# Predictive functional control: to prestabilise or not?

by J A Rossiter

RESEARCH REPORT NO. 810

October 2001

ACC02-IEEE1032

# Predictive Functional Control: to prestabilise or not ?

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

**Abstract**

It is possible to prestabilise the predictions used within Predictive Functional Control in order to increase the likelihood of a stabilising control design for unstable open-loop plant. However, it is not clear that such a procedure will always give better closed-loop performance. This paper compares strategies with and without prestabilisation.

**Keywords:** Predictive functional control, stability, prestabilisation, performance

## 1 Introduction

Predictive control is an intuitive control design method whereby one predicts the expected effects of differing control trajectories and then selects the trajectory which causes the most desirable expected behaviour. Such a procedure fits well with human based control and can lead to easier design. For instance such questions as over what horizon should one predict behaviour, what sort of future control trajectories should one use, can all be answered fairly easily. In academia there has been a tendency to link these decisions to optimal control [1, 17] as this enables one to use well understood theoretical results. In particular apriori analysis of stability is straightforward (e.g. [4, 7, 12]). However, the link to optimal control takes the technique further away from engineering intuition which was key in early industrial variants of MPC (predictive control), e.g. [2, 8]. In this paper we focus on one commerical product [8] Predictive Functional Control (PFC) which has sought to relate controller design as much as possible to well understood engineering concepts. This simplification is at the expense of some potential optimality and power, nevertheless its extensive success in practice demonstrates that such issues are rarely a problem.

There is however some classes of problem for which PFC will often fail, that is unstable open-loop processes with factors of the type $(s-a)/(s-ra)$, $r>1$ and multivariable processes. Here we concentrate on the former of these. Some recent work [14, 15] has shown that it is possible to transcribe some of the work on guaranteed stability (e.g. [4, 11]) using prestabilisation before optimisation (other works tend to use post stabilisation). By prestabilising the predictions before using them in a PFC algorithm, one is able to stabilise processes that previously could not be stabilised with such a simple approach. This development can even be extended to constraint handling and feasibility issues [16].

The interesting work that remains is to investigate the use of prestabilisation over a wide class of unstable processes. It is known that PFC coped well with some unstable processes, hence one ask if it is better to stick with the original algorithm in those cases, rather than using prestabilisation. There is good reason to investigate this question because prestabilisation has links to dead-beat control. Moreover, early variants of MPC with guaranteed stability (e.g. [4, 6]) were known to be overtuned unless one used large control and output horizons. PFC uses a control horizon of only one and hence the prestabilised variant could well suffer from being overtuned (near dead-beat in character).

In this paper a brief summary of PFC algorithms based on prestabilised and non prestabilised predictions will be given. It will then be shown how to compute implied closed-loop poles (which is non-trivial due to the nature of the independent model). An examples section will then contrast closed-loop poles and closed-loop simulations for each algorithm with a variety of scenarios and processes. Finally some conclusions will be drawn.

## 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 substitu-

tion 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], 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 time constant, say $T_{PFC}$ and (ii) a single prediction horizon say $T_h$ (denoted the coincidence horizon). The control move is 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 $T_h$ correspond to $n_y$ samples (i.e $n_y T = T_h$, $T$ the sample period), then the online computation reduces to solving:

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

Although apparently simple, this strategy has achieved great success in practice e.g. [10].

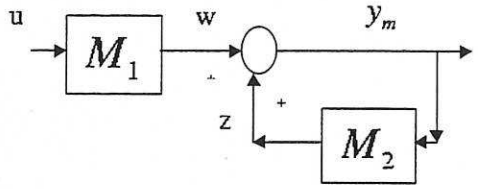


**Figure 1.** Independent model used for prediction

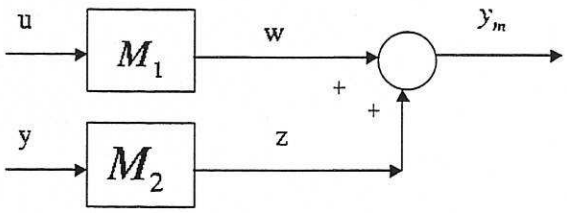


**Figure 2.** Independent model for simulation

### 2.2 Independent models

In PFC it is usual to use an IM (independent model) [3] for prediction. This can give significant improvements in sensitivity to measurement noise over the alternative of state realignment [13]. 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, $y \neq y_m$.

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

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. [9]) is to decompose the model into two parts as in figures 1,2 where for a process modelled by $G$:

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

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:

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

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

Note that for ease of notation any process dead-time is assumed to be absorbed into $n$ and $n_+$.

### 2.3 Prestabilisation and prediction

Using unstable predictions as a basis for a predictive control law design is generally unwise [12]. 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 [5] and instability is almost inevitable due to constraints (i.e. unnoticed constraint violations beyond the horizon). There is a need therefore to parameterise the degrees of freedom in such a way that the predictions are stable. Here (see [14]) we use the basic philosophy of [7, 11], 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)

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

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

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

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

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

Also define $\Gamma_n$ 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 Open-loop predictions:** Set up consistency conditions around $M_1$ and $M_2$ at each future time instance and solve as simultaneous equations. Hence the predictions are given by (see [14] for details):

$$\underset{\rightarrow}{y} = H\Delta\underset{\rightarrow}{u} + K_u\underset{\leftarrow}{u} + K_w\underset{\leftarrow}{w} + K_y\underset{\leftarrow}{y} + K_z\underset{\leftarrow}{z} \quad (5)$$

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

$$\begin{aligned} P &= C_{d-}^{-1}C_{\Delta}^{-1}C_{d+}^{-1} \\ H &= P\Gamma_n \\ K_u &= PC_{\Delta}(C_nL + C_{n-}H_{n+}) \\ K_w &= -PC_{\Delta}(C_{n-}H_{d-} + C_{d-}C_{b_2}L) - L \\ K_y &= PC_{\Delta}C_{d-}(C_{b_2}L + H_{b_2}) + L \\ K_z &= -PC_{\Delta}C_{d-}(C_{b_2}L + H_{n-}) - L \end{aligned}$$

**2.3.3 Prestabilised predictions:** It is straightforward [14] to form a parameterisation of future inputs that stabilises the predictions of (5). In essence this reduces to the constraint that $\Delta\underset{\rightarrow}{u}$ be selected such that

$$P^{-1}\underset{\rightarrow}{y} = \Gamma_{d+}\gamma \quad (6)$$

with $\gamma$ stable. This can be solved analytically via a suitable diophantine equation. More details are supplied in [14, 15] but are not given here to save space. The resulting predictions take the form

$$\left[\begin{array}{c} \Delta\underset{\rightarrow}{u} \\ \hline \underset{\rightarrow}{y} \end{array}\right] = \left[\begin{array}{c|c|c|c} C_u & C_w & C_y & C_z \\ \hline C_{yu} & C_{yw} & C_{yy} & C_{yz} \end{array}\right] \left[\begin{array}{c} \underset{\leftarrow}{u} \\ \hline \underset{\leftarrow}{w} \\ \hline \underset{\leftarrow}{y} \\ \hline \underset{\leftarrow}{z} \end{array}\right] + \left[\begin{array}{c} \Gamma_{d+} \\ \hline H_1 \end{array}\right] \underset{\rightarrow}{c} \quad (7)$$

where $H_1 = [C_{\Delta}C_{d-}]^{-1}\Gamma_n$.

## 3 The PFC Algorithm and closed-loop poles

### 3.1 Algorithms

Here we give the PFC algorithms based on open-loop and prestabilised predictions. The aim is to solve the equations in (1) using the predictions. Define the variable

$$\Psi = e^{-\frac{T}{T_{PFC}}}; \quad \Psi^{n_y} = e^{-\frac{T_h}{T_{PFC}}} \quad (8)$$

and define $\mathbf{e}_{n_y}^T$ to be the $n_y^{th}$ standard basis vector. First form the algorithm with open-loop predictions.

**Algorithm 3.1** *PFC with open-loop (unstable) predictions. Substitute predictions (5) and (8) into (1):*

$$\begin{aligned} y_{k+n_y|k} &= \mathbf{e}_{n_y}^T[H\Delta\underset{\rightarrow}{u} + K_u\underset{\leftarrow}{u} + K_w\underset{\leftarrow}{w} + K_y\underset{\leftarrow}{y} + K_z\underset{\leftarrow}{z}] \\ &= (1 - \Psi^{n_y})r_{k+n_y} + \Psi^{n_y}y_k \end{aligned} \quad (9)$$

*Solving (9) gives the control law as*

$$\Delta u_k = P_r r_{k+n_y} - P_u\underset{\leftarrow}{u} - P_w\underset{\leftarrow}{w} - P_y\underset{\leftarrow}{y} - P_z\underset{\leftarrow}{z} \quad (10)$$

*where* $g = (\mathbf{e}_{n_y}^T H)^{-1}$, $P_u = -gK_u$, $P_w = -gK_w$, $P_y = [\Psi^{n_y}, 0, ...] - gK_y$, $P_z = -gK_z$, $P_r = g(1 - \Psi^{n_y})$.

Using prestabilised equations, the PFC algorithm is:

**Algorithm 3.2** *PFC with prestabilised predictions Assume* $\underset{\rightarrow}{c} = c_k$ *and substituting (7) and (8) into (1) implies*

$$\begin{aligned} y_{k+n_y|k} &= \mathbf{e}_{n_y}^T[H_1c_k + C_{yu}\underset{\leftarrow}{u} + C_{yw}\underset{\leftarrow}{w} + C_{yy}\underset{\leftarrow}{y} + C_{yz}\underset{\leftarrow}{z}] \\ &= (1 - \Psi^{n_y})r_{k+n_y} + \Psi^{n_y}y_k \end{aligned} \quad (11)$$

*Solving (11) gives*

$$\begin{aligned} \mathbf{e}_{n_y}^T H_1 c_k &= r_{k+n_y}(1 - \Psi^{n_y}) + y_k\Psi^{n_y} \\ &\quad -\mathbf{e}_{n_y}^T[C_{yu}\underset{\leftarrow}{u} + C_{yw}\underset{\leftarrow}{w} + C_{yy}\underset{\leftarrow}{y} + C_{yz}\underset{\leftarrow}{z}] \end{aligned} \quad (12)$$

*Substituting (12) into (7) gives the control as:*

$$\Delta u_k = P_r r_{k+n_y} - P_u\underset{\leftarrow}{u} - P_w\underset{\leftarrow}{w} - P_y\underset{\leftarrow}{y} - P_z\underset{\leftarrow}{z} \quad (13)$$

*where* $g = (\mathbf{e}_{n_y}^T H_1)^{-1}$, $h = \mathbf{e}_1^T\Gamma_{d+}$, $P_u = hg\mathbf{e}_{n_y}^T C_{yu} - \mathbf{e}_1^T K_u$, $P_w = hg\mathbf{e}_{n_y}^T C_{yw} - \mathbf{e}_1^T K_w$, $P_y = hg\mathbf{e}_{n_y}^T C_{yy} - \mathbf{e}_1^T K_y$, $P_z = hg\mathbf{e}_{n_y}^T C_{yz} - \mathbf{e}_1^T K_z$.

### 3.2 Closed-loop poles

It is convenient to rewrite the difference equations (10,13) representations of the control law in terms of z-transform polynomials $D_u$, $D_w$, $D_y$, $D_z$ as follows:

$$u_k = P_r(z)r_{k+n_y} - D_u(z)u_k - D_w(z)w_k - D_y(z)y_k - D_z(z)z_k \quad (14)$$

where $D_u(z) = P_u[z^{-1}, z^{-2}, ...]^T$, $D_w(z) = P_w[1, z^{-1}, z^{-2}, ...]^T$, $D_y(z) = P_y[1, z^{-1}, z^{-2}, ...]^T$, $D_z(z) = P_z[1, z^{-1}, z^{-2}, ...]^T$. The argument $(.)(z)$ is dropped hereafter to aid readability. To compute the implied closed-loop for the nominal case, one must combine the following equations: the process $y = Gu$, the IM (3), $w = M_1u$, $z = M_2y$ (figure 2) and the controller (14). Substituting eqn. (3) into the control action eqn. (14) gives the full controller as

$$[1 + D_u + D_w\frac{n_+}{d_-}]u = P_r r_{k+n_y} - [D_y + D_z\frac{b_2}{n_-}]y \quad (15)$$

Hence the controller in the loop can be represented as

$$K = \frac{N}{D} \quad (16)$$

$$N = \frac{d_- - d_- D_u + D_w n_+}{d_-}; \quad D = \frac{n_- D_y + D_z b_2}{n_-}$$

Combining the controller and plant model, the closed-loop poles are obtained from $1 + GK = 0$ which implies

$$(d_- - d_- D_u + D_w n_+) n_+ + (n_- D_y + D_z b_2) d_+ = 0 \quad (17)$$

## 4 Comparison of algorithms

As it is hard to generalise theoretically, algorithms 3.1,3.2 will be compared using an number of examples spanning a range of possible unstable processes. The comparison will be by way of closed-loop poles, which indicate stability and expected speed of response, and closed-loop simulations for selected values of tuning.

### 4.1 Examples used

Several processes with different types of unstable poles/zeros are trialed.

- Example 1 has just one unstable pole at $z = 1.5$.

$$G(z) = \frac{z^{-1} - 0.3z^{-2}}{1 - 1.9 + z^{-1}0.48z^{-2} + 0.18z^{-3}}$$

- Example 2 has an unstable pole at $z \approx 1.49$ and an unstable zero $z \approx 1.22$. Note the pole is greater than the zero.

$$G(z) = \frac{0.2126z^{-1} - 0.2594z^{-2}}{1 - 2.3967z^{-1} + 1.3499z^{-2}}$$

- Example 3 has an unstable pole at $z \approx 1.22$ and an unstable zero $z \approx 1.35$.

$$G(z) = \frac{0.18z^{-1} - 0.2432z^{-2}}{1 - 2.1262z^{-1} + 1.1052z^{-2}}$$

- Example 4 has 2 unstable poles at $z = 1.2068 \pm 0.1885i$

$$G(z) = \frac{0.2661z^{-1} - 0.2172z^{-2}}{1 - 2.4136z^{-1} + 1.4918z^{-2}}$$

### 4.2 Closed-loop poles

In this section, the implied nominal closed-loop poles that arise from a PFC design are computed for PFC algorithms 3.1 and 3.2. This computation is carried out for various choices of coincidence horizon and target poles $\Psi$ (denoted as *Psi* in the figures). The modulus of the poles for 4 different $\Psi$ is plotted (y-axis) against coincidence horizon (x-axis) in figures 3-6 for examples 1-4 respectively. The poles for algorithm 3.1 are in dashed line and for algorithm 3.2 in solid line. The dotted line (modulus 0, 1) denotes the origin and stability boundary respectively. Clearly if any pole is modulus greater than 1, then the closed-loop will be unstable for the given coincidence horizon.

What is immediately clear from figures 3-6 is that overall the prestabilised approach to PFC (algorithm 3.2) is far more likely to give a stablising control law. That is the solid lines are far more often of modulus less than one and notably for coincidence horizons large enough algorithm 3.2 stabilised all the examples. However, for small coincidence horizons it could be unstable.

Ironically, PFC based on non-prestabilised predictions (algorithm 3.1) was more likely to be stable with small coincidence horizons (though not always, see examples 2,3) and was usually unstable for large coincidence horizons. Moroever, for large coincidence horizons one of the poles always tended to one, i.e. the closed loop would become very slow. Algorithm 3.1 could not stabilise example 2 at all!

In summary, for a simple instability (examples 1,4) algorithm 3.1 could be stabilising and give good poles for small coincidence horizons. Algorithm 3.2 on the otherhand was always stable with good poles for large horizons (including the troublesome example 2 on which algorithm 3.1 failed) and was sometimes also good for small horizons. Example 3 was not stabilised with a small horizon by either algorithm, perhaps due to the non-mimimum phase characteristic. There was some insignificant variability of the details with $\Psi$.

### 4.3 Simulation examples

The conclusions above can be readily illustrated with some closed-loop simulations. These are important because there is not a direct correspondence between closed-loop poles and performance. The following simulations are produced; as before dashed lines denote algorithm 3.1 and solid lines algorithm 3.2. .

- Figure 7. Example 1 with $n_y = 1, \Psi = .7$ and $n_y = 10,\ \Psi = 0.7$.
- Figure 8. Example 2 with $n_y = 7, \Psi = .5$ and $n_y = 20,\ \Psi = 0.5$.
- Figure 9. Example 3 with $n_y = 10, \Psi = .5$ and $n_y = 15,\ \Psi = 0.5$.
- Figure 10. Example 4 with $n_y = 2, \Psi = .6$ and $n_y = 8,\ \Psi = 0.6$.

It is clear from these that sometimes algorithm 1 gives the best performance and sometimes algorithm 2 and also that the simulations illustrate the results of the pole analysis. In Figure 7 it is seen that algorithm 3.2 gives good responses (but has a jerky input for low $n_y$) whereas algorithm 3.1 gives good response only for small $n_y$. Figure 8 suggests that algorithm 3.2 gives better response with larger $n_y$, algorithm 3.1 is unstable. Figure 9 shows similar performance for $n_y = 10$, but again shows the degradation of algorithm 3.1 for

large $n_y$. Figure 10 shows that contrary to the pole analysis of figure 6, good performance can only be obtained with algorithm 3.1 - again small $n_y$ only. Although algorithm 3.2 is stable for large $n_y$, the response is very poor (almost dead-beat behaviour).

## 5 Conclusions

It is clear that the variants of PFC based on unstable predictions and prestabilised predictions have different strengths and weaknesses. One would want to have both algorithms available to tackle any unknown process. It is noted that the algorithm based on open-loop predictions cannot stabilise every process, but when it works (usually in a narrow range of horizons) it can give good performance. The algorithm based on prestabilised predictions gives much better stability assurances and seems to stabilise all processes for large horizons, however the performance can be over active and resemble dead-beat behaviour for some processes. Future work will seek to explain this weakness in the algorithm and propose some improvements to increase its flexibility and appeal. Preliminary work based on [11] has given encouraging improvements.

## References


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

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

[3] 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, 308-323

[4] Kouvaritakis, B., J.A. Rossiter and A.O.T.Chang (1992), Stable Generalized predictive control, Proc IEE, 139, 4, pp349-362

[5] Kouvaritakis, B., J.R. Gossner and J.A. Rossiter (1996), Apriori stability condition for an arbitrary number of unstable poles, Automatica, 32, 10, 1441-1446

[6] Mosca, E. and J. Zhang (1992), Stable redesign of predictive control, Automatica, 28, 1229-1233

[7] Rawlings, J.B. and K.R. Muske (1993), The stability of constrained receding horizon control, Trans IEEE AC, 38, pp1512-1516

[8] Richalet, J., A. Rault, J.L. Testud and J. Papon (1978), Model predictive heuristic control: applications to industrial processes, Automatica, 14, 5, pp413-428

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

[10] Richalet, J., Plenary lecture, UKACC 2000

[11] Rossiter, J.A., J.R. Gossner and B. Kouvaritakis (1996), Infinite horizon stable predictive control Trans. IEEE AC, 41, 10, pp1522-1527.

[12] Rossiter, J.A., M.J. Rice and B.Kouvaritakis (1998). A numerically robust state-space approach to stable predictive control strategies, Automatica, 38, 1, 65-73

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

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

[15] Rossiter, J.A., (2001), Predictive functional control of unstable processes, submitted to IFAC 2002

[16] Rossiter, J.A., (2001), Handling constraints with predictive functional control of unstable processes, submitted to ACC 2002

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


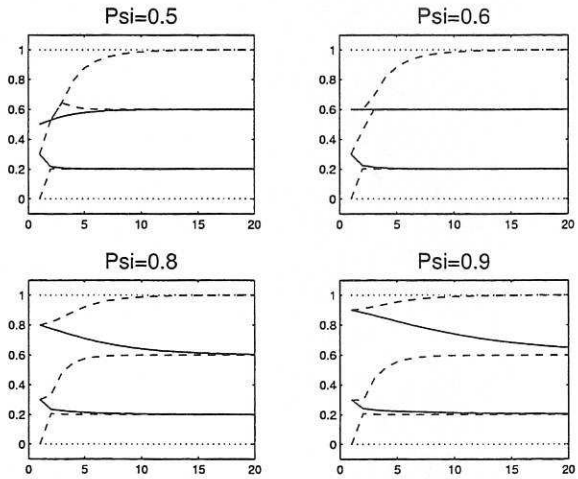


**Figure 3.** Modulus of closed-loop poles for example 1

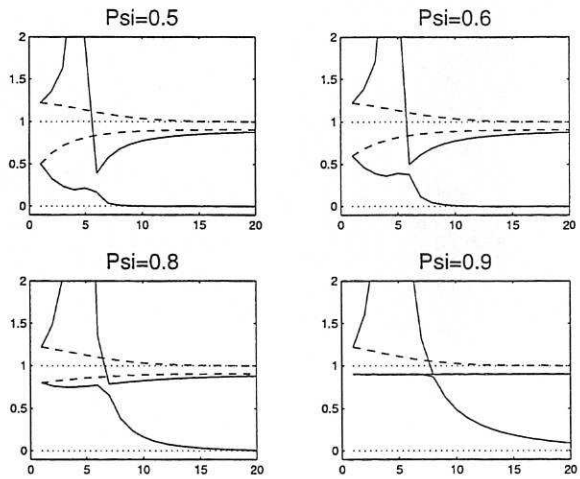


**Figure 4.** Modulus of closed-loop poles for example 2

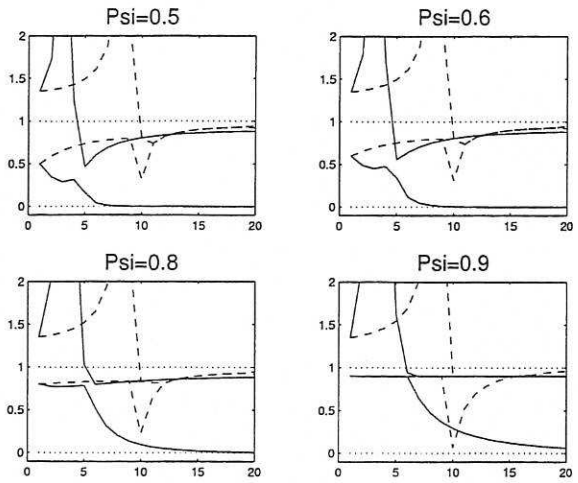


**Figure 5.** Modulus of closed-loop poles for example 3

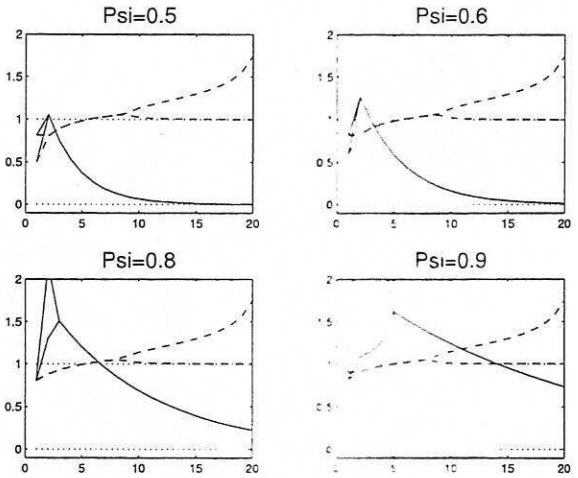


**Figure 6.** Modulus of closed-loop poles for example 4

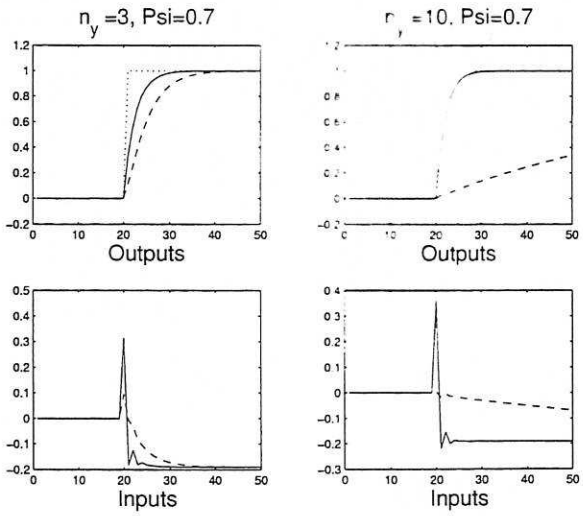


**Figure 7.** Simulations for example 1

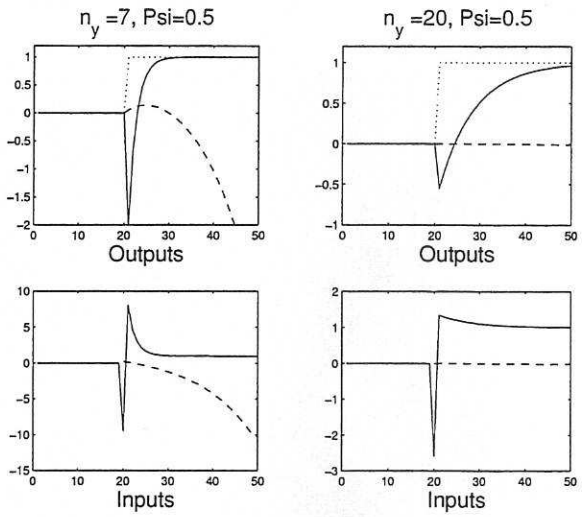


**Figure 8.** Simulations for example 2

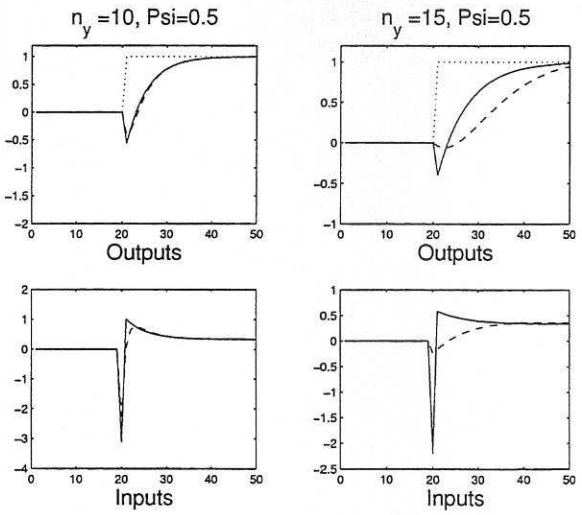


**Figure 9.** Simulations for example 3

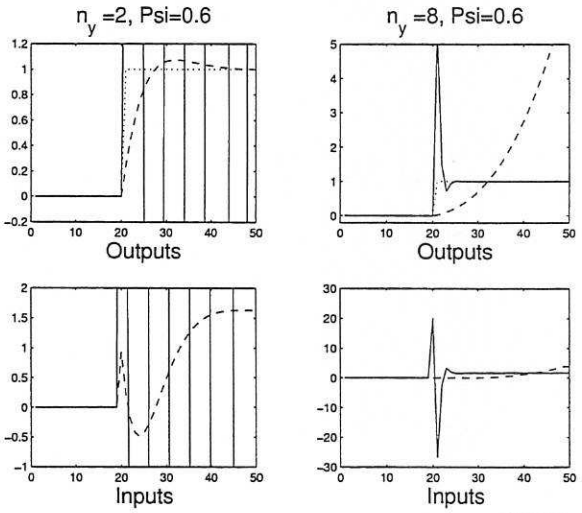


**Figure 10.** Simulations for example 4