop of Fig. 1 are never actually used and need not be computed.

Remark 2.2 It is noted that $P_r(z)$ is anticausal as it operates on future set-point changes.

The SGPC control law is summarised in algorithm form in section 3.5.

3 Stable predictive control laws from necessary or sufficient conditions

3.1 Class of stable predictions

Stable predictive control laws work by first selecting a set of stable input/output predictions and using these predictions in a suitable performance index. However, the predictions used by SGPC eqn. (11-12) in fact define only a subset of the whole class of stable predictions. This is so because of the requirement (eqn. 6) that the predicted output reach and remain at the set point within n_y samples and the control increments used to bring this about are themselves zero beyond n_u sampling instants. The widest class of stable predictions comprises weighting sequences with stable poles, not all of which lie at the origin.

To define the whole class of stable input/output predictions we use the notation

$$m(z) = m^+(z)m^-(z) \tag{17}$$

where the roots of $m^+(z)$ are outside or on the unit circle and the roots of $m^-(z)$ are strictly inside the unit circle. Hence $m^+(z)$ contains the unstable roots of $m(z)$ and $m^-(z)$ contains the stable roots of $m(z)$. Define n_{m^+}, n_{m^-} to be the orders of $m^+(z), m^-(z)$ respectively.

Theorem 3.1 *The entire set of stable input/output prediction pairs for the plant* $z^{-1}b(z)/a(z)$ *is given by*

$$u(z) = \frac{a^+(z)}{b^-(z)}c(z) \quad y(z) = \frac{z^{-1}b^+(z)}{a^-(z)}c(z) \quad c(z) \in RH_\infty \tag{18}$$

Proof: Consider the plant equation

$$y(z) = \frac{z^{-1}b(z)}{a(z)}u(z) \quad \Rightarrow \quad u(z) = \frac{a(z)}{z^{-1}b(z)}y(z) \tag{19}$$

If $y(z)$ is to be stable then it is clear that the input $u(z)$ must both be stable and cancel any unstable poles, i.e.

$$u(z) = a^+(z)f(z) \quad f(z) \in RH_\infty \tag{20}$$

However, alternatively it can be argued from eqn. (19b) that given that $y(z)$ is stable, $u(z)$ is stable iff $y(z)$ cancels any unstable zeros, i.e.

$$y(z) = b^+(z)e(z) \quad e(z) \in RH_\infty \tag{21}$$

For a stable input/output pair both constraints (20) and (21) must hold simultaneously which implies that

$$u(z) = a^+(z)f(z) \quad \text{and} \quad u(z) = \frac{a(z)}{b^-(z)}e(z) \tag{22}$$

and hence

$$b^-(z)f(z) = a^-(z)e(z) = c(z), \quad c(z) \in RH_\infty \tag{23}$$

Substituting this back into (20) and (21) gives eqn. (18). □

Remark 3.1 It is noted that eqn. (18) can be realised by the configuration of Figure 1 with $X(z), Y(z)$ defined from

$$a(z)\Delta(z)Y(z) + z^{-1}b(z)X(z) = a^-(z)b^-(z) \tag{24}$$

rather than the bezout identity of eqn. (8)

3.2 Cautious SC

A sensible predictive control law should employ necessary conditions for stable predictions rather than conditions which are only sufficient because this will release as many degres of freedom as possible for dealing with system constraints and optimising perfomance. The entire class of stable predictions is defined through eqn. (18). However, because the input/output predictions arising from such a class are infinite impulse responses, the stability results of Theorem 2.1 are not readily applied. Stability in SC is guaranteed because the predicted errors are zero beyond n_y samples; this implies that finite horizons are equivalent to infinite horizons and hence the performance index is a stable Lyapunov function as with the usual LQG. Such a condition can only be reintroduced with the predictions of eqn. (18) if the performance index is altered from that used in standard GPC (eqn. 4); this is discussed below.

Theorem 3.2 *The performance index*

$$J_{CaSC} = \|\underset{\rightarrow}{\tilde{r}} - \underset{\rightarrow}{\tilde{y}}\|_2^2 + \lambda\|\Delta\underset{\rightarrow}{\hat{u}}\|_2^2 \tag{25}$$

where $\underset{\rightarrow}{\tilde{r}}, \underset{\rightarrow}{\tilde{y}}$ *are the vectors of future values of* $\tilde{r}(z) = a^-(z)r(z), \tilde{y}(z) = a_-(z)y(z)$ *and* $\Delta\underset{\rightarrow}{\hat{u}}$ *is the vector of future values of* $\Delta(z)\hat{u}(z) = b^-\Delta(z)u(z)$ *and the predictions for* $y(z)$, $\Delta(z)u(z)$ *are taken from eqn. (18) with*

$$\underset{\rightarrow}{c}(z) = c_t + \ldots + c_{t+n_c-1}z^{1-n_c} + \frac{r_{t+n_y}a^-(1)z^{-n_c}}{b^+(1)\Delta(z)} \tag{26}$$

($n_y = n_{b^+} + n_c, n_u = n_{a^+} + n_c + 1$*) results in a stabilising control law.*

Proof: Substituting in for $y(z), u(z)$ from eqn. (18) it is clear that

$$\tilde{y}(z) = z^{-1}b^+(z)c(z) \quad \text{and} \quad \Delta\hat{u}(z) = A^+(z)c(z) \tag{27}$$

Then substituting in for $\underset{\rightarrow}{c}(z)$ from eqn. (26) gives

$$\tilde{r}_{t+i} - \tilde{y}_{t+i} = 0, (i > n_y) \; \Delta\hat{u}_{t+i-1} = 0, (i > n_u) \tag{28}$$

Clearly then, eqn. (28) is equivalent to conditions (6) and hence by the same arguments as those in Theorem (2.1), J_{CaSC} will be monotonically decreasing. Therefore the resulting control law will be stabilising. □

3.3 Mean-Level SC

CaSC uses a necessary condition for stable predictions and hence will have more degrees of freedom available for handling constraints and performance than SC. Recently [6], it was also demonstrated that a mean-level type of result could be applied even to unstable plant by expanding the class of stable predictions. Here we show that the mean-level (ML) algorithm uses a sufficient but not necesary condition for stable predictions and so, although it uses a wider class of predictions than SC, it uses a smaller class than defined in eqn. (18).

The mean-level algorithm [6] considers predictions where Lim $y_{t+i} = r, i \rightarrow \infty$, but restricts itself to a finite $\Delta u(z)$, i.e it assumes

$$\Delta u_{t+i-1} = 0 \quad i > n_u \tag{29}$$

In general we expect CaSC to be better than ML, as it employs more degrees of freedom, however ML control allows explicit input constraint handling because the future input trajectory, $\Delta\underset{\rightarrow}{u}$, is finite and therefore can be computed explicitly at each sample instant. CaSC on the other hand computes the optimum $\Delta\underset{\rightarrow}{\hat{u}}$ and $\Delta\underset{\rightarrow}{u}(z) = \Delta\underset{\rightarrow}{\hat{u}}(z)/b^-(z)$ and so explicit input constraint handling requires a horizon longer than n_u to allow for the poles implicit in $1/b^-(z)$. In practice however CaSC is less likely to drive the system against constraints and so this apparent advantage of ML often does not materialise. Moreover, both algorithms are equivalent in the handling of output constraints.

The prediction equations employed by a mean-level control law are described next.

Lemma 3.1 *The class of input/output pairs of stable predictions with infinite impulse response output predictions and finite inpulse response input predictions are given as:*

$$y(z) = \frac{z^{-1}b(z)}{a^-(z)}c(z);\ \Delta u(z) = A^+(z)c(z);\ c(z) \in RH_\infty \tag{30}$$

Proof: Using the conditions of eqn. (20) and using $y(z) = z^{-1}b(z)e(z)$ in place of eqn. (21) gives

$$a^+(z)f(z) = a(z)e(z) \tag{31}$$

Therefore $e(z) = f(z)/a^-(z)$, so selecting $c(z) = f(z)$ gives rise to eqns. (30) where $\Delta\underset{\rightarrow}{u}$ is a FIR if $c(z)$ is selected to have the same form as in eqn. (26). □

Theorem 3.3 *The mean-level control defined by the minimisation over the coefficients of* $\underset{\rightarrow}{c}(z)$ *of*

$$J_{ML} = \|\Delta\underset{\rightarrow}{u}\|_2^2 \tag{32}$$

where $\Delta\underset{\rightarrow}{u}$ *is defined in eqn. (30b) and*

$$\underset{\rightarrow}{c}(z) = c_t + ... + c_{t+n_c-1}z^{1-n_c} + \frac{ra^-(1)}{b(1)\Delta(z)}z^{-n_c} \tag{33}$$

($n_u = n_{a^+} + n_c + 1$) is stabilising.

Proof: The proof is analogous to that used in Theorem (2.1). □

3.4 Cautious mean-level SC

Mean-level SC allows for explicit input constraint handling, but does not place any weighting on transient output errors, which therefore could be very poor. Rather, it only requires the end-point constraint that Lim $y_{t+i} = r, i \to \infty$. These transient output errors can be incorporated by a slight modification to the standard GPC performance index similar to that used in CaSC; the resulting algorithm will be called CaML.

Theorem 3.4 *The performance index*

$$J_{CaML} = \|\underset{\rightarrow}{\tilde{r}} - \underset{\rightarrow}{\tilde{y}}\|_2^2 + \lambda\|\Delta\underset{\rightarrow}{u}\|_2^2 \tag{34}$$

gives rise to a stable control law if the predictions are those derived from eqn. (30), with $c(z)$ as in eqn. (33) and $\underset{\rightarrow}{\tilde{r}}, \underset{\rightarrow}{\tilde{y}}$ defined as in Theorem 3.2 and $n_y = n_b + n_c$, $n_u = n_{a^+} + n_c + 1$.

Proof: This parallels that of Theorem 3.2.

3.5 SC algorithms

For convenience the algorithms in this paper have been presented as arising from an optimisation over the future values of $c(z)$ (i.e. $\underset{\rightarrow}{c}(z)$) which is defined as the input to the configuration shown in Figure 1 but with the implied $X(z), Y(z)$ different for each controller discussed. In practice the system predictions would not be computed as given in eqn.(11), but by simulating the model forward in time. However, as mentioned in Remark 3.1, doing full closed-loop system predictions on the configuration of Figure 1 for SGPC, and using these predictions in the performance index results in a closed-loop controller of configuration 2. Conveniently this controller can be computed far more efficiently as discussed in Theorem 2.2. This methodolgy carries over to CaSC, ML, and CaML and so without proof (the reader is referred to [4] for full details), the efficient algorithms for computing the control laws presented in this paper are given next. In each case the controller $K(z) = N_k(z)/D_k(z)\Delta(z)$ and prefilter $P_r(z)$ are applied in the configuration of Figure 2.

Remark 3.2 It is noted that while GPC, SGPC and mean-level control give rise to a strictly anti-causal prefilter $P_r(z)$, CaSC and CaML give rise to a bi-causal prefilter. For convenience the causal part will be derived from a vector P_{rc} (which multiplies $\underset{\leftarrow}{r}$) and the anti-causal part will be derived from a vector P_{ra} (which multiplies $\underset{\rightarrow}{r}$). i.e.

$$\begin{aligned} P_r(z) &= P_{rc1} + P_{rc2}z^{-2} + ... + P_{rca^-}z^{-n_{a^-}} + \\ &\quad P_{ra1}z + P_{ra2}z^2 + ... + P_{ran_y}z^{n_y} \end{aligned} \tag{35}$$

Define O as an $n_y - 1$ row vector of zeros and $P_c(z) = P_{c1} + P_{c2}z^{-1} + ...$ where $P_c = [P_{c1}, P_{c2}, ...]$.

Algorithm 3.1 *The SGPC algorithm is defined through the diophantine equation*

$$a(z)\Delta(z)D_k(z) + z^{-1}b(z)N_k(z) = P_c(z) \tag{36}$$

where

$$\begin{aligned} P &= e^T(\Gamma_b^T\Gamma_b + \lambda\Gamma_A^T\Gamma_A)^{-1} \\ P_c &= [1, P[\Gamma_b^T H_b + \lambda\Gamma_A^T H_A)] \\ P_r &= P\Gamma_b^T - [O, P(\Gamma_b^T M_b\mathbf{l} + \lambda\Gamma_A^T M_A\mathbf{l})/b(1)] \end{aligned} \tag{37}$$

Algorithm 3.2 *The ML algorithm is defined by*

$$a(z)\Delta(z)D_k(z) + z^{-1}b(z)N_k(z) = a^-(z)P_c(z) \tag{38}$$

and

$$\begin{aligned} P &= e^T(\Gamma_{A^+}^T\Gamma_{A^+})^{-1} \\ P_c &= [1, P\Gamma_{A^+}^T H_{A^+}] \\ P_r &= -P\Gamma_{A^+}^T M_{A^+}\mathbf{l}a^-(1)/b(1) \end{aligned} \tag{39}$$

Algorithm 3.3 *The CaML algorithm is defined by*

$$a(z)\Delta(z)D_k(z) + z^{-1}b(z)N_k(z) = a^-(z)P_c(z) \quad (40)$$

and

$$\begin{aligned} P &= e^T(\Gamma_b^T\Gamma_b + \lambda\Gamma_{A+}^T\Gamma_{A+})^{-1} \\ P_c &= [1, P[\Gamma_b^T H_b + \lambda\Gamma_{A+}^T H_{A+})] \\ P_{rc} &= P\Gamma_b^T H_{a-} \\ P_{ra} &= P\Gamma_b^T - [O, P(\Gamma_b^T M_b \mathbf{1} + \lambda\Gamma_{A+}^T M_{A+}\mathbf{1})\tfrac{a^-(1)}{b(1)}] \end{aligned} \quad (41)$$

Algorithm 3.4 *CaSC is defined by*

$$a(z)\Delta(z)D_k(z) + z^{-1}b(z)N_k(z) = a^-(z)b^-(z)P_c(z) \quad (42)$$

and

$$\begin{aligned} P &= e^T(\Gamma_{b+}^T\Gamma_{b+} + \lambda\Gamma_{A+}^T\Gamma_{A+})^{-1} \\ P_c &= [1, P(\Gamma_{b+}^T H_{b+} + \lambda\Gamma_{A+}^T H_{A+})] \\ P_{rc} &= P\Gamma_b^T H_{a-} \\ P_{ra} &= P\Gamma_{b+}^T - [O, P(\Gamma_{b+}^T M_{b+}\mathbf{1} + \lambda\Gamma_{A+}^T M_{A+}\mathbf{1})\tfrac{a^-(1)}{b^-(1)}] \end{aligned} \quad (43)$$

Remark 3.3 In the definition of $m^+(z)m^-(z)$ (eqn. (17) it was assumed that $m^+(z)$ contained all the strictly unstable roots of $m(z)$ and $m^-(z)$ all the stable roots. However, it may be that it is better in some cases to include in $m^+(z)$ all the roots outside a circle of radius R, where $R < 1$. This is because the CaSC, CaML and ML controllers automatically include $a^-(z)$ in the closed-loop pole polynomial (see algorithms above); hence if one desired all the closed-loop poles to be at least as fast as R, then one would need to redefine $a^-(z)$ to exclude those roots with modulus greater than R, and hence include them in $a^+(z)$.

4 Simulation Comparisons

The motivation behind designing new stable predictive control algorithms was to use only necessary conditions for stable predictions. Such a set of predictions would be a wider class than those currently used by either SGPC or ML and hence would be less likely to use active inputs or to give over-active output responses. Here, two examples will be presented with the closed-loop responses to a step demand in $r(z)$ from each of the four algorithms, SGPC, ML, CaML, CaSC. As expected, it will be seen that SGPC drives the system very hard and hence uses highly active inputs. ML only costs the input activity and hence the transient errors may be poor, though the responses are slower. However, as it is still restricted to a FIR for $\Delta\underrightarrow{u}$, the inputs are still more active than they need to be. The same problem is shared by CaML, but as transient output errors are included, its performance is generally better though the difference can be small. Finally, CaSC demonstrates reasonable performance with low input activity.

In many real problems there are input constraints and failure to take these constraints into account can lead to instability, especially if the plant is open-loop unstable. CaSC because it has a less exacting end-point constraint than SGPC, CaML, and ML, is less likely to demand input signals exceeding the input constraints and hence will perform better and more safely in the presence of constraints. This is illustrated in example 1.

CaSC is less highly tuned than ML,CaML and SGPC and hence we would expect it to give a more robust controller. In this section we also compare the robustness of the four controllers given in this paper in Algorithms 3.1-3.4. A measure of robustness is given by

$$R = \frac{N_k(z)a(z)}{a(z)\Delta(z)D_k(z) + z^{-1}b(z)N_k(z)} \quad (44)$$

For a closed-loop with good robustness $|R|$ should be small.

Remark 4.1 For convenience it is assumed in the following simulations that the advance knowledge of the set-point is limited to P_r^n, ie. $r_{t+i} = r_{t+P_r^n}, \forall i > P_r^n$.

4.1 Example 1

The model is given by

$$\begin{aligned} a(z) &= 1 - 1.6z^{-1} + 0.13z^{-2} + 0.21z^{-3} \\ b(z) &= 1 - 2.7z^{-1} + 1.4z^{-2} \end{aligned} \quad (45)$$

and has an unstable pole at $z = 1.4$ and an unstable zero at $z = 2$. The control parameters are taken to be $n_c = 2, \lambda = 1, P_r^n = 3$ (n_y, n_u are determined automatically from n_c and the plant model). The output/input responses are plotted in Fig. 3a,b respectively and illustrate the expected characteristics. SGPC is the most highly tuned and has very active input and output responses. CaML and ML are very similar though as expected CaML is quicker than ML and slightly more active. CaSC is the least active and gives the best performance; in fact it settles as quickly as SGPC.

The robustness $|R|$ eqn. (44), for each controller, is plotted as a function of θ where $z = e^{j\theta}$ for $0 \leq \theta \leq \pi$. in Fig. 3c. The same ordering is illustrated with CaSC being the most robust while SGPC is the worst. ML is slightly more robust than CaML since CaML has a more stringent objective in

that its performance index includes a measure of the transient predicted errors.

Figure 4a,b,c are simulations on the same model, but now with input constraints

$$|u_t| < 1 \quad \text{and} \quad |\Delta u_t| < 0.3 \tag{46}$$

It is noted that only CaSC gives rise to a stable control law. The other algorithms try to use more rate (Fig. 4c) than is available and as a result fail. Typically a quadratic programming approach to solving a control optimisation with constraints might be used, however in this example ML, CaML and SGPC fail to find a feasible solution for $n_c < 6$ whereas CaSC succeeds with $n_c \geq 2$.

4.2 Example 2

For this example the model is

$$\begin{aligned} a(z) &= 1 - 1z^{-1} + 0.01z^{-2} + 0.12z^{-3} \\ b(z) &= 1 - 2.4z^{-1} + 0.8z^{-1} \end{aligned} \tag{47}$$

which has one unstable zero at $z = 2$ and no unstable poles - however the slowest pole is at $z = 0.8$. Selecting the control parameters $n_c = 2, \lambda = 1, P_r^n = 1$ gives the responses in Fig. 5a,b. Here it is noted that ML, CaML and CaSC are detuned so much that the output responses become unsatisfactory though the improvement in robustness over SGPC (Fig. 5c) is between 4 and 5 times.

The reason why the responses in Fig. 5a,b are so slow is the open-loop pole at $z = 0.8$ which is automatically included as a closed-loop pole, however as mentionned in remark 3.3 the definition of $m^+(z)$ and $m^-(z)$ (eqn. 17)) allows for $R \neq 1$, and so here it is prudent to choose $R = 0.75$ so that the pole at $z = 0.8$ was placed in $a^+(z)$. The resulting responses are given in Figs 6a,b. It is observed that the responses are now satisfactory and once again CaSC has caused a significant improvement in control activity. The non-minimum phase behaviour of CaSC (-0.25) is half that of SGPC (-0.55). Also the maximum amplitude control in CaSC (0.35) is much less than that in SGPC (0.55). CaML and ML fall between the two. The robustness plots Fig. 6c also show a significant improvement with SGPC upto twice as bad as CaSC.

4.3 Conclusions

It has been illustrated that it is possible to design predictive control laws with guaranteed stability which use necessary conditions for stable input/output predictions as opposed to the sufficient only conditions of earlier predictive controllers with guaranteed stability. This gives rise to control laws with less active input and output responses.

One key advantage of predictive control laws is their ability to deal systematically and optimally with system input constraints. However, the SC problem as specified can be infeasible, i.e. it may not be possible to satisfy constraint (6) and the input constraints simultaneously. In this paper only a necessary endpoint constraint has been used and hence the resulting predictive control problem is more likely to be feasible and therefore admit a solution. However the system predictions are now infinite impulse responses as opposed to FIR and thus longer constraint horizons will be required.

The CaSC controllers being less highly tuned than the SC variants in general give a more robust closed-loop. However, the work in this paper does not consider T-filters [1] or use of a $Q(z)$ polynomial [4] to improve robustness, though one would expect these to offer similar benefits to each control strategy. Q and T are essentially observer based, they do not affect the nominal tuning and therefore one would still expect a less highly tuned algorithm to be more robust. Furthermore, Q and T will have nothing to offer to the issues of constraint handling.

References

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

[2] E.Mosca and J.Zhang. *Stable redesign of predictive control* Automatica, Vol.28, No. 6, pp1229-1233, 1992

[3] D.W.Clarke and R.Scattolini. *Constrained receding horizon predictive control* Proc. IEE, Pt. D, Vol.138, No.4, pp347-354, 1992

[4] B.Kouvaritakis, J.A.Rossiter and A.O.T.Chang *Stable Generalized predictive control: an algorithm with guarenteed stability* Proc. IEE, Vol.139, No.4, pp349-262, 1992

[5] Rossiter, J.A. and Kouvaritakis, B. *Robustness and efficiency of generalized predictive control algorithms with guaranteed stability* Proceedings IEE Conference CONTROL 94, Warwick, pp1017-1022, 1994

[6] Rossiter, J.A. *GPC controllers with guaranteed tability and mean-level control of unstable plant*, 33rd CDC, Orlando, pp 3579-3580,1994

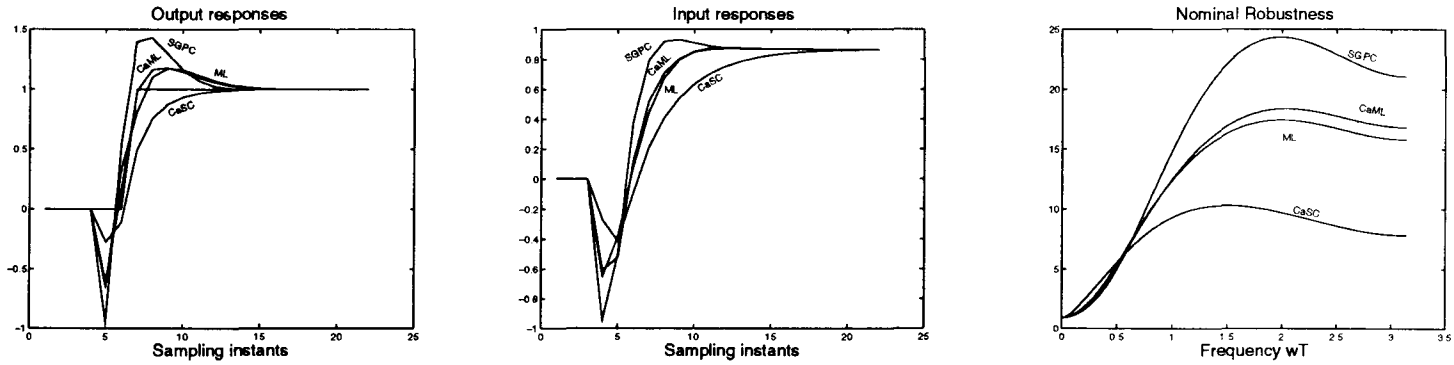

Figure 3a,b,c

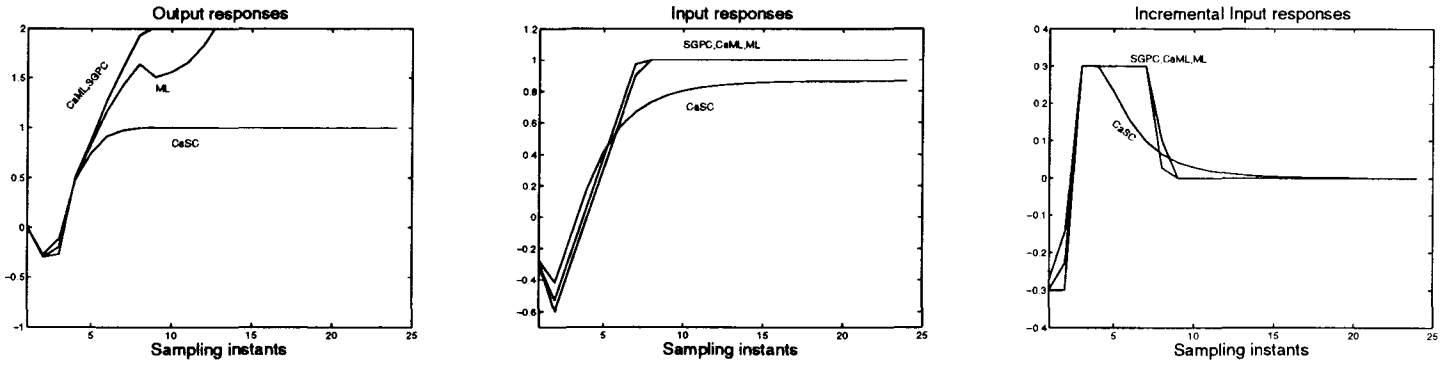

Figure 4a,b,c

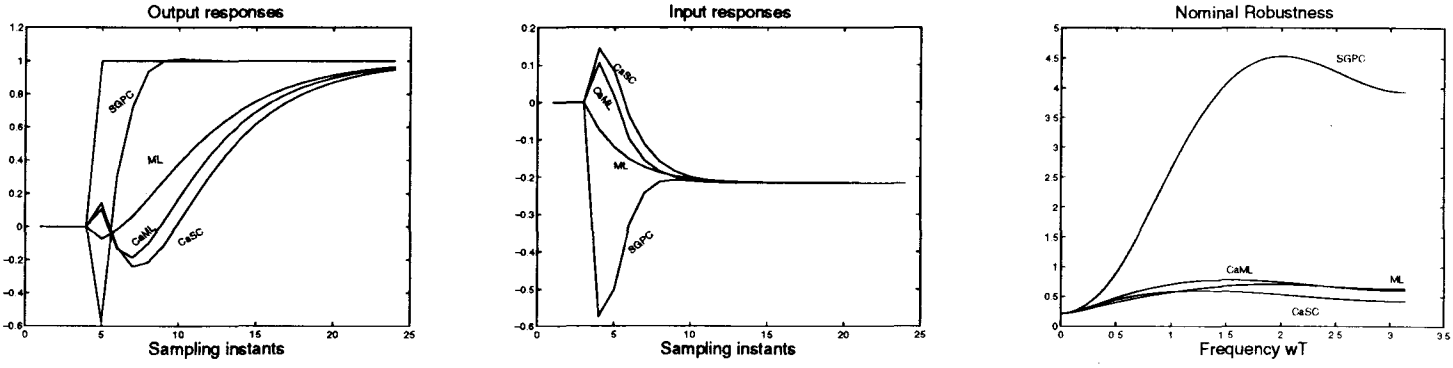

Figure 5a,b,c

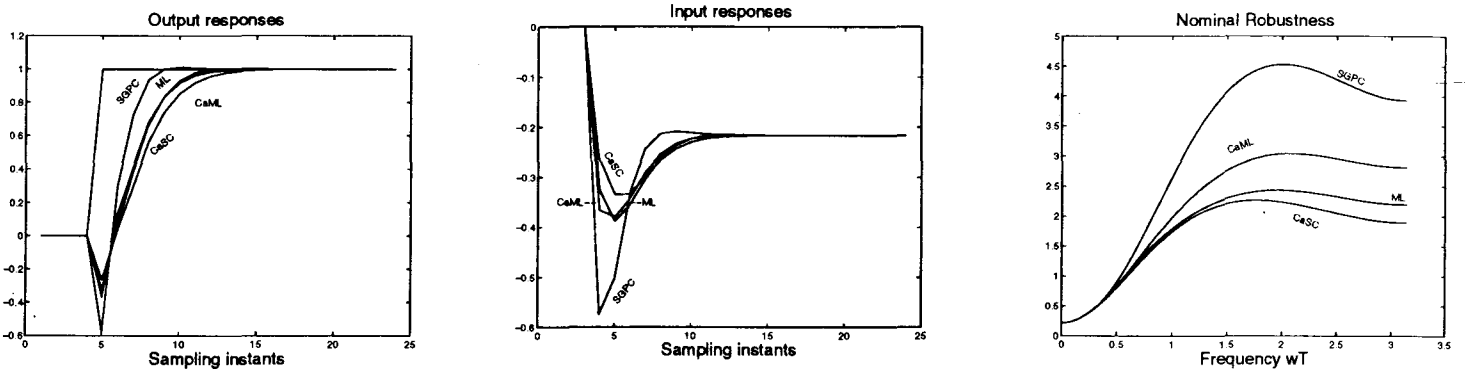

Figure 6a,b,c