Handling constraints with Predictive Functional Control of unstable processes.

by J A Rossiter and J Richalet

RESEARCH REPORT NO. 807

October 2001

ACC02-INV5202

Handling constraints with Predictive Functional Control of unstable processes

J.A. Rossiter
Dept. of Automatic Control & Systems Engineering
Sheffield University, Mappin Street
Sheffield, S1 3JD
email: J.A.Rossiter@sheffield.ac.uk
Tel. 44 114 2225685

J. Richalet
ADERSA
10 Rue de la Croix Martre
91873 PALAISEAU CEDEX
France
email: jacques.richalet@adersa.com

Abstract

This paper tackles 2 major issues. The first is to develop a new means of including constraint handling for unstable processes into the industrial successful and computationally efficient algorithm, predictive functional control (PFC); the benefits are illustrated. The second is to show how the technique to be proposed has recursive feasibility in the nominal case, that is feasibility now implies feasibility at the next sampling instant. This is known to be essential for robust stabilisation of unstable processes.

Keywords: Predictive functional control, computational efficiency, constraint handling

1 Introduction

Unstable processes provide a particularly hard challenge to the control engineer. Although for many processes, designing a control law to give nominal closed-loop stability is by now fairly straightforward, there are significant exceptions to this. For instance take the popular Model predictive control (MPC) strategy of GPC (Generalised predictive control, [2]). This has strong links to optimal control and hence for choices of input and control horizon that approach infinity, nominal closed-loop stability can be assured for any controllable and observable process. However, in practice [5] high values of control horizon are not used, partly because they imply significant online computation but also they can result in an overtuned control law which is less robust. If the horizons are restricted to be small, (say to reduce computational load) then closed-loop stability is not so straightforward, and indeed for many popular default selections of tuning GPC will fail to stabilise a plant with factors of the form $(s-a)/(s-ra)$, $r>1$. It is known that there are solutions to this problem (e.g. [8, 11, 16, 23]) however these algorithms involve endpoint constraints of some form which can imply the need for many degrees of freedom. Also, the online computation involves a quadratic program. The need for a demanding online computation limits the applicability of predictive control.

Much recent work (e.g. [24, 22, 10]) has considered how to produce an algorithm with a low on line comptuational burden, while still retaining benefits such as online optimisation and constraint handling. PFC (Predictive functional control [12]) algorithm is one such algorithm which achieves computational simplicity by using simpler but more intuitive design guidelines and has achieved widespread success in industrial applications. The success of its applications to unstable processes was however more variable and hence recent work [19, 20] has shown how, in the absence of constraints, PFC can be extended systematically to cater for the unstable case. The purpose of this paper is to develop, to validate and to illustrate the efficacy of some computationally efficient constraint handling tools for PFC.

One challenging complication with constrained unstable processes is how to ensure feasibility (e.g.[9, 14]). A MPC algorithm will be taken to be feasible if, with the degress of freedom specified, all constraints can be satisfied. In many commercial MPC packages this issue is well recognised and in the event of infeasibility there are embedded rules for relaxing or even removing so called *soft constraints* to regain feasibility. It is not our intent to consider such approaches as to some extent they are commercial rather than control decisions. There is a more important consideration with unstable processes, that is not only can constraints (soft and/or hard) be satisfied but more importantly can the process still be stabilised with allowable controls. In other words, if one ignored all soft constraints including the desirable set point, can the process be stabilised with the degrees of freedom available. In fact this problem can be solved backwards in a straightforward manner using admissible sets e.g. [6, 22, 24]. For a MPC algorithm deploying a terminal mode (e.g. [23]) and a some degrees of freedom, it is easy to compute the region within which the process will be stabilised. As long as the state is with that set, all is fine. If the state goes outside that set, the algorithm becomes undefined

and any behaviour could result. Hence our objective is to develop a PFC algorithm whose stabilisable region (or invariant set) is as large as possible in the nominal case.

In summary, recursive feasibility (maintaining the state inside the admissible set) implies invariance ([1, 6, 9]) and hence stability. For the nominal case one can ensure recursive feasibility if: (i) the predicted input trajectory is stabilising and meets constraints over the entire future and (ii) at the next sampling instant the same trajectory is reachable with the available degrees of freedom.

So then, what is the contribution of this paper ?

- Firstly, the PFC algorithm is based on independent models and hence the standard results in the literature do not transcribe in a straightforward fashion. Some work is required to show how such a transferal can be made and this paper provides a possible starting point.
- Secondly, by design PFC is far simpler than most MPC algorithms so one cannot necessarily use the same high powered mathematical machinery for online control selection and this in turn has repercussions on what can be said about recursive feasibility (or invariance). There is a need first to show how the PFC algorithm proposed in [20] can be reworked to include constraint handling and then secondly to modify this so as to ensure recursive feasibility.

This paper gives some background on PFC, independent models and prestabilisation, develops two constraint handling algorithms, shows how recursive feasibility can be established and illustrates the algorithm with examples.

2 Background

2.1 The PFC algorithm

In this paper we will adopt the notation of y, u, r for process outputs, inputs and setpoint respectively. z^{-1} is the unit delay operator such that $z^{-1}y_k = y_{k-1}$, y_k is the value of y at the kth sample and $y_{k+i|k}$ is the predicted value of y_{k+i} computed at sample k. PFC makes use of a system model to generate predictions of the process behaviour in terms of the current state and future inputs. The current input is selected by substitution of the predictions into a performance specification. The performance specification is defined by a desired closed-loop response in terms of a target first order lag.

Although more involved variants exist[1], in order to avoid over complicating this brief paper we concentrate on a PFC variant with just one degree of freedom. Hence in PFC one chooses: (i) the lag (that is the desired closed-loop time constant, say T_{PFC} and (ii) a single prediction horizon say T_h (denoted the coincidence horizon). The control move is then selected as the control which will cause the predicted plant output to coincide with the response of a target 1st order lag T_h seconds ahead. Let T_h seconds correspond to n_y samples (i.e $n_yT = T_h$, T the sample period), then the online computation reduces to solving:

$$\begin{aligned} \text{target} &= y_k + (r_{k+n_y} - y_k)(1 - e^{-\frac{T_h}{T_{PFC}}}) \\ y_{k+n_y|k} &= \text{target} \end{aligned} \tag{1}$$

Remark 2.1 *The implied computational burden is clearly trivial. In the unconstrained case this reduces to a fixed linear controller.*

2.2 Independent models

In PFC it is usual to use an IM (independent model) [4] for prediction. This can give significant improvements in sensitivity to measurement noise over the alternative of state realignment [18]. Also it is equivalent to a FIR model which is favoured in industry and hence this article will adopt an IM structure.

An independent model is intended to represent the process as closely as possible so that it has matching inputs and outputs. Let y_m be the output of the independent models (IM). The norm is to simulate the IM in parallel with the process, using the same inputs u. In general, due to uncertainty, $y \neq y_m$.

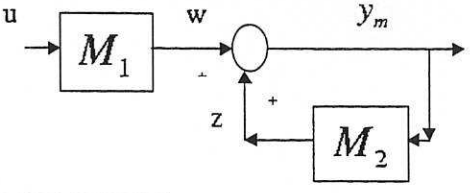
With unstable processes a parallel simulation cannot work because the same input will not stabilise the IM and an uncertain plant. A typical solution (e.g. [13]) is to decompose the model into two parts as in figures 1,2 where for a process modelled by G:

$$G = (I + M_2)^{-1}M_1 = \frac{n}{d} \tag{2}$$

where both M_1 and M_2 are stable. Figure 1 is used for prediction and figure 2 for online parallel simulation. A convenient decomposition in the SISO case is as follows:

$$G = \frac{n_+n_-}{d_+d_-}; \quad M_1 = \frac{n_+}{d_-}; \quad M_2 = \frac{b_2}{n_-}; \quad b_2 = n_- - d_+ \tag{3}$$

where n_+, d_+ are the factors containing unstable roots.



[1] Usually these are used to cater for setpoints with high order dynamics

Figure 1. Independent model used for prediction

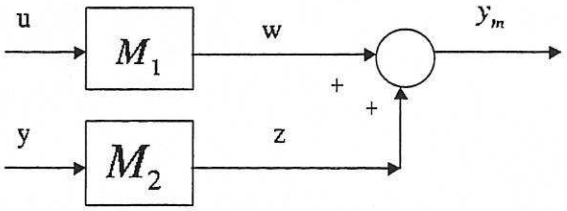


Figure 2. Independent model for simulation

2.3 Prestabilisation and prediction

Using unstable predictions as a basis for a predictive control law design is unwise [17]. Even if the behaviour is predicted to be good within the horizon, it would be divergent thereafter and hence one can not make recursive feasibility claims [9] and instability is almost inevitable due to constraints. There is a need therefore to parameterise the degrees of freedom in such a way that the predictions are stable. Here (see [19]) we use the basic philosophy of [11, 16], but without endpoint constraints. That is place structure into the predicted future control trajectory to bring the unstable dynamics under control.

2.3.1 Notation: Define vectors of future (arrow pointing right) and past values (arrow pointing left)

$$\underset{\rightarrow}{\Delta u} = \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+n_u-1} \end{bmatrix}; \quad \underset{\rightarrow}{y} = \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+n_y} \end{bmatrix}$$

$$\underset{\leftarrow}{\Delta u} = \begin{bmatrix} \Delta u_{k-1} \\ \Delta u_{k-2} \\ \vdots \end{bmatrix}; \quad \underset{\leftarrow}{y} = \begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \end{bmatrix}$$

The vector of future values can be any length, but n_y corresponds to the coincidence horizon and n_u the input horizon (often one in PFC). We will use the Toeplitz/Hankel notation to compute predictions. For a given polynomial $n(z) = n_0 + n_1 z^{-1} + \ldots$, define

$$C_n = \begin{bmatrix} n_0 & 0 & 0 & \ldots \\ n_1 & n_0 & 0 & \ldots \\ n_2 & n_1 & n_0 & \ldots \\ \vdots & \vdots & \vdots & \vdots \\ n_m & n_{m-1} & n_{m-2} & \vdots \end{bmatrix} \tag{4}$$

$$H_n = \begin{bmatrix} n_1 & \ldots & n_{m-1} & n_m \\ n_2 & \ldots & n_m & 0 \\ \vdots & \vdots & \vdots & \vdots \\ n_m & 0 & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Also define Γ_n as a tall and thin submatrix of C_n so that $[1, z^{-1}, z^{-2}...]\Gamma_n b = n(z)[1, z^{-1}, ...]b$ and note that dimensions are flexible to fit the context.

2.3.2 Nominal predictions: Set up consistency conditions around M_1 and M_2 (Figs. 1,2) at each future time instance and solve as simultaneous equations. Hence the predictions are given by ([19]):

$$\begin{array}{rcl} \underset{\rightarrow}{y} & = & \underset{\rightarrow m}{y} + L(\underset{\leftarrow}{y} - \underset{\leftarrow}{z} - \underset{\leftarrow}{w}) \\ \underset{\rightarrow m}{y} & = & C_{d_-}^{-1} C_\Delta^{-1} C_{d_+}^{-1} \{\Gamma_n \Delta \underset{\rightarrow}{u} + M_s v\} \end{array} \tag{5}$$

where

$$\begin{array}{rcl} M_s & = & [K_u, K_w, K_y, K_z]; \\ K_u & = & C_\Delta (C_n L + C_{n_-} H_{n_+}) \\ K_w & = & -C_\Delta (C_{n_-} H_{d_-} + C_{d_-} C_{b_2} L) \\ K_y & = & C_\Delta C_{d_-} (C_{b_2} L + H_{b_2}) \\ K_z & = & -C_\Delta C_{d_-} (C_{b_2} L + H_{n_-}) \end{array} \quad ; \quad v = \begin{bmatrix} \underset{\leftarrow}{u} \\ \underset{\leftarrow}{w} \\ \underset{\leftarrow}{y} \\ \underset{\leftarrow}{z} \end{bmatrix}$$

and L is a vector of ones and $\Delta = 1 - z^{-1}$.

2.3.3 Prestabilised predictions: It is straightforward to form an FIR[2] parameterisation of future inputs that stabilises the predictions of (5). Clearly stability implies the following constraint on $\Delta \underset{\rightarrow}{u}$:

$$[\Gamma_n, -\Gamma_{d_+}] \begin{bmatrix} \Delta \underset{\rightarrow}{u} \\ \gamma \end{bmatrix} = M_s v \tag{6}$$

Define K_1, K_2 such that this constraint is satisfied by:

$$\begin{bmatrix} \Delta \underset{\rightarrow}{u} \\ \gamma \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \end{bmatrix} v + \begin{bmatrix} \Gamma_{d_+} \\ 0 \end{bmatrix} \underset{\rightarrow}{c} \tag{7}$$

giving the corresponding output prediction as

$$\begin{array}{l} \underset{\rightarrow}{y} = H_1 \underset{\rightarrow}{c} + M_y v \\ H_1 = [C_\Delta C_{d_-}]^{-1} \Gamma_n \\ M_y \equiv [C_\Delta C_{d_-}]^{-1} K_2 + [0, -L, L, -L] \end{array} \tag{8}$$

2.3.4 The PFC algorithm with prestabilised predictions: Using the algorithm of eqn.(1) as a basis, the description of a PFC algorithm based on predictions (8) is elementary. Choose $\underset{\rightarrow}{c}$ to have just one variable c_k and define $\mathbf{e}_{n_y}^T$ to be the n_y^{th} standard basis vector. Then

$$y_{k+n_y|k} = \mathbf{e}_{n_y}^T [H_1 c_k + M_y v] \tag{9}$$

Solving for coincidence (1) gives

$$c_k = \frac{y_k + (r_{k+n_y} - y_k)(1 - e^{-\frac{T_h}{T_{PFC}}}) - \mathbf{e}_{n_y}^T M_y v}{\mathbf{e}_{n_y}^T H_1} \tag{10}$$

Substituting (10) into (7) gives the control increment.

$$\Delta u_k = \mathbf{e}_1^T [K_1 v + \Gamma_{d_+} c_k] \tag{11}$$

[2]Current work is looking at IIR paramterisations

3 Prestabilised PFC and constraint handling

3.1 An algorithm based on usual practice in PFC

The normal practice for constraint handling in many succesful industrial applications of PFC is to tune up two separate control strategies. One would be well tuned and the other detuned in such a way that it has low input activity and output overshoot is not expected. Then, one would predict the effect of using the well-tuned strategy in the closed-loop. If constraint violations are predicted, then one would resort to the detuned control law at that sampling instant. This clearly has analogies to [24, 22]. A proof of convergence, if required, could be tackled using invariant sets ([21]). Here we develop an algorithm which meshs such a policy with the prestabilised prediction equations.

Algorithm 3.1 Offline tasks: *Select control laws well tuned (S_1) and safe (S_2)*

Online tasks at each sampling point: *Find closed-loop predictions with S_1.*

1. *If violations predicted use S_2*
2. *If no violations predicted use S_1*

Remark 3.1 *It is noted that the success of this philosophy depends very much on whether a safe control law is easy to find. Nevertheless one can find the limit of applicability using admissible sets. One practical means of selecting the detuned control law is to use a slow, but not too slow, lag as the target.*

3.2 A novel constraint handling algorithm for PFC

The approach proposed developed next has more in common with the one degree of freedom algorithms (e.g. [10, 21]) as it allows a smoother movement between alternatives control laws rather than a simple switching [24]. Also it has analogies with reference governor approaches (e.g. [7, 15]) in that it has an implicit capacity to ignore/modify set point changes where this is judicious. This capacity will be seen to be essential and comes at no extra complication.

Consider the prediction equation of (8). This comprises of 2 parts: (i) the part depending on the current state and (ii) the totally free part. If one uses a minimal order solution to (6), then the part $\Delta \underset{\rightarrow}{u} = K_1 v$ has the minimum number of control moves within which the process can be stabilised, regardless of the setpoint. One should also note that $\Delta \underset{\rightarrow}{u} = K_1 v + \Gamma_{d+} \underset{\rightarrow}{c}$ is the whole class of solutions, hence all possible solutions can take $K_1 v$ as a base. The degrees of freedom in the solution are contained in $\Gamma_{d+} \underset{\rightarrow}{c}$; for PFC $\underset{\rightarrow}{c}$ is taken to have just one element although this is not a restriction to the prediction class.

Lemma 3.1 *Using the minimal order solution, that is selecting $\underset{\rightarrow}{c} = 0$ will cause the predictions to behave identically to those given at the previous sampling instant (in the nominal case).*

Proof: If there are n unstable poles, then the minimal order solution $K_1 v$ must have n terms whereas $\Delta \underset{\rightarrow}{u}$ has $n+1$ terms because Γ_{d+} has $n+1$ rows. At sampling instant $k+1$, the part of $\Delta \underset{\rightarrow}{u}_k$ yet to be implemented (the n terms remaining not including Δu_k, e.g. $\Delta \underset{\rightarrow}{u}_{k+1|k}$), has n terms and hence due to uniqueness must match the current minimal order solution $K_1 v$. □

Corollary 3.1 *One can only change predicted behaviour by using non-zero $\underset{\rightarrow}{c}$. Clearly there is a direct link between non-zero $\underset{\rightarrow}{c}$ and set point changes, that is a direct link to reference governing.*

Corollary 3.2 *All possible choices of target lag are subsumed in the variable $\underset{\rightarrow}{c}$. That is all solutions have the same base $K_1 v$ and the choice of target affects only how $\underset{\rightarrow}{c}$ is selected (e.g. see eqn.(10)). Hence, if $\underset{\rightarrow}{c}$ is left as a degree of freedom during constraint handling, then one can acheive identical control to a PFC algorithm based on any target lag !*

We can now propose a PFC algorithm for constraint handling.

Algorithm 3.2 Offline: *Set up the prediction equations (7,8).*

Online: *Compute $\underset{\rightarrow}{c}$ (e.g. 10) for the well tuned target lag and check for predicted violations.*

1. *If no violations, use well tuned control*
2. *If violations scale $\underset{\rightarrow}{c}$ as follows:*

$$\underset{\rightarrow}{c} \rightarrow \underset{\rightarrow}{c} - \gamma \underset{\rightarrow}{c}; \quad 0 \leq \gamma \leq 1 \qquad (12)$$

Substitute (12) into (7,8) and select the minimum γ that ensures no violations. Use resulting $\underset{\rightarrow}{c}$ in (11) to compute new control.

Remark 3.2 *The 2nd step of the algorithm above is a trivial computation that reduces to a set of inequality checks in one variable.*

Remark 3.3 *Whereas algorithm (3.1) allows only one value of γ, that associated to the choice of $\underset{\rightarrow}{c}$ with the detuned control, algorithm (3.2) allows far more flexibility in the choice of $\underset{\rightarrow}{c}$ and hence less detuning will be required in general, but also: (i) more detuning is available when needed and (ii) implicity set-point changes can be ignored if they cause infeasibility.*

Theorem 3.1 *In the nominal case the PFC algorithm of (3.2) is guaranteed to be feasible.*

Proof: This follows automatically from Lemma (3.1). As the solution of the previous sampling instant was feasible and is a subset of the solutions available now, therefore a feasible solution is available and can be selected with $\underset{\rightarrow}{c} = 0$. Clearly other choices of $\underset{\rightarrow}{c}$ may also be feasible and allow faster convergence. □

Note, as a corollary recursive feasibility implies no instability though a more formal proof of this is required and consitutes future work.

4 Examples

The following unstable process was used

$$G(z) = \frac{1 - 1.5z^{-1} + .36z^{-2}}{1 - 3.3z^{-1} + 3z^{-2} - 0.8z^{-3}} \qquad (13)$$

which has an unstable pole at 2 and an unstable zero at 1.2. The constraints are as follows

$$|\Delta u| \leq 0.5; \quad -1 \leq u \leq 1; \quad -0.3 \leq y \leq 2 \qquad (14)$$

The target discrete poles for PFC design are given as 0.6 for well tuned control and 0.8 for the detuned control. The coincidence horizon is 10.

For comparison (to demonstrate that constraints, denoted by dotted lines, are exceeded by a large amount), the unconstrained simulation is given in figure 3.

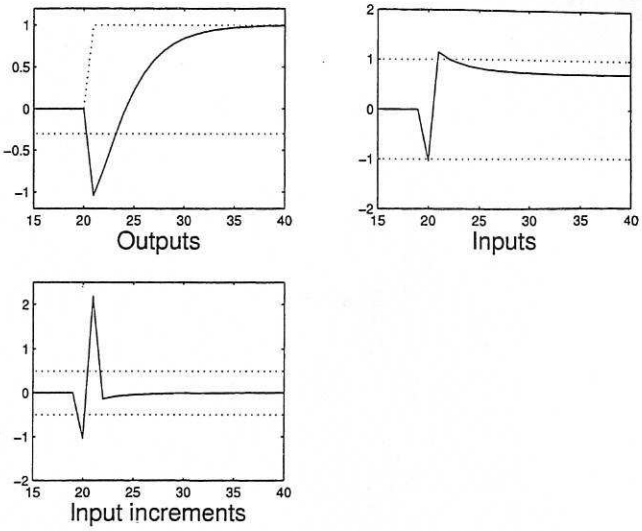


Figure 3. Simulation with no constraints

With constraints three simulations were performed:

- Simulation 1: Constraint handling by saturation (not displayed).
- Simulation 2: With algorithm (3.1) - dashed line.
- Simulation 3: With algorithm (3.2) - solid line.

The plots appear in figure 4 where the set point and the constraints are denoted by dotted lines and γ is denoted as *Gamma*. Also displayed on the same plot as γ is the control choice for simulation 2, '0' means the tuned (S_1) was selected and and '1' for the detuned (S_2).

It is clear from simulation 2 that simply selecting offline a fixed detuned law to avoid constraint violations has not worked here. In fact for all target poles from 0.9 to 0.95 a detuned law gave large violations. Moreover if saturation is enforced both simulations 1,2 gave unstable closed-loop responses. Therefore the approach of algorithm 3.1 is not straightforward in this case (although often successful in practice with stable plant).

Allowing $\underset{\rightarrow}{c}$ to be free (simulation 3) has given excellent responses and with no constraint violations. Also the values of scaling γ deployed are seen to be reasonable. One obvious conclusion is that for constraint handling to be effective for PFC, then some form of reference governing action must be allowed; this is implicit in algorithm (3.2) hence its success and moroever it comes at not extra computation. Putting reference governing into algorithm 3.1 would imply more computation and therefore is not as attractive.

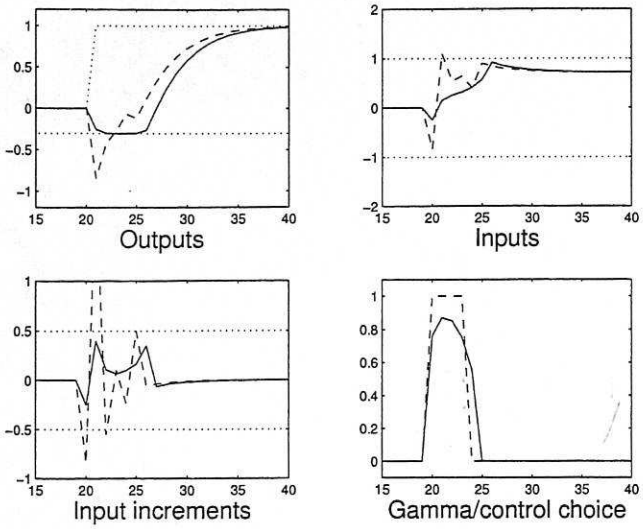


Figure 4. Simulations with constraints

5 Conclusion

This paper has shown how one means of extending the computationally efficient PFC algorithm proposed in [20] to take account of constraints. Moreover it has been shown that the algorithm has the important property of recursive feasibility so that stability now assures stability thereafter. Also it implicitly allows some reference governor action where this is beneficial. The efficacy of the algorithm is demonstrated by examples. Future work will look at extensions to cater for model uncertainty, disturbances and measurement noise and also alternative classes of stabilising predictions.

References

[1] Blachini, F. (1999), Set invariance in control, Automatica, 35, pp1747-1767

[2] Clarke, D.W., C. Mohtadi and P.S. Tuffs (1987). Generalised predictive control, Parts 1 and 2, Automatica, 23, pp. 137-160

[3] Cutler, C.R. and B.L. Ramaker (1980), Dynamic matrix control - a computer control algorithm, Proc. ACC, San Fransisco

[4] Garcia, C.E. and M. Morari (1982), Internal Model control 1. A unifying review and some new results, I&EC Proc. Design and Dev., 21, pp308-323

[5] Garcia, C.E. , D.M. Prett and M. Morari (1989), Model predictive control: theory and practice, a survey, Automatica, 25, p335-348

[6] Gilbert, E.G. and K. T. Tan (1991), Linear systems with state and control constriants: the theory and application of maximal output admissable sets, IEEE Trans AC, 36, 9, pp1008-1020

[7] Gilbert, E.G. and I. Kolmanovsky (1995), Discrete time reference governors for systems with state and control constraints and disturbance inputs, Proc. CDC, pp1189-1194

[8] Kouvaritakis, B., J.A. Rossiter and A.O.T.Chang (1992), Stable Generalized predictive control, Proc IEE, 139, pp349-362

[9] Kouvaritakis, B., J.R. Gossner and J.A. Rossiter (1996), Apriori stability condition for an arbitrary number of unstable poles, Automatica, 32, pp. 1441-1446

[10] Kouvaritakis, B., J.A. Rossiter and M. Cannon (1998), Linear quadratic feasible predictive control, Automatica, 34, 1583-1592

[11] Rawlings, J.B. and K.R. Muske (1993), The stability of constrained receding horizon control, Trans IEEE AC, 38, 1512-1516

[12] Richalet, J., A. Rault, J.L. Testud and J. Papon (1978), Model predictive heuristic control: applications to industrial processes, Automatica, 14, 413-428

[13] Richalet, J., Commande predictive, R 7 423

[14] Rossiter, J.A., B.Kouvaritakis and J.R. Gossner (1995), Feasibility and stability results for constrained stable generalised predictive control, Automatica, 31, 863-877

[15] Rossiter, J.A., J.R. Gossner and B.Kouvaritakis (1996), Guaranteeing feasibility in constrained stable generalised predictive control, Proc. IEE Pt.D, 143, 463-469

[16] Rossiter, J.A., J.R. Gossner and B. Kouvaritakis (1996), Infinite horizon stable predictive control Trans. IEEE AC, 41, 1522-1527.

[17] Rossiter,J.A., M.Rice and B.Kouvaritakis (1998). A numerically robust state-space approach to stable predictive control strategies, Automatica, 38, 65-73

[18] Rossiter, J.A. and J. Richalet (2001), Re-aligned models for prediction in MPC: a good thing or not ? APC6, York, UK

[19] Rossiter, J.A. (2001), Stable prediction for unstable independent models, submitted

[20] Rossiter, J.A. and J. Richalet (2001), Predictive functional control of unstable processes, submitted

[21] Rossiter, J.A., (2001), Extending the stability guarantees for an efficient predictive control law, ISSC, Maynooth, Ireland, 84-89

[22] Rossiter, J.A., and B.Kouvaritakis (2000), Computationally efficient algorithms for constraint handling with guaranteed stability and near optimality, to appear IJC

[23] Scokaert, P.O.M. and J. B. Rawlings (1996). Infinite horizon linear quadratic control with constraints, Proc. IFAC'96, vol. M, pp. 109-114

[24] Tan, K.T. and E.G. Gilbert (1992), Multimode controllers for linear discrete time systems with general state and control constraints, in Optim. techniques and applications (World Scientific Pub. Co.), 433-442