Automated Synthesis of Fixed Structure QFT Prefilters using Interval Constraint Satisfaction Techniques

P. S. V. Nataraj and M. M. Deshpande
Indian Institute of Technology Bombay, Mumbai 400 076, India.
nataraj@sc.iitb.ac.in, mmdeshpande@sc.iitb.ac.in

Abstract. A new, computationally efficient approach for automation of the prefilter design step of Horowitz's quantitative feedback theory (QFT) to robust feedback system synthesis is proposed. In the proposed approach, we pose the prefilter design problem as an interval constraint satisfaction problem (ICSP) and solve it using well-established interval constraint satisfaction techniques (ICST). To validate the above design approach we apply this to a benchmark problem to obtain simple, low order prefilters in quick time.

Keywords: Interval Analysis; Interval Constraint Satisfaction; Prefilter Design; Robust Control System; QFT.

1. Introduction

Quantitative feedback theory (QFT) is well-known technique for designing robust feedback systems for the plants having large parametric uncertainties. In the two-degree-of-freedom structure used in QFT, a prefilter is required to meet the desired tracking specifications. The synthesis of prefilter in QFT is usually carried out manually, and largely depend on the designers skill and experience. Thus, a good design is not always assured by the manual design process. The benefits of automatic design of prefilter are quite obvious. For instance, in the $n \times n$ MIMO design case of QFT, where n^2 number of prefilters are designed sequentially, any overdesign of the prefilter in a loop may lead to controller overdesign in the subsequent steps. In the QFT literature, no automated method is available for the synthesis of prefilter, with the exception of (Tharewal, 2005). However, the method in (Tharewal, 2005) does not find all feasible prefilters of a specified structure.

In this paper, we pose QFT prefilter synthesis problem as an interval constraint satisfaction problem and solve using existing efficient interval constraint satisfaction techniques (ICST) like HC4 and BC5. In the proposed approach, if a solution of the specified structure exists for the given parameter domain, then all prefilter solutions lying in the domain are generated. It also computationally verify the existence (or non-existence) of a QFT prefilter solution, for a specified prefilter structure and an initial domain of prefilter parameter values. To validate the above design approach we apply this to a benchmark problem to obtain simple, low order prefilters in quick time. The main features of the proposed approach are as follows:

1. The QFT prefilter synthesis is fully automated.
2. The structure of the prefilter transfer function can be pre-specified by the designer.

4th International Workshop on Reliable Engineering Computing (REC 2010)
Edited by Michael Beer, Rafi L. Muhanna and Robert L. Mullen

ISBN: 978-981-08-5118-7. *Published by* Research Publishing Services.
doi:10.3850/978-981-08-5118-7_055

3. The method uses interval analysis techniques to reliably find all feasible prefilter parameters in a given initial search domain.

4. The method uses, in particular, existing ICST-based algorithm for efficient computation of the feasible prefilter parameters.

The rest of paper is organized as follows: In Section 2, we outline the background of the essentials. In section 3, we first formulate the QFT prefilter design problem as an ICSP, and then solve it using existing ICST-based algorithm. In Section 4, we demonstrate the approach described in Section 3 on a benchmark problem. In Section 5, we demonstrate the non-existence verification of prefilter solution. Conclusions of the work is given in Section 6.

2. Background

We give a brief outline of the essentials of interval analysis, interval constraint satisfaction techniques, and quantitative feedback theory.

2.1. Interval Analysis

Interval analysis deals with the use of interval sets and the arithmetic calculations involved with the interval set operations. Outwardly rounded interval computations allow rigorous enclosures to be found for the ranges of elementary operations and functions. Thus, interval computations are used as a tool for the so-called validated computations, i.e., computations with guaranteed accuracy taking into account all kinds of computational errors.

$\mathbb{R}$ denotes the field of real numbers and $\mathbb{R}^n$ the vector space of column vectors of length n with real entries. A real, closed, nonempty interval is a pair $\mathbf{x} = [\underline{x}, \overline{x}]$ consisting of two real numbers $\underline{x}$ and $\overline{x}$ with $\underline{x} < \overline{x}$. The set of all such intervals is denoted by $\mathbb{IR}$. An interval vector $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_n)^T$ with components $\mathbf{x}_k = [\underline{x}_k, \overline{x}_k]$ is called as a box $\mathbf{x}$. The set of all boxes of dimension n is denoted by $\mathbb{IR}^n$.

The *lower bound* of an interval $\mathbf{x}$ is $\inf(\mathbf{x}) := \underline{x}$, its *upper bound* is $\sup(\mathbf{x}) := \overline{x}$, and its midpoint is $\mathrm{mid}(\mathbf{x}) := \frac{1}{2}(\underline{x} + \overline{x})$. The *width* of a interval $\mathbf{x}$ is denoted by $\mathrm{wid}(\mathbf{x}) := \overline{x} - \underline{x} \geq 0$. A set inclusion $\mathbf{x} \subseteq \mathbf{y}$ is true only when $\underline{y} \leq \underline{x}$ and $\overline{y} \geq \overline{x}$.

Interval arithmetic (IA) is an arithmetic operations over intervals. It is extension of real arithmetic operations. Consider two intervals $\mathbf{x} = [a, b]$ and $\mathbf{y} = [c, d]$, then we have:

$$\mathbf{x} + \mathbf{y} = [a + c, b + d]$$

$$\mathbf{x} - \mathbf{y} = [a - d, b - c]$$

$$\mathbf{x} \times \mathbf{y} = [min(ac, ad, bc, bd), max(ac, ad, bc, bd)]$$

$$\mathbf{x} \div \mathbf{y} = [min(a/c, a/d, b/c, b/d), max(a/c, a/d, b/c, b/d)] \text{if} 0 \ni [c, d]$$

Interval arithmetic is associative and commutative but not distributive. It instead has a subdistributive property i.e., for $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z} \in \mathbb{R}$,

$$\mathbf{x} \times (\mathbf{y} + \mathbf{z}) \subseteq \mathbf{x} \times \mathbf{y} + \mathbf{x} \times \mathbf{z}$$

The left and right side interval evaluations of the above expression may not have equivalent natural extensions due to dependency problem (i.e., occurrence of a variable more than once).

Interval analysis can represent outer approximations of real numbers. The range of a function f over $\mathbf{x}$ is denoted as

$$\text{range}(f, \mathbf{x}) = \{f(x) \mid x \in \mathbf{x}\},$$

can be obtained by interval extension. An interval extension of a real function $f : \mathbb{R}^n \to \mathbb{R}$ is a function $F : \mathbb{I}^n \to \mathbb{I}$ such that

$$\forall \mathbf{x} \in \mathbb{I}^n, (\{f(x) \mid x \in \mathbf{x}\} \subseteq F(\mathbf{x})).$$

This is considered as fundamental theorem in IA. Interval extensions are also known as interval functions or interval forms. The definition of interval extension implies existence of many interval extensions of a given real function.

A natural interval extension of a real function f, whose value over a box $\mathbf{x}$ is denoted by $f(\mathbf{x})$, is obtained by replacing each occurrence of each component x_i of x by the corresponding interval component $\mathbf{x}_i$ of $\mathbf{x}$ and, by executing all operations according to interval arithmetic rules. This evaluation is called as natural interval evaluation.

Different representations of the same real function f generally lead to different natural interval extensions. The overestimation problem, is due to occurrence of a variable more than once in interval evaluation. This problem is known as dependency problem of IA.

The reader is referred to (Moore, 1966; Hansen and Walster, 2005) for details of interval analysis.

2.2. Interval constraint satisfaction techniques

Constraint satisfaction problem (CSP) A constraint is a relation between the variables or unknowns of a problem, each taking a value in a domain. A constraint satisfaction problem (CSP) is given by a set of constraints (expressing the relations between the unknowns of a problem i.e., variables,) and search space defined as the Cartesian product of variable domains i.e., search box.

$$c_i(x_1, x_2) \geq 0, \text{for}(i = 1, ..., n),$$

The solution set of a CSP, c_i is the set of all elements from the search space, $x = x_1, x_2$ that satisfy all the constraints.

The nonlinear and non-convex constraint satisfaction over real numbers is generally NP-hard problem due to the limitation of machine arithmetic. Hence, constraint satisfaction over intervals is used.

2.2.1. *Interval constraint satisfaction problem (ICSP)*

The term interval constraint is a generic one denoting a constraint in which variables are associated with intervals denoting their domains of possible values. When interval are defined over reals, interval constraint sets are often called as continuous or numeric constraint satisfaction problem (Granvilliers and Benhamou, 2001). An interval constraint satisfaction problem (ICSP) can be stated as follows.

$$\mathcal{C}_i(x) \geq 0, for(i = 1, ..., n), x \in \mathbf{x}$$

where x is the one dimensional parameter vector belonging to a given search box $\mathbf{x}$ and f_i are the nonlinear, non-convex constraints to be satisfied. Constraint solving is obtaining those values of the variables from its domain which satisfy the given constraint. Classical techniques for solving ICSPs are based on a branch and bound algorithm (Granvilliers and Benhamou, 2001). In this method, a box is first evaluated and then split in two sub-boxes and f_i is computed for both the sub-boxes. The boxes which do not satisfy the constraints are discarded. The algorithm terminates when further splitting does not result in an increase in the accuracy under estimate or all the box in the list are over. Though splitting of the box helps in removing infeasible solution from domain, it makes interval algorithms very slow.

2.2.2. *Constraint solver*

Interval techniques are inherently slow due to two reasons: (a) globalness of the algorithm and (b) the bisections involved in these algorithms. The latter can be reduced with the help of techniques such as constraint consistency techniques (Jaulin, 2001).

To speed up the algorithm, pruning techniques such as constraint consistency techniques are used along with splitting. The box pruning is a reduction procedure, removing values that do not belong to the solution set. In particular, a box is eliminated if at least one constraint is proved to be violated for all points from the box. These techniques usually try to satisfy some form of local consistency, described as a common fixed point of domain reduction functions. The reduction functions (also called as ICST, contractors, filtering functions) for interval domains can be found in (Benhamou and Older, 1997; Jaulin, 2001). Jaulin *et al.* (Jaulin, 2001) describe various contractors based on Gauss elimination, Gauss-seidel algorithm, Krawczyk method, and Newton methods.

The pruning step combines different techniques to narrow down the domains by removing locally inconsistent intervals containing no solution of some constraint. There are different types of consistencies but the main ones are box and hull consistency. We can apply these consistency to each constraint in the constraint set to eliminate subboxes of the given box $\mathbf{x}$, that does not contain the solution. Some important notions used in consistency techniques are given below:

Consider a n-ary constraint c, a box $x_1 \times ... \times x_n \in \mathbb{I}^n$. Let $i \in \{1, ...n\}$ be a natural. The i-th projection of c is the set

$$\pi_i(c) = \{x_i | \exists x_1 \in \mathbf{x_1}, ..., \exists x_n \in \mathbf{x_n}, (x_1, .., x_n) \in c\}.$$

Consider a constraint $c(x_1, .., x_n)$ and a box $\mathbf{x}$. Let C be the natural extension of c. Let $i \in \{1, ...n\}$ be a natural. We say x_i is box consistent w. r. t. i-th projection of c if

$$\mathbf{x}_i = \text{Hull}(\{a_i \in \mathbf{x}_i | (\mathbf{x}_1, ...\text{Hull}(a_i), ...\mathbf{x}_n) \in C\}).$$

Consider a constraint $c(x_1, .., x_n)$ and a box $\mathbf{x}$. Let $i \in \{1, ...n\}$ be a natural. We say x_i is hull consistent w. r. t. i-th projection of c if

$$\mathbf{x}_i = \text{Hull}\pi_i(c, \mathbf{x}).$$

In hull consistency, the structure of the initial constraints is broken. The new variables are introduced by decomposing the original constraints. For the constraints having occurrence of a variable once in a constraint, this is a very fast and effective approach but domain tightening is hindered due

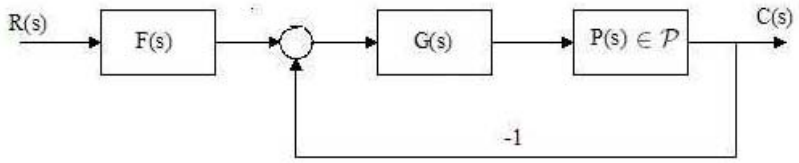

Figure 1. The two degree-of-freedom structure in QFT.

to occurrence of a variable more than once in a constraint (dependency problem). Domain pruning based on hull consistency (Benhamou, 1999) can be seen as a constraint inversion. The refinement in hull consistency results in HC4revise algorithm.

A numerical constraint inversion algorithm HC4revise (Benhamou, 1999) has been used for processing complex constraints. This is a direct method that is optimal if variables occur once in constraints. This implementation does not require any decomposition of the constraints. All the projection functions are evaluated by traversing a tree-structured representation of the constraints from bottom to top and conversely.

The box consistency has been introduced to avoid decomposing constraints, thus tackling the dependency problem. The implementation of box consistency involves the application of a one dimensional Newton method to solve a single equation for single variable. If variable occur more than once, box consistency is used. Domain pruning based on box consistency (Granvilliers and Benhamou, 2001) combines splitting of one variable domain and interval evaluation with respect to one constraint.

The interval Newton method (Hansen and Walster, 2005) is more specifically applied to square systems of equations. The interval Newton method can be decomposed in two steps: the first step generates a linear relaxation of the system through Taylor expansions; the second step implements the interval Gauss-Seidel algorithm to handle the linear interval system.

2.3. Quantitative feedback theory

Horowitz's approach of quantitative feedback theory (QFT) (Horowitz, 1993) to robust control system design has been gaining popularity in the control literature. It is based on extended classical control theory and uses well established tools such as Bode and Nichols plots. The QFT approach can handle single-input single-output (SISO) and multi-input multi-output (MIMO), linear and non-linear, time-varying and time-invariant, and lumped and distributed parameter systems.

A typical linear time-invariant (LTI) SISO two-degree-of-freedom feedback structure used in the QFT formulation is shown in Figure 1, where $P(s)$ is the plant belonging to an uncertain plant set $\mathcal{P}$, $G(s)$ is the controller, and $F(s)$ is the prefilter. Basically, QFT is a frequency-domain loop-shaping design technique. The objective is to synthesize $G(s)$ and $F(s)$, such that the given stability and performance specifications are met for all $P \in \mathcal{P}$.

In practice, the objective is to satisfy the given specifications over a finite design frequency set Ω. The main steps of the QFT design procedure are:

Template Generation: For a plant with parametric uncertainty, at each design frequency $\omega_i \in \Omega$, calculate the template or value set of $\mathcal{P}$ in the complex plane.
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 $B_i(\omega_i)$.
Design of Controller: Synthesize or 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.

Design of Prefilter: Design a prefilter $F(s)$ such that the robust tracking specifications are satisfied.
Details of the QFT design procedure are given in (Horowitz, 1993).

3. Synthesis of QFT prefilters using ICST

Consider a two-degree-of-freedom feedback system configuration shown in Figure 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 $\lambda \in \Re^l$ is a vector of plant parameters whose values vary over a parameter box $\boldsymbol{\lambda}$

$$\boldsymbol{\lambda} = \{\lambda \in \Re^l : \lambda_i \in [\underline{\lambda_i}, \overline{\lambda_i}],\ \underline{\lambda_i} \leq \overline{\lambda_i},\ i = 1, ..., l\}$$

This gives rise to a parametric uncertain plant family or set

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

Consider a plant $P(s,\lambda)$ from the plant set $\mathcal{P}$. The open loop transmission function is defined as

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

where $g(j\omega)$ and $\phi(j\omega)$ are magnitude and phase of the controller $G(s)$ and $p(j\omega)$ and $\theta(j\omega)$ are magnitude and phase of the plant $P(s)$ at frequency ω. The objective in QFT is to synthesize $G(s)$ and $F(s)$ such that the various stability and performance specifications are met for all $P(s) \in \mathcal{P}$ and for all frequencies ω. The controller $G(s)$ just stabilize the system. It does not guarantee to achieve tracking specification:

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

where the transfer functions $T_L(s)$ and $T_U(s)$ are called the lower and upper tracking models (on the tracking specifications). This is achieved by the prefilter $F(s)$.

Let

$$T(j\omega) = \frac{L(j\omega,\lambda)}{1 + L(j\omega,\lambda)} = \frac{G(j\omega)P(j\omega,\lambda)}{1 + G(j\omega)P(j\omega,\lambda)} \tag{2}$$

is closed loop transmission function and let

$$T_R(j\omega) = F(j\omega)T(j\omega) \tag{3}$$

From (1) and (3),

$$|T_L(j\omega)| \leq |T_R(j\omega)| \leq |T_U(j\omega)|, \forall\omega, \forall P \in \mathcal{P} \tag{4}$$

At each design frequency ω, and for each $P \in \mathcal{P}$, the above results in two inequalities

$$|T_L(j\omega)| - |F(j\omega)||T(j\omega)| \leq 0, \tag{5}$$
$$|F(j\omega)||T(j\omega)| - |T_U(j\omega)| \leq 0 \tag{6}$$

The variables of the inequalities (5) and (6) are prefilter parameters. These inequalities form a constraint set $\mathcal{C}$.

Given the tracking specifications and a feasible controller (obtained, say, using the approach described (Nataraj and Deshpande, 2008)), the design of the prefilter can be carried out as follows:

1. Specify the prefilter as

$$F(s,x) = \frac{\prod_{i=1}^{n_z}(s/z_i + 1)}{\prod_{k=1}^{n_p}(s/p_k + 1)\prod_{k=1}^{n'_p}\left(s^2/\vartheta_{n_k}^2 + 2\xi_k s/\vartheta_{n_k} + 1\right)} \tag{7}$$

   where the prefilter parameter vector x is

$$x = (z_1, .., z_{n_z}, p_1, .., p_{n_p}, \xi_1, .., \xi_{n'_p}, \vartheta_{n_1}, .., \vartheta_{n_{n'_p}})$$

2. Construct an initial search box of prefilter parameters, such that the designed prefilter is assumed to be stable and minimum-phase. That is, choose the lower bound on the parameters corresponding to each corner frequency as zero or slightly greater than zero, so that the designed prefilter is stable and minimum phase. Fix the upper bound on these parameters based on the cutoff frequency given by the tracking specs.

3. At each design frequency, calculate the magnitude of $T_R(s)$ over all $\mathcal{P}$ and magnitude of $T_L(s)$, $T_U(s)$.

4. Find the values of the prefilter parameters lying in the given initial search region, which satisfy constraints (5) and (6) at each design frequency.

Thus, ICSP for prefilter synthesis can be summarized as follows:

– The constraints will be for each design frequency, ω_i, $i = 1, ...n$ where n is number of design frequency.

$$|T_L(j\omega_i)| - |F(j\omega_i)||T(j\omega_i)| \leq 0,$$
$$|F(j\omega_i)||T(j\omega_i)| - |T_U(j\omega_i)| \leq 0$$

– The variables will be the prefilter parameter vector depending on the structure of the prefilter to be designed i.e.,
$$x = (z_1, .., z_{n_z}, p_1, .., p_{n_p}, \xi_1, .., \xi_{n'_p}, \vartheta_{n_1}, .., \vartheta_{n_{n'_p}})$$

– The domains for each variable can be given as suggested above i.e., all real poles and zeros can have initial domain as $[0, 200]$, all damping coefficients ξ can have $[0.5, 2]$ and all natural frequencies ϑ can have initial domain as $[0.01, 25]$.

The abovementioned ICSP is now solved using existing ICST-based algorithm. The algorithm is as follow:

Algorithm for prefilter synthesis

Inputs: The constraint set $\mathcal{C}$ resulting from (5) and (5), the initial parameter search box $\mathbf{x}^0$, and an accuracy parameter ϵ.

Output: The set $\mathcal{L}^{sol}$ of all certain prefilter solution, the set $\mathcal{L}^{Qsol}$ of uncertain prefilter solution, computed to the prescribed accuracy tolerance ϵ or "No prefilter design solution exists in the given initial search box $\mathbf{x}^0$"

BEGIN Algorithm

1. **Initialization part** Set $\mathcal{L}^{sol} \leftarrow \emptyset$, $\mathcal{L}^{Qsol} \leftarrow \emptyset$, $\mathcal{L} \leftarrow \mathbf{x}^0$.
2. **WHILE** ($\mathcal{L} \neq \emptyset$) **DO**
3. Fetch the first box $\mathbf{x}$ from the list $\mathcal{L}$
4. $\mathbf{x} \leftarrow$ **Prune**$(\mathcal{C}, \mathbf{x})$ (prune using BC5 filtering algorithm described in Section 2.)
5. **IF** ($\mathbf{x} \neq \emptyset$) **THEN**
6. **IF** all(inf($\mathcal{C}_i(\mathbf{x})$))$\geq 0$ **THEN**
7. $\mathcal{L}^{sol} \leftarrow \mathcal{L}^{sol} \cup \mathbf{x}$ % $\mathbf{x}$ is certain solution.
8. **ELSEIF** wid($\mathbf{x}$) $\leq \epsilon$ **THEN**
9. $\mathcal{L}^{Qsol} \leftarrow \mathcal{L}^{Qsol} \cup \mathbf{x}$ % $\mathbf{x}$ is quasi-solution.
10. **ELSE**
11. Bisect box $\mathbf{x}$ in maximum width direction into subboxes $\mathbf{x}^1$ and $\mathbf{x}^2$.
12. $\mathcal{L} \leftarrow \mathcal{L} \cup \mathbf{x}^1 \cup \mathbf{x}^2$
13. **END IF**
14. **END IF**
15. **END WHILE**

16. **IF** $((\mathcal{L}^{sol} \cup \mathcal{L}^{Qsol}) \neq \emptyset)$ **THEN**

17. Return $\mathcal{L}^{sol}$ and $\mathcal{L}^{Qsol}$

18. **ELSE**

19. Print("No prefilter design solution exists in the given initial search box $\mathbf{x}^0$")

20. **END IF**

END Algorithm

4. Design example

The approach proposed in Section 3 is tested on a benchmark example. The example has two different prefilter structures corresponding to two different controllers. All computations are carried out on a desktop PC with Intel Core-2-Duo, 2.4 GHz and 3 GB RAM using constraint modeling language RealPaver (Granvilliers, 2007).

Example This is a demo problem from the QFT toolbox (QFT tool-box, 1995). The plant $P(s)$ has the parametric uncertainty defined by

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

The controller $G(s)$ and prefilter $F(s)$ are to be designed such that the following specs are met for all $P \in \mathcal{P}$:

- Robust stability
$$\left| \frac{P(j\omega)G(j\omega)}{1 + P(j\omega)G(j\omega)} \right| < \infty \quad \text{for all} \quad \omega$$
- Robust stability margins: gain margin ≥ 2 dB, phase margin $\geq 50^0$ degrees.
$$\left| \frac{P(j\omega)G(j\omega)}{1 + P(j\omega)G(j\omega)} \right| \leq 1.26 \quad \text{for all} \quad \omega \geq 0.$$
- Robust tracking
$$|T_L(j\omega)| \leq \left| F(j\omega) \frac{P(j\omega)G(j\omega)}{1 + P(j\omega)G(j\omega)} \right| \leq |T_U(j\omega)|, \text{for all} \quad \omega \leq 10.$$

where

$$|T_L(j\omega)| = \left| \frac{120}{(j\omega)^3 + 17(j\omega)^2 + 82(j\omega) + 120} \right| \tag{8}$$

$$|T_U(j\omega)| = \left| \frac{0.6584(j\omega + 30)}{(j\omega)^2 + 4(j\omega) + 19.752} \right| \tag{9}$$

$\Rightarrow$ The design frequency set is chosen as

$$\Omega = [0.1, 0.5, 1.0, 2.0, 15, 100]\,.$$

We shall first use the controller reported in (QFT tool-box, 1995) as

$$G_a(s) = \frac{9.01\left(\frac{s}{113.8}+1\right)\left(\frac{s}{1.1}+1\right)}{\left(\frac{s}{42.81}+1\right)\left(\frac{s^2}{10^6}+\frac{1486s}{10^6}+1\right)} \tag{10}$$

For this controller, we attempt to synthesize a prefilter of the form (with a complex pole pair)

$$F_a(s) = \frac{1}{\left(\frac{s^2}{\omega_n^2}+\frac{2\xi s}{\omega_n}+1\right)} \tag{11}$$

The unknown variables are the parameters of the prefilter transfer function. The initial search box for the parameters $\mathbf{x} = (\xi, \omega_\mathbf{n})$ is constructed as

$$\mathbf{x} = \Big([0.5, 2], [10^{-4}, 25]\Big)$$

The constraints (5) and (6), and initial search box of the prefilter parameters (variables) now set up the ICSP. The algorithm described in Section 3 is then applied. All possible prefilters of the specified structure are obtained in about 0.2 seconds. Optimality of QFT prefilter is not discussed anywhere in the QFT literature. Any prefilter from the solution will achieve the desired tracking specification. Therefore, from among these solution, the following prefilter is chosen

$$F_a(s) = \frac{1}{\left(\frac{s^2}{(3.752)^2}+\frac{2*0.663s}{3.752}+1\right)} \tag{12}$$

The closed loop frequency responses of the entire plant set with the designed prefilter are shown in Figure 2. It shows that all the frequency responses clearly lie between those of the given closed loop tracking models, T_U and T_L.

To demonstrate further, we choose this time a different controller. We now use the controller designed using (Nataraj and Deshpande, 2008)

$$G_b(s) = \frac{10.95\left(\frac{s}{2.1}+1\right)}{\left(\frac{s}{950}+1\right)\left(\frac{s}{982}+1\right)} \tag{13}$$

We also now specify a different prefilter structure (with two real poles)

$$F_b(s) = \frac{1}{\left(\frac{s}{p_1}+1\right)\left(\frac{s}{p_2}+1\right)} \tag{14}$$

and construct the initial search box $\mathbf{x} = (\mathbf{p_1}, \mathbf{p_2})$ as

$$\mathbf{x} = ([0, 100], [0, 100])$$

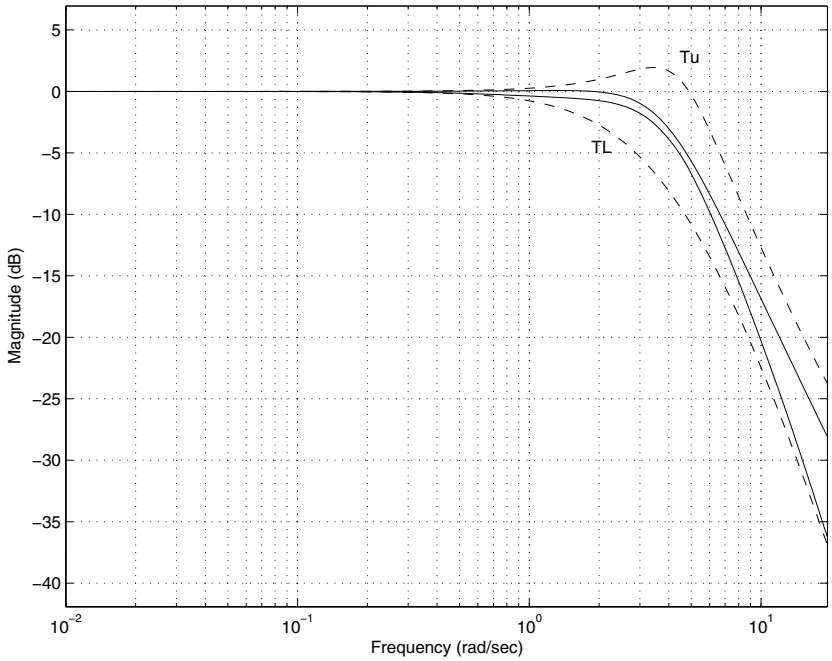

Figure 2. The closed loop frequency responses of the entire plant set with the synthesized filter $F_a(s)$, denoted as solid lines.

The algorithm in Section 3 is next applied, which gives all possible prefilter designs in about one second on a PC. From among these solutions, the following prefilter is chosen

$$F_b(s) = \frac{1}{\left(\frac{s}{3.197}+1\right)\left(\frac{s}{8.61}+1\right)} \tag{15}$$

The closed loop frequency responses of the entire plant set with the synthesized $F_b(s)$ are shown in Figure 3. It shows that all the responses clearly lie between those of the given closed loop tracking models, T_U and T_L.

5. Non-existence verification of prefilter solution

The proposed approach computationally verifies the non-existence of QFT preliter solution, for a specified prefilter structure over a specified search domain of prefilter parameter values. To demonstrate this capability, we consider Example in Section 4.

We use the controller reported in (QFT tool-box, 1995) as

$$G_a(s) = \frac{9.01\left(\frac{s}{113.8}+1\right)\left(\frac{s}{1.1}+1\right)}{\left(\frac{s}{42.81}+1\right)\left(\frac{s^2}{10^6}+\frac{1486s}{10^6}+1\right)} \tag{16}$$

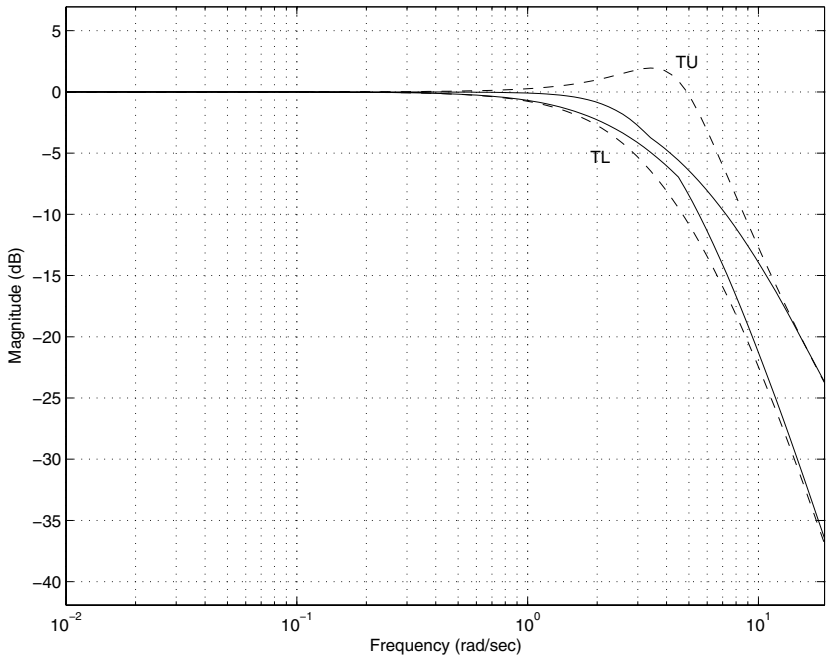

Figure 3. The closed loop frequency responses of the entire plant set with the synthesized filter $F_b(s)$, denoted as solid lines.

For the controller $G_a(s)$, this time we attempt to synthesize a prefilter of the form (with two real poles)

$$F_c(s) = \frac{1}{\left(\frac{s}{p_1}+1\right)\left(\frac{s}{p_2}+1\right)} \tag{17}$$

and construct the initial search box $\mathbf{x} = (\mathbf{p_1}, \mathbf{p_2})$ as

$$\mathbf{x} = ([0, 100], [0, 100])$$

The algorithm described in Section 3 is then applied. With the controller $G_a(s)$, prefilter structure $F_c(s)$ and the above initial search domain, the algorithm described in Section 3 terminates with the message "No solution in given initial search box ".

6. Conclusions

We propose an approach to automate the synthesis of fixed structure QFT prefilters. QFT prefilter synthesis problem is posed as an ICSP and solved using existing, computationally efficient ICST-based algorithm. The ICST-based algorithm uses box and hull consistency techniques to solve the nonlinear, non-convex constraints. For a given structure of the prefilter and initial search domain,

```
RealPaver v. 0.4 (c) LINA 2004

INITIAL BOX
  p1 in [0 , 100]
  p2 in [0 , 100]

END OF SOLVING
  Property:     no solution in the initial box
  Elapsed time: 0 ms
```

Figure 4. Output of the algorithm verifying the nonexistence of the prefilter solution.

if no feasible prefilter design exists, then the algorithm is guaranteed to computationally verify this fact. If a feasible prefilter does exist, then algorithm is guaranteed to find all feasible prefilter solutions in the initial search domain. The proposed approach is used successfully to design prefilters of two different structures for a benchmark problem.

The present work can further be extended to make the prefilter design completely reliable at all frequencies in the region of interest, by using interval frequency instead of using discrete design frequency set.

References

Benhamou, F., F. Goualard, L. Granvilliers, and J. F. Puget. Revising hull and box consistency. In *Proc. of 16th International Conference on Logic Programming*, 230–244, MIT Press, Las Cruces, NM, USA, 1999.

Benhamou, F. and W. Older. Applying interval arithemetic to real, integer and boolean constraints. *Journal of Logic Programming* , 32(1):1–24, 1997.

Borghesani, C., Y. Chait, and O. Yaniv. The Quantitative Feedback Theory toolbox for MATLAB, The MathWorks, Inc., MA, USA, 1995.

Granvilliers, L. and F. Benhamou. COCONUT deliverable D1 algorithms for nonlinear constrained and optimization problems: The state of the art. Technical report, University of Nantes, 2001.

Granvilliers, L. RealPaver: An interval solver using constraint satisfaction techniques. *ACM Transaction on Mathematical Software*, 32(1):138-156, 2006.

Granvilliers, L. *RealPaver (Version 0.4): Solving nonlinear constraints by interval computations. Users mannual.* LINA- University of Nantes, Faculty of Sciences, Nantes Cedex 3, France, 2007.

Hansen, E. and G. Walster. Global Optimization using Interval Analysis, Second Edition. Marcel Dekker, New York, 2005.

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

Jaulin, L., M. Kieffer, O. Didrit, and E. Walter. Applied Interval Analysis. Springer, London, 2001.

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

Nataraj, P. S. V. and M. M. Deshpande. Automated synthesis of fixed structure QFT controller using interval constraint satisfaction techniques. In *Proc. of 17th IFAC World Congress*, 4976-4981, Seoul, South Korea, 2008.

Tharewal, S. S. *Automated synthesis of QFT controllers and Pre-filters using Interval Global Optimazation Techniques.* PhD thesis, Systems and Control Engineering, IIT Bombay, India, 2005.