Systems & Control Letters 59 (2010) 139–146

Contents lists available at ScienceDirect

Systems & Control Letters

journal homepage: www.elsevier.com/locate/sysconle

Iterative robust control: Speeding up improvement through iterations

S. Garatti [a], M.C. Campi [b,*], S. Bittanti [a]

[a] Department of Electronics and Information, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133, Milano, Italy

[b] Department of Electrical Engineering for Automation, University of Brescia, Via Branze 38, 25123, Brescia, Italy

ARTICLE INFO

Article history:
Received 28 March 2008
Received in revised form
31 July 2009
Accepted 12 December 2009
Available online 31 December 2009

Keywords:
Iterative control
Identification for control
Robustness
Randomized methods
Adaptive systems

ABSTRACT

Iterative control has been widely studied in recent years as an efficient methodology for the design of highly performing controllers of complex plants. The idea behind iterative design is that, when the plant is exceedingly complex, the design of the controller in one shot is hazardous. Instead, one can perform a sequence of closed-loop identification and controller design steps, aiming at progressively learning how to increase the control performance through experience.

In this paper, we introduce a new iterative control scheme which explicitly accounts for the presence of uncertainty in the plant description (iterative *robust* control). Our contention is that introducing robustness in iterative schemes permits one to quickly improve performance through steps, while preserving the robust stability of the closed-loop system.

1. Introduction

1.1. Why iterative control?

Consider the problem of designing a highly performing controller C for an uncertain plant P (see Fig. 1). We suppose that the control performance is evaluated by means of a given control cost $J(C, P) \geq 0$ such that the lower the J the better the performance. The final objective is to find a controller C that guarantees a given performance level: $J(C, P) \leq \bar{J}$.

The main feature of the control problem under consideration is that the plant dynamics is assumed to be *unknown*, a situation which occurs in many practical engineering problems.

A typical way to deal with uncertainty in the plant dynamics is to resort to identification methods to obtain a model $\widehat{P}$ of the plant. Then, the controller is designed based on $\widehat{P}$. This way of proceeding, however, calls for some care. Indeed, it is well known, [1–3], that identifying the plant dynamics in one shot may often result in a model which is unsuitable for controller design purposes. The reason is that it is a priori difficult to select an appropriate model class which achieves a sensible compromise between plant complexity and the limitations posed by the finiteness of the data set (bias vs. variance error trade-off). Moreover, when the plant input can be manipulated, designing a suitable experiment can be difficult.

A well recognized fact, [4,1–3], is that not all the plant characteristics are important for closed-loop performance. Thus, the goal of the identification step is that of accurately identifying only those (usually few) plant dynamical features which are relevant to control design. Though the system–model mismatch may still remain large, highly performing controllers can then be designed as the identified model turns out to be properly tuned towards control objectives (*identification for control*, [5–7]).

Clearly, the problem here is how to perform the identification experiment so as to identify the "plant dynamical features which are relevant to control design". Iterative procedures prove powerful for this purpose, [1,8,3].

In general terms, an iterative control scheme goes as follows. Suppose that a controller C_{i-1} has been already designed, whose performance is however not good enough. The controller C_{i-1} is updated to C_i through the following steps:

Step 1. Data are collected in *closed-loop* with C_{i-1} in place, and a plant model $\widehat{P}_i$ is estimated;

Step 2. A new controller C_i is designed based on $\widehat{P}_i$;

Step 3. C_i is connected to the plant and the performance is checked: if $J(C_i, P) \leq \bar{J}$ then the procedure is halted; otherwise $i = i + 1$ and the procedure is repeated from step 1.

* Corresponding author. Tel.: +39 030 3715458; fax: +39 030 380014.
E-mail addresses: sgaratti@elet.polimi.it (S. Garatti), marco.campi@ing.unibs.it (M.C. Campi), bittanti@elet.polimi.it (S. Bittanti).
URLs: http://home.dei.polimi.it/sgaratti/ (S. Garatti), http://www.ing.unibs.it/~campi/ (M.C. Campi), http://home.dei.polimi.it/bittanti/ (S. Bittanti).

0167-6911/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysconle.2009.12.009

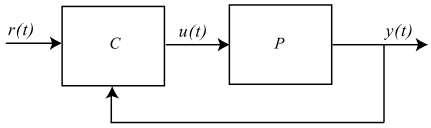

Fig. 1. Closed-loop system.

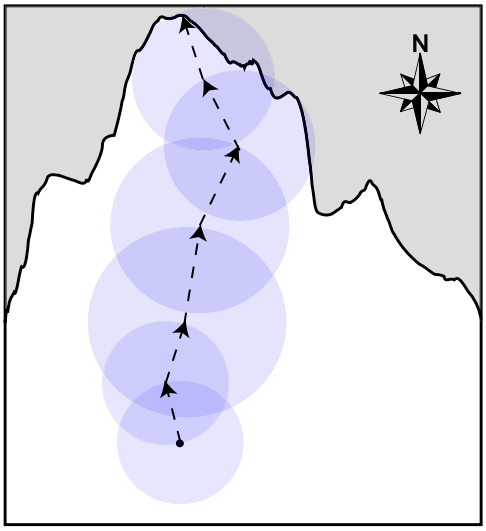

Fig. 2. An iterative procedure for the cliff problem.

The controller validation at step 3 is performed by means of experimental data collected while the real plant is operated with C_i.

As it appears, iterative control consists in a sequence of intertwined closed-loop identification and control design steps. The goal is to unveil the plant dynamical features relevant to control design through small adjustments. The effectiveness of this idea can be better understood through the following metaphor.

A metaphor that catches the essence of iterative control. Suppose a person is not too far from a cliff. It is dark and he can only light the scenery up by means of a few matches he has is his pocket. The objective is to get as close as possible to the northernmost point on the edge of the cliff without falling down (see Fig. 2).

A decision needs to be made as to whether to use the matches altogether or individually one after the other. By striking the matches altogether, the hope is to bring light into the whole route to the cliff's edge (one-shot full plant identification). There is, however, the risk that the light would be too dim to reveal where the edge is located exactly, and the matches would probably be wasted if used this way. A wiser strategy consists instead in lighting one match at a time so as to reveal the scene in the vicinity of the current position, and then moving a little step towards the north in the area that is visible. As the person moves in the new position, he cannot proceed any further safely in the darkness. At this point, another match is used so that the next step can be performed in a safe way, and so on and so forth until the northernmost point is reached. ∗

Iterative control works similarly. The closed-loop setting at step 1 is introduced to avoid a full plant identification. Indeed, in closed-loop only some features of the plant are excited and estimated. Thus, during the first iterations – when poorly performing controllers are connected to the plant – only certain plant dynamical features are unveiled. This information is then used to adjust in step 2 the current controller, so moving a little step in the direction of improving the control performance. Iterating this scheme, the desired performance can be eventually reached.

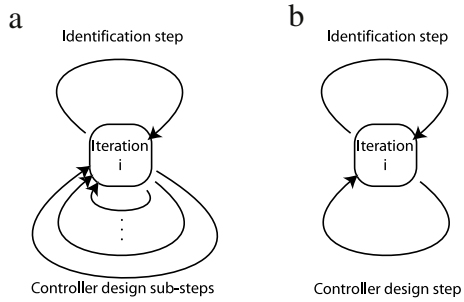

Fig. 3. Iterative schemes (a) vs. robust iterative schemes (b).

One important aspect which is worth emphasizing in the metaphor is that the northernmost point was reached by enlightening not all the cliff but only a path connecting the initial position to the target position. Therefore, the identification effort is spent towards achieving the final objective without learning "too much" of the system. Similarly, when improving the control system performance step by step in an iterative controller design scheme, we do not explore the whole plant dynamics to achieve the final result.

1.2. The need for robustness in iterative control

The goal of the present paper is to introduce and discuss iterative *robust* control. Our contention is that incorporating robustness in iterative schemes permits one to improve the performance through iterations more rapidly. The idea is better put in context by first emphasizing some drawbacks with classical (non-robust) iterative schemes.

Iterative control has been intensively studied in the last decade, [4,9–13], and there is a variety of iterative techniques with different and specific features, [14–17,1,18,8,3]. Yet, as shown in [19,20], a common feature of iterative methods is the need for cautious adjustments at each controller update step. This can be easily understood considering that, at each iteration i, only a partial description of the plant becomes available, and, consequently, the controller C_i has to be designed on a conservative ground (*cautious controller*). In many iterative schemes, the model $\widehat{P}_i$ at step 1 of the procedure is simply a nominal model of the plant with no concern for its reliability. Consequently, when the controller C_i is designed at step 2, one has no hints on its range of validity. This circumstance reflects into an over-conservative use of the model.

In e.g. the well known "windsurfer" approach, [21,22,8], the fact that the model reliability is unknown is taken care of by splitting step 2 into a number of sub-steps (Fig. 3. a): using the model identified at step 1 and *without updating it*, the controller is progressively tuned so as to improve the control performance (e.g. when the plant is linear, one typically tries to progressively increase the closed-loop bandwidth); at any sub-step, the designed controller is tested on the real plant to avoid reaching the closed-loop stability limit. When it is detected that a further performance improvement is likely to generate instability, the model $\widehat{P}_i$ is no longer deemed reliable and a new model is identified. This means that step 2 is halted and the procedure moves on to step 3.

Unfortunately, this way of proceeding has a drawback: each intermediate controller has to be tested on the real plant and this requires to access the plant many times for experiment. This results in a relatively long and expensive design procedure.

The drawbacks in the above approach can be alleviated by resorting to *robust* iterative control techniques, i.e. iterative schemes which explicitly account for the presence of uncertainty in the control design phase at step 2. In this way, it will be possible to design the controller at each iteration in a single shot (Fig. 3. b),

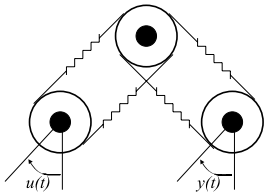

Fig. 4. The Grenoble transmission system.

so reaching high performances without experimental over-effort. Precisely, this can be obtained by replacing steps 1 and 2 with the following ones:

Step 1′. From the data collected in closed-loop:
1′. a Estimate a nominal model;
1′. b Estimate a model of uncertainty;
Step 2′. Design the *best possible robust controller* according to the existing level of uncertainty.

The idea behind points 1′ and 2′ above can be explained as follows. At iteration i, a sensible selection of the controller has to meet two different and conflicting objectives:

- on the one hand, the controller has to be cautious to avoid a possible destabilization of the control system;
- on the other hand, it should not be over-conservative, otherwise the corresponding performance improvement is not significant.

The robust controller design at step 2′ performs in a single step the best compromise between these two objectives according to the present level of uncertainty. In this way, the achieved performance rapidly improves through iterations, while preserving the robust stability of the closed-loop system. This is an important achievement, and we expect that the algorithm here introduced will impact on the real implementation of iterative control, by also fostering a more widespread acceptance of iterative schemes among practitioners.

1.3. Outline of the paper

The iterative algorithm outlined above describes the essential idea of iterative robust control as introduced in this paper. This idea is more concretely developed in the subsequent sections by performing specific choices for the implementation of steps 1′ and 2′.

For the sake of concreteness, all the choices are first presented in Section 2 for a simple example of an elastic transmission system. However, these choices carry a general applicability. Further discussion on alternative choices is provided in Section 3, while Section 4 contains some final conclusions.

2. Robust iterative control: The example of an elastic transmission system

2.1. Plant description

Consider the Grenoble transmission system of [23]. This system is formed by three pulleys connected by two elastic belts as shown in Fig. 4. The system input $u(t)$ is the angular position of the first pulley, while the output $y(t)$ is the angular position of the third pulley. The control objective is to make the angular position of the third pulley as close as possible, over a suitable bandwidth, to a given reference signal $r(t)$ (tracking control problem).

The discrete-time u to y transfer function of the system is (see [23]):

$$P(z) = \frac{0.033z + 0.054}{z^4 - 2.85z^3 + 3.72z^2 - 2.65z + 0.87}.$$

It has two pairs of complex conjugate stable poles, giving rise to two resonant peaks. A zero outside the unit circle (non-minimum phase zero) is also present. See Fig. 5 for the system Bode diagram and for a graphical representation of its poles and zeros.

A 1-degree-of-freedom control scheme as shown in Fig. 6 is considered. Initially, the plant was operated with the following PI controller:

$$C_0(z) = 0.01 \cdot \frac{z + 0.5}{z - 1},$$

which resulted in a stable but slow closed-loop system (Fig. 7).

2.2. Identification and uncertainty estimation (step 1′)

Closed-loop identification was performed with a square wave with period $T = 100$ and amplitude 1 as reference input, and data collection lasted $N = 3000$ instants. The system output was generated by simulation from system:

$$y(t) = P(z)u(t) + d(t),$$

where noise $d(t)$ was

$$d(t) = \frac{z - 2}{z - 0.9}e(t), \quad e(t) = WGN(0, 0.0001)$$

(WGN = White Gaussian Noise). Note that $d(t)$ is a highly correlated stochastic noise as it is typical of many real applications. Its standard deviation is 0.026.

As prescribed by the robust iterative scheme in steps 1′.*a* and 1′.*b*, we need to estimate a nominal model as well as a model of uncertainty. These models have to be used in the subsequent step 2′ to design a robust controller.

As we shall see (Section 2.3), the nominal model and the uncertainty description play very different roles in the control design step and, thus, it is advisable to consider different classes of models for the two steps 1′.*a* and 1′.*b*. Specifically, the nominal model of step 1′.*a* needs to be simple to facilitate the construction of a nominal controller, while the model for uncertainty description of step 1′.*b* is complex enough to capture all system parts.

For nominal model identification, we considered an ARMAX(4,2,4) model class:

$$\mathcal{M} = \left\{ y(t) = \frac{B(z, \lambda)}{A(z, \lambda)}u(t) + \frac{C(z, \lambda)}{A(z, \lambda)}\eta(t) \right\},$$

where $\eta(t)$ is white noise and λ is the vector of A, B, C coefficients, and the model $M(\widehat{\lambda}_i)$ at iteration i was obtained by resorting to Prediction Error Methods (PEM), [24,25].

The model class for uncertainty evaluation was instead:

$$\mathcal{P} = \{ y(t) = P(z, \vartheta)u(t) + v(t) \},$$

where $v(t)$ is a noise process and $P(z, \vartheta)$ is parameterized through a Finite Impulse Response (FIR) filter, i.e. $P(z, \vartheta) = \vartheta_1 z^{-1} + \vartheta_2 z^{-2} + \cdots + \vartheta_n z^{-n}$ with $n = 100$. The choice of n was based on an inspection of the plant impulse response.

The FIR structure of order 100 carries 100 parameters, a large amount. This choice was made because we supposed we had no clue on the real system structure other than that its transient drops off in approximately 100 instants, in which case the FIR structure is the most reasonable choice. Should some information on the system structure be known, such an information could be conveniently incorporated in the model $P(z, \vartheta)$.

The uncertainty evaluation was performed by means of the asymptotic theory of system identification, [24,25], that provides a probability density $f_i(\vartheta)$ describing the likelihood that the model corresponding to ϑ is the true system. This density is Gaussian with a mean and a variance estimated from data. Specifically, the asymptotic theory of Instrumental Variable (IV) methods was

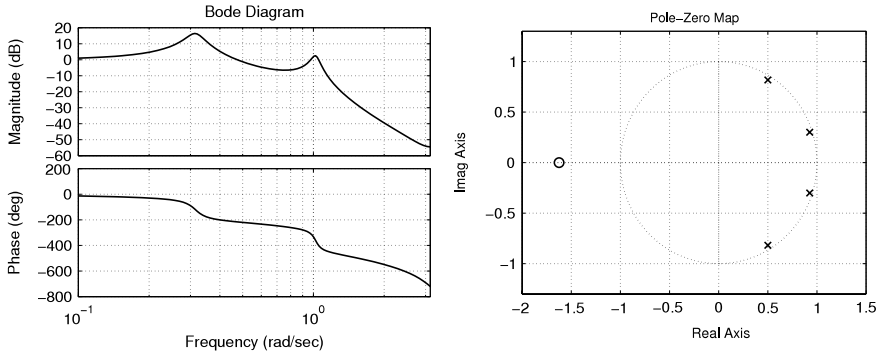

Fig. 5. Bode diagram and poles and zeros of $P(z)$.

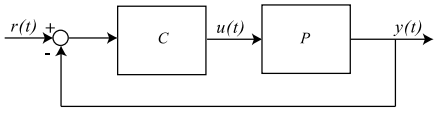

Fig. 6. Closed-loop system.

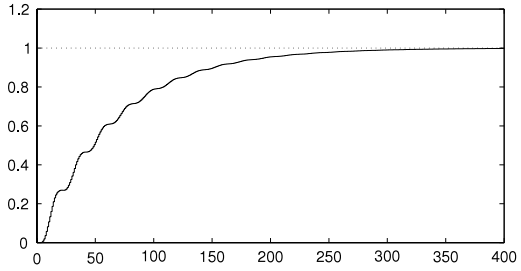

Fig. 7. Step response of the initial closed-loop system.

used (e.g. [24], Chapter 9.5) with $[u_r(t-1)\dots u_r(t-100)]'$ as instrument, where $u_r(t)$ was obtained by simulating noise-free the nominal model with the latest designed controller in the loop and with reference $r(t)$. This is a standard choice in IV implementation, see e.g. [26].

We feel advisable to remark that $f_i(\vartheta)$ is not a descriptor of the uncertainty in the nominal model $M(\widehat{\lambda}_i)$, it instead directly describes how reliable the full description of the real plant provided by $P(z,\vartheta)$ is, based on the available batch of data. This is the relevant information used in the controller design step 2′ to detune the nominal controller, as described in the next section.

2.3. Controller design (step 2′)

The objective of this section is to describe how the information supplied by step 1′ (i.e. $M(\widehat{\lambda}_i)$ and $f_i(\vartheta)$) can be used to design the "best possible robust controller" in step 2′.

A well known fact is that designing a controller with robust features is a hard task in general. We therefore introduce a specific two-stage easy-to-implement approach as follows:

a. First, a *nominal controller* C_i is designed based on the nominal model $M(\widehat{\lambda}_i)$;
b. The nominal controller is *detuned* in the light of the existing level of uncertainty as described by $f_i(\vartheta)$, so as to meet robustness requirements.

The big part of the controller is selected in point a. based on the nominal model with no concern for uncertainty, and this makes the design simple. Uncertainty comes into play only for detuning, which is much easier a robust design than determining the *whole* robust controller.

Nominal controller design and detuning are now discussed in turn.

As nominal controller we used a linear deadbeat controller which can be obtained from the identified u to y nominal transfer function $B(z,\widehat{\lambda}_i)/A(z,\widehat{\lambda}_i)$ as follows (see [27]):

$$C_i(z) = \frac{A(z,\widehat{\lambda}_i)}{B(1,\widehat{\lambda}_i)z^k - B(z,\widehat{\lambda}_i)}, \tag{1}$$

where $k = 4$ is the order of $A(z,\widehat{\lambda}_i)$.

When connected to $P(z)$, $C_i(z)$ yields the following complementary sensitivity function (here $B_P(z)$ and $A_P(z)$ are respectively the numerator and the denominator of $P(z)$):

$$F_i(z) = \frac{B_P(z)A(z,\widehat{\lambda}_i)}{A_P(z)B(1,\widehat{\lambda}_i)z^k - A_P(z)B(z,\widehat{\lambda}_i) + B_P(z)A(z,\widehat{\lambda}_i)}.$$

If $B(z,\widehat{\lambda}_i) = B_P(z)$ and $A(z,\widehat{\lambda}_i) = A_P(z)$ (i.e. the real plant $P(z)$ is exactly identified), $F_i(z) = B(z,\widehat{\lambda}_i)/B(1,\widehat{\lambda}_i)z^k$, a FIR system. This means that the reference signal is tracked in a finite number of steps (the controller is high-performing *in the nominal case*). However, $C_i(z)$ is also very sensitive to model inaccuracy and when $P(z)$ is not correctly identified, $C_i(z)$ may even lead to instability. The detuning is introduced to safeguard the control system against instability, at the price of a degradation of the nominal performance.

The detuning was obtained with a well known technique in the Internal Model Control (IMC) context, [28], which has been also used in the windsurfer iterative scheme, [8]. The idea is to augment the structure of $C_i(z)$ by forcing in a detuning parameter γ, through which one can incorporate robustness.

Precisely, the augmented controller $C_i(z,\gamma)$ is as follows:

$$C_i(z,\gamma) = \frac{A(z,\widehat{\lambda}_i)(1-\gamma)^k}{B(1,\widehat{\lambda}_i)(z-\gamma)^k - B(z,\widehat{\lambda}_i)(1-\gamma)^k},$$

where $\gamma \in \Gamma = [0,1)$. Note that $C_i(z,0) = C_i(z)$.

The complementary sensitivity function obtained when $P(z)$ is operated with $C_i(z,\gamma)$ is

$$F_i(z,\gamma) = \frac{B_P(z)A(z,\widehat{\lambda}_i)(1-\gamma)^k}{A_P(z)B(1,\widehat{\lambda}_i)(z-\gamma)^k - (A_P(z)B(z,\widehat{\lambda}_i) - B_P(z)A(z,\widehat{\lambda}_i))(1-\gamma)^k},$$

whose DC gain is always 1.

When $\gamma \to 1$, the poles of $F_i(z,\gamma)$ tend to the roots of $A_P(z)B_i(1,\widehat{\lambda}_i)(z-\gamma)^k$, which are stable provided that $P(z)$ is stable, independently of the mismatch between $P(z)$ and the identified model (*robust stability*). On the other hand, when $\gamma \to 1$, the dominant poles are those placed in γ, which means that the control system response is very slow. Thus, altogether, γ plays the role of a detuning parameter: as $\gamma \to 0$, the nominal performance is retrieved, whereas $\gamma \to 1$ leads to a guarantee of internal stability at the price of performance degradation.

The question now arises as to how γ should be selected based on the existing level of uncertainty as given by $f_i(\vartheta)$. Following [29], an approach robust in probability is adopted here as explained next.

Denote by (γ, ϑ) the closed-loop system formed by controller $C_i(z, \gamma)$ and model $P(z, \vartheta)$. The robust controller is then obtained as the solution of the following optimization program:

$$\begin{aligned} &\min_{\gamma \in \Gamma} \gamma \\ &\text{subject to } \mathbb{P}\{(\gamma, \vartheta) \text{ is not stable}\} \leq \alpha. \end{aligned} \tag{2}$$

Here, $\mathbb{P}\{A\}$ denotes the probability of event A with respect to the density function $f_i(\vartheta)$, and α is a robustness level chosen by the user. Note that $\mathbb{P}\{(\gamma, \vartheta) \text{ is not stable}\} = \int_{\Theta} \mathbf{1}_{(\gamma,\vartheta) \text{ is not stable}} \cdot f_i(\vartheta) \cdot \mathrm{d}\vartheta$, where $\mathbf{1}_{\cdot}$ denotes indicator function. $\mathbb{P}\{(\gamma, \vartheta) \text{ is not stable}\}$ is therefore a function of the sole variable γ. Thus, in the optimization program (2), the detuning effect is kept as moderate as possible, provided that a robust requirement on stability is satisfied. Since $f_i(\vartheta)$ is a Gaussian with the whole $\mathbb{R}^n$ as its support, a non-zero level α must be accepted. The degree of freedom in the choice of α has to be spent depending on the required level of robustness in the application at hand, also bearing in mind that the lower α the stronger the detuning effect.

To solve (2), we here suggest using randomized methods (see e.g. [30–34]) as explained in the next subsection.

2.4. Randomized methods

Randomized methods are Monte Carlo like methods through which an approximate solution to (2) can be computed at low computational cost.

Let $\{\gamma_1, \ldots, \gamma_p\}$ be a grid of points densely covering $[0, 1)$. In the randomized approach, one searches for the best detuning parameter among $\{\gamma_1, \ldots, \gamma_p\}$, rather than over the entire feasible set $[0, 1)$. One should note that the proposed detuning method has just 1 detuning parameter, so that a suitable covering of the domain for γ can be obtained with a reasonable amount of grid points.

Optimization over $\{\gamma_1, \ldots, \gamma_p\}$ is based on an empirical approximation of probability $\mathbb{P}$. Precisely, define

$$\widehat{\mathbb{P}}\{(\gamma_h, \vartheta) \text{ is not stable}\} = \frac{1}{q} \sum_{k=1}^{q} \mathbf{1}_{(\gamma_h, \vartheta_k) \text{ is not stable}}, \quad h = 1, \ldots, p,$$

where the ϑ_k's are q parameters extracted from Θ in an independent fashion according to the probability density $f_i(\vartheta)$. Note that $\widehat{\mathbb{P}}\{(\gamma_h, \vartheta) \text{ is not stable}\}$ is again a function of the sole γ_h.

Interestingly, the approximation introduced by using $\widehat{\mathbb{P}}$ in place of $\mathbb{P}$ can be kept moderate at will by suitably selecting the number q of the extracted ϑ_k's. The well known Hoeffding theorem [32,33] can be used to this aim:

Theorem 1 (*Hoeffding*). *Fix two real numbers $\epsilon > 0$ and $\delta > 0$. If*

$$q > (2\epsilon^2)^{-1} \ln(2p/\delta), \tag{3}$$

then we have that

$$\mathbb{P}\{(\gamma_h, \vartheta) \text{ is not stable}\} \leq \widehat{\mathbb{P}}\{(\gamma_h, \vartheta) \text{ is not stable}\} + \epsilon$$

for every $h = 1, \ldots, p$, with a probability greater than or equal to $1 - \delta$.

Remark 1. Theorem 1 says that $\mathbb{P}\{(\gamma_h, \vartheta) \text{ is not stable}\}$ can be approximated by $\widehat{\mathbb{P}}\{(\gamma_h, \vartheta) \text{ is not stable}\}$ with arbitrary precision as long as the number q of the ϑ_k extractions is sufficiently high. Note however that the theorem statement holds true with a certain probability $1 - \delta$ only. This is a consequence of the fact that $\widehat{\mathbb{P}}\{(\gamma_h, \vartheta) \text{ is not stable}\}$ is a random element depending on the extracted $\vartheta_1, \ldots, \vartheta_q$: $\mathbb{P}\{(\gamma_h, \vartheta) \text{ is not stable}\} - \widehat{\mathbb{P}}\{(\gamma_h, \vartheta) \text{ is not stable}\}$ can be smaller than ϵ for some multi-samples and not for others, and δ refers to the probability of extracting a "bad" multi-sample $\vartheta_1, \ldots, \vartheta_q$ such that $\mathbb{P}\{(\gamma_h, \vartheta) \text{ is not stable}\} - \widehat{\mathbb{P}}\{(\gamma_h, \vartheta) \text{ is not stable}\} \leq \epsilon$ does not hold for some h. It is also important to note that q in (3) depends on δ logarithmically so that a very small value of δ can be forced in without significantly affecting q.

Differently from other non-randomized numerical methods for computing $\mathbb{P}\{(\gamma_h, \vartheta) \text{ is not stable}\}$, q does not depend on the size n of ϑ. Therefore, using high order FIR models for uncertainty description (we used $n = 100$) does not adversely affect the computational complexity and this is the chief advantage of using the introduced randomized method over deterministic approaches.

Remark 2. Other randomized methods have been proposed in the literature. Among these, sequential methods have proven effective in a number of endeavors, see e.g. [35–39], and can be used for general randomized control design. We here have preferred to adopt the easier Hoeffding inequality since our randomized design reduces to just one parameter selection, in which case Hoeffding proves simple and effective.

Based on the above discussion, program (2) can be substituted by its randomized counterpart:

$$\begin{aligned} &\min_{\gamma \in \{\gamma_1, \ldots, \gamma_p\}} \gamma \\ &\text{subject to } \frac{1}{q} \sum_{k=1}^{q} \mathbf{1}_{(\gamma, \vartheta_k) \text{ is not stable}} \leq \alpha - \epsilon. \end{aligned}$$

Thanks to Theorem 1, the solution γ^o to this program is such that $\mathbb{P}\{(\gamma^o, \vartheta) \text{ is not stable}\} \leq \alpha$ holds with high probability $1 - \delta$. See Section 2.6 for a numerical example.

2.5. The complete iterative robust controller design scheme

In summary, the iterative scheme is so described:

0. (*Initialization step*) connect an initial controller $C_0(z)$ to the plant. Set $i = 1$;
1. From the data collected in closed-loop:
   1. a Identify a low order model $M(\widehat{\lambda}_i)$ in $\mathcal{M}$;
   1. b Estimate the probability density $f_i(\vartheta)$ over the high order model class $\mathcal{P}$;
2. Design the nominal controller $C_i(z)$ based on $M(\widehat{\lambda}_i)$, and from $C_i(z)$ construct $C_i(z, \gamma)$. Extract ϑ_k, $k = 1, \ldots, q$, according to $f_i(\vartheta)$ and let
   $$\gamma^o = \arg \min_{\gamma \in \{\gamma_1, \ldots, \gamma_p\}} \gamma$$
   $$\text{subject to } \frac{1}{q} \sum_{k=1}^{q} \mathbf{1}_{(\gamma, \vartheta_k) \text{ is not stable}} \leq \alpha - \epsilon;$$
3. Connect $C_i(z, \gamma^o)$ to the plant and check for the performance: if the performance is satisfactory then stop; else set $i = i + 1$ and go to 1.

2.6. Simulation results

By applying the iterative controller design scheme of Section 2.5 to the Grenoble system we achieved the following results.

Fig. 8 represents the reduced order u to y transfer function $B(z, \widehat{\lambda}_1)/A(z, \widehat{\lambda}_1)$ estimated at the first iteration. From Fig. 8, a large error between the estimated model and the plant is apparent. In Fig. 9, the Bode plot of some models extracted according to $f_1(\vartheta)$ are represented.

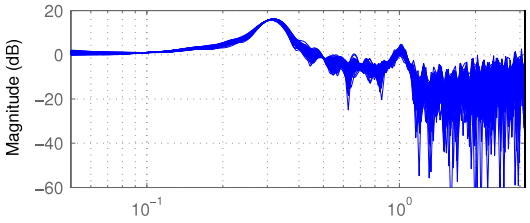

Fig. 8. Estimated nominal model at the first iteration (continuous line) and true system (dashed line).

Fig. 11. Uncertainty at the third iteration.

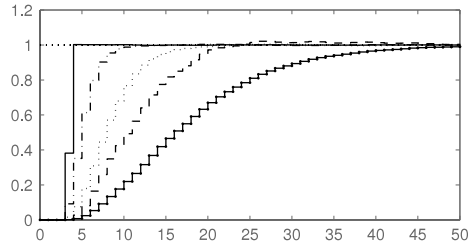

Fig. 9. Uncertainty at the first iteration: some models extracted from $f_1(\vartheta)$.

Fig. 12. Closed-loop step response for the first five iterations.

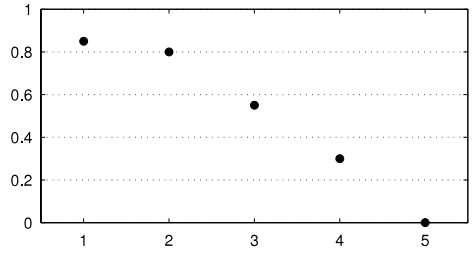

Fig. 10. Estimated nominal models at the second iteration (continuous line) and at the third iteration (dashed line).

Fig. 13. γ^o at each iteration.

The randomized algorithm for the detuning was applied by uniformly sampling $[0, 1)$ in $p = 20$ points: $\gamma_1 = 0, \gamma_2 = 0.05, \gamma_3 = 0.1, \ldots, \gamma_{20} = 0.95$.

The choices $\alpha = 0.04, \epsilon = 0.02$ and $\delta = 0.0001$ were made. By Eq. (3), the number q of models extracted according to the probability density $f_1(\vartheta)$ was 16 125. Large values of q are not a significant problem since the procedure is totally off-line.

The obtained detuning parameter γ^o turned out to be equal to 0.85. Its large value indicates a conservative choice which is justified by the high level of uncertainty. The step response of the obtained closed-loop system is the slowest curve depicted in Fig. 12.

Carrying on the iterative procedure, the identified nominal model became more and more accurate (see Fig. 10), and uncertainty concentrated around the true system (see Fig. 11). This led to the selection of the γ^o's as indicated in Fig. 13. Fig. 12 shows that the control performance rapidly improved through iterations, while preserving the robust stability.

3. Some further remarks on the implementation of the iterative scheme

Though applied to a specific example, the iterative control scheme presented in the previous section is of general applicability and it is intended as a complete and ready-to-use method. On the other hand, the user may want to stray from this scheme and make different controller/identification choices (i.e. different choices for $\mathcal{M}$, $\mathcal{P}$, $C_i(z)$, and $C_i(z, \gamma)$). In this section, we provide some hints on a number of specific aspects that the user should account for when making a selection.

Choice of the class of models $\mathcal{M}$ for nominal controller design. $\mathcal{M}$ should be a class of sufficiently low order models so as to obtain a simple enough nominal controller. As iterations progress, this model gets tuned to the system dynamics that are relevant for controller design. In general, the selection of $\mathcal{M}$ is not a critical one. Indeed, selecting too a simple $\mathcal{M}$ class does not cause any important problem with the developed iterative scheme because the detuning mechanism will automatically reveal the occurrence of this fact: the inability to describe relevant dynamics will result in that uncertainty will remain significant and the robust features will stop the detuning effect reduction from proceeding. When this happens, the user can be advised to increase the order of $\mathcal{M}$.

Choice of the class of models $\mathcal{P}$ for detuning. The user should be warned that not all model classes are suitable for uncertainty description, a fact recently pointed out in [40]. In fact, for some classes of models (e.g. Box–Jenkins models) and in condition of poor excitation (as is often the case in iterative control), the asymptotic theory of system identification may lead to unreliable results. Precisely, there are situations where the estimated density turns out to be peaky – suggesting that uncertainty is restricted – and yet the real plant is located quite afield from the peak.

To be more concrete, let us show what happens in the first iteration of our simulation of Section 2.6 if uncertainty is evaluated by means of a full order (i.e. the same order of the real plant) Box–Jenkins model class:

$$\mathcal{P}^{BJ} = \{y(t) = G(z, \vartheta)u(t) + H(z, \vartheta)\xi(t)\},$$

where $\xi(t)$ is white noise and ϑ is the vector of the numerator and denominator polynomial coefficients of G and H.

In Fig. 14, the Bode plot of some models extracted according to the density $f_1^{BJ}(\vartheta)$ obtained via the asymptotic theory of Prediction Error Methods, [24,25], are represented, showing that uncertainty concentrates around a model far from the true