Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2009, Article ID 671869, 21 pages
doi:10.1155/2009/671869

Research Article

Nonlinear Nonconvex Optimization by Evolutionary Algorithms Applied to Robust Control

Joaquín Cervera and Alfonso Baños

Computer and Systems Engineering Department, Faculty of Computer Engineering, University of Murcia, Campus de Espinardo, 30100 Murcia, Spain

Correspondence should be addressed to Joaquín Cervera, jcervera@um.es

Received 17 March 2009; Accepted 5 July 2009

Recommended by J. Rodellar

This work focuses on the problem of automatic loop shaping in the context of robust control. More specifically in the framework given by Quantitative Feedback Theory (QFT), traditionally the search of an optimum design, a non convex and nonlinear optimization problem, is simplified by linearizing and/or convexifying the problem. In this work, the authors propose a suboptimal solution using a fixed structure in the compensator and evolutionary optimization. The main idea in relation to previous work consists of the study of the use of fractional compensators, which give singular properties to automatically shape the open loop gain function with a minimum set of parameters, which is crucial for the success of evolutionary algorithms. Additional heuristics are proposed in order to guide evolutionary process towards close to optimum solutions, focusing on local optima avoidance.

1. Introduction

It is a well-known fact that there is no general procedure to exactly solve nonlinear nonconvex optimization problems when the solutions belong to continuous solution sets ([1]). In the case of solutions belonging to discrete solution sets, it is always possible to find the optimal solution by a branch and bound type exhaustive search ([2]). Branch and bound techniques can also be applied to continuous solution set problems, specially when combined with interval analysis ([3, 4]). The obtained solution is, in general, only close to optimal, according to a certain accuracy factor. A major drawback of branch and bound techniques is that they are a very costly approach in terms of computation time. The only way to quickly obtain an (approximate) solution to this sort of problem is to use some kind of randomized search algorithm, like random algorithms ([5]) or evolutionary algorithms ([6]) according to the particular problem to be solved.

Quantitative Feedback Theory is a robust frequency domain control design methodology which has been successfully applied in practical problems from different domains [7]. One of the key design steps in QFT, equivalent to controller design, is loop shaping of the open loop gain function to a set of restrictions (or *boundaries*) given by the design specifications and the (uncertain) model of the plant. Although this step has been traditionally performed by hand, the use of CACSD tools (e.g., the QFT MATLAB Toolbox [8]) has made the manual loop shaping much more simple. However, the problem of automatic loop shaping is of enormous interest in practice, since the manual loop shaping can be hard for the nonexperienced engineer, and thus it has received a considerable attention, specially in the last three decades.

Optimal QFT loop computation is a nonlinear nonconvex optimization problem, for which there is not yet an optimization algorithm which computes a globally optimum solution in a reasonable time, in terms of interactive design purposes. It must be noticed, however, that the work by Nataraj and others on this subject, based on deterministic optimization procedures, combining branch and bound optimization and interval analysis techniques, is very promising (see e.g., [9, 10]).

Other typical approaches to solve this problem have tried to find approximate solutions in different ways. For instance, some authors have simplified the problem somehow, in order to obtain a different optimization problem for which there exists a closed solution or an optimization algorithm which does guarantee a global optimum in a shorter computation time. A trade-off between necessarily conservative simplification of the problem and computational solvability has to be chosen. This is the approach in, for instance, [11, 12]. Some authors have investigated the loop shaping problem in terms of particular structures, with a certain degree of freedom, which can be shaped to the particular problem to be solved. This is the case in [13–15]. Another possibility is to use randomized search algorithms, able to directly face nonlinear and nonconvex optimization problems, at the cost of returning an only close to optimal solution, and not guaranteeing that, in general, the returned solution is not far from the optimum. This is the approach adopted in [16, 17], based on evolutionary algorithms.

Evolutionary algorithms are computationally demanding, specially as the dimension of the search space increases. In this paper the authors study the use of evolutionary algorithms-based optimization, proposing the addition, with respect to previous work, of some heuristics, very much specific to the particular problem under consideration, which help to improve obtained solutions accuracy and computation time needed to obtain these solutions. In this sense, a good structure for the compensator, in terms of using a reduced set of parameters, but with a rich frequency domain behavior, is of crucial importance. This is the main heuristic proposed in this paper: to use evolutionary algorithms together with a flexible structure, able to get a close to optimum solution, but with a reduced number of parameters. In previous work, the compensator has been fixed to a rational structure, with a finite (but no necessarily small) number of zeros and poles. In this work, the main contribution is to introduce a fractional compensator that, with a minimum number of parameters, gives a flexible structure in the frequency domain regarding automatic loop shaping. In fact, it can be approximated by a rational compensator, but with a considerably large number of parameters. This dramatic reduction in the number of parameters has shown to be of capital importance for the success of evolutionary algorithms in the solution of the automatic loop shaping problem. Other applied heuristics have to do with including some features in the objective function that guide the evolutionary search towards close to optimum solutions, paying special attention to prevent the search from

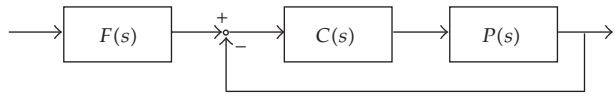

Figure 1: Two degrees of freedom control system configuration.

getting stacked in local minima, which is specially likely to happen in the problem under consideration.

In this work, following [18–21] it is considered the particular case of minimum phase open loop gain functions, for which the investigated compensators can give a good structure with a reduced set of parameters. Some first steps towards the nonminimum phase case are given in [18].

From here onwards, the structure of the paper stands as follows. In Section 2 a brief introduction to QFT is given; in Section 3 the proposed solution for the QFT automatic loop shaping problem by evolutionary algorithms is presented; in Section 4 this procedure is applied to a typical benchmark problem. Finally, Section 5 presents the conclusions.

2. Introduction to QFT

The basic idea in QFT ([7]) is to define and take into account, along the control design process, the quantitative relationship between the amount of uncertainty to deal with and the amount of control effort to use. Typically, the QFT control system configuration (see Figure 1) considers two degrees of freedom: a controller $C(s)$, in the closed loop, which manages uncertainty and disturbances affecting $u(t)$ or $y(t)$; and a precompensator, $F(s)$, designed after $C(s)$, used to satisfy tracking specifications.

Consider an uncertain plant, represented as the set of transfer functions $\mathbf{P} = \{P(s) = P_0(s)Q(s), Q \in \mathbf{Q}\}$, with $P_0(s)$ being the nominal plant and $\mathbf{Q}$ being a set of transfer functions representing uncertainty. For a certain frequency ω, the *template* $T_{\mathbf{P}}(j\omega)$ is defined as the Nichols chart (**NC**) region ([22]) given by

$$T_{\mathbf{P}}(\omega) \doteq \left\{ \left(\angle P(j\omega), \left| P(j\omega) \right|_{dB} \right) \in \mathbf{NC}, P \in \mathbf{P} \right\}. \tag{2.1}$$

The design of the controller $C(s)$ is accomplished in the Nichols chart, by shaping the nominal open loop transfer function, $L_0(s) = P_0(s)C(s)$. A discrete set of design frequencies Ω is chosen. Given quantitative specifications on robust stability and robust performance on the closed loop system, the set of *boundaries* $\mathbf{B} = \{B_\omega, \omega \in \Omega\}$ is computed. Each B_ω defines the allowed region $\mathbf{A}_\omega$ (conversely, forbidden region $\mathbf{F}_\omega = \mathbf{NC} \setminus \mathbf{A}_\omega$) in **NC** for $L_0(j\omega)$, so that $L_0(j\omega) \in \mathbf{A}_\omega$ (conversely, $L_0(j\omega) \notin \mathbf{F}_\omega$) for all $\omega \in \Omega$ (*boundaries satisfaction*) implies specification satisfaction by $L(s) = P(s)C(s)$ for all $P(s) \in \mathbf{P}$. For example, in Figure 9 we have the following.

(i) The open lines horizontally crossing the Nichols plane from $-360°$ to $0°$ are performance boundaries. In this example, for each $\omega \in \Omega = \{0.1, 0.5, 2, 15, 100\}$, $\mathbf{A}(j\omega)$ is defined as the region above B_ω.

(ii) The closed line around the point (0 dB, −180°) is a robust stability boundary, defining $\mathbf{A}(j\omega)$ (conversely $\mathbf{F}(j\omega)$) as the region outside (conversely inside) B_ω. In this example, this boundary is a *Universal High-Frequency Boundary* (UHFB), a stability boundary called *universal* because the region inside it, $\mathbf{F}_{\text{UHFB}}$, is not forbidden only for a particular frequency ω, but for all $\omega \in \mathbb{R}^+$, so it imposes $L_0(j\omega) \notin \mathbf{F}_{\text{UHFB}}$ for all $\omega \in \mathbb{R}^+$.

The basic step in the design process, *loop shaping*, consists of the following nonlinear nonconvex constrained optimization problem: design of $L_0(s)$ which satisfies boundaries (constraints) and minimizes the high-frequency gain (optimization criterion—[23]), defined as

$$K_{\text{hf}} \doteq \lim_{s\to 0} s^e L_0(s), \tag{2.2}$$

where e is L_0 excess of poles over zeros. The comparison in terms of this optimization criterion of two nominal open loops $L_{0_1}(s)$ and $L_{0_2}(s)$, with respective excess of poles over zeros e_1 and e_2, only makes sense if $e_1 = e_2$. The comparison is performed in terms of the loops high-frequency gains, K_{0_1} and K_{0_2}, respectively.

Note that this criterion is defined irrespective of the particular structure used for $L_0(s)$. It has been shown ([23]) that for minimum phase $L_0(s)$ this optimum exists, it is unique and, in general, has an infinite number of poles and zeros, and so it is not implementable. For this reason it is referred along this work as the QFT *theoretical optimum*, $L_{\text{qto}}(s)$, with optimal high-frequency gain K_{qto}.

Since $L_{\text{qto}}(s)$ is not practical as a loop design, a more practical definition of optimum can be stated. Define $X \doteq \{x = (x_1, \ldots, x_n)\}$, $x_i \in \mathbb{R}$, $i = 1 \cdots n$, as the set of controller structure parameters to be instantiated. Consider $L_X(s)$ an open loop structure parameterized by X, that is, each $x \in X$ defines $L_x(s)$, a particular instance of $L_X(s)$, with high-frequency gain K_{L_x}. Define $K_{L_O} \doteq \min\{K_{L_x}, x \in X\} \leq K_{\text{qto}}$, the minimum high-frequency gain that can be achieved by any $L_X(s)$ instance. An L-optimal transfer function or L_O is defined as an $L_X(s)$ instance L_a such that $K_{L_a} = K_{L_O}$.

For a given $L_X(s)$ open loop structure, K_{L_O} represents the best approach to K_{opt} that can be achieved with that structure. The ability of a structure $L_X(s)$ to achieve $L_{\text{qto}}(s)$ shape is called *close-to-optimality*, defined as $\text{CO}(L) \doteq K_{L_O} - K_{\text{qto}}$. Since computing L_O is in general still a hard optimization problem, and thus K_{L_O} is not known, it is common to compute instead suboptimal transfer functions L_{sOi} with high-frequency gain $K_{L_{\text{sOi}}}$ as close to K_{L_O} as possible, with $i \in \mathbb{N}$. This is the approach considered in this work, using evolutionary algorithms to compute an L_{sOi} with $K_{L_{\text{sOi}}}$ which approaches K_{L_O} as much as possible. A measure of how accurately a certain L_{sO1} approaches K_{L_O} can only be established in relative terms, that is, by comparing it to another L_{sO2} in terms of their high-frequency gains.

The meaning of this optimum definition is minimizing the cost of feedback related with sensor noise amplification at high frequencies by minimizing $|L_0(j\omega)|$ at high frequencies, which, due to phase/magnitude relations given by Bode integrals ([24]), is equivalent to minimize $\int_{\mathbb{R}^+} \angle L_0(j\omega) d\omega$, subject to $\mathbf{B}$ satisfaction. Figures 2 and 3 show the typical shape of a QFT optimal loop $L_{\text{qto}}(j\omega)$, with polygonal line simplified versions of $L_{\text{qto}}(j\omega)$ and boundaries in $\mathbf{B}$. For frequencies $\omega \in [0, \omega_{a2}]$, $\angle L_{\text{qto}}(j\omega)$ minimization is limited by performance boundaries. For frequencies $\omega \in [\omega_{a2}, \infty]$, $\angle L_{\text{qto}}(j\omega)$ minimization is only limited by the UHFB (which makes $L_{\text{qto}}(j\omega)$ tightly follow the right and bottom

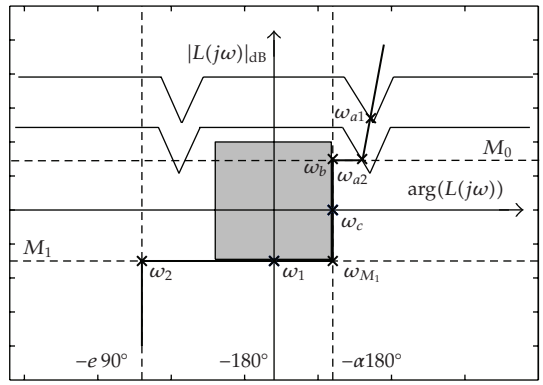

Figure 2: Nichols plot of $L_{\text{qto}}(j\omega)$, QFT theoretical optimal loop, exhibiting typical *Bode Step* in the frequency range $[\omega_{M1}, \omega_2]$.

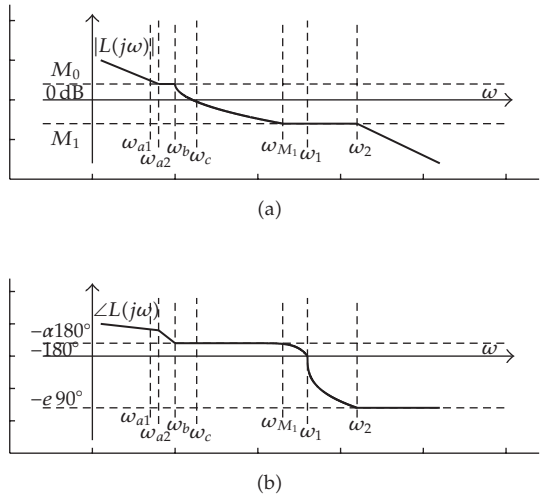

Figure 3: Bode plot of $L_{\text{qto}}(j\omega)$, QFT theoretical optimal loop, exhibiting typical *Bode Step* in the frequency range $[\omega_{M1}, \omega_2]$.

sides of the UHFB) and the specified excess of poles over zeros e. Note that ω_c is not directly established in the specifications, but indirectly, as a consequence of boundaries in **B**.

Note the shape of $L_{\text{qto}}(j\omega)$ in the frequency range $[\omega_{M1}, \omega_2]$; see Figures 2 and 3. It is called *Bode step* ([18, 23, 25], first defined in [25]), a frequency band where $L_{\text{qto}}(j\omega)$ has constant magnitude and a fast phase decrease (according to $\angle L_{\text{qto}}(j\omega)$ minimization objective, once the UHFB does not constrain this objective), surrounded by frequencies where $\angle L_{\text{qto}}(j\omega)$ is constant ($\alpha\pi$ and $-e\pi/2$, resp.) and $|L_{\text{qto}}(j\omega)|$ decreases. Bode step is very typical of QFT optimal loops, and when designing a suboptimal loop L_{sO}, the designer has to try to make L_{sO} exhibit a *Bode step-like* shape in order to succeed in accurately approaching L_{qto}. Note that a Bode step in $L_{\text{sO}}(j\omega)$ can be interpreted as $L_{\text{sO}}(j\omega)$ tightly following the bottom part of the UHFB.

3. QFT Automatic Controller Design by Evolutionary Optimization

The used method for automatic QFT controller design consists of using a fixed structure in the compensator with a certain number n of free parameters $x_i \in \mathbb{R}$, $i = 1 \cdots n$, which constitute the solution space $X \doteq \{x = (x_1, \ldots, x_n)\}$ of the nonconvex nonlinear global optimization to be performed by evolutionary algorithms. As it was said in the introduction, there are two critical factors which determine the efficiency of such an approach. One is the dimension of the search space, n, which exponentially increases the computational cost in terms of time. The second factor is the use of adequate heuristics which guide the evolutionary search towards close to optimum solutions.

The main contribution of this work has to do with the first factor: to use a fractional structure in the compensator is proposed as a key idea to get flexible structures, able to yield close to optimum solutions, but with a reduced n. Several structures from literature have been adapted to solve the QFT loop shaping problem, including some structures proposed by the authors in previous works. These structures are introduced in Sections 3.1.1–3.1.4.

The second contribution of this work has to do with the second factor. Some ad hoc heuristics, with features very much specific to the particular problem under consideration, have been developed in order to help evolutionary search to improve obtained solutions accuracy and computation time needed to obtain these solutions, specially in terms of local minima avoidance. These heuristics are presented in Section 3.2. Section 3.3 describes the algorithm used for evolutionary optimization, detailing the objective function, which includes these heuristics.

In Section 4 a design example is solved by using these heuristics and all the fractional structures. The results obtained by each fractional structure are compared.

3.1. Fractional Structures

3.1.1. TID

TID controller [26] is a modified version of PID controller, where the proportional term is replaced by a *tilted* (fractional) term, with transfer function s^{-e_T}, which permits a better approach to theoretical optimum. The resulting controller, including a low pass filter, is given by

$$C_{\mathrm{TID}}(s) = k\left(\frac{T}{s^{e_T}} + \frac{1}{s} + \frac{qDs}{q+s}\right)\frac{1}{(1+s/w_h)^{n_h}}. \tag{3.1}$$

3.1.2. $PI^{\lambda}D^{\mu}$

$\mathrm{PI}^{\lambda}\mathrm{D}^{\mu}$ controller [27, 28] is a PID generalization in which both integrator and derivative terms have real order, λ and μ, respectively. In this work, the multiplicative version of $\mathrm{PI}^{\lambda}\mathrm{D}^{\mu}$ given in [29] will be used, with transfer function

$$C_{\mathrm{PI}^{\lambda}\mathrm{D}^{\mu}}(s) = k_c x^{\mu}\left(\frac{\lambda_1 s + 1}{s}\right)^{\lambda}\left(\frac{\lambda_2 s + 1}{x\lambda_2 s + 1}\right)^{\mu}. \tag{3.2}$$

CRONE 2 (Section 3.1.3) is a particular case of (3.2), where some parameters are linked, and so flexibility is reduced compared to (3.2).

3.1.3. CRONE-Based Controllers

The CRONE approach [30, 31] defines three generations of fractional controllers based on the use of frequency-band noninteger differentiators. First and second generations (CRONE 1 and CRONE 2) use real non-integer differentiation, whereas third generation (CRONE 3) use complex non-integer differentiation. Three CRONE structures-based compensators are studied. The first one uses the transfer function structure common to CRONE 1 and 2. The basic component of this structure is an order n differentiator in the form of the implementable band-defined transfer function. An order n_I band-limited integrator and an order n_F lowpass filter are added to manage accuracy and robustness and control effort problems, being the open loop structure finally defined as

$$L_{\mathrm{CR2}}(s) = k\left(\frac{\omega_l}{s}+1\right)^{n_I}\left(\frac{1+s/\omega_h}{1+s/\omega_l}\right)^{n}\frac{1}{(s/\omega_h+1)^{n_F}}. \tag{3.3}$$

The application of the CRONE 2 structure to the QFT problem is quite straightforward.

The second compensator uses CRONE 3 structure, consisting of the substitution of the (real) order n integrodifferentiator in CRONE 2 for the real part $D_r(s)$ of a (complex) order $n = a + ib$ integrodifferentiator in the form of the implementable band-defined transfer function

$$D(s) = D_r(s) + iD_i(s) = \left(C_0\frac{1+s/\omega_h}{1+s/\omega_l}\right)^{a+ib} \tag{3.4}$$

with $C_0 = \omega_h/\omega_u = \omega_u/\omega_l$ and $\omega_u = \sqrt{\omega_l\omega_h}$. The open loop structure $L(s)$ in CRONE 3 is finally defined as

$$\begin{aligned} L_{\mathrm{CR3}}(s) = k\left(\frac{\omega_l}{s}+1\right)^{n_I}&\left(C_0\frac{1+s/\omega_h}{1+s/\omega_l}\right)^{a} \\ &\times\cos\left[b\ \mathrm{Log}\left(C_0\frac{1+s/\omega_h}{1+s/\omega_l}\right)\right]\frac{1}{(s/\omega_h+1)^{n_F}}. \end{aligned} \tag{3.5}$$

The third compensator is decoupled CRONE 3 ([32]), a modified version of CRONE 3 structure (3.5), where some parameters are decoupled in order to obtain higher flexibility. Frequencies ω_h and ω_l are decoupled by defining new frequencies ω'_h, ω'_l, and ω_{h4}. C_0 is decoupled by redefining C_0 in cos() as $C'_0 = cC_0$, where c is a new free parameter, $C_0 = \omega'_h/\omega_u = \omega_u/\omega'_l$, and $\omega_u = \sqrt{\omega'_l\omega'_h}$. The new structure to be shaped is

$$\begin{aligned} L_{\mathrm{decCR3}}(s) = k\left(\frac{\omega_l}{s}+1\right)^{n_I}&\left(C_0\frac{1+s/\omega_h}{1+s/\omega_l}\right)^{a} \\ &\times\cos\left[b\ \mathrm{Log}\left(cC_0\frac{1+s/\omega'_h}{1+s/\omega'_l}\right)\right]\frac{1}{(s/\omega_{h4}+1)^{n_F}}. \end{aligned} \tag{3.6}$$

For both CRONE 3 and decoupled CRONE 3, a detailed study of the parameters involved, its interrelations and their allowed ranks in order to obtain desired behavior, is developed in [32]. In [33] a CRONE-based structure is proposed for the design of open loops based on Bode optimum-like specifications.

3.1.4. FCT Terms

Fractional order Complex Terms (FCTs) [19, 34] are terms

$$T_i = \left(\frac{\omega_{ni}^2}{s^2 + 2\delta_i \omega_{ni} s + \omega_{ni}^2} \right)^{e_i} \tag{3.7}$$

with $e_i \in \mathbb{R}$, $e_i > 0$ corresponding to poles and $e_i > 0$ corresponding to zeros. Different combinations of these terms can be used depending on the problem to be solved. For a typical tracking problem with a minimum phase plant, the structure

$$L_0 = KT_1T_2T_3 \tag{3.8}$$

with $K \in \mathbb{R}^+$, two FCT poles and one FCT zero, achieves very good results.

3.2. Heuristics

Once the problem of a large number of parameters to be optimized has been solved by the use of fractional structures, the main problem is the fact that, due to the typically convex nature of the constraints in the solutions space, the evolutionary search can easily get stacked in local minima. The main goal for the heuristics design has been to help the evolutionary search to quickly avoid these local minima when they are detected.

The objective function used as the criterion for the natural selection of individuals during the evolutionary optimization process, $O(X)$, is a multiobjective real function which returns a to-be-minimized real value related with nonviolation of boundaries (constraint), stability (constraint), and minimization of K_{hf} (to-be-optimized value). Violation of boundaries and lack of stability are, in fact, constraints, but due to how evolutionary algorithms work, they have to be translated into to-be-optimized values, as K_{hf} originally is. A first approach could be to assign a large value to any solution which violates any boundary or is unstable. But this does not differentiate between solutions which violate a certain boundary completely, or only a little bit, which produces a blind $O(X)$ in the sense that is not able to look for the way to become out of the violation of a certain given constraint. In order to avoid the evolutionary optimization getting stack in such situations, it is necessary to guide it by establishing, by adequately shaping $O(X)$, that the violation of a certain constraint by a certain solution X_a is not as important as the violation produced by another solution X_b, so $O(X_a) < O(X_b)$. This way, the evolutionary algorithm will prefer the survival of X_a instead of X_b, so that better solutions survive, even if they do not satisfy constraints. This scheme, repeated generation after generation, leads to solutions that do respect constraints. These are some of the components of $O(X)$ intended to guide the evolutionary optimization towards solutions satisfying constraints.

(i) For open boundaries, let B_ω be an open boundary, and consider that it is defined as $B_\omega(p) : \mathbb{R} \to \mathbb{R}$, a function that assigns to every phase p in **NC** the magnitude of B_ω at that phase. For certain solutions X_a and X_b, it should happen $|L_{X_i}(j\omega)| \geq B_\omega(\angle L_{X_i}(j\omega))$, $i = \{a, b\}$. Assume that this constrain is not satisfied; that is, B_ω is violated, by both X_a and X_b. X_a is considered better than X_b when $L_{X_a}(j\omega)$ is closer to B_ω than L_{X_b} is, in terms of magnitude. More formally, the penalty for this

violation included in $O(X)$ is a monotonically increasing function of $B_\omega(\angle L_X(j\omega)) - |L_X(j\omega)|$. This choice, made generation after generation, contributes to get solutions X such that $L_X(j\omega)$ is over B_ω, thus satisfying B_ω constraint.

(ii) For closed boundaries, let B_ω be a closed boundary, and consider that it is defined by $Bi_\omega(p) : \mathbb{R} \to \mathbb{R}$, $i = u, l$, two functions that assign to every phase p in **NC** the upper and lower magnitudes of B_ω at that phase, respectively (bivalued boundary). For certain solutions X_a and X_b, it should happen $|L_{X_i}(j\omega)| \geq Bu_\omega(\angle L_{X_i}(j\omega))$ and $|L_{X_i}(j\omega)| \leq Bl_\omega(\angle L_{X_i}(j\omega))$, $i = \{a, b\}$. Assume that this constrain is not satisfied; that is, B_ω is violated, by both X_a and X_b. X_a is considered better than X_b when $L_{X_a}(j\omega)$ is closer to Bu_ω or Bl_ω than L_{X_b} is, in terms of **NC** Euclidean distance. This choice, made generation after generation, contributes to get solutions X such that $L_X(j\omega)$ is out of B_ω, thus satisfying B_ω constraint.

Another important consideration, in order to avoid the evolutionary search getting stacked, is an $O(X)$ which gives more importance to constraints satisfaction than to k_{hf} minimization. This way, the evolutionary search first searches for valid solutions, and then, once they have been achieved, it optimizes among them.

3.3. *Algorithm*

This section describes the optimization algorithm which has been implemented for controllers synthesis. It consists on the use of commercial evolutionary algorithm software (the *Genetic and evolutionary algorithm toolbox for use with MATLAB* (GEATbx, [35])) together with an ad hoc objective function, programmed by the authors, implementing the heuristics described in Section 3.2.

The evolutionary algorithm is, in particular, a multiple subpopulations evolutionary search algorithm. In this kind of evolutionary search, each subpopulation evolves in an isolated way for a few generations (as it happens in a single population evolutionary algorithm). After that, one or more individuals are exchanged between subpopulations. The way this process models species evolution is more similar to nature, compared to single population evolutionary algorithms, which helps to avoid local minima. The basic structure of this kind of algorithm is the following (adapted from [35]):

```
PROCEDURE Multipopulation_Evolutionary_Search_Algorithm (search_space)

BEGIN PROCEDURE

  INITIALIZATION, consisting on:
    creation of initial population // set of randomly chosen
                                   // points from search space,
                                   // grouped in species
    evaluation_of_individuals =
                      OBJECTIVE_FUNCTION(initial_population);
```

```
  WHILE NOT(any termination criteria is met)
  BEGIN WHILE

    NEW POPULATION GENERATION: // consisting on...
        fitness assignment selection;
        recombination;
        mutation;
        evaluation_of_offspring = OBJECTIVE_FUNCTION(offspring);
        reinsertion;
        migration;
        competition;

    END WHILE;

    RESULT = best individual in last population;

END PROCEDURE;
```

This is the code which implements the objective function $O(X)$ described in Section 3.2:

```
PROCEDURE Objective_Function (set_of_individuals)

BEGIN PROCEDURE

  FOR EACH individual IN set_of_individuals, indexed as i
    SBV = Compute_Stability_Boundary_Violation(individual);
    SRV = Compute_Stability_Ray_Violation(individual);
    PBV = Compute_Performance_Boundary_Violation(individual);
    NBV = Compute_Noise_Boundary_Violation(individual);

    PSP = Compute_Phases_Separation_Penalization(individual);
    MPP = Compute_Maximum_Phase_Penalization(individual);

    HFG = Compute_High_Frequency_Gain(individual);
```

```
    Constraints_Violation =
       BIG_VALUE * (weights[1] * SBV
                    + weights[2] * SRV
                    + weights[3] * PBV
                    + weights[4] * NBV);
    Technical_Penalizations =
       BIG_VALUE * (weights[5] * PSP
                    + weights[6] * MPP);
    Optimization_Criterion =
                    weights[7] * HFG;

    RESULT[i] = Constraints_Violation
                + Technical_Penalizations
                + Optimization_Criterion;
  END FOR;

END PROCEDURE;
```

Values *SBV*, *SRV*, *PBV* and *NBV* are related with a direct violation of any of the problem constraints. This is the reason they are amplified by *BIG_VALUE* so that any solution satisfying constraints is preferred (has a lower objective value) compared to any other which does not satisfy any of the constraints.

Values PSP and MPP are related with technical issues, which are necessary to consider in order to avoid *nonreal* solutions. Nonreal solutions are solutions that, due to intrinsic limitations of discrete computation, such as the need to consider a finite number of points to represent boundaries and loops, would satisfy the algorithm discrete translation of continuous constraints, but not the original continuous constraints. These values are also amplified by *BIG_VALUE*.

Value HFG is directly related with the optimization criterion, high-frequency gain minimization. It is not amplified by *BIG_VALUE*, and so it is only considered, in practice, when no constraint is violated.

Weights are conceived to give more importance to a certain penalization compared to others, but have not been used for the moment; that is, all of them are equal.

As explained in Section 3.2, *SBV*, *SRV*, *PBV*, and *NBV* values correspond to a gradient, related with how much a certain boundary is violated, so that the objective function distinguishes different degrees of violation of a certain constraint, so that it helps the evolutionary algorithm to approach solutions which do not violate that constraint. For instance, consider the following MATLAB code, corresponding to the implementation of the function *Compute Performance Boundary Violation* (CPBV).

```
PROCEDURE Compute_Perform_Boundary_Violation(loop_points,boundaries)

  loop_phases = loop_points.phases;
```

```
loop_magnitudes = loop_points.magnitudes;
bnd_phases = boundaries.phases;
bnd_magnitudes = boundaries.magnitudes;
nphases = length(bnd_phases);
bnd_phases_indexes =
        mod(round(loop_phases*(nphases/360)+nphases),nphases);
bnd_phases_indexes = nphases - bnd_phases_indexes;
minimum_allowed_magnitudes = bnd_magnitudes(bnd_phases_indexes);
magnitude_differences =
        loop_magnitudes - minimum_allowed_magnitudes;
negative_differences =
        magnitude_differences .* (magnitude_differences < 0);
squared_differences = negative_differences.^2;

RESULT = sum(squared_differences);

END PROCEDURE;
```

For each design frequency ω, this procedure checks whether the loop is below the performance boundary at that frequency, B_ω, and in that case quantifies how much below it is, which is called *magnitude difference* in the code. These magnitude differences are squared, so that bigger magnitude differences are considered much worse than small differences. After that, these values are summed up to obtain the final boundary violation indicator, so that this indicator doubles when the loop violates boundaries at two frequencies, compared to a single violation, is triple when there are three violating frequencies, and so on. For an example of CPBV function behaviour, consider Figures 4, 5, and 6, representing loops L_{ex1}, L_{ex2}, and L_{ex3}, respectively, corresponding to the first loop design in Section 4 (based on a PID controller). In Figure 4, performance boundaries are violated by L_{ex1} at design frequencies $w_1 = 0.1\,\text{rad/s}$ and $w_2 = 0.5\,\text{rad/s}$, by about 5 dB. The function yields a boundary violation indicator $\text{CPBV}(L_{ex1}) = 4.1e3$. Using *BIG_VALUE* $= 1e6$, this produces an objective function value around $O(L_{ex1}) = 4.1e9$, which is a big penalization. The evolutionary algorithm will prefer individuals (loops) with lower objective function values. For instance, in Figure 5 performance boundaries are violated only at design frequency $w_1 = 0.1\,\text{rad/s}$, by 1 or 2 dB. CPBV function yields a boundary violation index $\text{CPBV}(L_{ex2}) = 1.5e3$, which corresponds to an objective function value $O(L_{ex2}) = 1.5e9$, still a big penalization, but lower than $O(L_{ex1})$. This way, the evolutionary algorithm tends to move the loop, at each frequency $\omega \in \Omega$, out of $F(j\omega)$ (forbidden area) and towards $A(j\omega)$ (allowed area), producing a design like L_{ex3} in Figure 6, where no boundary is violated, and so $\text{CPBV}(L_{ex3}) = 0$ and $O(L_{ex3})$ is contributed only by the optimization criterion, with no penalization by constraints violation, yielding $O(L_{ex3}) = 152$.

Figure 7 shows the evolution, for an optimization of the loop structure shown in Figures 4, 5 and 6, of $O(L_i)$, where i is an index for each generation in the evolutionary

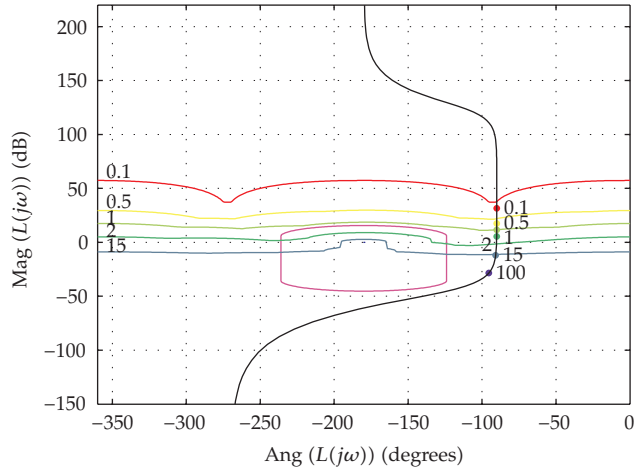

Figure 4: L_{ex1} loop, with CPBV(L_{ex1}) = 4.1e3 and $O(L_{ex1})$ = 4.1e9.

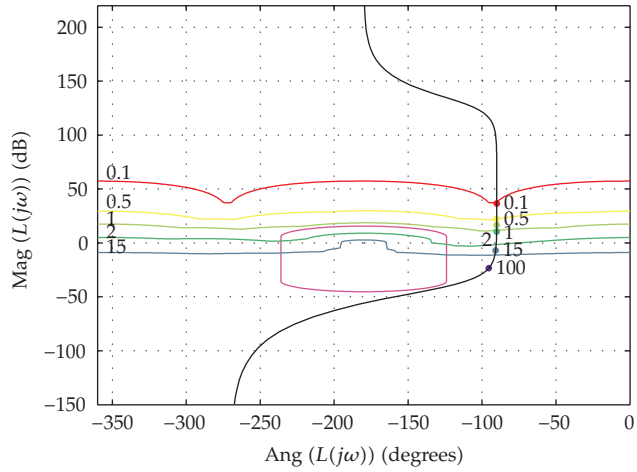

Figure 5: L_{ex2} loop, with CPBV(L_{ex2}) = 1.5e3 and $O(L_{ex2})$ = 1.5e9.

algorithm execution, and L_i is the best individual (loop) in generation i, best in the sense of minimizing the objective function. During the first three generations there are some boundary violations, which produce big objective function values. These values decrease as generation index i increases, due to the effect of the implemented heuristics. From fourth generation there is no boundary violation, so *BIG_VALUE* is not applied any more, and so from that point the task of the evolutionary algorithm is reduced to get better and better values for the optimization criterion. This process can be better visualized in Figure 8, where scale has been adapted for this purpose. This figure permits checking how the objective function converges to a certain value, in this case around 152, from a certain generation, in this case around generation 130.

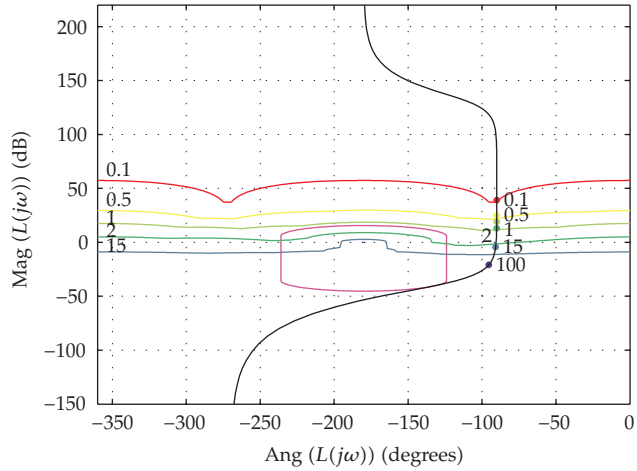

Figure 6: L_{ex3} loop, with CPBV(L_{ex3}) = 0 and $O(L_{ex3})$ = 152.

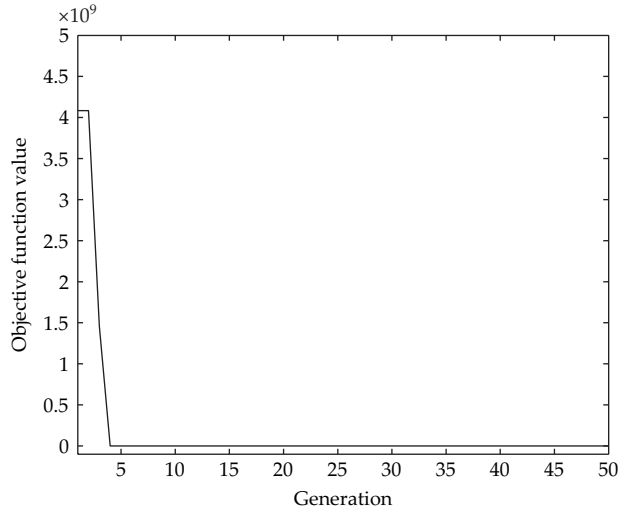

Figure 7: Evolution of $O(L_i)$ with generation index i, big scale, showing the effect of boundaries being violated at first generations.

4. Design Example

To illustrate the behavior of the proposed optimization method, the QFT Toolbox for MATLAB [8] Benchmark Example number 2 is used. It has also been used, for instance, in [16, 17]. In [16], two rational loops are designed, a second-order one, with K_{hf} = 136.6 dB, and a third-order one, with K_{hf} = 130 dB, with n_{pe} = 3 in both cases. A common n_{pe} = 3 will be used along this section, so that loops can be compared in terms of K_{hf}. In those structures which cannot get n_{pe} = 3 by themselves, a term

$$H(s) = \frac{1}{(1 + s/\omega_h)^{n_h}} \tag{4.1}$$

is added to fix n_{pe} = 3.

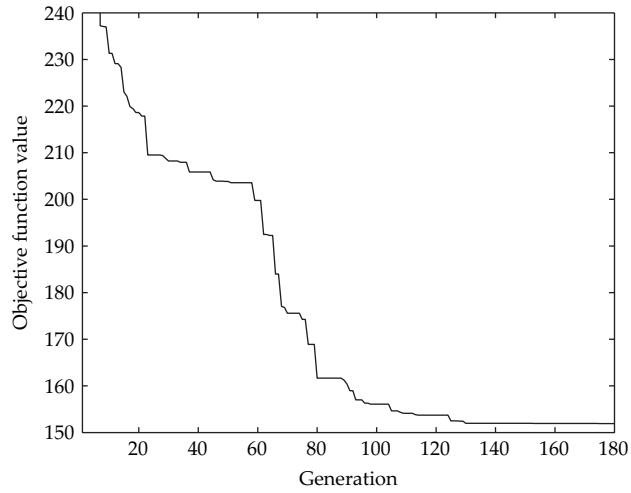

Figure 8: Evolution of $O(L_i)$ with generation index i, reduced scale, showing the minimization of optimization criterion, once constraints are satisfied.

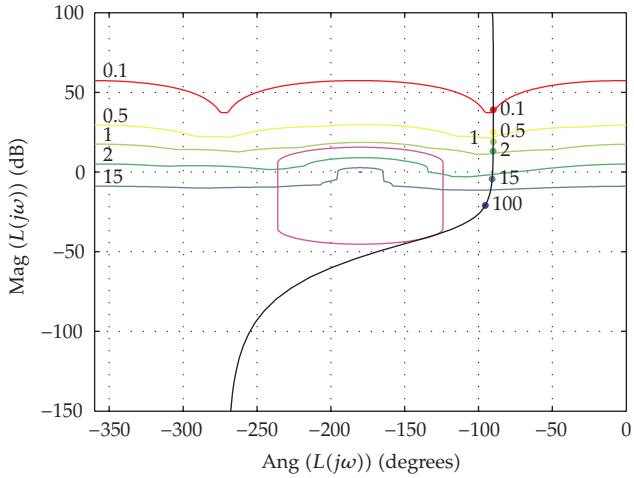

Figure 9: PID-based loop design.

For comparison purposes, a classical PID (TID with $e_T = 0$) based design (Figure 9) is also considered, with $K_{\text{hf}} = 152\,\text{dB}$ and parameters in (3.1): $k = 9.08 * 10^6$, $T = 9.9 * 10^5$, $q = 184.9$, $D = 1.97$, $w_h = 2091.3$, $n_h = 2$.

The result obtained with TID controller is shown in Figure 10, with $K_{\text{hf}} = 140\,\text{dB}$, improving PID. Parameters in (3.1) are $k = 3.86 * 10^{-5}$, $T = 4.4 * 10^5$, $e_T = 0.36$, $q = 5685.4$, $D = 0.77$, $w_h = 7625.2$, $n_h = 2$.

$\text{PI}^\lambda\text{D}^\mu$ improves TID result by another 12 dB, with $K_{\text{hf}} = 128\,\text{dB}$, Figure 11. Parameters in (3.2), with term (4.1), are $k_c = 2.79$, $x = 10^1\ 0$, $\mu = 0$, $\lambda = 0.38$, $\lambda_1 = 0.0083$, $\lambda_2 = 10^10$, $\omega_h = 951.87$, $n_h = 2$.

CRONE 2 structure yields the loop shown in Figure 12, with corresponding $K_{\text{hf}} = 129.5\,\text{dB}$, and parameters in (3.3), are $k = 2.98 * 10^6$, $\omega_l = 2588$, $n_I = 1.28$, $\omega_h = 1.65 * 10^7$, $n = 3.16$, $n_F = 3$. This result is slightly worse than $\text{PI}^\lambda\text{D}^\mu$'s, which could be expected, since

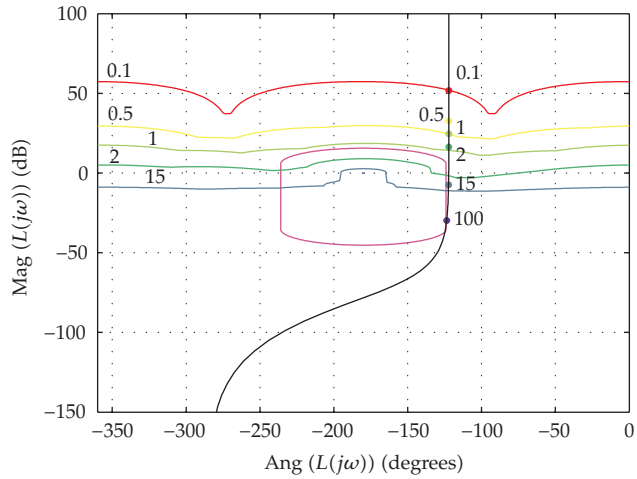

Figure 10: TID-based loop design.

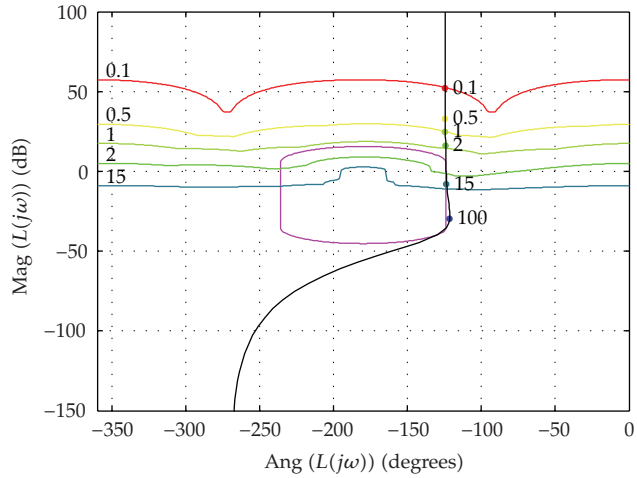

Figure 11: $PI^{\lambda}D^{\mu}$-based loop design.

CRONE 2 is the same structure as $PI^{\lambda}D^{\mu}$, but with some parameters linked, which implies less flexibility.

In Figure 13 it is shown a design using CRONE 3 structure, with $K_{\text{hf}} = 126.8\,\text{dB}$, and parameters in (3.5), are $k = 0.0084$, $\omega_l = 153$, $n_I = 1.4$, $C_0 = 2.3$, $\omega_h = 825$, $a = 0.85$, $b = 0.34$, $n_F = 3$. The result is slightly better than using CRONE 2.

In Figure 14 it is shown a design using decoupled CRONE 3 structure, with $K_{\text{hf}} = 105.3\,\text{dB}$, and parameters in (3.6), are $k = 1.46$, $\omega_l = 7$, $n_I = 1.07$, $C_0 = 11.2$, $\omega_h = 256.9$, $\omega_l = 7.1$, $a = 1.5$, $b = 0.45$, $c = 0.7$, $\omega'_h = 2 * 10^4$, $\omega'_l = 166$, $\omega_{h4} = 446$, $n_F = 3$. This result significantly improves the original CRONE 3 design (in more than 20 dB).

Finally, in Figure 15 it is shown the result obtained with the FCT structure (3.8), with $K_{\text{hf}} = 94.2\,\text{dB}$, which improves decoupled CRONE 3 result by more than 10 dB. Parameters for (3.8), are $K = 8.3$, $\omega_{n1} = 2.02$, $e_1 = 0.16$, $\delta_1 = 2.32$, $\omega_{n2} = 743.6$, $e_2 = 1.4$, $\delta_2 = 0.28$, $\omega_{n3} = 7145$, $e_3 = -0.57$, $\delta_3 = 2.55$.

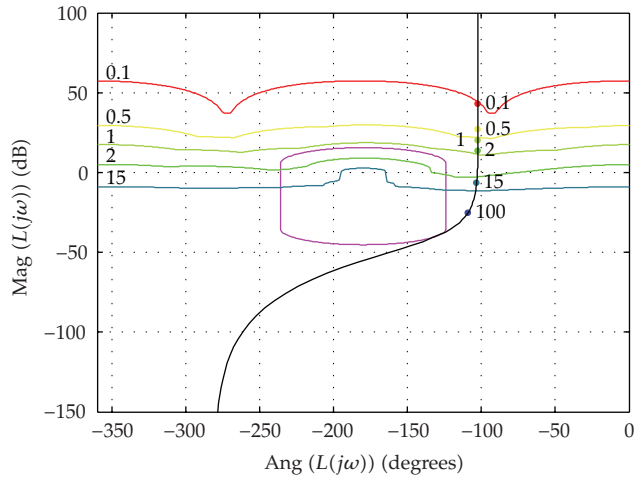

Figure 12: CRONE-2-based loop design.

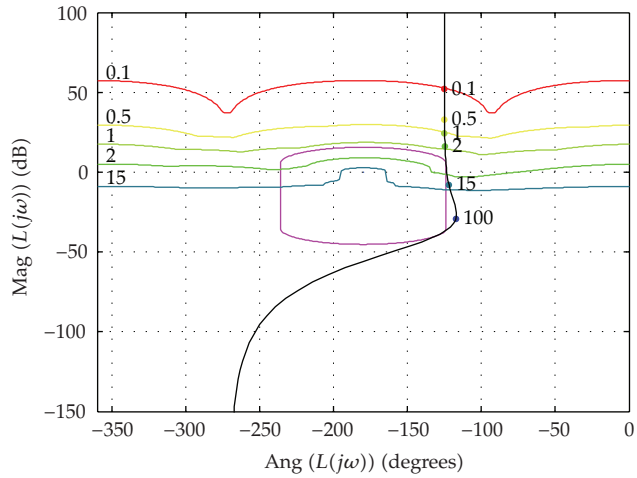

Figure 13: CRONE-3-based loop design.

Table 1: K_{hf} comparison.

CONTROLLER	K_{hf} (dB)
PID	152
TID	140
$\text{PI}^{\lambda}\text{D}^{\mu}$	128
CRONE 2	129.5
CRONE 3	126.8
decoupled CRONE 3	105.3
FCT	94.2

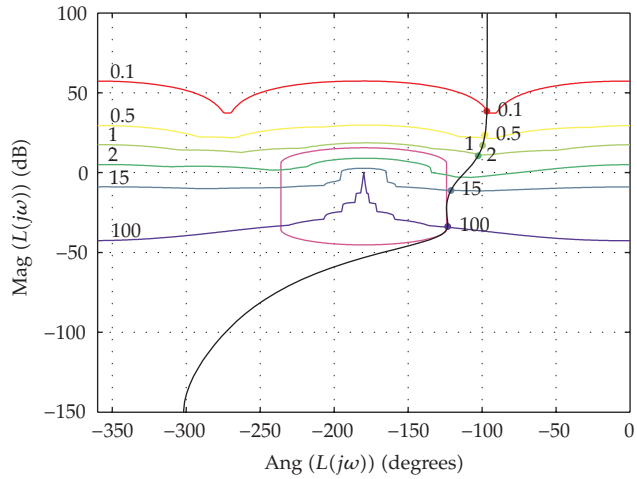

Figure 14: Decoupled CRONE-3-based loop design.

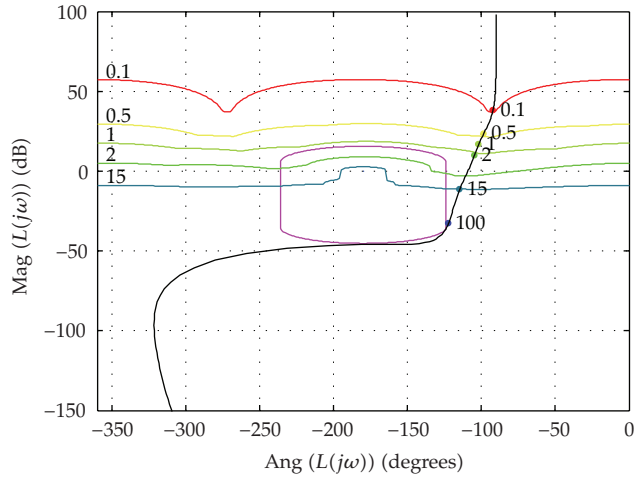

Figure 15: FCT-based loop design.

Figure 15 shows a comparison of the noise amplification at the plant input, $T_N(s) = -C/(1+L)$, achieved by each design. Table 1 summarizes the results obtained in terms of K_{hf}. As it can be easily checked, there is a direct correlation between K_{hf} and $T_N(s)$. Note how the most flexible the used structure is (and so the best K_{hf} it achieves) and the better Bode step-like shape its associated loop achieves.

5. Conclusions

An automatic QFT controller design procedure, based on evolutionary algorithms optimization on the parameters of a fixed structure, has been proposed. The key idea behind this proposal is the introduction of a structure with few parameters (a must in order to get good results from evolutionary optimization) but, at the same time, flexible enough, thanks to

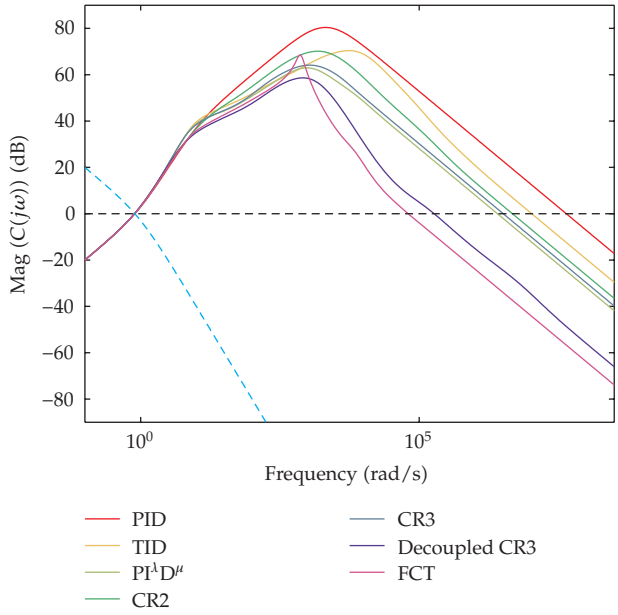

Figure 16: $T_N(s)$ comparison. From left to right, in terms of ω_{cg}, FCT, decoupled CRONE 3, $PI^{\lambda}D^{\mu}$, CRONE 3, CRONE 2, TID and PID.

its fractional nature, to get results which are close to the optimum. Fractional structures have been proposed as ideal candidates. Additional heuristics, focused on guiding the evolutionary search to prevent it from getting stacked in local minima, have been proposed. These structures and heuristics have achieved very good results in terms of QFT classical optimization criterion.

Acknowledgment

This work was partially supported by the Spanish Government DPI2007-66455-C02-01 project, which is greatly appreciated by the authors.

References

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, Wiley-Interscience, New York, NY, USA, 3rd edition, 2006.

[2] R. G. Parker and R. L. Rardin, *Discrete Optimization*, Academic Press, New York, NY, USA, 1989.

[3] R. B. Kearfott, "An interval branch and bound algorithm for bound constrained optimization problems," *Journal of Global Optimization*, vol. 2, no. 3, pp. 259–280, 1992.

[4] M. A. Wolfe, "Interval methods for global optimization," *Applied Mathematics and Computation*, vol. 75, no. 2-3, pp. 179–206, 1996.

[5] A. Zhigljavsky and A. Zilinskas, *Stochastic Global Optimization*, vol. 9, Springer, New York, NY, USA, 2008.

[6] W. M. Spears, K. A. De Jong, T. Bck, D. B. Fogel, and H. de Garis, "An overview of evolutionary computation," in *Proceedings of the European Conference on Machine Learning*, vol. 667, pp. 442–459, 1993.

[7] I. Horowitz, *Quantitative Feedback Design Theory (QFT)*, vol. 1, QFT Press, Boulder, Colo, USA, 1993.

[8] C. Borghesani, Y. Chait, and O. Yaniv, *Quantitative Feedback Theory Toolbox*, The MathWorks, Natick, Mass, USA, 1995.

[9] P. S. V. Nataraj and N. Kubal, "Automatic loop shaping in QFT using hybrid optimization and constraint propagation techniques," *International Journal of Robust and Nonlinear Control*, vol. 17, pp. 251–264, 2006.

[10] P. S. V. Nataraj and S. Tharewal, "An interval analysis algorithm for automated controller synthesis in QFT designs," in *Proceedings of the NSF Workshop on Reliable Engineering Computing*, Savannah, Ga, USA, 2004.

[11] A. Gera and I. Horowitz, "Optimization of the loop transfer function," *International Journal of Control*, vol. 31, no. 2, pp. 389–398, 1980.

[12] D. F. Thomson, *Optimal and sub-optimal loop shaping in quantitative feedback theory*, Ph.D. thesis, School of Mechanical Engineering, Purdue University, West Lafayette, Ind, USA, 1990.

[13] Y. Chait, Q. Chen, and C. V. Hollot, "Automatic loop-shaping of QFT controllers via linear programming," *Journal of Dynamic Systems, Measurement, and Control*, vol. 121, pp. 351–357, 1999.

[14] C. M. Fransson, B. Lennartson, T. Wik, K. Holmstrm, M. Saunders, and P. O. Gutman, "Global controller optimizacion using horowitz bounds," in *Proceedings of the 15th IFAC Trienial World Congress*, 2002.

[15] O. Yaniv and M. Nagurka, "Automatic loop shaping of structured controllers satisfying QFT performance," Tech. Rep., 2004.

[16] W. H. Chen, D. J. Ballance, and Y. Li, "Automatic loop-shaping in QFT using genetic algorithms," Tech. Rep., Centre for Systems and Control, University of Glasgow, Glasglow, UK, 1998.

[17] C. Raimúndez, A. Baños, and A. Barreiro, "QFT controller synthesis using evolutive strategies," in *Proceedings of the 5th International QFT Symposium on Quantitative Feedback Theory and Robust Frequency Domain Methods*, pp. 291–296, Pamplona, Spain, 2001.

[18] J. Cervera, *Ajuste automático de controladores en QFT mediante estructuras fraccionales*, Ph.D. thesis, Facultatea de Informática, Universidad de Murcia, Murcia, Spain, 2006.

[19] J. Cervera and A. Baños, "Automatic loop shaping in QFT by using a complex fractional order terms controller," in *Proceedings of the 7th QFT and Robust Domain Methods Symposium*, 2005.

[20] J. Cervera and A. Bañnos, "Automatic loop shaping in QFT by using crone structures," in *Proceedings of the 2nd IFAC Workshop on Fractional Differentiation and Its Applications*, IFAC, Porto, Portugal, 2006.

[21] J. Cervera, A. Baños, C. A. Monje, and B. M. Vinagre, "Tuning of fractional pid controllers by using QFT," in *Proceedings of the 32nd Annual Conference of the IEEE Industrial Electronics Society (IECON '06)*, IEEE, Paris, France, 2006.

[22] W. H. Chen and D. J. Ballance, "Stability analysis on the Nichols chart and its application in QFT," Tech. Rep., Centre for Systems and Control, University of Glasgow, Glasglow, UK, 1998.

[23] I. Horowitz, "Optimum loop transfer function in single-loop minimum-phase feedback systems," *International Journal of Control*, vol. 18, pp. 97–113, 1973.

[24] H. W. Bode, *Network Analysis and Feedback Amplifier Design*, Van Nostrand, New York, NY, USA, 1945.

[25] B. J. Lurie and P. J. Enright, *Classical Feedback Control with MATLAB*, Marcel Dekker, New York, NY, USA, 2000.

[26] B. J. Lurie, "Tunable tid controller," US patent no. 5371670, 1994.

[27] I. Podlubny, "Fractional-order systems and fractional-order controllers," in *UEF-03-94*, The Academy of Sciences Institute of Experimental Physics, Kosice, Slovakia, 1994.

[28] I. Podlubny, "Fractional-order systems and $PI^{\lambda}D^{\mu}$-controllers," *IEEE Transactions on Automatic Control*, vol. 44, no. 1, pp. 208–214, 1999.

[29] C. A. Monje, B. M. Vinagre, V. Feliu, and Y. Q. Chen, "On autotuning of fractional order $PI^{\lambda}D^{\mu}$ controllers," in *Proceedings of the 2nd FRAC Workshop on Fractional Differentiation and Its Application*, Porto, Portugal, 2006.

[30] A. Oustaloup, *La Commande CRONE*, Hermes, Paris, France, 1991.

[31] A. Oustaloup, J. Sabatier, and X. Moreau, "From fractal robustness to the crone approach," in *Proceedings of the Colloquium Fractional Differential Systems: Models, Methods and Applications (FDS '98)*, vol. 5, pp. 177–192, ESAIM, 1998.

[32] J. Cervera and A. Baños, "Automatic loop shaping in QFT using CRONE structures," *Journal of Vibration and Control*, vol. 14, no. 9-10, pp. 1513–1529, 2008.

[33] A. Baños, J. Cervera, P. Lanusse, and J. Sabatier, "Bode optimal loop shaping with CRONE compensators," in *Proceedings of the 14th IEEE Mediterranean Electrotechnical Conference*, Ajaccio, France, 2008.

[34] J. Cervera and A. Baños, "QFT loop shaping with fractional order complex pole-based terms," submitted to *International Journal of Robust and Nonlinear Control*.

[35] H. Pohlheim, "Geatbx documentation," 2004, http://www.geatbx.com/.