Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

# Accelerated QFT interval automatic loop shaping algorithm

Isaac Martínez-Forte | Joaquín Cervera*

[1]Dpto. de Informática y Sistemas, Universidad de Murcia, Murcia, Spain

**Correspondence**
*Joaquín Cervera, Facultad de Informática, Campus de Espinardo, 30100 Murcia, Spain.
Email: jcervera@um.es

**Summary**

The quest for an efficient QFT Automatic Loop Shaping algorithm is an open research problem. Different approaches have been adopted in the literature to efficiently solve this optimization problem, which is computationally hard due to its nonlinear and non-convex nature. The first algorithms focused on different forms of simplification of the original problem, leading to faster but not accurate results. Stochastic algorithms have been another popular way of dealing with the complexity of this problem. These algorithms are faster than an exhaustive search, but do not guarantee, in general, the globally optimal solution. A third, more recent, approach, consists of using interval analysis global search algorithms, which are able to accurately solve the original problem, and are very suitable for execution speed optimization. In this work, one of these algorithms is taken as a basis to propose and develop an improved algorithm which is more efficient in terms of the execution time needed to solve a given problem. This improved efficiency is achieved by using two new information sources: Phase information and feasible boxes information. The main contribution of this work is to propose the use of these two new information sources to reduce execution time and to integrate this idea in the original algorithm. Their individual and joint associated speedups are measured by solving two classical QFT example problems: Matlab® QFT Toolbox example 2 and ACC '90 benchmark problem. The results show that the new algorithm's performance is significantly better.

**KEYWORDS:**
Control design, uncertain systems, robust control, Quantitative Feedback Theory, automatic loop shaping, interval optimization, CACSD

## 1 | INTRODUCTION

Quantitative Feedback Theory (QFT) is a robust frequency domain control design technique, successfully applied in different domains ([1],[2],[3],[4]). *Loop shaping* of the nominal open loop gain function $L_0(s)$ is a key design step. It is performed considering a set of restrictions (*boundaries*) resulting from design specifications and the (uncertain) model of the plant (template), plus the QFT traditional design objective of minimizing the high-frequency gain ($K_{hf}$) ([5]). Loop shaping constitutes the QFT optimization problem: design of $L_0(s)$ satisfying boundaries (constraints) and minimizing $K_{hf}$.

In QFT origins, Loop shaping was performed by hand. The subsequent use of Computer Aided Control System Design tools (e.g., the classical Matlab® QFT Toolbox[6] or the more modern QFT Control Toolbox for Matlab® – QFTCT[7]) made manual loop shaping much simpler, but still a certain degree of expertise and some time is needed. Thus, the problem of automatic loop shaping (QFT ALS) is of enormous practical interest and has been addressed from various viewpoints, especially in the last

four decades. Optimal loop computation is a nonlinear and non-convex optimization problem for which it is difficult to find a computationally efficient solution. Various approaches to solve it have been adopted in the literature.

The first approach, four decades ago, was to consider a simplified version of the problem, and it yielded conservative results. Some examples are:[8], where the problem is convexified by using rectangular templates,[9], which uses linear approximations of boundaries, or[10], where the problem in modeled in terms of LMIs. Another way of simplifying the problem is to consider particular controller structures with a certain (limited) degree of freedom, e.g.[11,12,13].

A second, more recent, approach, has been the use of stochastic optimization algorithms, which are able to directly handle nonlinear and non-convex optimization problems. For instance, evolutionary algorithms are used in[14,15,16]. The main drawback of these algorithms is that they do not guarantee, in general, an optimal solution, including the possibility of not finding any solution or finding a local optimum. Furthermore, their computational burden is significantly higher as the number of controller parameters increases. An interesting approach to reduce this number is the use of fractional structures, e.g.[17,18,19].

The third approach is the use of interval arithmetic based search in order to ensure that the whole space of solutions is explored. Thus, for a certain QFT problem, finding the global optimum is guaranteed if there is any solution; and, if not, the algorithm will be able to state it. These features can be very important. For example, if no solution is found the designer knows that the specifications should be relaxed. Another framework in which accurately obtaining the global optimum is essential is the comparison of different controller structures, as performed in[17,18,19]. This exhaustive search is computationally slow, with execution times depending exponentially on the number of controller parameters. But the structure of these algorithms make them very suitable for execution time acceleration. The first algorithm of this kind was introduced in[20] (algorithm *NT*). In[21] (algorithm *NK*) *NT* acceleration is achieved by combining geometric constraints (GCP), applied to all controller parameters, with a hybrid optimization scheme able to launch local searches from certain points of the global search. Algorithm *NK* obtains better results but is still slow when the number of controller parameters grow.

Based on these ideas, several algorithms using an ICSP (Interval Constraint Satisfaction Problem) scheme have been developed ([22],[23],[24]). This approach is sophisticated but computationally too intense. In[25] this approach is accelerated by an interval consistency technique. In these proposals there is an equation to be solved at each search iteration for each template representative considered, so increasing accuracy in uncertainty modeling results in slower computation times.

The aim of this work is to start from algorithms *NT* and *NK*, both simple, general, and where arbitrary template accuracy can be obtained without affecting execution performance and to develop a faster algorithm which is able to solve the same QFT problems in a fraction of *NK* execution time. The ultimate goal would be to compute controllers with tens of parameters in a few seconds, allowing the designer to fluently interact with the computer aided control design tool: for a certain problem, the designer could establish a set of specs, a controller structure, and immediately obtain the results, so that specs or controller structure could be interactively modified accordingly. This work is a step towards that goal.

This work is structured as follows. In section 2 some basic material about QFT, interval arithmetics and interval global search, is presented, including algorithms *NT* and *NK*. Section 3 presents two speedup proposals, together with a new algorithm, *MC*, corresponding to their application to the *NK* algorithm. In section 4 the *MC* algorithm is applied to a couple of traditional QFT example problems, and its execution time is analyzed and compared to *NT*'s and *NK*'s. Finally, in section 5 the results obtained are discussed and some potential future directions are suggested.

## 2 | PRELIMINARIES

### 2.1 | Quantitative Feedback Theory

QFT ([1]) focuses on an accurate plant uncertainty model and its qualitative implications in terms of the control effort needed to cope with this uncertainty. The usual QFT control system configuration (Fig. 1) has 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, $F(s)$, devoted to satisfy tracking specifications.

Let P be an uncertain plant defined as a set of transfer functions $\mathrm{P} \doteq \{P(s)\}$ representing uncertainty, with $P_0(s) \in \mathrm{P}$ the (arbitrarily chosen) nominal plant. For a certain frequency $\omega$, the *template* $T_\mathrm{P}(\omega)$ is defined as the Nichols Chart (NC) region

$$T_\mathrm{P}(\omega) \doteq \{(\angle P(j\omega), |P(j\omega)|_{dB}) \in \mathrm{NC}, P \in \mathrm{P}\}, \tag{1}$$

with $\mathrm{NC} = \{(\angle p, |p|_{dB}), p \in \mathbb{C}\}$. The open loop transfer function is defined as $L(s) \doteq P(s)C(s)$ and the nominal open loop transfer function as $L_0(s) \doteq P_0(s)C(s)$ .

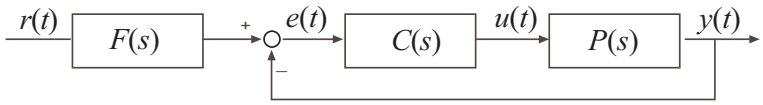


**FIGURE 1** Two degrees of freedom control system configuration.

The controller $C(s)$ is designed in the Nichols Chart by shaping $L_0(s)$. The designer establishes a discrete set $\Omega$ of design frequencies and a set $S_\Omega$ of quantitative specifications on robust stability and robust performance on the closed loop system. Typically $S_\Omega$ includes the following specs:

- Robust stability[1]:
$$\left|\frac{L(j\omega)}{1+L(j\omega)}\right| \leq \lambda, \forall\omega \in \Omega \tag{2}$$
It is usually convenient to use a *Universal High Frequency Boundary* (UHFB), a special robust stability boundary called *universal* because the region inside it should be avoided by $L_0(j\omega)\ \forall\omega \in \mathbb{R}^+$.

- Robust tracking performance:
$$\alpha(\omega) \leq \left|\frac{L(j\omega)}{1+L(j\omega)}F(j\omega)\right| \leq \beta(\omega)\ , \forall\omega \in \Omega \tag{3}$$

- Robust output disturbance rejection performance:
$$\left|\frac{1}{1+L(j\omega)}\right| \leq \delta_o(\omega)\ , \forall\omega \in \Omega \tag{4}$$

- Robust input disturbance rejection performance:
$$\left|\frac{P(j\omega)}{1+L(j\omega)}\right| \leq \delta_i(\omega)\ , \forall\omega \in \Omega \tag{5}$$

where $\alpha$, $\beta$, $\delta_o$ and $\delta_i$ are chosen by the designer.

Based on $S_\Omega$ and the set of templates $T_\Omega \doteq \{T_P(\omega), \omega \in \Omega\}$, a set of *boundaries* $B_\Omega = \{B_\omega, \omega \in \Omega\}$ is computed. Each $B_\omega$ is a curve in NC which separates allowed ($A_\omega$) and forbidden ($F_\omega$) regions for $L_0(j\omega)$. $L_0(j\omega) \in A_\omega \forall\omega \in \Omega$ (*boundaries satisfaction*) implies specification satisfaction by $L(s)\ \forall P(s) \in P$. Conversely, $\exists\omega \in \Omega \mid L_0(j\omega) \in F_\omega$ (*boundaries violation*) implies specifications are not met.

On Fig. 5 a typical example of $B_\Omega$ can be observed, where both *open* and *closed* boundaries are present [2]:

- *Open boundaries*: The open lines horizontally crossing NC from $-360^o$ to $0^o$, in this example (tracking) performance boundaries. For each $\omega \in \Omega = \{0.1, 0.2, 1, 2, 15\}$, $A_\omega$ is defined as the allowed region, above $B_\omega$, and $F_\omega$ as the forbidden region, below.

- *Closed boundaries*: The closed line around (0 dB, $-180^o$) is a robust stability boundary, more precisely, an UHFB, defining $A_\omega$ as the region outside $B_\omega$, and $F_\omega$ as the region inside, $\forall\omega \in \mathbb{R}^+$.

*Loop shaping*, the main QFT design stage, consists of a constrained optimization problem: Design of $L_0(s)$ satisfying boundaries (constraints) and minimizing
$$K_{hf} \doteq \lim_{s\to\infty} s^{pe} L_0(s), \tag{6}$$
where *pe* is $L_0$ excess of poles over zeros. Note this optimization criterion focuses on reduction of sensor noise at high frequencies. However, its application can result in controllers that are worse regarding other criteria. Hence, the designer should choose the most adequate criterion depending on the specific problem characteristics. The algorithm proposed in this work uses $K_{hf}$ minimization as optimization criterion. Adapting this algorithm to other criteria could be considered elsewhere.

[1] Note this definition of robust stability, based on the complementary sensitivity function, common in some QFT literature, is adopted here. But robust stability margin, defined in terms of the distance between $L$ and the critical point $(-180^o, 0_{dB}) \in$ NC is described by the sensitivity function as $\left|\frac{1}{1+L(j\omega)}\right| \leq \lambda$.

[2] Note both open and closed NC boundaries are closed curves in the arithmetic complex plane (Nyquist diagram).

Since $P_0(s)$ is fixed, $L_0(s)$ design is equivalent to $C(s)$ design. A rational controller general structure can be expressed as

$$NLC(s,x) = k\frac{\prod_{i=1}^{n_{rz}}(s+z_i)\prod_{j=1}^{n_{cz}}(s^2+2\alpha_j v_j s+v_j^2)}{\prod_{l=1}^{n_{rp}}(s+p_l)\prod_{m=1}^{n_{cp}}(s^2+2\beta_m u_m s+u_m^2)} \tag{7}$$

where

$$x = (k, z_1, \dots, z_{n_{rz}}, \alpha_1, \dots, \alpha_{n_{cz}}, v_1, \dots, v_{n_{cz}}, p_1, \dots, p_{n_{rp}}, \beta_1, \dots, \beta_{n_{cp}}, u_1, \dots, u_{n_{cp}}) \tag{8}$$

is a vector containing the $n$ controller parameters, $n = 1 + n_{rz} + 2n_{cz} + n_{rp} + 2n_{cp}$. In the particular way of expressing $C(s)$ given by (7), it happens $k = K_{hf}$, which, without loss of generality, is quite convenient for optimization, because it simplifies objective function computation. A particular pole-zero structure (values for $n_{rz}$, $n_{cz}$, $n_{rp}$ and $n_{cp}$) is a priori chosen by the designer. The optimization variables are the controller parameters $x$ for that particular structure, and $L_0$ can thus be expressed as

$$L_0(s,x) = P_0(s)C(s,x). \tag{9}$$

## 2.2 | Interval arithmetics

For the following basic interval arithmetics definitions and properties,[26] and[27] are used; further details can be obtained there. The notation has been slightly adapted:

- An *interval number* $X$ is defined as the real interval $[\underline{X}, \overline{X}] = \{x | \underline{X} \leq x \leq \overline{X}\}$.
- A real number $x$ is equivalent to a zero width interval $X = [x, x]$, called *degenerate interval*.
- An *n-dimensional interval vector* $\mathbf{x}$, or *n-dimensional box*, is an ordered column $n$-tuple of intervals: $\mathbf{x} = (X_1, \dots, X_n)^T$.
- These operators are defined:
  - $\sup(X) \doteq \overline{X}$ and $\sup(\mathbf{x}) \doteq (\overline{X}_1, \dots, \overline{X}_n)^T$
  - $\inf(X) \doteq \underline{X}$ and $\inf(\mathbf{x}) \doteq (\underline{X}_1, \dots, \underline{X}_n)^T$
  - $\text{subs}(\mathbf{x}, j, S) \doteq (X_1, \dots, X_{j-1}, S, X_{j+1}, \dots, X_n)^T$ (It returns a new version of $\mathbf{x}$ where $X_j$ has been substituted by $S$.)
- An *interval function* $F$ is an interval-valued function of one or more interval arguments $X_1, \dots, X_n$, i.e., it maps the value of one or more interval arguments onto an interval. This can also be regarded as mapping a box argument $\mathbf{x} = (X_1, \dots, X_n)^T$ onto an interval: $F(\mathbf{x}) = F((X_1, \dots, X_n)^T)$. For convenience and readability, both interpretations will be interchangeably used, so that $F(\mathbf{x}) = F((X_1, \dots, X_n)^T) = F(X_1, \dots, X_n)$, written preferably as $F(\mathbf{x})$ or $F(X_1, \dots, X_n)$.
- In the following definitions, let $f(x_1, \dots, x_n)$ be a real multivariate function with real codomain, i.e., $f : \mathbb{R}^n \to \mathbb{R}$.
- An *interval extension* of $f$ is an interval function $F^I(\mathbf{x})$ such that $F^I(x_1, \dots, x_n) = f(x_1, \dots, x_n), \forall x_1, \dots, x_n$, i.e., if the arguments of $F^I$ are degenerate intervals, then $F^I(x_1, \dots, x_n)$ is a degenerate interval equal to $f(x_1, \dots, x_n)$.
- The *natural interval extension* (NIE) of $f$, $F^{\text{NIE}}$, is an interval extension obtained by replacing, in the expression for $f(x_1, \dots, x_n)$, every real variable $x_i$, $i = 1, \dots, n$, with a corresponding interval variable $X_i$, and every real operator or function with corresponding interval operations and functions.
- An interval function $F$ is *inclusion isotonic* if, for any pair of boxes $\mathbf{x}$ and $\mathbf{y}$,

$$\mathbf{x} \subseteq \mathbf{y} \;\Rightarrow\; F(\mathbf{x}) \subseteq F(\mathbf{y}). \tag{10}$$

- The *range* of a function $f : \mathbb{R}^n \to \mathbb{R}$ *over a box* $\mathbf{x}$ is defined as the real numbers set $range(f, \mathbf{x}) \doteq \{f(x) | x \in \mathbf{x}\}$. $range(f, \mathbf{x})$ can also be expressed as $f(\mathbf{x})$ for short.
- The *Fundamental Theorem of Interval Analysis* ([26]) establishes that, if $F$ is an *inclusion isotonic* interval extension of $f : \mathbb{R}^n \to \mathbb{R}$, then

$$range(f, \mathbf{x}) \subseteq F(\mathbf{x}). \tag{11}$$

- With the NIE being an inclusion isotonic interval extension, by direct application of (11), for any $f : \mathbb{R}^n \to \mathbb{R}$,

$$range(f, \mathbf{x}) \subseteq F^{\text{NIE}}(\mathbf{x}), \tag{12}$$

  which constitutes the base for algorithms *NT* and *NK*.

## 2.3 | Interval global search

A solution for QFT ALS optimization problem consists of a $n$-tuple of values instantiating the $n$ controller parameters in $x$ (8), such that $C(s, x)$ (7) is instantiated to a particular $C(s)$. Since it is a nonlinear and non-convex problem, in general a global exhaustive search in the solutions space (a certain box $\mathbf{x} \subset \mathbb{R}^n$) is required in order to guarantee an optimal result, or to otherwise be able to state no solution exists. Since an exhaustive search in this infinite space is not possible, a possible approach could be to discretize $\mathbf{x}$, for instance choosing $e$ uniformly distributed points in each $\mathbf{x}$ dimension. This idea has two main drawbacks: In general, a) how far the solution obtained is from the real optimum is not known; and, b) the fact that no solution is obtained does not necessarily mean no solution exists.

### 2.3.1 | Algorithm *NT*

Algorithm *NT* ([20]), and its improved version *NK* ([21]), introduce, for the first time, the use of an interval global branch and bound search ([27],[28]) to solve this problem. The basic idea is to use the inclusion isotonicity property (12) in the following sense. Assume:

- a QFT ALS problem is given by specifying a plant P, a set of design frequencies $\Omega$, a set of specifications $\mathrm{S}_\Omega$, and a controller structure $(n_{rz}, n_{cz}, n_{rp}, n_{cp})$;
- corresponding $\mathrm{T}_\Omega$ and $\mathrm{B}_\Omega$ are computed;
- an initial search box of controller parameter values $\mathbf{x}$ is given [3];
- expressions $\mathrm{mag}(\mathbf{x}, \omega)$ and $\mathrm{ang}(\mathbf{x}, \omega)$, natural interval extensions of functions $|L_0(j\omega, x)|$ and $\angle L_0(j\omega, x)$, magnitude and angle of $L_0(j\omega, x)$ (9), respectively, are given.

Application of $\mathrm{mag}(\mathbf{x}, \omega)$ and $\mathrm{ang}(\mathbf{x}, \omega)$ to $\mathbf{x}$ at a certain frequency $\omega$ yields $M$ and $H$, magnitude and phase intervals, respectively, which define a *rectangle* (two-dimensional box in NC) $\mathbf{R}_{\mathbf{x}\omega}$ containing, as (12) guarantees, all the possible values of $L_0(j\omega, x) \forall x \in \mathbf{x}$:[4]

$$\mathbf{R}_{\mathbf{x}\omega} \doteq (M, H) \leftarrow (\mathrm{mag}(\mathbf{x}, \omega), \mathrm{ang}(\mathbf{x}, \omega)) \tag{13}$$

Depending on the relation between $\mathbf{R}_{\mathbf{x}\omega}$ and boundaries $B_\omega$, three situations can happen regarding $\mathbf{x}$:

- $\mathbf{R}_{\mathbf{x}\omega} \subset \mathrm{A}_\omega$, meaning $\mathbf{x}$ is *feasible* for $\omega$, i.e., $L_0(j\omega, x)$ respects boundaries $B_\omega \forall x \in \mathbf{x}$.
- $\mathbf{R}_{\mathbf{x}\omega} \subset \mathrm{F}_\omega$, meaning $\mathbf{x}$ is *infeasible* for $\omega$, i.e., $L_0(j\omega, x)$ violates boundaries $B_\omega \forall x \in \mathbf{x}$.
- Otherwise $\mathbf{x}$ is *ambiguous* for $\omega$, i.e., some part of $\mathbf{x}$ is feasible and some is infeasible.

Given a box $\mathbf{x}$ and frequency $\omega$, their associated

- *Infeasible rectangle*, $\mathbf{iR}_{\mathbf{x}\omega}$, is defined as the largest [5] infeasible rectangle contained in $\mathbf{R}_{\mathbf{x}\omega}$.
- *Feasible rectangle*, $\mathbf{fR}_{\mathbf{x}\omega}$, is defined as the largest feasible rectangle contained in $\mathbf{R}_{\mathbf{x}\omega}$.

Fig. 3 shows some examples of rectangles, infeasible rectangles and feasible rectangles for different boxes and frequencies.

The concept of feasibility can be generalized to: $\mathbf{x}$ is *feasible* if it is feasible $\forall \omega \in \Omega$ and *infeasible* if it is infeasible for any $\omega \in \Omega$. Otherwise it is *ambiguous*. A feasibility test procedure (FT) can be created to check the feasibility of a given box $\mathbf{x}$: It receives as arguments $\mathbf{R}_{\mathbf{x}\Omega} \doteq \{\mathbf{R}_{\mathbf{x}\omega} | \omega \in \Omega\}$ (13) and $\mathrm{B}_\Omega$, and returns $flag_{\mathbf{x}}$, a label with value *infeasible*, *feasible*, or *ambiguous*. If $\mathbf{x}$ is infeasible, no solution exists and the algorithm ends. If it is feasible, any $x \in \mathbf{x}$ is a solution, and any $x \in \mathbf{x}$ such that $x[1] = \inf(\mathbf{x}[1])$ (any $x$ with minimum $K_{hf}$) can be used as solution. If $\mathbf{x}$ is ambiguous, then it should be split into smaller pieces (subboxes) to individually check their feasibility until a suitable solution (feasible or ambiguous with a designer prescribed $\epsilon$ accuracy) is reached. This is precisely how *NT* algorithm proceeds. Each parameters subbox $\mathbf{z}$ is a node in an implicit search tree, implemented by means of a to-be-processed nodes list *NL*. *NL* elements (nodes) are triples $(\mathbf{z}, z, flag_{\mathbf{z}})$, being $\mathbf{z}$ the parameter box itself, $\mathbf{z}$, $z$ the minimum of the objective function over $\mathbf{z}$, i.e., $z = \inf(\mathbf{z}[1])$, and $flag_{\mathbf{z}}$ a flag representing $\mathbf{z}$ feasibility.

Its pseudocode is listed as **Algorithm 1** (page 6). Note this kind of search tree typically leads to exponential order execution times with respect to the number of controller parameters.

[3] Note parameters in $\mathbf{x}$ are restricted to be greater than zero for an stable and minimum phase controller and to guarantee mag and ang images are finite intervals.
[4] Note $\mathbf{R}_{\mathbf{x}\omega}$ is an overbounding of the set $\{L_0(j\omega, x), x \in \mathbf{x}\}$.
[5] Largest NC area, using dB and degrees as units for magnitud and phase, respectively.

**Algorithm 1** Pseudocode for *NT* algorithm

Inputs: $B_\Omega$ and $\mathbf{x}$, initial search box.
Output: $\mathbf{z}$, box of computed optimal values of controller parameters, or message "No feasible solution in $\mathbf{x}$".

1. Check feasibility of the initial search box:
   (a) $\mathbf{z} \leftarrow \mathbf{x}$ (initialize *current box* to initial box).
   (b) $\mathbf{R}_{\mathbf{z}\Omega} \leftarrow (\text{mag}(\mathbf{z}, \omega), \text{ang}(\mathbf{z}, \omega)), \forall \omega \in \Omega$.
   (c) $flag_{\mathbf{z}} \leftarrow \text{FT}(\mathbf{R}_{\mathbf{z}\Omega}, B_\Omega)$.
   (d) IF $flag_{\mathbf{z}} = infeasible$, PRINT "No feasible solution in $\mathbf{z}$" and EXIT.
2. Initialize the list *NL*:
   (a) $z \leftarrow \inf(\mathbf{z}[1])$ ($z$ initialization).
   (b) $NL \leftarrow \{(\mathbf{z}, z, flag_{\mathbf{z}})\}$.
3. IF $flag_{\mathbf{z}} = feasible$, PRINT "Optimal controller parameter box is $\mathbf{z}$" and EXIT.
4. Bisect $\mathbf{z}$ along its maximum width coordinate direction $d$ into two subboxes $\mathbf{v}^1$ and $\mathbf{v}^2$ such that $\mathbf{z} = \mathbf{v}^1 \cup \mathbf{v}^2$.
5. Do the feasibility test for the new subboxes and discard any infeasible ones, i.e., for $l = 1, 2$:
   (a) $\mathbf{R}_{\mathbf{v}^l\Omega} \leftarrow (\text{mag}(\mathbf{v}^l, \omega), \text{ang}(\mathbf{v}^l, \omega)), \forall \omega \in \Omega$
   (b) $flag_{\mathbf{v}^l} \leftarrow \text{FT}(\mathbf{R}_{\mathbf{v}^l\Omega}, B_\Omega)$
   (c) $v^l \leftarrow \inf(\mathbf{v}^l[1])$
   (d) Form the triple $(\mathbf{v}^l, v^l, flag_{\mathbf{v}^l})$.
   (e) IF $flag_{\mathbf{v}^l} = infeasible$, discard $(\mathbf{v}^l, v^l, flag^l_{\mathbf{v}})$.
6. Sort the list *NL*
   (a) $NL \leftarrow NL - \{(\mathbf{z}, z, flag_{\mathbf{z}})\}$ .
   (b) Add any remaining triple(s) from step 5.(e) to *NL*.
   (c) IF *NL* is empty, there is no feasible solution in $\mathbf{x}$, so PRINT "No feasible solution in $\mathbf{x}$" and EXIT.
   (d) Sort *NL* such that the second members of all triples in *NL* do not decrease in value.
   (e) Denote the first item of list *NL* as $(\mathbf{z}, z, flag_{\mathbf{z}})$.
7. Go to step 3.

### 2.3.2 | Algorithm *NK*

In order to improve the speed of the *NT* algorithm, the *NK* algorithm was developed. It is very similar to the *NT* algorithm except for some small details and especially because of the addition of two features: a) A parameter box reduction, called *Quick Solution* (*QS*) algorithm, executed for every new box at its creation, and b) a *Local Optimization* (*LO*) algorithm, executed only sometimes, according to a heuristic decision rule (executing *LO* for every iteration would slow down algorithm *NK*, see ([21]) for details). Note that the use of this heuristic rule does not imply discarding any solution, it only affects algorithm acceleration, so finding the global optimum is still guaranteed. *LO* addition converts the *NK* algorithm into a hybrid optimization algorithm, i.e., one that combines global and local search. This work focuses on the first addition, *QS*, described in section 2.3.3, since the authors found its improvement to be the most promising way to speedup the *NK* algorithm. Since *NK* is very similar to *NT*, instead of listing *NK* completely, which would be highly redundant, the following approach is adopted in **Algorithm 2** (page 7):

the *NT* algorithm's structure is used as a basis, and the only pseudocode lines that are listed are those corresponding to places where *QS* is added to *NK*.

**Algorithm 2** Pseudocode for *NK* algorithm

Inputs: $B_\Omega$ and $\mathbf{x}$, initial search box.
Output: $\mathbf{z}$, box of computed optimal values of controller parameters, or message "No feasible solution in $\mathbf{x}$".

1. ...
...
1.(b). ...
1.(b-bis). $\mathbf{z} \leftarrow \text{QS}(\mathbf{R}_{\mathbf{z}\Omega}, B_\Omega, \Omega, \mathbf{z})$
1.(c). ...
...
4. ...
4.bis. For $l = 1, 2$: $\mathbf{v}^l \leftarrow \text{QS}(\mathbf{R}_{\mathbf{v}^l\Omega}, B_\Omega, \Omega, \mathbf{v}^l)$
5. ...
...

### 2.3.3 | Algorithm *QS*

This algorithm is restricted to controllers containing only real poles and zeros, whose values are greater than 0, and so it is algorithm *NK*, because it uses *QS*. The controller general structure given in (7) is reduced to

$$C(s, x) = k\frac{\prod_{i=1}^{n_{rz}}(s + z_i)}{\prod_{l=1}^{n_{rp}}(s + p_l)} \tag{14}$$

and the parameters vector $x$ is also reduced accordingly:

$$x = (k, z_1, \dots, z_{n_{rz}}, p_1, \dots, p_{n_{rp}}). \tag{15}$$

The main idea behind this algorithm is to use the information about the relative vertical (magnitude) position of the NC rectangles given by $\mathbf{R}_{\mathbf{z}\Omega}$ and the boundaries $B_\Omega$. *QS* takes advantage of the fact that $|L_0(j\omega, x)|$ varies monotonically [6] with each individual parameter in $x$ (15), i.e., with gain and each pole and zero parameter, to decrease the size of the parameters box $\mathbf{z}$. *QS* algorithm is explained in detail and exemplified in [21]. A simplified explanation, for the particular case (without loss of generality) of open boundaries with $A_\omega$ above $B_\omega$ (as examples in Fig. 5 for $\omega \in \{0.1, 0.2, 1, 2\}$) is:

For each $\omega \in \Omega$, and for each $x(j)$ in $x$, $j = 1, ..., n$, perform the following steps:

- For each $\lambda = 1...n$, $\lambda \neq j$, assign to $x(\lambda)$ the value in $\mathbf{z}[\lambda]$ maximizing $x(\lambda)$ contribution to $|L_0(j\omega, x)|$.
- Let $S$ be the largest $\mathbf{z}[j]$ subinterval such that subs$(\mathbf{z}, j, S)$ is infeasible for $\omega$.
- $\mathbf{iz} \leftarrow \mathbf{i}^j_{\mathbf{z}\omega}$, being $\mathbf{i}^j_{\mathbf{z}\omega}$ defined as subs$(\mathbf{z}, j, S)$.
- $\mathbf{z}$ shrinks by subtracting $S$ from $\mathbf{z}[j]$, i.e., $\mathbf{z} \leftarrow \mathbf{z} - \mathbf{iz}$.

Remark: Note that $\mathbf{R}_{\mathbf{iz}\omega} \subseteq \mathbf{iR}_{\mathbf{z}\omega}$, so $L_0(j\omega, x)$ violates $B_\omega$ $\forall x \in \mathbf{iz}$, which allows us to ignore solutions contained in $\mathbf{iz}$.

The formal expression of the described algorithm is given as pseudocode as in **Algorithm 3** (page 8).

[6] For real poles and zeros with values greater than 0.

**Algorithm 3** Pseudocode for *QS* algorithm

Inputs: $\mathbf{R}_{\mathbf{z}\Omega}$, $\mathrm{B}_\Omega$, $\Omega$ and $\mathbf{z}$.
Output: $\mathbf{z}$, reduced version of original $\mathbf{z}$.

For $\omega \in \Omega$,
  For $j \in \{1, ..., n\}$
    $\mathbf{y} \leftarrow \mathbf{z}$  // $\mathbf{y}$ is a copy of $\mathbf{z}$
    For $\lambda \in \{1, ..., n\} - \{j\}$  // In this *for*, $\mathbf{y}[\lambda]$ receives the value $r$ in $\mathbf{z}[\lambda]$ that maximizes $x(\lambda)$ contribution to $|L_0(j\omega, x)|$.
      If $1 \leq \lambda \leq 1 + n_{rz} + 2n_{cz}$  // gain ($k$) or zero
        $r \leftarrow \sup(\mathbf{z}[\lambda])$
      Else  // pole
        $r \leftarrow \inf(\mathbf{z}[\lambda])$
      Endif
      $\mathbf{y} \leftarrow \mathrm{subs}(\mathbf{y}, \lambda, [r, r])$  // interval $\mathbf{y}[\lambda] \leftarrow r$
    Endfor
    $S \leftarrow$ largest subinterval of $\mathbf{y}[j]$ such that $\mathrm{subs}(\mathbf{y}, j, S)$ is infeasible for $\omega$
    $\mathbf{iz} \leftarrow \mathrm{subs}(\mathbf{z}, j, S)$
    $\mathbf{z} \leftarrow \mathbf{z} - \mathbf{iz}$
  Endfor
Endfor

## 3 | *NK* ALGORITHM SPEEDUP

Being faster than the *NT* algorithm, *NK* algorithm is still very costly in terms of computation burden. Thus, there is a clear interest in speeding it up. Two ways of reducing its computation time are proposed here: a) Improving the *QS* algorithm by using phase information (apart from magnitude information); and b) by using information about feasible boxes (apart from about infeasible ones).

As previously stated, the main idea used by *QS* to accelerate *NT* was to study the relation between NC boxes given by $\mathbf{R}_{\mathbf{z}\Omega}$ and boundaries $\mathrm{B}_\Omega$.

These proposals focus on this idea and take it further, in the sense of exploiting more information. They are explained in the following subsections, using boundaries from the Matlab® QFT Toolbox ([6]) design example 2 (section 4.1). Once the individual speedup proposals have been explained, in section 4 they are applied to the resolution of two typical QFT problems and their individual and combined contributions to *NK* algorithm speedup are studied.

### 3.1 | Using phase information

As previously explained, the *NK* algorithm, by means of *QS*, uses information about the vertical (magnitude) relation of the rectangles in $\mathbf{R}_{\mathbf{z}\Omega}$ and the boundaries $\mathrm{B}_\Omega$ in order to decrease the size of a certain parameters box. For an example, consider Fig. 2, where for a certain parameter box $\mathbf{a}$ and frequency $\omega = 0.1$, their associated rectangle $\mathbf{R}_{\mathbf{a}\,0.1}$ and infeasible rectangle $\mathbf{iR}_{\mathbf{a}0.1}$ are shown. Thanks to the use of *QS* in *NK*, for each parameter $x(j)$, $j = 1...n$, a certain portion $\mathbf{ia} = \mathbf{i}^j_{\mathbf{a}0.1}$ of $\mathbf{a}$ can be removed from $\mathbf{a}$. As previously stated, corresponding rectangles $\mathbf{R}_{\mathbf{ia}\,0.1}$ are subsets of $\mathbf{iR}_{\mathbf{a}0.1}$, which guarantees each $\mathbf{ia}$ is an infeasible box and thus can be safely removed from $\mathbf{a}$.

This kind of parameter box reduction is very important, and the most intuitive, because: a) It especially affects controller parameter $k$, directly connected to the optimization goal, $K_{hf}$ minimization; and b) Because open boundaries are similar to horizontal lines in the sense that they naturally act as a limit for the vertical extension of rectangles.

However, information about the horizontal relation between the rectangles in $\mathbf{R}_{\mathbf{z}\Omega}$ and the boundaries $\mathrm{B}_\Omega$, i.e., their relation in terms of phase, could also contribute to $\mathbf{z}$ reduction. This happens, trivially, for the vertical parts of closed boundaries, for instance the UHFB in Fig. 2: Some $\mathbf{R}_{\mathbf{ic}100}$ subsets of $\mathbf{iR}_{\mathbf{c}100}$ (in gray), could be used to shrink $\mathbf{c}$. But this also happens with open boundaries: Since they are not completely horizontal lines, when rectangles become smaller after some subdivisions in step 4 of

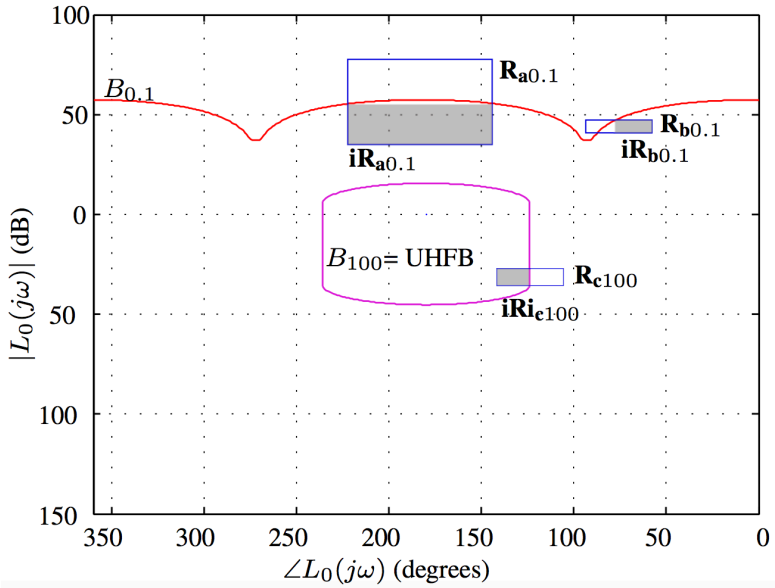


**FIGURE 2** Boundaries $B_{0.1}$ and $B_{100}$ and rectangles and infeasible rectangles (light grey) corresponding to given boxes **a** and **b** at $\omega = 0.1$ rad/s, and box **c** and $\omega = 100$ rad/s.

algorithms *NT* and *NK*, the situation of $\mathbf{R}_{\mathbf{b}0.1}$ rectangle, with an infeasible (gray) part that can be separated by a phase reduction, becomes common, and phase information can be used to shrink **b**.

Note that, in order to be able to perform this phase based boxes reductions, as happened with magnitude, it is necessary that $\angle L_0(j\omega, x)$ varies monotonically over the intervals of the parameters in $x$ (8). There are three types of terms in $C(s, x)$ (7). It can be easily checked that the contributions of the first and the second type to $\angle L_0(j\omega, x)$ are monotonic:

- $k$ has no effect on $\angle L_0(j\omega, x)$, so its effect is monotonic.
- Real poles or zeros, with structure $j\omega + q$, whose contribution to $\angle L_0(j\omega, x)$ is given by $\arctan(\omega/q)$ for poles and $-\arctan(\omega/q)$ for zeros, which are monotonic with respect to $q$.

The contribution to $\angle L_0(j\omega, x)$ of the third type of term, complex poles or zeros, is not monotonic in general. Their structure, $(j\omega)^2 + 2rtj\omega + t^2$, yields a contribution to $\angle L_0(j\omega, x)$ described by $\arctan(2rt\omega/(t^2 - \omega^2))$, monotonic respect to $r$, but not with respect to $t$. However, this difficulty can be overcome by splitting, for each $\omega \in \Omega$, the interval corresponding to $t$, $T$, into two subintervals $T_1 = [\underline{T}, \omega]$ and $T_2 = [\omega, \overline{T}]$ such that there is monotonicity respect to $t$ in both subintervals. This leads to splitting the parameters box **z** into two boxes at each $\omega \in \Omega$, which complicates the algorithm. For simplicity, and without loss of generality, this mechanism is not included in this work, so the programmed algorithm is restricted to real zeros and poles, as algorithm *NK* is, so the controller structure will be also reduced to (14) and the parameters vector to (15).

## 3.2 | Using feasible boxes information

Fig. 3 shows some examples of feasible rectangles: It is a second version of Fig. 2 where feasible rectangles $\mathbf{fR}_{\mathbf{a}0.1}$, $\mathbf{fR}_{\mathbf{b}0.1}$ and $\mathbf{fR}_{\mathbf{c}100}$ have been added, corresponding to boxes **a** and **b** and $\omega = 0.1$ rad/s, and box **c** and $\omega = 100$ rad/s. Given an arbitrary box **z** and frequency $\omega$, any **z'** $\subseteq$ **z** such that $\mathbf{R}_{\mathbf{z}',\omega} \subseteq \mathbf{fR}_{\mathbf{z}\omega}$, **z'** is feasible. To extract information from the existence of feasible subrectangles and, consequently, **z'** feasible subboxes of **z**, in order to reduce **z**, is an interesting possibility. But there are some important differences with respect to the use of infeasible boxes in the *QS* algorithm that must be considered. In *QS* algorithm, pieces of **z** are removed in an iterative process, returning a new **z**, which can be expressed as infeasible subboxes $\mathbf{i}^j_{\mathbf{z}\omega}$ at each $\omega \in \Omega$ and parameter $x(j)$, $j = 1, ..., n$, which is joined and then subtracted from **z**, which can be expressed as

$$\mathbf{z} \leftarrow \mathbf{z} - \bigcup_{\omega \in \Omega} \mathbf{i}_{\mathbf{z}\omega} \tag{16}$$

with $\mathbf{i}_{\mathbf{z}\omega} \doteq \bigcup_{j=1...n} \mathbf{i}^j_{\mathbf{z}\omega}$.

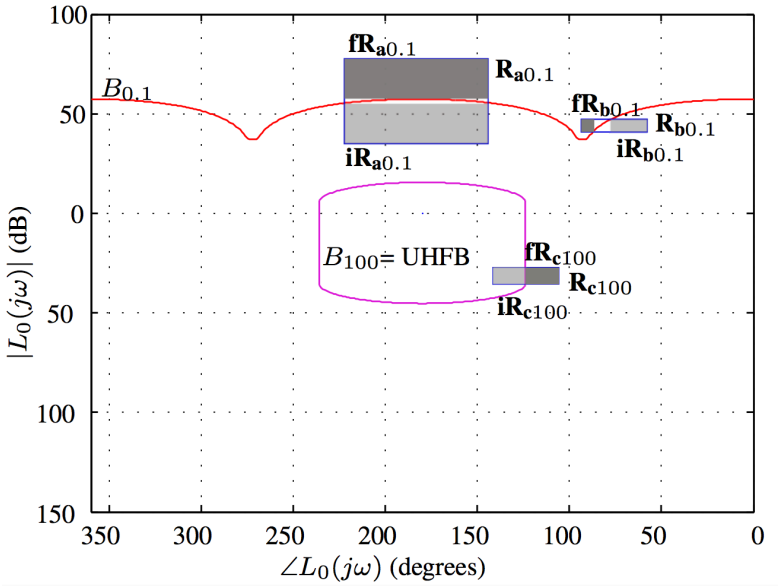


**FIGURE 3** Second version of Fig. 2 (infeasible rectangles in light grey) where feasible rectangles (dark grey) have been added.

Again, the input is a box, $\mathbf{z}$, and the output is a reduced version of that box where the subtracted parts are discarded. However, when dealing with infeasible subboxes:

- Compared to (16), feasible subboxes at each frequency $\omega$, $\mathbf{f}_{\mathbf{z}\omega}$, should be intersected instead of joined, since a feasible box must be so $\forall\omega \in \Omega$:

$$\mathbf{z'} \leftarrow \bigcap_{\omega\in\Omega} \mathbf{f}_{\mathbf{z}\omega}. \tag{17}$$

- In a similar fashion, computation of $\mathbf{f}_{\mathbf{z}\omega}$ at each $\omega$ also involves intersection instead of junction:

$$\mathbf{f}_{\mathbf{z}\omega} \doteq \bigcap_{j=1\ldots n} \mathbf{f}^j_{\mathbf{z}\omega}, \tag{18}$$

  where each $\mathbf{f}^j_{\mathbf{z}\omega}$ is computed in a $QS$-like way:

  - For each $\lambda = 1...n$, $\lambda \neq j$, assign to $x(\lambda)$ the value in $\mathbf{z}[\lambda]$ minimizing $x(\lambda)$ contribution to $|L_0(j\omega, x)|$.
  - Let $F$ be the largest $\mathbf{z}[j]$ subinterval such that subs($\mathbf{z}, j, F$) is feasible for $\omega$.
  - $\mathbf{f}^j_{\mathbf{z}\omega} \doteq$ subs($\mathbf{z}, j, F$).

- $QS$ algorithm receives a box $\mathbf{z}$ and returns a shrunk version of $\mathbf{z}$. However, the result of the process of exploiting feasible subboxes information is double: a) $\mathbf{z'}$, a feasible box, and b)

$$\mathrm{U} = \{\mathbf{u}_1, ..., \mathbf{u}_m\}, \tag{19}$$

  a set of subboxes of $\mathbf{z}$, with $m = 2^n - 1$, such that

$$\bigcup_{\mathbf{u}\in\mathrm{U}} = \mathbf{z} - \mathbf{z'}. \tag{20}$$

  Each of these $m$ boxes in U can be feasible, infeasible or ambiguous, and so they should go through steps 5 and 6 in $NT$ and $NK$ algorithms in order to be discarded or included in list $NL$ in their right position, as any ordinary box from step 4 would, depending on their feasibility. Note that $m$ grows exponentially with the number of parameters $n$, so a good trade-off between the better prune achieved and extra computation cost must be chosen. In this work, as a first approach, the choice was to use only the first parameter in $x$ ($x(1) = k$), for computing $\mathbf{z'}$. The idea, which has provided good speedup results, is to focus on this parameter because it is unique in the sense that it directly represents the objective function. This approach simplifies U, with $m = 1$:

$$\mathrm{U} = \{\mathbf{u}\} \doteq \mathbf{z} - \mathbf{z'}. \tag{21}$$

Note that, being **z** ambiguous, in general **u** is an ambiguos box. Otherwise it is infeasible, but this can only happen under very unusual circumstances, like completely straight boundaries.

- Compared to **iz** treatment, **z'** should not be discarded because it can include the optimal solution. In fact a feasible box like **z'** is a very special box, because it certainly contains a solution. Moreover, any $x \in$ **z'** is a solution, and the best solutions are those with $k = x(1) = \inf(\mathbf{z'}[1])$. This very valuable information can be used for a better prune in the branch and bound process underlying *NT* and *NK* algorithms. This can be implemented by using a prune variable $C$ which stores the value $\inf(\mathbf{z'}[1])$ of the best (minimum $\inf(\mathbf{z'}[1])$) feasible box found. Additionally, this new step 3bis is added to *NK*:

  3bis Prune and boxes reduction by $C$ information:

  3bis.(a) If $C \leq z$,

  - discard $(\mathbf{z}, z, flag_{\mathbf{z}})$: $NL \leftarrow NL - \{(\mathbf{z}, z, flag_{\mathbf{z}})\}$.

  - If *NL* is empty, there is no feasible solution in **x** (initial search box), so PRINT "No feasible solution in **x**" and EXIT.

  3bis.(b) Else reduce **z** using $C$ information:

  - $K \leftarrow [z, C]$ (initialize interval variable $K$)

  - $\mathbf{z} \leftarrow \text{subs}(\mathbf{z}, 1, K)$

  - Update $(\mathbf{z}, z, flag_{\mathbf{z}})$ with updated **z**.

Remarks: Note condition in step 3bis.(a) will be never satisfied when using the particular sorting criterion given by step 6.(d) in *NT* and *NK* algorithms. So this step could be omitted in this particular implementation. But it is still included in the text in order to emphasize the additional acceleration that could be achieved in the case of using a different sorting criterion. Note that, in that case, step 3 should be reformulated, since the solution analyzed at that step would be, in general, not optimal.

## 3.3 | Proposed algorithm - MC

In this section the proposed algorithm, *MC*, which formalizes the proposals in the previous subsections, is presented. The source code is available at https://github.com/Isaac-Martinez-Forte/QFTbx. A new *QS* function, *QS2*, must be implemented (**Algorithm 4**, page 12). It receives the same information as *QS*, applies the procedures described in sections 3.1 and 3.2, and as a result returns:

- **z'**, a feasible box, or $\emptyset$, indicating not found.
- **uz**, a box whose feasibility is unknown, or $\emptyset$ if $\mathbf{z'} = \mathbf{z}$.

For the sake of simplicity, and without loss of generality, algorithm *QS2* is written for these particular cases in each of its three stages:

- Stages 1 and 3, open boundaries with $A_\omega$ above $B_\omega$, crossing the vertical sides of considered rectangle, as exemplified by $\mathbf{fR}_{\mathbf{a}0.1}$ in Fig. 3.
- Stage 2, open or closed boundary, with $A_\omega$ on the right of $B_\omega$, crossing horizontal sides of considered rectangle, as exemplified by $\mathbf{fR}_{\mathbf{b}0.1}$ or $\mathbf{fR}_{\mathbf{c}100}$ in Fig. 3.

*MC* algorithm is detailed and commented in **Algorithm 5** on page 13.

**Algorithm 4** Pseudocode for *QS2* algorithm

Inputs: $\mathbf{R}_{\mathbf{z}\Omega}$, $\mathrm{B}_{\Omega}$, $\Omega$ and $\mathbf{z}$.
Output: $\mathbf{z'}$ and $\mathbf{u}$.

// STAGE 1, *QS* is used to reduce $\mathbf{z}$ using magnitude info
  $\mathbf{z} \leftarrow \mathrm{QS}(\mathbf{R}_{\mathbf{z}\Omega}, \mathrm{B}_{\Omega}, \Omega, \mathbf{z})$
// STAGE 2, reduce $\mathbf{z}$ using phase information
  For $\omega \in \Omega$,
    For $j \in \{2, ..., n\}$ // $k$ does not influence phase
      $\mathbf{y} \leftarrow \mathbf{z}$  // $\mathbf{y}$ is a copy of $\mathbf{z}$
      For $\lambda \in \{2, ..., n\} - \{j\}$  // In this *for*, $\mathbf{y}$ receives the value $r$ in $\mathbf{z}[\lambda]$ that maximizes $x(\lambda)$ contribution to $\angle L_0(j\omega, x)$...
        If $2 \le j \le 1 + n_{rz} + 2n_{cz}$  // zero
          $r \leftarrow \sup(\mathbf{z}[\lambda])$
        Else  // pole
          $r \leftarrow \inf(\mathbf{z}[\lambda])$
        Endif
        $\mathbf{y} \leftarrow \mathrm{subs}(\mathbf{y}, \lambda, [r, r])$  // interval $\mathbf{y}[\lambda] \leftarrow r$
      Endfor
      $S \leftarrow$ largest subinterval of $\mathbf{y}[j]$ such that $\mathrm{subs}(\mathbf{y}, j, S)$ is infeasible for $\omega$
      $\mathbf{iz} \leftarrow \mathrm{subs}(\mathbf{z}, j, S)$
      $\mathbf{z} \leftarrow \mathbf{z} - \mathbf{iz}$
    Endfor
  Endfor
// STAGE 3, use feasible boxes information to split $\mathbf{z}$ into $\mathbf{z'}$ and $\mathbf{uz}$ in the direction of $k$ (first parameter in $x$)
  $\mathbf{y} \leftarrow \mathbf{z}$  // $\mathbf{y}$ is a copy of $\mathbf{z}$
  For $\lambda \in \{2, ..., n\}$
    If $2 \le \lambda \le 1 + n_{rz} + 2n_{cz}$  // zero
      $r \leftarrow \inf(\mathbf{z}[\lambda])$
    Else  // pole
      $r \leftarrow \sup(\mathbf{z}[\lambda])$
    Endif
    $\mathbf{y} \leftarrow \mathrm{subs}(\mathbf{y}, \lambda, [r, r])$  // interval $\mathbf{y}[\lambda] \leftarrow r$
  Endfor
  $k_{\min} \leftarrow \inf(\mathbf{z}[1])$,  $k_{\max} \leftarrow \sup(\mathbf{z}[1])$,  $k_f \leftarrow k_{\min}$
  For $\omega \in \Omega$,
    $k_\omega \leftarrow$ minimum value in $\mathbf{z}[1]$ such that $\mathrm{subs}(\mathbf{y}, 1, [k_{\mathrm{f}}, k_{\max}])$ is feasible for $\omega$, or $\infty$ if no value exists.
    $k_f \leftarrow max(k_\omega, k_f)$  // intersection
  Endfor
  If $k_f = \infty$  // there is no feasible box
    $\mathbf{z'} \leftarrow \varnothing$
  Else
    $\mathbf{z'} \leftarrow \mathrm{subs}(\mathbf{z},1,[k_f, k_{\max}])$
  Endif
  $\mathbf{u} \leftarrow \mathbf{z} - \mathbf{z'}$

**Algorithm 5** Pseudocode for *MC* algorithm

Inputs: $\mathrm{B}_\Omega$ and $\mathbf{x}$, initial search box.
Output: $\mathbf{z}$, box of computed optimal values of controller parameters, or message "No feasible solution in $\mathbf{x}$".

1. Check feasibility of the initial search box
   (a) $\mathbf{z} \leftarrow \mathbf{x}$ (initialize *current box* to initial box).
   (b) $\mathbf{R}_{\mathbf{z}\Omega} \leftarrow (\mathrm{mag}(\mathbf{z},\omega), \mathrm{ang}(\mathbf{z},\omega)), \forall\omega \in \Omega$.
   (b-bis) $(\mathbf{z'}, \mathbf{z}) \leftarrow \mathrm{QS2}(\mathbf{R}_{\mathbf{z}\Omega}, \mathrm{B}_\Omega, \Omega, \mathbf{z})$ // Note $\mathbf{z}$ receives the result $\mathbf{u}$ from *QS2*.
   (c) $flag_{\mathbf{z}} \leftarrow \mathrm{FT}(\mathbf{R}_{\mathbf{z}\Omega}, \mathrm{B}_\Omega)$.
   (d) IF $flag_{\mathbf{z}} = infeasible$ and $\mathbf{z'} = \varnothing$, PRINT "No feasible solution in $\mathbf{z}$" and EXIT.
2. Initialize the list *NL*
   (a) Initialization of $z$, minimum achievable $K_{hf}$, and prune variable $C$:
   If $\mathbf{z'} = \varnothing$, $C \leftarrow \infty$, $z \leftarrow \inf(\mathbf{z}[1])$
   Else $C \leftarrow \mathbf{z'}[1]$, $z \leftarrow \min(\inf(\mathbf{z}[1]), \inf(\mathbf{z'}[1]))$
   Endif
   (b) List *NL* initialization:
   If $\mathbf{z'} = \varnothing$, $NL \leftarrow \{(\mathbf{z}, z, flag_{\mathbf{z}})\}$,
   Else
   If $\inf(\mathbf{z}[1]) < \inf(\mathbf{z'}[1])$, $NL \leftarrow \{(\mathbf{z}, z, flag_{\mathbf{z}}), (\mathbf{z'}, \inf(\mathbf{z'}[1]), feasible)\}$
   Else, $NL \leftarrow \{(\mathbf{z'}, \inf(\mathbf{z'}[1], feasible), (\mathbf{z}, z, flag_{\mathbf{z}}))\}$, $\mathbf{z} \leftarrow \mathbf{z'}$, $flag_{\mathbf{z}} \leftarrow feasible$
   Endif
   Endif
3. IF $flag_{\mathbf{z}} = feasible$, PRINT "Optimal controller parameter box is, $\mathbf{z}$" and EXIT.
3bis. Prune and boxes reduction by $C$ information:
   3bis.(a) If $C < z$, [1]
   - $NL \leftarrow NL - \{(\mathbf{z}, z, flag_{\mathbf{z}})\}$ // discard $(\mathbf{z}, z, flag_{\mathbf{z}})$
   - If *NL* is empty, there is no feasible solution in $\mathbf{x}$, so PRINT "No feasible solution in $\mathbf{x}$" and EXIT.
   3bis.(b) Else reduce $\mathbf{z}$ using $C$ information:
   - $K \leftarrow [z, C]$
   - $\mathbf{z} \leftarrow \mathrm{subs}(\mathbf{z}, 1, K)$
   - Update $(\mathbf{z}, z, flag_{\mathbf{z}})$ with updated $\mathbf{z}$.
4. Bisect $\mathbf{z}$ along its maximum width coordinate direction $d$ into two subboxes $\mathbf{v}^1$ and $\mathbf{v}^2$ such that $\mathbf{z} = \mathbf{v}^1 \cup \mathbf{v}^2$.
4.bis. For $j = 1, 2$: $(\mathbf{z}^j, \mathbf{v}^j) \leftarrow \mathrm{QS2}(\mathbf{R}_{\mathbf{v}^j\Omega}, \mathrm{B}_\Omega, \Omega, \mathbf{v}^j)$
5. Do the feasibility test for the new subboxes $\mathbf{v}^1$ and $\mathbf{v}^2$ and discard any infeasible ones, i.e., for $j = 1, 2$:
   (a) $\mathbf{R}_{\mathbf{v}^j\Omega} \leftarrow (\mathrm{mag}(\mathbf{v}^j, \omega), \mathrm{ang}(\mathbf{v}^j, \omega)), \forall\omega \in \Omega$
   (b) $flag_{\mathbf{v}^j} \leftarrow \mathrm{FT}(\mathbf{R}_{\mathbf{v}^j\Omega}, \mathrm{B}_\Omega)$
   (c) $v^j \leftarrow \inf(\mathbf{v}^j[1])$
   (d) Form the triple $(\mathbf{v}^j, v^j, flag_{\mathbf{v}^j})$.
   (e) IF $flag_{\mathbf{v}^j} = infeasible$, drop $(\mathbf{v}^j, v^j, flag^j_{\mathbf{v}})$.
5bis. Process $\mathbf{z}^1$ and $\mathbf{z}^2$ boxes, i.e., for $j = 1, 2$:
   If $\mathbf{z}^j \neq \varnothing$,
   (a) $C = \min(C, \inf(\mathbf{z}^j[1])$
   (b) Form the triple $(\mathbf{z}^j, \inf(\mathbf{z}^j[1], feasible)$.
   Endif
6. Sort the list *NL*
   (a) $NL \leftarrow NL - \{(\mathbf{z}, z, flag_{\mathbf{z}})\}$ .
   (b) Add any remaining triple(s) from steps 5.(e), and triples from step 5bis, to *NL*.
   (c) IF *NL* is empty, there is no feasible solution in $\mathbf{x}$, so PRINT "No feasible solution in $\mathbf{x}$" and EXIT.
   (d) Sort *NL* such that the second members of all triples in *NL* do not decrease in value.
   (e) Denote the first item of list *NL* as $(\mathbf{z}, z, flag_{\mathbf{z}})$.

# 4 | DESIGN EXAMPLES AND RESULTS

In the following two subsections two example problems typically used in QFT literature examples are introduced and solved by using these five algorithms:

- *NT*
- *NK*
- ***MC***
- *MC2*: *MC* algorithm, deactivating stage 3
- *MC3*: *MC* algorithm, deactivating stage 2

Algorithms *MC2* and *MC3* are included in order to observe individual contributions of phase information and feasible boxes information, respectively. *NK* is used as a baseline to determine the speedup achieved by *MC*. *NT* is included in order to observe *NK* speedup respect to *NT*.

For a given problem and algorithm, the fundamental parameter determining execution time is $n$, the number of controller parameters. So, in order to compare how execution time evolves with respect to $n$, for both problems, the five algorithms are executed for real poles/zeros controller structures with an increasing $n$, up to the $n$ value where the faster algorithm's execution time is below 1 minute. An initial search box of controller parameter values, $\mathbf{x}$, common to the five algorithms, has been chosen in each example such that the slower algorithm can shape the 5 parameters controller in a maximum of 5 minutes. These limits were chosen so that the experiments size was enough to observe the differences between the five algorithms and, at the same time, the experiments remain agile. The experiments were performed on an Intel$^{\text{R}}$ Core$^{\text{TM}}$ i7-6700HQ CPU at 2.60GHz, with 8 GB RAM, running ArchLinux. *gcc* 8.2 C++ compiler was used, with optimization -o2 activated. All the code involving QFT functions was programmed from scratch in C++ and Qt.

For any value of $n$, the controller structure used in the following examples is given by

$$C(s,x) = k\frac{\prod_{i=1}^{n_{rz}}(s+z_i)}{\prod_{j=1}^{n_{rp}}(s+p_j)} \tag{22}$$

with

$$x = (k, z_1, \dots, z_{n_{rz}}, p_1, \dots, p_{n_{rp}}) \tag{23}$$

and where $n_{rz} = \lfloor \frac{n-1}{2} \rfloor$ and $n_{rp} = \lceil \frac{n-1}{2} \rceil$. Note $C(s,x)$ is strictly proper for odd $n$ and proper for even $n$.

## 4.1 | Matlab QFT Toolbox design example 2

This example is introduced in[6] as the second design example used to introduce Matlab® QFT Toolbox (from now on *QFT Toolbox* for short). It is an easy to solve by hand example, suitable to train beginners:

$$\mathrm{P} = \left\{ P(s) = \frac{ka}{s(s+a)}, k \in [1,10], a \in [1,10] \right\}$$

$\Omega = \{0.1, 0.5, 1, 2, 15, 100\}$ is defined, and $\mathrm{S}_\Omega$ is given by this robust stability spec:

$$\left| \frac{P(j\omega)C(j\omega)}{1 + P(j\omega)C(j\omega)} \right| \leq \gamma = 1.2, \ \forall P \in \mathrm{P}, \ \omega \in \mathbb{R}^+$$

and this robust performance tracking spec:

$$\alpha(\omega) \leq \left| F(j\omega)\frac{P(j\omega)C(j\omega)}{1 + P(j\omega)C(j\omega)} \right| \leq \beta(\omega), \forall \omega \in \Omega$$

with

$$\alpha(\omega) = \left| \frac{0.6584(j\omega + 30)}{(j\omega)^2 + 4j\omega + 19.752} \right|$$

and

$$\beta(\omega) = \left| \frac{120}{(j\omega)^3 + 17(j\omega)^2 + 82j\omega + 120} \right|.$$

**TABLE 1** Execution time (seconds) for each algorithm, for $n = 1..9$, for QFT toolbox design example 2. Blank: >5m.

| *n* | *NT* | *NK* | *MC2* | *MC3* | *MC* |
|---|---|---|---|---|---|
| 1 | 0.01 | 0.03 | 0.03 | 0.03 | 0.03 |
| 2 | 0.10 | 0.11 | 0.11 | 0.11 | 0.10 |
| 3 | 0.88 | 0.66 | 0.59 | 0.40 | 0.35 |
| 4 | 15.46 | 3,90 | 3.51 | 2.56 | 2.40 |
| 5 | 38.98 | 8.88 | 8.30 | 2.68 | 2.29 |
| 6 | | 21.10 | 16.69 | 7.57 | 5.01 |
| 7 | | 48.41 | 40.99 | 15.90 | 12.10 |
| 8 | | 98.07 | 86.34 | 32.10 | 24.79 |
| 9 | | | | 62.99 | 50.49 |
| 10 | | | | | 101.09 |

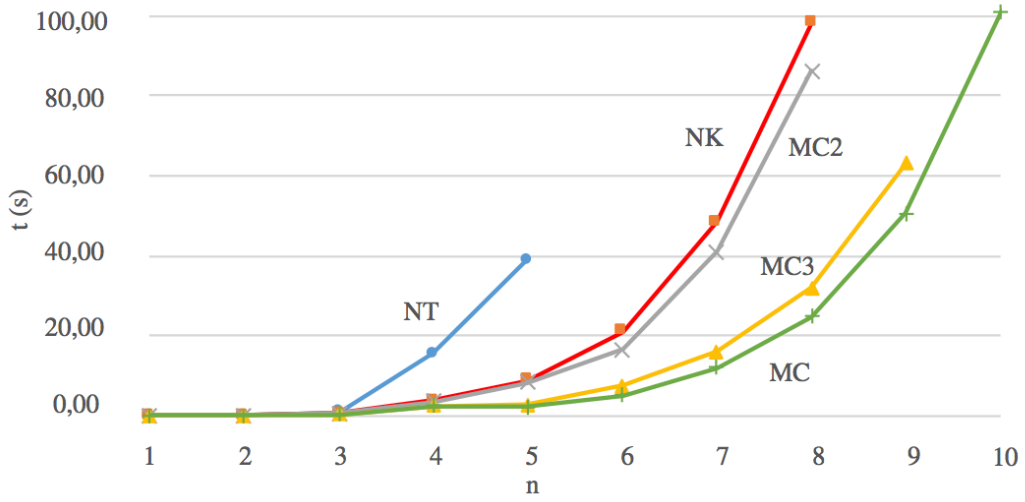


**FIGURE 4** Execution time (seconds) for each algorithm, for $n = 1..10$, for QFT toolbox design example 2.

Table 1 shows the execution times for the different algorithms for $n = 1...10$. A blank cell indicates no time was measured because the execution took more than 5 minutes. The same information is graphically represented in Fig. 4.

In order to allow some form of quantitative comparison between execution times, an exponential regression has been performed for each of the execution times in Fig. 4, obtaining a coefficient of determination $R^2 > 0.96$ in all cases, and values $\alpha$ and $\beta$ for the expression $y(x) = \alpha + \beta^x$ shown in Table 2.

**TABLE 2** Exponential regression $y(x) = \alpha + \beta^x$ coefficients for execution times for solving QFT toolbox design example 2.

| parameter | *NT* | *NK* | *MC2* | *MC3* | *MC* |
|---|---|---|---|---|---|
| $\alpha$ | 0.0011 | 0.014 | 0.0167 | 0.021 | 0.022 |
| $\beta$ | 9.055 | 3.27 | 3.15 | 2.56 | 2.45 |

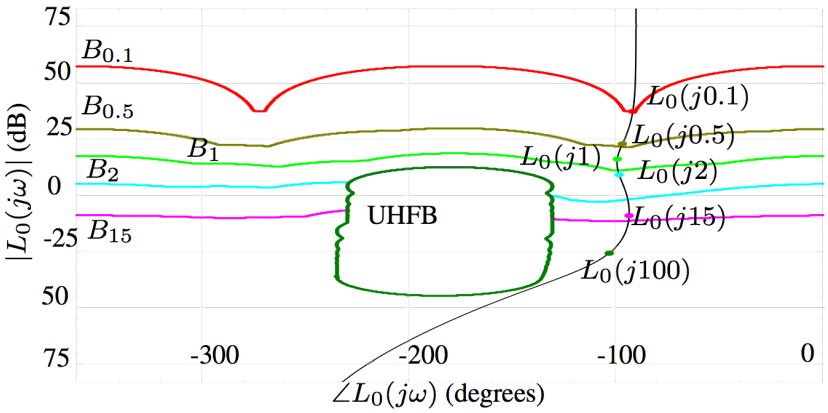


**FIGURE 5** QFT toolbox example problem 2, boundaries $B_\omega$, and example of $L_0(s)$ obtained as solution from *MC* algorithm.

As an example of the results obtained by the *MC* algorithm, Fig. 5 shows open loop $L_0(s)$ given as solution by the algorithm executed with $n = 10$, together with associated boundaries $B_\Omega$. The corresponding controller $C_{\text{QFTex2}}(s)$ is given by structure (22) with $x = (7.012e6, 1.47, 1089, 1987.5, 3212.9, 575, 998, 1995, 2700, 3010)$.

## 4.2 | ACC '90 benchmark example

This example corresponds to the ACC '90 benchmark problem proposed in[29], a non-trivial problem, previously used in, for instance,[30],[1] and[21].

The plant is given by

$$P(s) = \frac{Y(s)}{U(s)} = \frac{e}{m_1 s^2 \left( m_2 s^2 + e \left( 1 + \frac{m_1}{m_2} \right) \right)}, \tag{24}$$

where $m_1 = m_2 = 1$ and $e \in [0.5, 2]$. It is an undamped spring-mass system (Fig. 6).

The control objective is to obtain robust stability with a reasonable control effort. Impulse response settling time should be around $t_s = 15$ s. The main difficulty comes from the pair of complex undamped poles at $s = \pm\sqrt{-2e}$ , because they lead to an infinite magnitude resonance peak at frequency $\omega_p$, dependent on $e$ value. In order to be able to plot $L_0(s)$ in a finite area of NC, these poles are considered as lightly damped. $e_0 = 0.5$ is chosen as the nominal value for $e$, which leads $\omega_p = 1$ rad/s. So $\Omega$ is chosen with special emphasis around that frequency: $\Omega = \{0.1, 0.98, 0.99, 1, 2, 5, 7, 8.5, 10, 15, 20, 100\}$. The corresponding set of boundaries $\mathrm{B}_\Omega$, constituted by only (closed) stability boundaries, is depicted in Fig. 8.

Table 3 shows the execution times for the different algorithms for $n = 1...12$. Again blank cells indicate no time was measured because the execution took more than 5 minutes. The same information is graphically represented in Fig. 7.

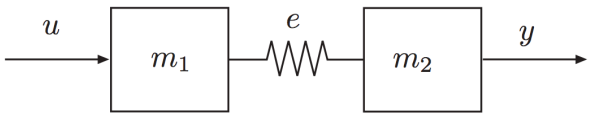


**FIGURE 6** ACC '90 benchmark problem.

**TABLE 3** Execution time (seconds) for each algorithm, for $n = 1..12$, for ACC '90 benchmark problem. Blank: >5m.

| ***n*** | ***NT*** | ***NK*** | ***MC2*** | ***MC3*** | ***MC*** |
|---|---|---|---|---|---|
| 1 | 0.03 | 0.03 | 0.08 | 0.02 | 0.02 |
| 2 | 0.26 | 0.03 | 0.09 | 0.05 | 0.03 |
| 3 | 0.47 | 0.04 | 0.14 | 0.07 | 0.07 |
| 4 | 9.06 | 0.11 | 0.31 | 0.19 | 0.13 |
| 5 | 50.84 | 0.27 | 0.53 | 0.54 | 0.58 |
| 6 | | 0.55 | 0.99 | 1.05 | 1.07 |
| 7 | | 2.79 | 1.51 | 2.91 | 1.76 |
| 8 | | 44.57 | 10.08 | 18.45 | 2.24 |
| 9 | | | 15.76 | 36.75 | 5.30 |
| 10 | | | 57.63 | | 8.65 |
| 11 | | | | | 25.44 |
| 12 | | | | | 60.85 |

**TABLE 4** Exponential regression $y(x) = \alpha + \beta^x$ coefficients for execution times for solving ACC '90 benchmark problem.

| **parameter** | ***NT*** | ***NK*** | ***MC2*** | ***MC3*** | ***MC*** |
|---|---|---|---|---|---|
| $\alpha$ | 0.0046 | 0.0026 | 0.0183 | 0.0057 | 0.088 |
| $\beta$ | 6.26 | 2.96 | 2.1 | 2.56 | 2.07 |

Again, an exponential regression has been performed for execution times in Fig. 7, obtaining $R^2 > 0.96$ in all cases, except for *NK*, with $R^2 > 0.92$, and values $\alpha$ and $\beta$ for the expression $y(x) = \alpha + \beta^x$ shown in Table 4.

As an example of the results obtained by the *MC* algorithm, Fig. 8 shows open loop $L_0(s)$ given as solution by the algorithm executed with $n = 12$, together with associated boundaries $B_\Omega$.

The corresponding controller $C_{\text{ACC90}}(s)$ is given by structure (22) with $x = (1.14e7, 0.075, 0.35, 0.39, 25, 34.87, 7.2, 30.1, 39.8, 95.9, 96.25)$.

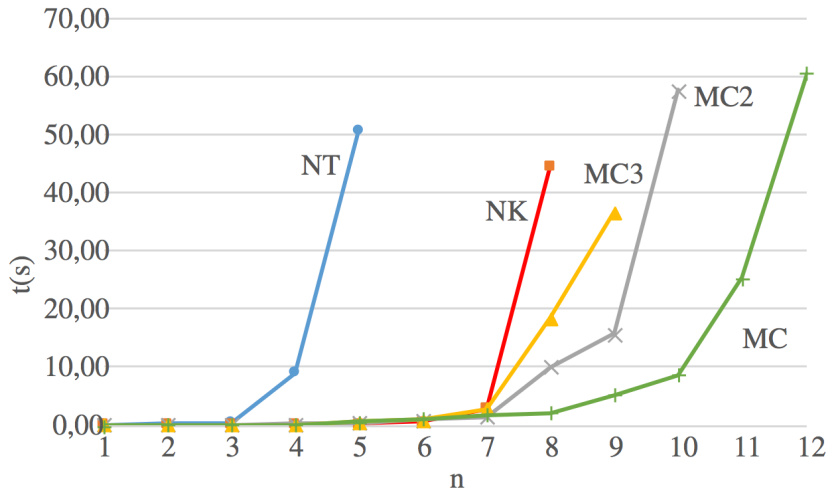


**FIGURE 7** Execution time (seconds) for each algorithm, for $n = 1..12$, for ACC '90 design example.

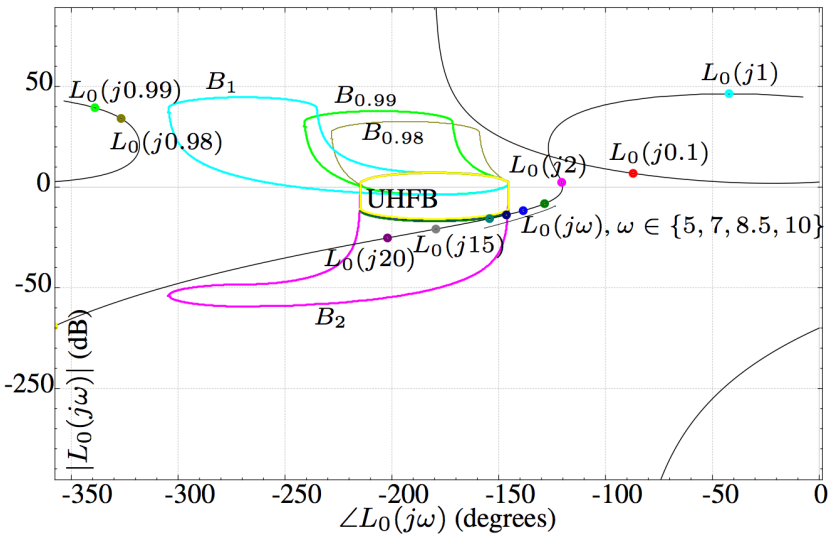


**FIGURE 8** ACC '90 benchmark problem, boundaries $B_\omega$, and example of $L_0(s)$ obtained as solution from *MC* algorittm.

## 4.3 | Results analysis

In both example problems algorithm *MC* significantly improved the results obtained by the base algorithm, *NK*:

- QFT toolbox example 2: This improvement can be observed graphically in Fig. 4. The *NK* algorithm takes less than one minute up to 7 parameters, while the *MC* obtains the same results for 9 parameters. Table 2 shows how the *MC* decreases the base of the exponential time ($\beta$) from 3.27 to 2.45, which is a large difference, especially for a higher number of parameters.
- ACC '90 benchmark: In this example the improvement is higher, which is apparent in Fig. 7. The *NK* algorithm takes less than one minute for up to 8 parameters, while the *MC* raises this value to 11 to 12 parameters. Table 4 shows how the *MC* decreases $\beta$ even more than in the previous problem, from 2.96 to 2.07.

An interesting result that can be observed is the different individual effects of the use of phase information (*MC2*) and feasible boxes information (*MC3*), in each example:

- QFT toolbox example 2: In Fig. 5 *MC2* if very close to *NK*, with a very slight $\beta$ improvement (from 3.27 to 3.15). However, *MC3* if very close to *MC*, i.e., *MC* improvement in this problem has to do basically with the addition of information about feasible boxes, and phase information makes almost no difference. This behavior is coherent with the type of boundaries on this problem (Fig. 5): Open boundaries (which allow feasible boxes information use), with almost no vertical parts (except for the UHFB and part of $B_{0.1}$), which makes phase information use almost impossible.
- ACC '90 benchmark: In contrast, in this problem in Fig. 8 *MC2* improves the algorithm performance more than *MC3*, although both lead to a significant improvement in *NK*, from $\beta = 2.96$ to 2.1 and 2.56 respectively. Again, this behavior is coherent with the type of boundaries on this problem (Fig. 8): Closed boundaries, with large vertical parts, which permit phase information use, and also with large horizontal parts, which also allow feasible boxes information use.

## 5 | CONCLUSIONS

Two possible ways of speeding up *NK* are proposed: Using phase information and using feasible boxes information. Algorithm *MC*, *NK* plus both improvements, was proposed and implemented. In order to be able to compare the algorithms, and in particular to measure the speedup obtained by the *MC*, algorithms *NT* and *NK* were also implemented, and all the algorithms were used to solve two classical QFT example problems: QFT toolbox example 2 and ACC '90 benchmark problem. The results obtained show that the *MC*, like the *NT* and *NK*, exhibits an exponential execution time (with respect to $n$, number of controller parameters), but significantly reduces the base of the exponential time function. In practice this means that, for the example problems used, the number of controller parameters for which the algorithm needs less than one minute is increased by 2 to 4 parameters.

Algorithms *MC2* and *MC3* have been defined as the *MC* deactivating stage 3 or stage 2, respectively, in order to detect which part of the speedup was due to the use of phase information of feasible boxes information. In QFT toolbox example 2 feasible boxes information became more important, whereas in ACC '90 benchmark problem phase information was the most important factor. An explanation for this phenomenon has been given. These results suggest that both improvements are complementary, although each of them can become more important depending on the shape of the boundaries.

Note this work is limited to the case of a real, right half complex plane, poles and zeros controller. The algorithm modifications needed to include complex poles and zeros have been mentioned, but not developed. This issue only affects the use of feasible boxes information, and not the use of phase information. The possibility for unstable or non minimum phase controllers will be included in future versions of the algorithm.

Another limitation is that the proposed version of the *MC* is specific for $K_{hf}$ minimization, although it could be reformulated to handle different optimization criterions.

There are various ways in which the obtained speedup can be increased. One has to do with the fact that, regarding the use of feasible boxes information, only the $k$ controller parameter has been considered. A promising path to explore is to study the use of other parameters, including heuristics in order to determine which parameters or parameter combinations are the most convenient to use at each point of the algorithm execution. Another interesting possibility has to do with how the list *NL* is sorted. In algorithms *NK* and *MC* it is sorted by the minimum $k$ of the box $\mathbf{z}$, i.e., by $\inf(\mathbf{z}[1])$, which is the best possible result achievable from that box. But the most common case is that this result cannot really be achieved. So it would be very interesting to develop some heuristically estimated benefit that can be gained from a certain box, and to sort *NL* accordingly. In this way the algorithm would be guided to explore better solutions first, so those solutions could be obtained sooner and then could be used to prune worse solutions, making the overall algorithm become faster.

## References

1. Horowitz I. *Quantitative Feedback Design Theory - QFT (Vol.1)*. Boulder, Colorado, USA: QFT Press . 1993.
2. Yaniv O. *Quantitative Feedback Design of Linear and Non-linear Control Systems*. Norwell, MA.: Kluwer Academic Publisher . 1999.

3. Sidi M. *Design of Robust Control Systems: From Classical to Modern Practical Approaches*. Malabar, FL.: Krieger Publishing . 2002.

4. García-Sanz M. *Robust Control Engineering: Practical QFT Solutions*. USA: CRC Press, Taylor and Francis . 2017.

5. Horowitz I. Optimum Loop Transfer Function in Sigle-Loop Minimum-Phase Feedback Systems. *Int. J. of Control* 1973; 18: 97-113.

6. Borghesani C, Chait Y, Yaniv O. *Quantitative Feedback Theory Toolbox*. The MathWorks, Inc.; Natick, MA, USA: 1995.

7. García-Sanz M. The QFT Control Toolbox for Matlab–QFTCT. http://codypower.com/CDP_QFTCT.htm; 2020.

8. Thomson D. *Optimal and Sub-Optimal Loop Shaping in Quantitative Feedback Theory*. PhD thesis. School of Mechanical Eng., Purdue University, West Lafayette, IN, USA; 1990.

9. Gera A, Horowitz I. Optimization of the Loop Transfer Function. *Int. J. of Control* 1980; 31: 389-398.

10. Bokharaie V, Khaki-Sedigh A. An LMI Approach to Automatic Loop-Shaping of QFT Controllers. In: ; 2006; Glasgow, UK.

11. Chait Y, Chen Q, Hollot CV. Automatic Loop-Shaping of QFT Controllers Via Linear Programming. *ASME J. Dynamic Systems, Measurement, and Control* 1999; 121: 351-357.

12. Fransson C, Lennartson B, Wik T, Holmström K, Saunders M, Gutman P. Global Controller Optimizacion Using Horowitz Bounds. In: IFAC. ; 2002.

13. Yaniv O, Nagurka M. Automatic loop shaping of structured controllers satisfying QFT performance. In: . 127(3). ASME. ; 2005; Glasgow, UK: 472-477.

14. Chen W, Ballance D, Li Y. Automatic loop-shaping in QFT using genetic algorithms. tech. rep., Centre for Systems and Control, University of Glasgow; Glasglow, UK: 1998.

15. García-Sanz M, Guillén J. Automatic Loop Shaping of QFT Controllers Via Genetic Algorithm. In: . 2. IFAC. Elsevier Sci.; 2000; Kidlington, UK: 603-608.

16. Raimúndez C, Baños A, Barreiro A. QFT Controller Synthesis Using Evolutive Strategies. In: ; 2001; Pamplona, Spain: 291-296.

17. Cervera J, Baños A. Automatic Loop Shaping in QFT by Using CRONE Structures. In: IFAC. ; 2006; Porto, Portugal.

18. Cervera J, Baños A. Automatic loop shaping in QFT by using CRONE structures. *Journal of Vibration and Control* 2008; 14: 1513-1529.

19. Cervera J, Baños A. QFT loop shaping with fractional order complex pole-based terms. *Journal of Vibration and Control* 2013; 19: 294-308.

20. Nataraj P, Tharewal S. An interval analysis algorithm for automated controller synthesis in QFT designs. *Journal of Dynamic Systems Measurement and Control-Transactions of the ASME.* 2007; 129: 311-321.

21. Nataraj P, Kubal N. Automatic loop shaping in QFT using hybrid optimisation and constraint propagation techniques. *International Journal of Robust and Nonlinear Control - Special Issue: Quantitative Feedback Theory. In memoriam of Isaac Horowitz.* 2006; 17: 251-264.

22. Manoj M, Nataraj P. Automated Synthesis of fixed structure QFT controller using Interval Constraint satisfaction techniques. In: IFAC. ; 2008; Seoul, Korea: 4976-4981.

23. Rambabu K, Nataraj P. Synthesis of fractional order QFT controllers using interval constraint satisfaction technique. In: IFAC. ; 2010; Badajoz, Spain: 4976-4981.

24. Mukesh D, Nataraj P. Automated synthesis of multivariable QFT controller using interval constraint satisfaction technique. *Journal of Process Control* 2012; 22(8): 751–765.

25. Jeyasenthil R, Nataraj P. An interval-consistency-based hybrid optimization algorithm for automatic loop shaping in quantitative feedback theory design. *Journal of Vibration and Control* 2015; 23(3): 414–431.

26. Moore RE. *Methods and applications of interval analysis*. Philadelphia, PA, EE.UU.: SIAM . 1979.

27. Hansen E, Walster G. *Global optimization using interval analysis: revised and expanded*. New York, NY: CRC Press . 2003.

28. Kearfott R. *Rigorous Global Search: Continuous Problems*. Dordrecht, The Netherlands: Kluwer Academic Publishers . 2003.

29. Wie B, Bernstein D. A benchmark problem for robust control system design. In: ACC. ; 1990; San Diego, CA, USA.

30. Chiang Y, Safonov R. *Robust Control Toolbox for use with MatLab*. Natick, MA, USA: The MathWorks, Inc. . 2001.

## AUTHOR BIOGRAPHY

**Isaac Martínez-Forte.** Isaac Martínez-Forte was born in Yecla (Murcia), Spain, in 1989. He received his graduate degree in Computer Engineering from the University of Murcia, Spain, in 2015. From 2014 to 2019 he has worked as a predoc student on QFT automatic loop shaping. During 2018 and 2019 he worked as lecturer at the Computer Engineering Faculty of the University of Murcia, and since 2020 at the Computer Engineering Polytechnic School of the University of Alicante. His research interests include robust control, especially QFT, with applications in process control.

**Joaquín Cervera López.** Born in Cartagena (Murcia), Spain, in 1976, he received the graduate and Ph.D. degrees in Computer Engineering from the University of Murcia, Spain, in 1999 and 2006, respectively. From 1999 to 2001 he held a grant from the Séneca Foundation (Spain) as a Ph.D. Student, working on nonlinear QFT at the University of Murcia and on port-Hamiltonian systems on the Applied Mathematics Faculty of the University of Twente (Netherlands). Since 2001 he has worked as lecturer and researcher at the Computer Engineering Faculty of the University of Murcia, and as assistant professor sinc 2019. His research interests include robust control, applied to process control, and CACSD tools, especially QFT ALS.

**How to cite this article:** I. Martínez-Forte, and J. Cervera (2019), Speedup of QFT interval automatic loop shaping algorithm, *IJRNC*, *2019;xx:x–x.*