Synthesis of Fractional-order QFT Controllers using Interval Constraint Satisfaction Technique

Rambabu Kalla * **P. S. V. Nataraj** **

* *System & Control Engineering, Indian Institute of Technology Bombay, Mumbai, India (e-mail: ram@sc.iitb.ac.in).*
** *System & Control Engineering, Indian Institute of Technology Bombay, Mumbai, India (e-mail: nataraj@sc.iitb.ac.in).*

Abstract: A method is proposed for automatic synthesis of quantitative feedback theory (QFT) based robust fractional-order controllers. The problem of QFT controller synthesis is formulated as an interval constraint satisfaction problem, where the nonlinear and non-convex robust stability and tracking bound specifications form the constraints. The problem is solved using the branch and prune algorithm. Among all the obtained feasible controllers, the one with least high frequency gain is selected for control. The method is demonstrated to design a PD^{β} and a generic fractional-order controller for a DC motor plant transfer function having parametric uncertainties.

Keywords: Constraint propagation, Constraint satisfaction problems, Fractional-order systems, Interval analysis, Quantitative feedback theory, Parameter uncertainty, Reliable computation.

1. INTRODUCTION

The QFT approach to robust control synthesis is appealing as an engineering based approach, and can be considered as an extension of classical control. In this paper, the QFT controller synthesis problem is formulated as an ICSP based on quadratic inequalities, where the constraint set comprises of the non-convex nonlinear magnitude-phase QFT bounds at the various design frequencies. The QFT bounds arise from quadratic inequalities for the robust stability and performance specifications. A fixed-structure fractional-order controller is assumed, and the ICSP is solved efficiently using the branch and prune technique described in section 2.4.

The proposed approach finds all feasible controller solutions (in a given search box). From among the feasible solutions, the optimal solution (in the QFT sense) can be picked as the one with the minimum high frequency gain (Horowitz [1993]). The reliability of the design follows from the usage of interval arithmetic, and robustness of the design from QFT.

The salient features of the proposed approach are as follows:

- It is a fully automated approach.
- It enables the designer to specify in advance the structure of the fractional-order controller to be synthesized.
- It can deal directly with the numerical values of the possibly non-convex, nonlinear QFT bounds at each design frequency.
- If, for the specified structure and the given search box of controller parameter values, a feasible controller does exist, then the method is guaranteed to find all controllers lying within the search box.
- It is a reliable approach (i.e., it takes into account all kinds of computational errors). This property comes from usage of interval arithmetic in all computations.
- It can be used to readily find the "optimal" controller solution from the generated set of feasible controller solutions. The optimal controller in the QFT sense is the one having minimum high frequency gain (for the controller structure in (12) below, the high frequency gain is the controller parameter k). The optimal QFT controller can be thus found by simple sorting based on the obtained values of high frequency gain k.
- It finds controller solutions that are robust to the given parametric uncertainty. This follows from the satisfaction of all the QFT design inequalities at each design frequency.

The proposed approach is demonstrated on an example of electric motor with large parametric uncertainty. For this example, fractional-order controllers of two different structures are synthesized: a fractional PD, and a more general one.

The rest of the paper is organized as follows: The background of QFT, interval arithmetic, and interval constraint processing is given in section 2. The QFT controller synthesis problem is formulated in section 3. In section 4, the methodology to solve the problem is given. In section 5, the proposed approach is demonstrated on an electric motor example. Section 6 gives the conclusions of this paper. The HC4 pruning (filtering) is illustrated in appendix A.

2. BACKGROUND

In this section, we give a brief outline of the essentials of QFT, interval arithmetic, interval constraint processing and a branch and prune algorithm to solve ICSPs. We start with the outline of QFT.

2.1 Quantitative feedback theory

Quantitative Feedback Theory (QFT) (Horowitz [1991]) has been applied in many engineering systems successfully. The basic idea in QFT is to convert the given design specifications and plant uncertainties into robust stability and performance bounds in the Nichols chart. Then, a controller is designed to satisfy the bounds using gain-phase loop shaping techniques. The main objective is to design a controller for an uncertain plant such that the cost of feedback is minimized, and all robust stability and performance specifications are satisfied. The most important feature of QFT is that it is able to deal with fairly complicated uncertain plants, for large uncertainties.

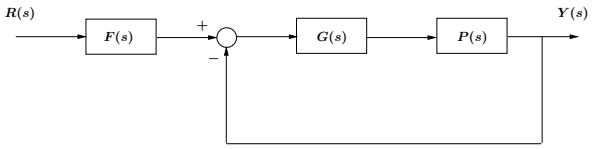

Fig. 1. The two degree of freedom structure in QFT.

Consider the two degree of freedom configuration shown in Fig. 1, where $G(s)$ and $F(s)$ are the controller and prefilter respectively. The uncertain linear time-invariant plant $P(s)$ is given by $P(s) \in \{P(s,\lambda) : \lambda \in \boldsymbol{\lambda}\}$, where $\lambda \in \mathcal{R}^l$ is a vector of plant parameters whose values vary over a parameter box $\boldsymbol{\lambda}$ given by

$$\lambda = \{\lambda \in \mathcal{R}^l : \lambda_i \in [\underline{\lambda_i}, \overline{\lambda_i}], \underline{\lambda_i} \leq \overline{\lambda_i}, i = 1, ..., l\} \quad (1)$$

This gives rise to a parametric plant family or set

$$\mathcal{P} = \{P(s,\lambda) : \lambda \in \boldsymbol{\lambda}\}$$

The open loop transmission function is defined as

$$L(s,\lambda) = G(s)P(s,\lambda) = g(j\omega)e^{j\phi(j\omega)}p(j\omega)e^{j\theta(j\omega)} \quad (2)$$

where $G(s)$ is controller transfer function. The nominal open loop transmission function is

$$\begin{aligned} L_0(s) &= G(s)P(s,\lambda_0) = g(j\omega)e^{j\phi(j\omega)}p_0(j\omega)e^{j\theta_0(j\omega)} \\ &= l_0(j\omega)e^{j\psi_0(j\omega)} \end{aligned} \quad (3)$$

where

$$l_0(j\omega) = g(j\omega)p_0(j\omega)$$

and

$$\psi_0(j\omega) = \phi_(j\omega) + \theta_0(j\omega)$$

The objective in QFT is to synthesize $G(s)$ and $F(s)$ such that the various stability and performance specifications are met for all $P(s) \in \mathcal{P}$. In general, the following specifications are considered in QFT:

(1) Robust stability

$$\left|\frac{L(j\omega)}{1+L(j\omega)}\right| \leq \infty \quad (4)$$

(2) Robust stability margin

$$\left|\frac{L(j\omega)}{1+L(j\omega)}\right| \leq w_s \quad (5)$$

(3) Robust tracking performance

$$|T_L(j\omega)| \leq \left|\frac{F(j\omega)L(j\omega)}{1+L(j\omega)}\right| \leq |T_U(j\omega)| \quad (6)$$

(4) Robust input disturbance rejection performance

$$\left|\frac{P(j\omega)}{1+L(j\omega)}\right| \leq w_{d_i}(\omega) \quad (7)$$

(5) Robust output disturbance rejection performance

$$\left|\frac{1}{1+L(j\omega)}\right| \leq w_{d_o}(\omega) \quad (8)$$

where w_s is the stability margin specification, $T_L(j\omega)$ and $T_U(j\omega)$ are the lower and upper tracking performance specifications, while w_{d_i} and w_{d_o} are the input and output rejection performance specifications.

In practice, the objective is to satisfy the given specifications over a finite design frequency set Ω.

The main steps of the QFT design process are

(1) **Generating plant templates**: For a given uncertain plant $P(s) \in \mathcal{P}$, at each design frequency $\omega_i \in \Omega$, calculate the template or value set of the plant $P(j\omega)$ in the complex plane.
(2) **Computation of QFT bounds**: At each design frequency ω_i, translate the stability and performance specifications using the plant templates to obtain the stability and performance bounds in the Nichols chart. The bound at ω_i is denoted as $B_i(\angle L_0(j\omega), \omega_i)$, or simply $B_i(\omega_i)$.
(3) **Design of controller**: Design a controller $G(s)$ such that
   - The bound constraints at each design frequency ω_i are satisfied.
   - The nominal closed loop system is stable.
(4) **Design of prefilter**: Design a prefilter $F(s)$ such that the robust tracking specifications are satisfied.

For a detailed description of the QFT design procedure, the reader is referred to (Horowitz [1993]).

Robust fractional-order controller synthesis using QFT principles has been addressed only recently. Robust fractional PID controllers are designed using QFT principles in (Cervera and Banos [2006b]) and (Nataraj and Tharewal [2007]). Automatic loop shaping based on QFT using CRONE structures is dealt in (Banos and Barreiro [2006], Cervera and Banos [2008, 2006a]).

2.2 Interval arithmetic

The key idea behind interval arithmetic as given in (Moore et al. [2009]) is the approximation of real numbers by intervals to quantify the errors introduced with finite precision arithmetic. In addition, interval computations provide an appropriate framework to deal with uncertain data.

Basic calculus on intervals with care for rounding errors was first developed by M. Warmus in (Warmus [1956]) and (Warmus [1961]). Modern interval arithmetic was developed independently in late 1950s by several researchers, including M. Warmus (Warmus [1956]), T. Sunaga (Sunaga [1958]) and R. E. Moore (Moore [1959]). Later on, R.

E. Moore enriched the research in this direction with his PhD thesis (Moore [1962]) and later on came up with the first foundational book on interval analysis (Moore [1966]) and many further publications. Owing to his excellent contributions, R. E. Moore is regarded as a founding father of interval arithmetic and interval analysis. Traditional mathematical methods sometimes produce erroneous results because of the presence of rounding errors in computers. This type of errors can be handled by using interval arithmetic as an alternative to finite precision arithmetic. So the interest in interval arithmetic has been growing and has been successfully used in numerous computing methods including real world applications.

The essential notations of interval analysis are given below.

(1) A real interval $\mathbf{x}$ is a closed and bounded set $\mathbf{x} = [\underline{x}, \overline{x}] = \{x \in \mathbb{R} : \underline{x} \leq x \leq \overline{x}\}$, where $\underline{x}$ and $\overline{x}$ are lower and upper endpoints of the interval.
(2) The set of all intervals in denoted by $\mathbb{IR}$.
(3) An interval vector is a vector whose elements are intervals. Let n be the number of elements of the real vector $(x_1, x_2, ..., x_n) \in \mathbb{R}^n$. Then, $\mathbf{x}$ denotes the the n-dimensional interval vector $(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$.
(4) An interval vector with $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$ with components $\mathbf{x_k} = [\underline{x}_k, \overline{x}_k]$ is called as box $\mathbf{x}$. The term box is used as a synonym for interval vector and rectangle for a 2-dimensional interval vector.
(5) The set of all boxes of dimension n is denoted by $\mathbb{IR}^n$.
(6) Two intervals are equal if their corresponding endpoints are equal.
(7) The width of an interval $\mathbf{x}$ is defined and denoted by $w(\mathbf{x}) = \overline{x} - \underline{x}$
(8) The midpoint of $\mathbf{x}$ is given by $mid(\mathbf{x}) = m(\mathbf{x}) = \frac{1}{2}(\underline{x} + \overline{x})$.
(9) The radius of $\mathbf{x}$ is given by $rad(\mathbf{x}) = r(\mathbf{x}) = \frac{1}{2}(\overline{x} - \underline{x})$.
(10) The intersection of two intervals $\mathbf{x}$ $\mathbf{y}$ is empty if $\underline{x} > \overline{y}$ or $\overline{x} < \underline{y}$.
(11) The relations $\in$, $\subset$, $\cap$, $\cup$, etc., are all defined component-wise.

Other definitions and general results of interval arithmetic like natural inclusion function, hull, union, projection, etc., can be found in the books (Moore [1979]) and (Moore et al. [2009]). Interval methods for solving system of equations can be found in (Neumaier [1990]) and methods for optimization problems in (Hansen and Walster [2004]).

The finite nature of computers prevents an exact representation of many real numbers. In practice, the real set, $\mathbb{R}$, is therefore approximated by a finite set $\mathbb{F}_\infty \equiv \mathbb{F} \cup \{-\infty, +\infty\}$, where $\mathbb{F}$ is the set of floating-point numbers (Goldberg [1991]). The set of real intervals is then approximated by the set, $\mathbb{I}_\diamond$, of closed floating-point intervals with bounds in $\mathbb{F}_\infty$. The concept of an interval hull is extended to that of an $\mathbb{F}$-hull, the smallest interval box in $\mathbb{I}_\diamond^n$ containing $S \subseteq \mathbb{R}^n$, and is denoted by $\square^{\mathbb{F}} S$. The other concepts are also extended with respect to the inclusion property. The power of interval arithmetic lies in its implementation on computers. In particular, outwardly rounded interval arithmetic allows computing rigorous enclosures for the ranges of operations/functions. The arithmetic operations $(+, -, *, /)$ can be made rigorous by adding suitable rounding controls. This makes a qualitative difference in scientific computations because the results are now intervals in which the exact result must lie. Interval arithmetic can be carried out for virtually any arithmetic expression that can be evaluated with floating-point arithmetic. However, expressions that are equivalent in real arithmetic differ in interval arithmetic because interval arithmetic is only subdistributive. Therefore, computations should be arranged so that the overestimation of the ranges of functions is minimized.

2.3 Interval constraint processing

The two areas that have had the greatest impact on a modern theory of constraints and their use in automated problem solving are operations research (OR) and artificial intelligence (AI). Operations research is concerned with building mathematical models of real world situations to allow the experimental analysis of problems. Artificial intelligence is concerned with intelligently accelerating automated problem solution techniques. The concept of solving constraints by using constraint propagation was developed independently by a number of researchers to solve arithmetic and boolean constraints. The term generally is attributed to Sussman and Steele (Sussman and Steele [1980]), who used constraint propagation to solve constraints in the constraint language CONSTRAINTS.

Numerical constraint satisfaction problems accept as input only problems represented by exact numerical values and correspondingly produce only crisp solutions as output. This limitation can be removed by implementing generalized constraint propagation schemes based on interval arithmetic instead of conventional arithmetic. By using intervals instead of exact values, we may express inexact numerical constraints in a well-defined way and compute necessary conditions for consistency in inconsistent situations.

An interval constraint satisfaction problem (ICSP) as given in (Hyvonen [1992]) is composed of

(1) A set of real valued variables, e.g., $v = \{v_1, ..., v_n\}$;
(2) A set of interval domains of the variables, e.g., $\mathbf{x} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$;
(3) A set of constraints, e.g., $c = \{c_1, ..., c_m\}$ over the given set of variables.

The problem is to find in the initial box $\mathbf{x}_1 \times ... \times \mathbf{x}_n$ all the consistent values with respect to all constraints. A variable $v_i \leftarrow \mathbf{x}_i$ is consistent if and only if each interpretation $v_i \leftarrow x \in \mathbf{x}_i$, can be satisfied with respect to all constraints by some extension: $\forall x \in \mathbf{x}_i \; \exists \{v_1 \leftarrow x_1 \in \mathbf{x}_1, ..., v_i \leftarrow x, ..., v_m \leftarrow x_m \in \mathbf{x}_m\} : c_1, ..., c_m$ are satisfied. The set of variables of the constraint c_i is denoted by $\mathbb{V}_{c_i}$

There are two steps in solving an ICSP, constraint propagation and constraint branching. The basic idea of constraint propagation algorithms (also called filtering or narrowing or consistency algorithms or narrowing operators) consists of removing, from the domains associated to the constraint variables, inconsistent values that can never be part of the solution. This process reduces significantly the search tree and possibly the computational effort to find a solution if one exists or to demonstrate that there is no solution. In general, the results are propagated through the whole constraint set, and the process is repeated until

a stable set is obtained. Research in the area of solving interval constraint satisfaction problems (Benhamou et al. [1999]) is devoted to finding correct and (near) optimal interval propagation techniques that can be efficiently implemented. A constraint narrowing algorithm transforms the domains of those variables involved in it into tighter intervals such that:

(1) Resulting intervals are always included in the original ones (*contractance* property).
(2) All values in the original intervals verifying the associated constraint of the narrowing operator, belong to the resulting intervals (*soundness* or *correctness*).
(3) The subset interval relation is conserved by the transformation (*monotonicity*).

Well known examples of constraint narrowing operators are hull and box consistency (Benhamou et al. [1999]) and kBConsistency operators (Lhomme [1993]). In our problem, we make use of an efficient implementation of hull consistency, known as HC4, as the the narrowing operator. HC4 filter is described below.

HC4 filter: The HC4 filter was proposed in (Benhamou et al. [1999]). Inputs to the HC4 filter are the constraint in user form (i.e. without decomposing it in several equations) and the set of interval domains (box). The algorithm efficiently computes an interval extension of the equation, narrowing intervals of the variables involved. Inside the HC4 filter the input equation is represented as an attribute tree where the root node is a p-ary relation symbol, and terms in the equation form sub-trees rooted at nodes containing either a variable, a constant or an operation symbol.

The HC4 filter works in two phases called forward evaluation and backward propagation. The forward phase is a tree traversal going from the leaves to the root, evaluating at each node the natural interval extension of that subterm of the constraint. The backward phase traverses the tree from the root to the leaves, projecting on each node the effects of interval narrowing already performed on its parent node. In the backward propagation phase, an interval may become empty. When this happens the constraint is inconsistent with respect to the initial domains. HC4 algorithm is explained by means of a simple example in Appendix A. Refer to (Benhamou et al. [1999]) for an extended description.

Until now we have been dealing with the first step in solving the ICSP, i.e., constraint propagation. Constraint propagation algorithms alone are not sufficient for solving an ICSP, that is to say, they do not eliminate all the non-solution elements from the domains. As a consequence, it is necessary to employ some additional strategy to solve it. One complementary method is the so-called constraint branching that divides the variable domains to construct new sub-problems, i.e., branches in the search tree on which constraint propagation is reactivated. The process is also called as splitting or sub-division process.

2.4 Branch and prune algorithm of ICST

In this section, the branch and prune algorithm to solve an ICSP as given in (Benhamou et al. [1999], Granvilliers and Benhamou [2006]) is described.

Algorithm: Branch and prune using HC4 filter.

Inputs: The variable set $v = \{v_1, ..., v_n\}$ of the ICSP, set of initial interval domains of the variables $\mathbf{x^0} = \{\mathbf{x_1^0}, ...\mathbf{x_n^0}\}$ of the ICSP, the constraint set $c = \{c_1, ..., c_m\}$ of the ICSP, and the specified accuracy tolerance ϵ of the solution set.

Output: The solution set $\mathcal{L}^{sol}$ (boxes) of the ICSP computed to the prescribed accuracy tolerance ϵ.

Algorithm 1 (Branch and prune algorithm)

1: Construct the natural inclusion functions for each constraint in the constraint set $c = \{c_1, ..., c_m\}$.
2: Initialize the solution list $\mathcal{L}^{sol} \leftarrow \{\}$ and the working list $\mathcal{L} \leftarrow \mathbf{x^0}$.
3: **while** $\mathcal{L} \neq \{\}$ **do**
4: Extract $\mathbf{x}$ from $\mathcal{L}$.
5: $s \leftarrow c$
6: **while** $s \neq \{\}$ **AND** $\mathbf{x} \neq \{\}$ **do**
7: Extract c_i from s.
8: Narrow the box $\mathbf{x}$ using the HC4 filter (as explained in the section 2.3) to $\mathbf{x}'$.
9: **if** $\mathbf{x} \neq \mathbf{x}'$ **then**
10: $s \leftarrow s \cup \{c_j \mid \exists v_k \in \mathbb{V}_{c_j} \wedge \mathbf{x_k} \neq \mathbf{x_k'}\}$ {Add to the set s all the constraints containing the variables whose search domains are narrowed.}
11: $\mathbf{x} \leftarrow \mathbf{x}'$
12: **else**
13: $s \leftarrow s \setminus \{c_i\}$
14: **end if**
15: **end while**
16: **if** $\mathbf{x} \neq \{\}$ **then**
17: **if** $w(\mathbf{x}) \leq \epsilon$ **then**
18: $\mathcal{L}^{sol} \leftarrow \mathcal{L}^{sol} \cup \mathbf{x}$
19: $\mathcal{L} \leftarrow \mathcal{L} \setminus \{\mathbf{x}\}$
20: **else**
21: Subdivide the box $\mathbf{x}$, along the variable whose width is the largest, into two sub-boxes $\mathbf{x}^1$ and $\mathbf{x}^2$.
22: $\mathcal{L} \leftarrow \mathcal{L} \cup \mathbf{x}^1 \cup \mathbf{x}^2$
23: **end if**
24: **end if**
25: **end while**
26: Output the solution set $\mathcal{L}^{sol}$ and EXIT.

3. PROBLEM FORMULATION

Consider the two degree-of-freedom structure shown in Fig. 1, where, $G(s) = G(s, x)$ and $F(s)$ are the fractional-order controller and prefilter respectively, x is the vector of unknown controller parameters, and $P(s) \in \mathcal{P}$ is the uncertain linear time-invariant plant. The objective in QFT is to synthesize $G(s)$ and $F(s)$ such that the various stability and performance specifications are met for all $P(s) \in \mathcal{P}$. In terms of quadratic inequalities, these specifications can be written as (Chait and Tsypkin [1993], Chait and Yaniv [1993]) (see notation of section 2.1):

- Robust stability specification, see (4),

$$g^2p^2 + 2gp\cos(\phi + \theta) + 1 \geq 0 \tag{9}$$

- Robust gain-phase margin specification, see (5),

$$g^2p^2\left(1-\frac{1}{{w_s}^2}\right)+2gp\cos(\phi+\theta)+1\geq 0 \qquad (10)$$

- Robust tracking specification, see (6),

$$g^2{p_k}^2{p_i}^2\left(1-\frac{1}{\delta^2(\omega)}\right)+2gp_kp_i\left[p_k\cos(\phi+\theta_i)-\frac{p_i}{\delta^2(\omega)}\cos(\phi+\theta_k)\right]+\left({p_k}^2-\frac{{p_i}^2}{\delta^2(\omega)}\right)\geq 0 \qquad (11)$$

where

$$\delta(\omega)=\left|\frac{T_U(j\omega)}{T_L(j\omega)}\right|$$

The last inequality should be met for each possible pair $P_i(s)=p_i(j\omega)e^{j\theta_i(j\omega)}, P_k(s)=p_k(j\omega)e^{j\theta_k(j\omega)}$ of plants from the uncertain plant set $\mathcal{P}$. The objective is to satisfy the above specifications, over the plant set $\mathcal{P}$, and over the design frequency set Ω. In practice, a finite number of plants and frequencies are chosen from these sets.

In our work, the controller $G(s,x)$ is represented with the fractional-order structure

$$G(s,x)=\frac{k\prod_{i=1}^{n_z}(s^\beta+z_i)}{\prod_{k=1}^{n_p}(s^\alpha+p_k)} \qquad (12)$$

where the controller parameter vector x is

$$x=(k,\beta_1,..,\beta_{n_z},\alpha_1,\alpha_{n_p},z_1,...,z_{n_z},p_1,...,p_{n_p}) \qquad (13)$$

with αs and βs as fractional (positive) powers. Another fractional-order structure that we use is the fractional-order $PI^\alpha D^\beta$ structure of Podlubny (Podlubny [1999]):

$$G(s,x)=k_p+k_is^{-\alpha}+k_ds^\beta \qquad (14)$$

In this case, the controller parameter vector x is

$$x=(k_p,k_i,k_d,\alpha,\beta) \qquad (15)$$

The magnitude and angle functions of $G(s,x)$ are defined as

$$g_{mag}(\omega,x)=|G(s=j\omega,x)|\,;g_{ang}=\angle G(s=j\omega,x) \qquad (16)$$

Note that in the proposed approach, one can choose any fractional-order structure for the controller, as long as one can obtain the expressions for the magnitude and angle functions (this puts virtually no restriction on the controller structure that can be used).

The control design problem is to solve the above set of inequality constraints, derived for all plants in $\mathcal{P}$, and for all design frequencies in Ω. With an initial search box for x specified, this becomes an ICSP. Its solution gives the feasible set of controller parameters x. If desired, the optimal QFT controller can be extracted from the obtained feasible solution set by picking the controller having the minimum gain k (a more efficient approach would use some kind of optimization technique, but this is left for future research).

For implementation of the fractional-order controller in frequency domain, it has to be approximated to integer-order one using Oustaloup's recursive approximation technique given in (Oustaloup et al. [2000]). The digital implementation of the integer-order controller so obtained can be be done by discretizing it using well known Euler or Tustin method.

4. METHODOLOGY

The proposed approach to solve the ICSP described above is as follows:

(1) Choose a structure for the fractional-order controller, say the one in (12).
(2) Form the variable set comprising of the controller parameters $v=x$ in (13) for the ICSP.
(3) Construct the initial search box $\mathbf{x^0}$ of the controller parameters x.
(4) For the given problem specifications, construct the quadratic inequalities given by (9) to (11) for each plant and at each design frequency. From these inequalities, form the constraints of the ICSP: $c \leftarrow \{c_1,..c_i\}$.
(5) Solve the ICSP using the branch and prune algorithm described in section 2.4.
(6) The output of the algorithm is the set of all feasible controller parameters satisfying the QFT bound constraints to the prescribed accuracy ϵ. If desired, one can sort the obtained feasible solution set to obtain optimal QFT controller having minimum gain k (this controller solution is recommended from a QFT viewpoint for actual) implementation.

5. DESIGN EXAMPLE

The proposed approach is demonstrated on an example of a DC motor with uncertain plant parameters. Two different controllers are designed, to show the capabilities of the approach. The design is executed on a computer with Intel® Core™2 Duo 2.4 GHz processor and 2 GB of RAM, running Linux Fedora Core-7. The method is implemented using *RealPaver* (Granvilliers and Benhamou [2006]).

Example 5.1. Consider the uncertain plant transfer function for the speed loop of a DC Motor (D'Azzo and Houpis [1995])

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

The design specifications for this problem are as follows: stability, gain margin of at least 3 dB, no overshoot, settling time between 2 and 4.5 seconds, and zero steady state error. These specifications are translated into the frequency domain as (see notation of section 2.1)

- Robust stability margin spec, $w_s=1.2$
- Tracking performance spec,

$$T_U(s)=\frac{1.5}{(s+1.5)}$$
$$T_L(s)=\frac{1}{(s+1)^2}$$

The nominal plant parameters are $k_0=1$ and $a_0=1$.

Using the proposed approach, first a fractional PD^β controller of the form $K_p+K_ds^\beta$is synthesized. The initial search domain for the controller parameters are taken as

$$K_p = [0, 1000], K_d = [1.96, 2.95], \beta = [0.524, 0.787]$$

The controller solutions are to be found to an accuracy $\epsilon = 0.001$.

The parametric plant uncertainty is captured by a set of 9 plants obtained from the combination of the minimum, mean, and maximum values of the k and a parameter intervals. The design frequency set is

$$\Omega = [0.001, 0.015, 0.25, 3.84, 60]$$

. The above specifications are converted into a set of quadratic inequalities, for each plant and each design frequency. This gives a set of inequality constraints, where the controller parameters K_p, K_d, β are the unknown variables. The ICSP is solved using the branch and prune algorithm given in section 2.4, and all feasible controllers in the search region are obtained in about 24. The controller with minimum high frequency gain is chosen (by simple sorting of all the feasible controller solutions) as

$$G(s) = 2.785 + 1.968 s^{0.787} \tag{17}$$

Fig. 2 shows that the obtained controller achieves all the given specifications - the QFT bounds representing the inequalities are respected at each frequency by the designed $L(s)$.

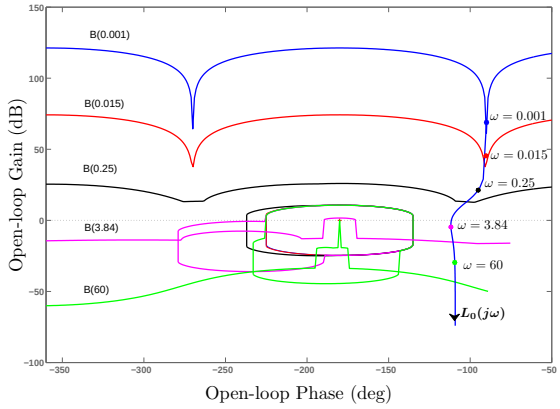

Fig. 2. Example 5.1. Plot of nominal loop transmission function corresponding to the PD^β controller in (17).

Example 5.2. The proposed approach is next demonstrated on the same DC motor plant, but for designing a slightly more general fractional-order controller structure, with different tracking specifications. The fractional-order controller structure is chosen now as

$$G(s) = \frac{k(s^\beta + z_1)}{(s + p_1)(s^\alpha + p_2)}$$

with the frequency domain specifications as

- Robust stability margin spec, $w_s = 1.2$
- Tracking performance specifications as,

$$T_U(s) = \frac{0.6854(s + 30)}{(s^2 + 4s + 19.752)}$$
$$T_L(s) = \frac{120}{s^3 + 17s^2 + 82 + 120}$$

The initial search domain for the controller parameters are taken as

$$k = [0, 10^8], \beta = [0.6, 0.9], \alpha = [0.7, 1],$$
$$z_1 = [1, 2], p_1 = [1300, 1350], p_2 = [1000, 1200]$$

The controller solutions are to be found to an accuracy $\epsilon = 0.1$.

As before, the parametric plant uncertainty is captured by a set of 9 plants obtained from the combination of the minimum, mean, and maximum values of the k and a parameter intervals. The design frequency set is

$$\Omega = [0.1, 0.5, 2, 15, 100].$$

The above given specifications are converted into a set of quadratic inequalities, for each plant and each design frequency. This gives a set of inequality constraints, where the controller parameters are the unknown variables. The ICSP is solved using the branch and prune algorithm given in section 2.4, and all feasible controllers in the search region are obtained in about 115 seconds. The controller with minimum high frequency gain is chosen (by simple sorting of all the feasible controller solutions) as

$$G(s) = \frac{10067845(s^{0.78} + 1.22)}{(s + 1312)(s^{0.995} + 1110)} \tag{18}$$

Fig. 3 shows graphically that the obtained controller respects all the QFT bound constraints.

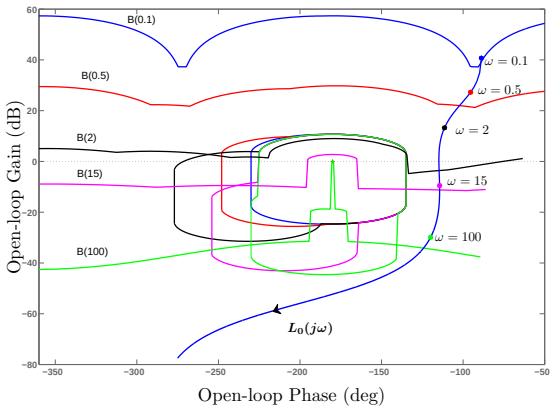

Fig. 3. Example 5.2. Plot of nominal loop transmission function corresponding to the controller in (18).

6. CONCLUSIONS

A new computationally efficient approach has been proposed for the automatic synthesis of fixed structure fractional-order QFT controllers. The approach uses the various QFT quadratic inequalities describing the robust stability and performance specifications, over the set of uncertain plants and design frequencies. The unknown variables in the inequalities are the parameters of the fractional-order controller. This approach leads to an interval constraint satisfaction problem, which is efficiently solved using the branch and prune algorithm of ICST.

The proposed approach has several notable features. It deals directly with the numerical values of the possibly non-convex, nonlinear QFT bounds, thereby avoiding the over design arising from any approximation of QFT bounds. The issues of robust stability and performance

are all rigorously taken into consideration in the proposed approach. For a given structure of controller and initial search domain, the proposed method is guaranteed to find all feasible fractional-order controllers.

REFERENCES

A. Banos and A. Barreiro. Automatic loop shaping in QFT by fractional structures. In *Proc. of IFAC Workshop on Fractional Differentiation and its Applications*, Porto, Portugal, 2006.

F. Benhamou, F. Goualard, and L. Granvilliers. Revising hull and box consistency. In *Proc. of ICLP'99 - MIT Press*, pages 230–244, 1999.

J. Cervera and A. Banos. Automatic loop shaping in QFT by fractional structures. In *Proc. of IFAC Workshop on Fractional Differentiation and its Applications*, Porto, Portugal, 2006a.

J. Cervera and A. Banos. Tuning of fractional PID controllers by using QFT. In *Proceedings IECON '06- 32nd Annual Conference of the IEEE Industrial Electronics Society*, Paris, 2006b.

J. Cervera and A. Banos. Automatic loop shaping in QFT using crone structures. *Journal of Vibration and Control*, 14(9-10):1513–1529, 2008.

Y. Chait and Y. Tsypkin. SISO QFT design with non-parametric uncertainties. In *Proc. of American Control Conference*, pages 1694–1695, Masachusetts, USA, 1993.

Y. Chait and O. Yaniv. Multi-input/single-output computer aided control design using Quantitative Feedback Theory. *International Journal of Robust and Nonlinear Control*, 3(1):47–54, 1993.

J. J. D'Azzo and C. H. Houpis. *Linear Control System Analysis and Design,4th edition*. McGraw-Hill, New York, 1995.

D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

L. Granvilliers and F. Benhamou. Realpaver: An Interval Solver using Constraint Satisfaction Techniques. *ACM Transaction on Mathematical Software*, 32(1):138–156, 2006.

E. Hansen and G. Walster. *Global Optimization using Interval Analysis, second edition*. Marcel Dekker, New York, 2004.

I. M. Horowitz. *Quantitative Feedback Theory (QFT)*. QFT Publications, Boulder, Colorado, 1993.

I. M. Horowitz. Survey of quantitative feedback theory (QFT). *Int. J. Control*, 53(2):255–291, 1991.

E. Hyvonen. Constraint reasoning based on interval arithmetic - the tolerance propagation approach. *Artificial Intelligence*, 58:71–112, 1992.

O. Lhomme. Consistency techniques for numeric CSPs. In *Proceedings of the 13th IJCAI, IEEE Computer Society Press*, pages 232–238, 1993.

R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

R. E. Moore. *Interval Arithmetic and Automatic Error Analysis in Digital Computing*. PhD dissertation, Department of Computer Science, Stanford University, 1962.

R. E. Moore. Automatic error analysis in digital computation. Technical Report LMSD- 84821, Missiles and Space Division, Lockheed Aircraft Corporationa, Sunnyvale, California, USA, 1959.

R. E. Moore. *Methods and Applications of Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1979.

R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

P. S. V. Nataraj and Sachin Tharewal. On fractional-order QFT controllers. *Trans. ASME J. Dynamic Systems, Measurement, and Control*, 129:212–218, 2007.

A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge Univ. Press, Cambridge, 1990.

A. Oustaloup, F. Levron, B. Mathieu, and F. M. Nanot. Frequency-band complex noninteger differentiator: Characterization and synthesis. *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, 47(1):25–39, 2000.

I. Podlubny. Fractional order systems and P I(lambda) D(mu)-controller. *IEEE Transactions on Automatic Control*, 44(1):208–214, 1999.

Teruo Sunaga. Theory of an interval algebra and its applications to numerical analysis. *RAAG Memoirs*, 2: 29–46, 1958.

G. Sussman and G. Steele. CONSTRAINTS: A language for expressing almost-hierarchical descriptions. *Artificial Intelligence*, 14(1):1–39, 1980.

M. Warmus. Calculus of approximations. *Bulletin de l'Acad mie Polonaise des Sciences*, 5(4):253–259, 1956.

M. Warmus. Approximations and inequalities in the calculus of approximations. *Bulletin de l'Acad mie Polonaise des Sciences*, 9(4):241–245, 1961.

Appendix A. ILLUSTRATION OF HC4 FILTER

Let us consider a constraint $C : x^2 + y^2 = 1$. Here the variable set is $\{x, y\}$. Let the initial search domain for x and y be $[-10, 10]$. First step is to form the tree using constraint as shown in Fig. A.1. The operands and constants occupy the leaf nodes whereas the operators are be placed at the parent nodes.

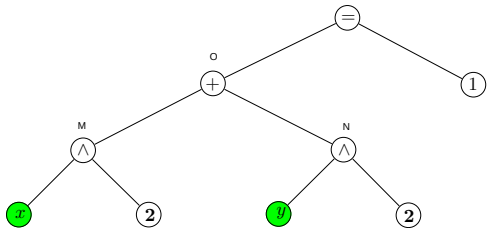

Fig. A.1. Binary tree constructed using the constraint $C : x^2 + y^2 = 1$

The forward evaluation is done using the left-hand and right-hand parts of the equation using interval arithmetic, saving at each node the result of the local evaluation as shown in Fig. A.2. In the backward propagation step the expression tree is swept from top to bottom (see Figs. A.3, A.4, and A.5), the domains computed during the forward evaluation are used to project the relation at each node on the remaining variables. The values evaluated during backward propagation are shown in left side of the node in *bold*. First we start at the root node and from the constraint the right hand side should be equal to left hand

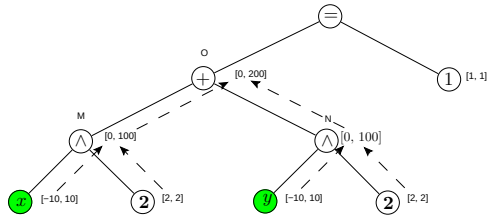

Fig. A.2. Forward evaluation

side, so the value at the node O should be equal to $[1, 1]$. The updated tree is shown in Fig. A.3. The old values are shown in ellipses on the right side of the node and the new values are shown in bold on the left side of the node.

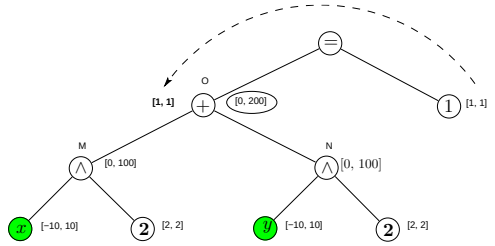

Fig. A.3. Backward propagation - 1

New value at node M is evaluated as given below.

$$\begin{aligned}\mathbf{M_{new}} &= \mathbf{M_{old}} \cap (\mathbf{O_{new}} - \mathbf{N_{old}}) \\ &= [0, 100] \cap ([1, 1] - [0, 100]) \\ &= [0, 100] \cap [-99, 1] \\ &= [0, 1]\end{aligned}$$

New value at the node N is evaluated using the new value of node M

$$\begin{aligned}\mathbf{N_{new}} &= \mathbf{N_{old}} \cap (\mathbf{O_{new}} - \mathbf{M_{new}}) \\ &= [0, 100] \cap ([1, 1] - [0, 1]) \\ &= [0, 100] \cap [0, 1] \\ &= [0, 1]\end{aligned}$$

The updated tree is as shown in Fig. A.4 below.

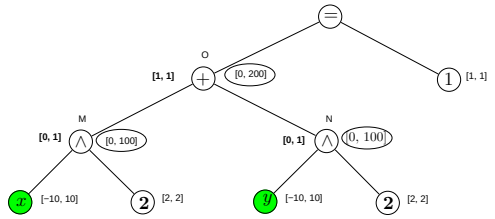

Fig. A.4. Backward propagation - 2

The new evaluation of $\mathbf{x^2}$ is the new value at the node M

$$\begin{aligned}\mathbf{x_{new}} &= \mathbf{x_{old}} \cap (\mathbf{M_{new}})^{\mathbf{0.5}} \\ &= [-10, 10] \cap ([0, 1]^{0.5}) \\ &= [-10, 10] \cap [-1, 1] \\ &= [-1, 1]\end{aligned}$$

Similarly the new value of $\mathbf{y}$ is evaluated as shown below.

$$\begin{aligned}\mathbf{y_{new}} &= \mathbf{y_{old}} \cap (\mathbf{N_{new}})^{\mathbf{0.5}} \\ &= [-10, 10] \cap ([0, 1]^{0.5}) \\ &= [-10, 10] \cap [-1, 1] \\ &= [-1, 1]\end{aligned}$$

The updated tree is as shown in Fig. A.5

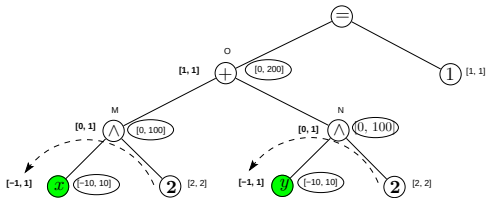

Fig. A.5. Backward propagation - 3

After backward propagation is complete, the search domain contracts i.e., the inconsistent search domain will be thrown out. The updated values of both $\mathbf{x}$ and $\mathbf{y}$ are [-1, 1]. The updated tree after complete backward propagation is shown in Fig. A.6.

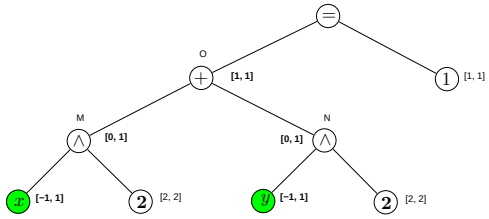

Fig. A.6. Updated tree after complete backward propagation

So the search domain (box) $\mathbf{x} \times \mathbf{y}$ shrinks from $[-10, 10] \times [-10, 10]$ to $[-1, 1] \times [-1, 1]$ after one iteration of HC4.