INTERVAL QFT : A MATHEMATICAL AND COMPUTATIONAL ENHANCEMENT OF QFT

P. S. V. NATARAJ

Dedicated to Prof. I. Horowitz

ABSTRACT. The paper presents an overview of a mathematical and computational enhancement of Horowitz's QFT design procedure. The enhancement uses methods of interval analysis and is called as interval QFT, or IQFT. IQFT addresses and solves some of the fundamental issues in QFT, concerning selection of design frequencies, selection of controller phases in bound generation, approximation of plant templates with finite plant sets, and generation of plant templates and controller bounds with reliability and to a prescribed accuracy. An example is presented to illustrate the key features of IQFT.

1. INTRODUCTION

Over the last decade or so, Horowitz's quantitative feedback theory (QFT) approach [9] to robust control system design has been gaining popularity among control researchers. The QFT approach comprises of a collection of techniques for dealing with several classes of uncertain plants: linear and nonlinear, time-invariant and time-varying, lumped and distributed, single input-output and multi input-output, single-loop and multiple-loop, etc. Recently, several researchers have shown that the QFT technique is more general and powerful than other robust control approaches, see for example, [4], [11], [24], [35], [36].

The QFT approach (like all other robust control approaches) is based on mappings involving point numbers. It is the thesis of this paper that several key enhancements to the *point* QFT approach can be obtained by adopting a fundamentally different approach based on mappings involving *intervals*. The interval based QFT approach, or simply, the interval QFT (IQFT) approach, provides guarantees on the reliability and accuracy of the generated plant templates and controller bounds, and automatically produces error estimates. Further, the key issues of finite frequency selection in design frequency set formulation, finite plant approximation in template generation, and finite phase selection in bound generation are resolved in the interval QFT approach.

The idea behind interval approaches is to design algorithms, which in a single computation, do the approximation and a rigorous error analysis. With interval approaches, one can directly deal with interval sets containing infinitely many points, and perform set operations such as subdivisions, unions, intersections, finding convex hulls, testing for set inclusion, testing for disjointedness of sets, etc. For a general introduction to the interval analysis (IA) approach, the reader may consult [15].

The IQFT approach is being pursued in the interval mathematics - QFT group at IIT Bombay. IQFT can be applied to any problem that can be solved with point QFT methods, but is surprisingly richer in its consequences:

Interval Analysis based Template Generation (IATG) algorithms provide several guarantees: the templates are (a) guaranteed to be of prescribed accuracy, (b) guaranteed to be reliable - that is, an assurance of correctness of the computed numerical results is provided, in face of all kinds of computational errors, such as round-off, truncation, and approximation, (c) guaranteed to enclose all actual template points, thereby avoiding any loss of robustness due to template approximation errors.

Systems and Control Engineering Group, Department of Electrical Engg., IIT Bombay 400 076, India. Email: nataraj@ee.iitb.ernet.in.

Interval Analysis based Bound Generation (IABG) algorithms also provide several guarantees: the bounds are (a) guaranteed to be robust against template inaccuracies, in the sense that regardless of the accuracy of the interval plant template used these can never lead to violation of the specifications, (b) guaranteed to be robust against phase discretization, so that no blip in the bounds can lead to violation of the specifications, (c) guaranteed to be reliable, despite all kinds of computational errors, such as round off, truncation, and approximation; the algorithms also offer key improvements: (d) the bounds are obtainable usually in much less time, and (e) *a posteriori* error estimates are readily available from the bounds.

Before proceeding to describe IQFT methods, the various ways in which the design problems can be posed in QFT are first categorized.

1.1. **Classes of QFT Design Problems. Class A problem** : "Given the design frequency. Then, how does the QFT designer generate the plant template and compute the controller bounds so that the latter are guaranteed to be reliable over designer-selected controller phase intervals ? "

On the other hand, at the given frequency, the QFT designer may wish to specify *a priori* the desired accuracy of the controller bounds, and leave it to the algorithm to automatically arrive at the appropriate controller phase intervals. This situation leads to the following problem class:

Class B problem: "Given the design frequency. Then, how does the QFT designer generate the plant template, and the controller phase intervals with their corresponding controller bounds, so that the bounds are guaranteed to be reliable and have a prescribed accuracy ?"

At an even more advanced level, the designer may wish to specify only the desired accuracy of the controller bounds, the range of design frequencies along with the desired spacing between the bounds, and leave it to the algorithm to automatically arrive at whatever frequency and controller phase intervals are appropriate. This situation leads to the following problem:

Class C problem : "Given a range of design frequencies. Then, how does the QFT designer generate the frequency intervals and the plant templates, and the controller phase intervals with their corresponding controller bounds, so that the bounds are guaranteed to be reliable, have a prescribed accuracy, and be spaced apart as desired ? "

To solve problems of class A, the QFT designer can execute an IATG algorithm for generating the interval plant template and then apply an IABG algorithm to compute the controller bounds over designer-selected phase intervals.

To solve problems of classes B and C, the QFT designer can execute the appropriate *unified* procedure (cf. section 4). The unified procedures combine template and bound generation steps for generating bounds of prescribed accuracy, and appropriate controller phase and design frequency intervals are selected automatically in the procedures.

The issues of finite frequency and finite phase selection in point QFT are resolved in the unified procedures, along with the issue of finite plant approximation that troubles all point-based methods of template and bound generation. It may be pertinent to note here that "*point* methods and computations with ordinary *floating-point* numbers have no direct way of dealing with sets containing infinitely many or uncountably many points" [16, section 5], and further, "using *point* methods, there may be no indication, let alone guarantee, of the accuracy or completeness of the results" [8].

The rest of this paper is organized as follows. In sections 2 and 3, an overview of the various IATG and IABG algorithms is given. In section 4, a procedure to solve design problems of class C is outlined. In section 5, concluding remarks on IQFT are given.

All computations in this work are done on a PC / Pentium III 550 MHz machine using the interval arithmetic toolbox INTLAB that runs under MATLAB [28].

2. Interval Template Generation

Consider a plant represented by the transfer function $\mathfrak{g}(s,\lambda)$, where $\lambda = \{\lambda_1, ... \lambda_n\}$ is a real vector of the fundamental system parameters and s is the Laplace variable. Suppose the parameters λ_i vary independently over given real intervals Λ_i^0, so that we have a box $\Lambda^0 = \{\Lambda_1^0, ...\Lambda_n^0\}$ of plant parameters.

Denote the phase angle and magnitude functions of $\mathfrak{g}(s,\lambda)$ as $f_{ang}(\omega,\lambda) = \arg \mathfrak{g}(j\omega,\lambda)$; $f_{mag}(\omega,\lambda) = |\mathfrak{g}(j\omega,\lambda)|$, where ω is a given frequency. Define the angle-magnitude function f as $f(\omega,\lambda) = (f_{ang}(\omega,\lambda), f_{mag}(\omega,\lambda))$. Then, the set $\mathcal{G}(\omega) := \{f(\omega,\lambda), \lambda \in \Lambda^0\}$ defines a region in the angle-magnitude plane (i.e., in the Nichols chart), called the template of $\mathfrak{g}(s,\lambda)$ at the given ω.

Definition 2.1. *The expression which arises if each occurrence of λ in $f(\omega,\lambda)$ is replaced by Λ, if each occurrence of a predeclared function (like* $\sin, \cos, \exp$*,etc.) is replaced by the corresponding predeclared interval function, and if the arithmetic operations in $f(\omega,\lambda)$ are replaced by the corresponding interval arithmetic operations, is called as the natural interval extension of $f(\omega,\lambda)$ to Λ. The natural interval extension of f to Λ is denoted as $F(\omega,\Lambda)$.*

Example 2.1. *(Natural interval extension) If $f(\lambda) = 1-5\lambda+1/3\lambda^2$, then $F(\Lambda) = 1-5\Lambda+1/3\Lambda^2$ is the natural interval extension of $f(\lambda)$. As another example, if $f(\lambda) = \lambda_1 * \sin\lambda_2 - \lambda_3$, then $F(\Lambda) = \Lambda_1 * ISIN(\Lambda_2) - \Lambda_3$ is the natural interval extension of $f(\lambda)$, where $ISIN$ is the pre-declared interval* sin *function in some programming language.*

One can compute $F(\omega,\Lambda)$ and obtain, with a single evaluation of F, a template comprising of a single angle-magnitude rectangle. By inclusion property of natural interval extensions [15, Theorem 3.1], $F(\omega,\Lambda)$ encloses the actual template $\mathcal{G}(\omega)$. However, this angle-magnitude rectangle $F(\omega,\Lambda)$ usually has a width that considerably exceeds the specified width ε.

Therefore, one may repeatedly subdivide (or partition) the parameter box, find the evaluations of F over the subboxes using interval arithmetic, and take the union of the results to get templates comprising of smaller and smaller angle-magnitude rectangles which give increasingly accurate information about the actual phase-magnitude values. It is a fundamental result in interval analysis that as the partition of the parameter box is refined, these templates will converge to the actual template. The partition or subdivision process can be stopped when the widths of all the angle-magnitude rectangles is less than ε.

Depending on the way the subdivision is done, IATG algorithms can be cast into three categories:

2.1. **IATG using Uniform Subdivision.** The uniform subdivision process has been originally suggested by Moore [15, sec. 4.1] in a general setting of finding the range of a function, and for QFT template generation in [29] as follows.

First, find a uniform subdivision factor N for all parameter intervals, making use of an inequality relation in [15, equation 4.5]. Then, subdivide each parameter interval into N equal subintervals with this subdivision factor, and create a so-called *uniform* subdivision partition. Lastly, evaluate $F(\omega,\Lambda)$ over the subboxes of the uniform partition, in a *parallel* manner, using vectorized operations. It can be shown from the work of Moore [15, sec. 4.1] that every resulting angle-magnitude rectangle is of width at most ε, and that the collection of all these rectangles constitutes the required template which necessarily contains the actual one.

The below example illustrates the uniform subdivision process.

Example 2.2. *(Uniform Subdivision): Let $f(\lambda_1,\lambda_2) = \lambda_1(1-\lambda_1)+\lambda_2$, with $\lambda_1 \in \Lambda_1 = [0, 0.25], \lambda_2 \in \Lambda_2 = [2,4]$. Subdividing Λ_1 and Λ_2 into $N = 2$ subintervals of equal width gives four resulting subboxes. Evaluating the natural interval extension F over the four subboxes gives the maximum width of F over all these subboxes as 1.125. If $N = 3$ is chosen, the maximum width of F over all the nine resulting subboxes is found to be 0.75, whereas for $N = 10, 100$ the same is respectively $0.225, 0.0225$. However, as discussed below, a major difficulty in this approach is that one usually does not know in advance the* exact *value of N required to achieve a specified maximum width of F.*

The advantage of the uniform subdivision based algorithm is that it is a single step algorithm - the template can be generated with a single interval evaluation of F for each subbox of the partition, and moreover, this can be done in a *parallel* manner for all the subboxes using vectorized interval arithmetic operations. The disadvantage of this algorithm is that N is usually heavily overestimated, due to overestimation in the subdivision factor calculation [25] and in Moore's inequality referred above. Consequently, a much larger number of subboxes than required will be generated with such a N. Therefore, it is clear that the amount of computational effort may be rather large in this algorithm, which may lead in turn to correspondingly large computation times.

2.2. **IATG using Adaptive Subdivision.** The adaptive subdivision process is also well-known in general setting of range finding in IA literature, see, for example [13]. It has also been recently used (together with the interval Gauss-Seidel method) for template generation in [20] as follows.

First, evaluate F over the current parameter box, and check the width of the resulting angle-magnitude rectangle against the specified maximum width ε. If the specified width is exceeded, then bisect the box into two subboxes by cutting in the coordinate direction in which the box is longest. Discard the original box. Pick any one of the two subboxes, and put the other subbox in a processing list. Successively subdivide the picked (current) subbox till the width of the resulting angle-magnitude rectangle is at most ε. Then, write the angle-magnitude information to a solution list, and discard the current subbox. Repeat the above for all subboxes created in this process. Finally, output the generated template as the collection of all angle-magnitude rectangles present in the solution list.

The below example illustrates the adaptive subdivision process.

Example 2.3. *(Adaptive Subdivision): Let* $f(\lambda_1, \lambda_2) = \lambda_1(1-\lambda_1) + \lambda_2$, *with* $\lambda_1 \in \Lambda_1 = [0, 0.25], \lambda_2 \in \Lambda_2 = [2, 4]$, *as in previous example. Further suppose that the specified maximum width of* F *is* $\varepsilon = 0.1$. *Let* $w(F)$ *denote the width of* F. *Evaluating the natural interval extension* $F(\Lambda_1, \Lambda_2) = \Lambda_1(1-\Lambda_1) + \Lambda_2$ *over the given* Λ *gives* $F = [2, 4.25]$, $w(F) = 2.25$. *As* $w(F) > \varepsilon$, Λ *is subdivided along its longest component direction, which is here component direction* 2. *This gives two subboxes denoted as* $\Lambda^a = ([0, 0.25], [2, 3])$ *and* $\Lambda^b = ([0, 0.25], [3, 4])$. *The first subbox* Λ^a *is next processed, while the other subbox* Λ^b *is sent to a list for later processing. Next,* F *is evaluated on* Λ^a *to get* $F(\Lambda^a) = [2, 3.25], w(F) = 1.25$. *As* $w(F) > \varepsilon$, Λ^a *is subdivided along its longest component direction, which is again component direction* 2. *This gives two subboxes denoted as* $\Lambda^{aa} = ([0, 0.25], [2, 2.5])$ *and* $\Lambda^{ab} = ([0, 0.25], [2.5, 3])$. *The first subbox* Λ^{aa} *is processed next, while* Λ^{ab} *is sent to the processing list. The subdivision is repeated on* Λ^{aa} *and its siblings till* $w(F) < \varepsilon$. *Then, the next box from the processing list is picked and successively subdivided till its* $w(F) < \varepsilon$, *and so on.*

The advantage of this algorithm is that it generates a considerably much smaller number of angle-magnitude rectangles in the template, because a subbox is successively subdivided into smaller subboxes only as long as the angle-magnitude rectangle width is unacceptable. This can therefore be viewed as an adaptive subdivision process. However, each subbox is processed *sequentially*. Therefore, the algorithm generally takes considerably more time than the above one involving parallel computations.

2.3. **IATG using Parallel-Adaptive Subdivision.** The attractive feature of the uniform subdivision based approach is parallel evaluation of F over all subboxes of a partition; while that of the adaptive subdivision based approach is adaptive subdivision. An algorithm that combines these two advantageous features i.e., *parallel* functional evaluation and *adaptive* subdivision is the so-called parallel-adaptive (PA) algorithm in [22], [30]. The PA algorithm for template generation runs as follows.

In the first or initial iteration, create a partition by subdividing all parameter intervals into subintervals of nearly equal width. Then, perform parallel evaluation of F over the subboxes of the partition, and save all angle-magnitude rectangles whose widths are less than ε in a solution list, discarding the corresponding subboxes from any further consideration. In the second and succeeding iterations (only those subboxes for which the

corresponding angle-magnitude rectangles have unacceptable widths are left), cut simultaneously all subboxes in the *longest* direction, discard the original subboxes used for cutting, and perform parallel evaluation of F over the subboxes resulting from cutting. Save all angle-magnitude rectangles whose widths are less than ε in the solution list, discarding the corresponding parameter subboxes from any further consideration. Terminate the iterations when there are no more subboxes to deal with.

The advantage of the PA algorithm over the uniform subdivision algorithm is that the former usually generates a much smaller number of angle-magnitude rectangles. This is due to the usage of adaptive subdivision process in the former algorithm, which keeps the amount of computations to reasonable limits, and in turn results in significant reductions in computation times (in contrast, a much large number of subboxes is typically created in the uniform subdivision partition, due to heavy overestimation in the subdivision factor. This thereby leads to large execution times). The advantage of the PA algorithm over the adaptive subdivision algorithm is that the former usually executes much faster since all subboxes under consideration are cut in parallel (i.e., simultaneously), F is evaluated in parallel over all the resulting subboxes. Recall that in the adaptive subdivision algorithm, the subboxes are adaptively cut and processed sequentially, which slows down the processing due to its sequential nature.

In several examples, the PA algorithm is found to be faster than the above two categories by $1-2$ orders of magnitude [22].

The efficiency of the PA algorithm, in terms of the final number of template rectangles generated and the algorithm execution time, is greatly influenced by the selection of the coordinate direction along which boxes are subdivided in each iteration. In the version of the PA algorithm just described, the rule of subdivision is to cut each box along its longest direction. However, in many examples it is observed that this subdivision rule merely increases the number of template rectangles without yielding any significant reduction in their widths. Therefore, to further improve upon the efficiency of the PA algorithm, more efficient subdivision rules need to be explored. These are next described.

2.4. **IATG using Parallel-Adaptive with other subdivision rules.** In the so-called parallel-adaptive with back-tracking algorithm in [31], the subdivision rule is to adaptively cut the boxes along a *favorable* direction, i.e., in a direction along which the width of F over the subboxes of partition is reduced at least by some acceptable percentage.

In the so-called parallel-adaptive with first-box algorithm in [18], the subdivision rule is as follows. In a given iteration, pick the first box from the list of parameter boxes to be processed and bisect it along the first coordinate direction to get two subboxes. Then, evaluate F over these subboxes, and find the maximum width of F. Repeat the process in all other coordinate directions, and determine the coordinate direction in which the corresponding maximum width of F is the least. This coordinate direction is then used as the direction for subdividing all boxes in the current iteration.

The parallel-adaptive algorithm can also be based on other subdivision rules found in the literature on interval branch and bound algorithms for global optimization (see, for instance, [26]).

2.5. **Illustrative examples.** The PA algorithm with the various subdivision rules is applied to generate the interval plant templates on a suite of eleven transfer function examples taken from the QFT literature. The templates of all transfer functions are generated to an accuracy of 1 deg and 1 dB.

The findings of this study are summarized in Table 1. In this Table, Rule A is the rule of subdivision along the longest coordinate direction used in the basic version of the PA algorithm. Rule B is the maximum smear rule in [26], commonly used for global optimization. In column 3, the number of uncertain parameters n is given. A "-" in the Table indicates that the computer runs out of memory.

The performances with the various subdivision rules are compared, in terms of number of template rectangles (also called as solution boxes in the Table) and the execution time in seconds taken to generate these boxes.

	Example	n	Solution	IATG Algorithm PA, with subdivision rule as Rule A [22]	Rule B [26]	Back-tracking [31]	First-box [18]	Boundary Extraction [21]
1	Simple poles [19]	3	boxes	7,712	627	459	627	128
			time(s)	3.27	1.88	0.74	0.46	0.08
2	Underdamped [27]	2	boxes	30,223	281,814	25,526	32,201	1,053
			time(s)	7.24	109	3.5	4.86	16
3	DC Motor [1]	2	boxes	–	3,048	2,451	2,860	285
			time(s)		4.11	1.5	1.11	0.4
4	NMP [32]	3	boxes	512	808	504	512	138
			time(s)	1.33	1.59	0.3	0.44	0.08
5	Non-rational [10, pp.129]	3	boxes	6,759	35,404	6,759	6,891	436
			time(s)	7.69	35.23	3	3.55	2.9
6	DC Motor -2 [7]	3	boxes	–	–	84,466	92,608	1,604
			time(s)	–	–	16	21.99	84
7	Vehicle clutch [6]	3	boxes	30,424	3,109	2,644	2,844	319
			time(s)	6.09	2.15	0.79	1.81	0.6
8	Multiple lags [20]	4	boxes	235,139	411,118	170,448	189,903	2369
			time(s)	1.23×10^3	1.27×10^3	633	1×10^3	70
9	Mechanical [10, pp.222]	5	boxes	17,320	–	1,751	1,981	579
			time(s)	10.89	–	1	0.99	0.8
10	Aircraft [33]	5	boxes	40,002	966	642	1,508	131
			time(s)	10.73	1.92	1	1.06	0.1
11	Inv. pendulum [2]	7	boxes	–	208	194	208	101
			time(s)	–	1.312	0.7	0.64	0.05

TABLE 1. Performances of Parallel-adaptive IATG and boundary extraction algorithms.

Summarizing the results of the numerical tests, the first-box subdivision rule is found to be the overall best choice, in terms of consistency, number of template rectangles, and time taken.

2.6. **Properties of IATG Algorithms. Mathematical reliability** of the IATG algorithms readily follows from the existence of well-known theorems in interval analysis and usage of exact interval arithmetic in all computations. For instance, it follows immediately from the inclusion property of natural interval extensions [15, Theorem 3.1] that, for any $\varepsilon > 0$, the generated templates indeed enclose the exact templates. Moreover, it is a fundamental result in interval analysis [15, Theorem 4.1] that as the partitions are refined, the enclosures will converge to the actual ranges of the values over the given domain sets. From this property, it follows that in the limiting case of the specified width equal to zero, IATG algorithms yield templates that converge to the exact templates. Further, for any finite $\varepsilon > 0$, IATG algorithms generate the required templates in a *finite* number of steps. This can be shown by proceeding on similar lines to [12, Theorem 2.10]. Lastly, rigorous justification of the subdivision processes in a general setting are given in [12], [15].

Computational reliability of the IATG algorithms means that the algorithms are stable when implemented on floating-point systems. The IATG algorithms can be made computationally reliable by implementing them

in any interval arithmetic compiler. An interval arithmetic compiler uses machine interval arithmetic (MIA) in all computations, see, for instance, [14].

Efficiency of implementation. IATG algorithms are efficient to implement as they are simple from a computer programmer's point of view, and as their performances are not sensitive to details of implementation or to any "tuning".

Range of application of IATG algorithms is very vast, as the algorithms are applicable to any programmable transfer function that is continuous in its parameters.

2.7. **Boundary extraction.** Similar to the case of the point template, it can be shown [19] that for a simply connected interval plant template only the boundary template rectangles need to be considered in designs. Now, the set of boundary rectangles of an interval plant template is obtainable as the union of upper, lower, left and right boundary rectangles in the Nichols chart (by upper boundary rectangles is meant the template rectangles with the maximum magnitude at each phase, and so on).

In [21], an algorithm is presented to extract all the boundary rectangles from a given interval plant template, without introducing any kind of boundary approximations. The procedure given therein to extract upper boundary rectangles from an interval plant template is as follows: Start always by marking the rectangle forming the left end of the template as a boundary rectangle, and set it as *current* rectangle. Then, journey along the top edge of the current rectangle towards the right end, checking if any upward jump to a higher magnitude occurs on this journey. **IF yes**, mark the rectangle associated with the jump as a boundary rectangle, set it as *current* rectangle, and take the jump along the left side of the current rectangle to reach its top edge. Next, continue the journey along the top edge of the current rectangle towards the right end, and so on, as before. **IF no**, complete the journey to reach the right end of the top edge, and when the right end of the entire template itself is thereby reached, print out all the boundary rectangles and exit the procedure; otherwise, journey downwards along the right side of current rectangle to the next lower magnitude rectangle, mark the latter rectangle as a boundary rectangle, set it as *current* rectangle, and journey along the top edge of the current rectangle towards the right end, and so on, as before.

The procedure to extract lower boundary rectangles is the same as to that for upper boundary rectangles, except for a few obvious modifications, such as journeying along the bottom (instead of top) edge of each current rectangle, and checking for any downward (instead of upward) jumps to a lower (instead of upper) magnitude on the journey. The procedure to extract left and right boundary rectangles is derived by simply interchanging the roles of magnitude and phase in the above procedures. Finally, the set of all boundary rectangles of the interval template is obtained by taking the union of the upper, lower, left and right boundary rectangles.

The results of boundary extraction obtained on the suite of eleven transfer function examples referred to earlier, are given in Table 1 in the last column. The Table shows that the proposed algorithm extracts the boundary rectangles quite efficiently in terms of computational time. Typically, the algorithm takes about $10-40$ milliseconds to extract the boundary rectangles from a set of 100 interval template rectangles - indicating the efficiency of the boundary extraction algorithm.

3. Interval Bound Generation

In this section, IABG algorithms for the various specifications are described. The IABG algorithms are basically extensions of the quadratic constraints approach given in [3], [5] to the interval case.

The robust sensitivity reduction case is first taken up.

3.1. **Robust sensitivity reduction.** Consider the plant family $\{\mathfrak{g}(s,\lambda), \lambda \in \Lambda^0\}$ embedded into a single-loop system with controller $\kappa(s)$. Then, the robust sensitivity reduction specification (spec) at a given frequency ω is

$$|1+\kappa(j\omega)\,\mathfrak{g}(j\omega,\lambda)|^{-1} \leq w_s(\omega), \quad \forall \lambda \in \Lambda^0$$

In the quadratic constraints approach [5], the bounds are computed at ω as follows. Substituting the polar forms $\kappa = ke^{j\theta}$ and $\mathfrak{g} = ge^{j\varphi}$ in the above spec and simplifying gives the quadratic inequality (dropping the argument $j\omega$)

$$g^2k^2 + 2gk\cos(x) + z_s \geq 0$$

where $z_s := 1 - 1/w_s^2$, $x := \theta + \varphi$. The quantity x can be interpreted as the phase of the loop transmission function. Setting the RHS of above inequality to zero and solving for the roots gives

$$k^u_{root}, k^l_{root} := \left(-\cos(x) \pm \sqrt{\cos^2(x) - z_s}\right)/g \tag{1}$$

Thus, for the plant $\mathfrak{g}$, the allowable magnitude range of the controller for satisfying the sensitivity reduction spec at θ is $[0, k^l_{root}] \bigcup [k^u_{root}, \infty)$. Then, for the entire template $\mathcal{G}$ the allowable magnitude range of the controller at θ is $[0, k^l_{bound}] \bigcup [k^u_{bound}, \infty)$, where

$$k^u_{bound} := \max\left\{k^u_{root}\right\}; \quad k^l_{bound} := \min\left\{k^l_{root}\right\}$$

The quantities k^u_{bound}, k^l_{bound} are called as the upper and lower bounds on the controller magnitude at θ. The procedure can be repeated for all $\theta \in [-2\pi, 0]$ to obtain the bounds over the entire controller phase range.

The quadratic constraints approach generates bounds at point values of controller phase and frequencies, and is based on point plant templates. The quadratic constraints approach can be extended for generating bounds at interval values of controller phase, and even for interval frequencies, using interval plant templates, as described below.

It is assumed that in the sequel that all interval plant templates are closed and simply connected.

3.1.1. *Computing the root intervals.* Consider a template rectangle picked from the interval plant template at a given frequency ω, and denote its magnitude interval as G and phase interval as Φ. Let the controller phase be also an interval Θ. Then, substituting all these interval quantities in the RHS of (1) gives the root *intervals* K^u_{root}, K^l_{root} defined as

$$K^u_{root} := \frac{-\cos X + \sqrt{\cos^2 X - z_s}}{G};\ K^l_{root} := \frac{-\cos X - \sqrt{\cos^2 X - z_s}}{G} \tag{2}$$

where

$$X := \Theta + \Phi \tag{3}$$

By analogy with the point case, X can be interpreted as the phase *interval* of the loop transmission function, and K^u_{root}, K^l_{root} as the upper and lower root intervals at phase interval X for the considered template rectangle.

Computation of these root intervals directly from (2) using interval arithmetic produces overestimation, due to multiple occurrences of the intervals Θ and Φ in the functional expressions and the interval dependency effect. However, if the computations are done as per [19, Theorem 2.2], then the exact values of the root intervals are obtained.

3.1.2. *Computing the working phase interval.* It is shown in [19, Theorem 2.2] that any bound computations for this spec need to be performed only for those X belonging to the so-called working phase interval Y defined as

$$Y := -\left(\arccos\left(-\sqrt{z_s}\right), \arccos\left(\sqrt{z_s}\right) + \pi\right) \subseteq [-2\pi, 0] \tag{4}$$

Elsewhere in the range $[-2\pi, 0]$, any controller magnitude is satisfactory to achieve the spec, so there is no need to compute the bounds at such phases.

3.1.3. *Composing the bound intervals.* Consider a template rectangle in the interval template. Using (2), the root intervals are computed at various X in the working phase range Y. From these X and using the phase interval Φ of template rectangle, the corresponding controller phase intervals Θ are back-calculated from (3) and the root intervals at X reassigned to the respective Θ. The obtained results are plotted in the Nichols chart as $2-$dim interval vectors or rectangles given by (Θ, K^u_{root}) and $\left(\Theta, K^l_{root}\right)$, with the controller phase intervals Θ on the x-axis and the root intervals on the y-axis.

The procedure is repeated for all template rectangles, to obtain a Nichols plot of the upper and lower root intervals versus controller phase intervals for the entire interval template.

At each Θ, the uppermost and lowermost root intervals are picked from this plot and designated as the upper and lower bound intervals. The remaining root intervals are discarded.

3.1.4. *Composing bounds from bound intervals.* At each Θ, the upper and lower bounds are obtained as the maximum and minimum of the upper and lower bound intervals. Now, for any interval, the maximum and minimum occur at the right and left endpoints of the interval. Hence, the upper and lower bounds are simply the left and right endpoints of the respective bound intervals at each Θ.

3.1.5. *Significance of root and bound intervals.* The interval plant template at ω contains all the points in the usual point template generated at ω.

The upper root interval for a template rectangle at a given Θ, contains the entire range of upper root values computed for all points in the template rectangle, at all $\theta \in \Theta$. Similarly for the lower root interval.

The upper bound interval at a given Θ contains the upper bound for all points in the entire template, at all $\theta \in \Theta$. Similarly for the lower bound interval.

3.1.6. *Boundary template rectangles.* For the case of point templates, it is shown in [5] that only the boundary plants of the template need to be considered in bound generation for the various specs. For the case of interval plant templates, it can be similarly shown that only the boundary template rectangles need to be considered in bound generation using IABG algorithms.

3.1.7. *Computational reliability of bounds.* All computations in this section (as in all of IQFT) are done using machine interval arithmetic, which automatically takes care of all kinds of computational errors, such as round off, truncation, and approximation. This produces bound values that are not only exact for the given interval plant template, but are also reliable.

3.1.8. *IABG algorithm for robust sensitivity reduction specification.* The main steps of the IABG algorithm in [19], called as Algorithm SRI, can now be outlined.

First, any of the IATG algorithms described in section 2 is used to generate the interval template at the given frequency, and the set of boundary template rectangles extracted from the template using the boundary extraction algorithm in [21].

Then, the working phase interval Y is found as per section 3.1.2, and subdivided into several phase subintervals of say, 5 deg phase widths.

Next, the root intervals are computed as per section 3.1.1 at each of these subintervals, and the bound intervals composed as per section 3.1.3.

Lastly, the bounds are composed from these bound intervals as per section 3.1.4.

3.2. **Robust gain-phase margin specification.** At a given frequency ω, the robust gain-phase margin specification can be written as

$$\left|\kappa\left(j\omega\right)\mathfrak{g}\left(j\omega,\lambda\right)\ /\ \left(1+\kappa\left(j\omega\right)\mathfrak{g}\left(j\omega,\lambda\right)\right)\right| \leq w_m\left(\omega\right), \quad \forall \lambda \in \Lambda^0$$

The IABG algorithm for this spec, called as Algorithm RSBI in [19], is identical to the one in the preceding subsection, except for very some minor changes in the RHS expressions in (2).

3.3. **Robust tracking specification.** At a given frequency ω, the robust tracking specification can be written as

$$(5) \qquad \left|\kappa\left(j\omega\right)\mathfrak{g}_i\left(j\omega,\lambda\right)\,/\,\left(1+\kappa\left(j\omega\right)\mathfrak{g}_i\left(j\omega,\lambda\right)\right)\right|\;/\;\left|\kappa\left(j\omega\right)\mathfrak{g}_k\left(j\omega,\lambda\right)\,/\,\left(1+\kappa\left(j\omega\right)\mathfrak{g}_k\left(j\omega,\lambda\right)\right)\right| \leq \delta\left(\omega\right),$$

for all pairs of plant elements $\mathfrak{g}_i, \mathfrak{g}_k$ in the plant family. Consider an arbitrary but fixed *pair* of boundary rectangles and recall that G denotes the magnitude interval and Φ the phase interval of a template rectangle. Then, by substituting these quantities in (5) squaring both sides and simplifying, gives the quadratic inequality

$$\left\{G_i^2 G_k^2\left(1-\frac{1}{\delta^2}\right)\right\}k^2 + 2\left\{G_k^2 G_i\cos\left(\Phi_i+\Theta\right) - \frac{G_k G_i^2}{\delta^2}\cos\left(\Phi_k+\Theta\right)\right\}k + \left\{G_k^2 - \frac{G_i^2}{\delta^2}\right\} \geq 0$$

Making the substitutions $X_1 := \frac{1}{G_i}; X_2 := \frac{1}{G_k}; X_3 := \Psi_i + \Theta; X_4 := \Psi_k + \Theta$ and solving the above with the equality sign gives

(6)

$$K_{root}^u, K_{root}^l := \left(\left(\frac{X_2\cos X_4}{\delta^2} - X_1\cos X_3\right) \pm \sqrt{\left(\frac{X_2\cos X_4}{\delta^2} - X_1\cos X_3\right)^2 - (1-\frac{1}{\delta^2})(X_1^2 - \frac{X_2^2}{\delta^2})}\right) / \left(1-\frac{1}{\delta^2}\right)$$

Thus, for the considered pair of boundary template rectangles, the allowable magnitude range of the controller at Θ is $[0, \min\left\{K_{root}^l\right\}] \bigcup [\max\left\{K_{root}^u\right\}, \infty)$. The allowable magnitude range of the controller at Θ for the entire interval plant template is obtained by taking the intersection of the magnitude ranges over all pairs of boundary template rectangles.

Just as is the case with (2), direct interval arithmetic evaluation of (6) produces overestimated values of K_{root}^u and K_{root}^l, due to multiple occurrences of the intervals $X_1, ..., X_4$ in the functional expressions and the interval dependency effect. However, by combining the concept of *monotonicity* with the tool of *subdivision*, the root intervals can be obtained directly and exactly, with no need for any initial guess values or algorithmic iterations (as in classical optimization methods). Then, the computed bounds also turn out to be exact for the given interval template. These ideas form the basis of the IABG algorithm in [17], called Algorithm TSI, that generates tracking bounds.

The procedure to solve QFT design problems in class C are next given (due to space limitations, the procedures for class A and B problems are not given in this paper).

4. Procedure for class C problems

Extensions of some key concepts are first outlined, followed by the procedure to solve class C problems.

Let Ω^0 denote the given design frequency range, and Ω denote a fixed but arbitrary frequency interval in Ω^0.

4.1. **Natural interval extensions.** The natural interval extension in Definition 2.1 is extended for the interval frequency case as follows: The expression which arises if each occurrence of λ and ω in $f_{mag}(\omega,\lambda)$ is replaced Λ and Ω, if each occurrence of a pre-declared function (like $\sin, \cos, \exp$, etc.) is replaced by the corresponding pre-declared interval function, and if the arithmetic operations in $f_{mag}(\omega,\lambda)$ are replaced by the corresponding interval arithmetic operations, is called as the natural interval extension of $f_{mag}(\omega,\lambda)$ to Ω and Λ. The natural interval extension of f_{mag} is denoted as $F_{mag}(\Omega,\Lambda)$. Similarly for $F_{ang}(\Omega,\Lambda)$ and $F(\Omega,\Lambda)$.

4.2. **Interval frequency plant template.** By extension of the definition for the point frequency case, the plant template at frequency interval Ω can be defined as $\mathcal{G}(\Omega) := \{f(\omega, \lambda), \lambda \in \Lambda^0, \omega \in \Omega\}$.

The plant template for the case of frequency interval can also be generated using any of the IATG algorithms in section 2, by considering Ω as just another interval parameter for purposes of subdivision. The resulting template is called as an interval *frequency* plant template. An interval frequency plant template also comprises of one or more phase-magnitude rectangles, called as template rectangles, just as for the point frequency case. A key property of the interval frequency plant template at Ω is that it always encloses the exact template $\mathcal{G}(\Omega)$, that is, the interval frequency plant template at Ω contains all the points in the usual (point frequency) templates generated at all $\omega \in \Omega$.

4.3. **Working phase interval.** The working phase interval Y is now a function of frequency, as z_s depends on frequency through the spec. The working phase interval Y corresponding to the Ω under consideration can be found from (4).

4.4. **Spacing the frequency intervals and the bounds.** The designer can choose among various criteria for obtaining frequency intervals from the given design frequency range. The following criterion is chosen here.

Spacing criterion : Let Ω' be an adjacent frequency interval to Ω. Then, the maximum of the upper bound at Ω and maximum of the upper bound at Ω' should be within some specified distance or spacing, denoted Υ dB (note that the maximums of the bounds need not occur at the same Θ). Further, the minimum of the lower bound at Ω and minimum of the lower bound at Ω' should also be within Υ dB.

The above criterion is chosen just to illustrate the methodology of the proposed procedure. Other criteria can be specified by the designer to suit the needs of the problem, and the proposed procedure can be suitably adapted to achieve the ends.

4.5. **Size of template rectangles for problem class C.** To solve problems in this class, firstly an acceptable size of template rectangles needs to be determined and specified to the IATG algorithms, as follows.

Let ε denote the desired accuracy of the bounds in dB, and let ε be split into two positive quantities ε_Q and ε_G such that their sum equals ε. Further, let the working phase interval Y be subdivided such that width (in dB) of function $Q_\pm$ defined as

$$Q_\pm := -\cos(X) \pm \sqrt{\cos^2(X) - z_s}$$

is at most ε_Q on every subdivision. Let $\widehat{w}$ denote the smallest width of these subdivisions. Then, it is shown in [23] that the required size of the interval plant template (for specifying to the IATG algorithms) is as follows: the magnitude and phase sides of each template rectangle have lengths at most ε_G and $\widehat{w}$, respectively.

4.6. **The procedure.** A unified procedure for obtaining sensitivity reduction bounds is outlined below. The procedure generates bounds which are of prescribed accuracy ε dB, and spaced such that bounds of adjacent frequency intervals have their maximum magnitudes within Υ dB, and similarly for the minimum magnitudes.

The steps of the unified procedure for robust sensitivity specification are

1. **Split the accuracy**: The accuracy ε is split into two positive quantities ε_Q and ε_G such that their sum equals ε.
2. **Subdivide the design frequency range** : The design frequency range is successively bisected till the spacing criterion (given earlier) is achieved at each frequency interval Ω.
3. **Subdivide the working phase intervals**: At each frequency interval Ω, the following is done. The working phase interval Y in (4) is first computed. Then, Y is successively bisected till over each subdivision X, the width (in dB) of function $Q_\pm = -\cos(X) \pm \sqrt{\cos^2(X) - z_s}$ is at most ε_Q. Lastly, the smallest width of these subdivisions is found and denoted as $\widehat{w}$.

4. **IATG at each frequency interval**: For each frequency interval Ω, using any IATG algorithm an interval frequency plant template comprising of template rectangles of prescribed size is generated: each template rectangle should have its magnitude side of length at most ε_G dB, and phase side of length at most $\widehat{w}$. Then, the boundary template rectangles are extracted, while the rest are discarded from each interval frequency plant template.
5. **IABG at each frequency interval**: At each frequency interval Ω, using the X intervals obtained in step 3 above, the bound intervals are composed as in section 3.1.3, and then the bounds are composed from these bound intervals as in section 3.1.4.

The procedure is identical for the robust gain-phase margin specification, except that the respective algorithm is used in the IABG step above.

4.7. Illustrative example for class C problem.

Example 4.1. *The transfer function of a system found in active noise and vibration control, with highly underdamped resonances, is*

$$g(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2},\ \omega_n \in [0.75, 1.25],\ \xi \in [0.02, 0.06]$$

The nominal parameters are $\omega_{n0} = 0.75$ and $\xi_0 = 0.02$. The robust gain-phase margin specification is $w_m = 1.2$ dB. The design frequency range is $\Omega^0 = [0.5, 1.5]$. The objective is to generate the bounds with an accuracy $\varepsilon = 2.5$ dB and spacing $\Upsilon = 7.5$ dB over the given design frequency range.

Notes: The design frequencies (as points or as intervals) and the controller phases at which bounds are to be generated are not selected here, but are rather left to be appropriately determined by the procedure. However, the design frequency range, the accuracy of the bounds, and the spacing between them are prescribed. This puts the design problem in class C.

The above unified procedure is applied to generate the robust stability bounds of prescribed accuracy and spacing. In Step 1, the prescribed accuracy $\varepsilon = 2.5$ dB is split as $\varepsilon_Q = 2.1, \varepsilon_G = 0.4$ dB, so that $\varepsilon = \varepsilon_Q + \varepsilon_G$. In Step 2, the given design frequency interval $\Omega^0 = [0.5, 1.5]$ is successively bisected till the spacing criterion is satisfied at each frequency interval Ω. This step gave 17 frequency intervals listed in Table 2. Then, steps 3 and 4 are executed at each frequency interval to generate the interval frequency template and extract the boundary rectangles. Lastly, the IABG algorithm for the robust gain-phase margin specification, Algorithm RSBI in [19], is applied to the boundary rectangles of the templates to generate the bounds at each frequency interval. The total time for these steps is about 300 seconds. For plotting purposes, the controller bounds are converted to those on the nominal loop transmission function $L_o(s)$. Figure 1 shows the plots of these bounds at selected frequency intervals Ω_4, Ω_5 and Ω_6 (bounds only for selected frequencies are plotted, to avoid cluttering the figure).

Comments on the automatic selection of frequency intervals: It is seen in Figure 1 that the frequency intervals are such that the spacing between the bounds (in the sense of section 4.4) is at most $\Upsilon = 7.5$ dB. Further, it is seen in Table 2 that the frequency intervals have different widths, with the maximum width as 0.13 and the minimum width as 0.03. The widths vary because the frequency intervals are automatically found in the unified procedure (in the subdivision Step 2), so that bounds of prescribed accuracy are generated.

In contrast, in the point QFT procedures, the designer (usually arbitrarily) selects the frequency values at which the design is to be performed. The unified procedure completely solves the difficulty of carefully analyzing and selecting design frequencies with the point QFT procedures. To emphasize this capability, note in Figure 1 that while proceeding from Ω_4 through Ω_6, the bounds shift considerably rightward at Ω_5. This phenomenon is important for loop-shaping, and the unified procedure captures it automatically. However, it is quite likely that this phenomenon would be missed in a designer-selected frequency set.

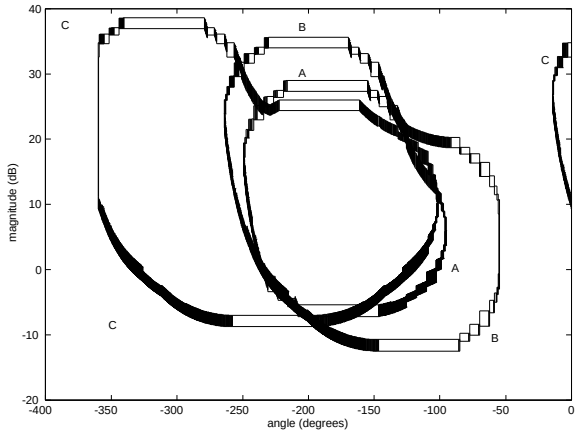

FIGURE 1. Robust gain-phase margin bounds for the underdamped system. The bounds are generated by the unified procedure for class C problems. 17 frequency intervals are automatically generated, and bounds for only 3 frequency intervals are plotted and marked as A - Ω_4, B - Ω_5,C - Ω_6. The specified bound accuracy is 2.5 dB.

TABLE 2. Frequency intervals obtained by subdividing the design frequency range.

$\Omega_1 = [0.50, 0.56]$	$\Omega_2 = [0.56, 0.62]$	$\Omega_3 = [0.62, 0.65]$	$\Omega_4 = [0.65, 0.68]$	$\Omega_5 = [0.68, 0.75]$	$\Omega_6 = [0.75, 0.78]$
$\Omega_7 = [0.78, 0.81]$	$\Omega_8 = [0.81, 0.84]$	$\Omega_9 = [0.84, 0.87]$	$\Omega_{10} = [0.87, 0.90]$	$\Omega_{11} = [0.90, 0.93]$	$\Omega_{12} = [0.93, 1.00]$
$\Omega_{13} = [1.00, 1.06]$	$\Omega_{14} = [1.06, 1.12]$	$\Omega_{15} = [1.12, 1.25]$	$\Omega_{16} = [1.24, 1.37]$	$\Omega_{17} = [1.37, 1.50]$	

Comments on the accuracy of the bounds: The maximum possible error in the generated bounds is readily obtainable from the widths of the plotted intervals in Figure 1. It is seen from the figure that this maximum error is less than the prescribed accuracy of $\varepsilon = 2.5$ dB.

Comments on the automatic selection of phase intervals: The controller phase intervals plotted in Figure 1 have different widths at a given frequency interval. Further, the widths also vary with the frequency interval. The widths vary in this manner because they are adaptively and automatically found in the unified procedure (in the subdivision Step 3), so that bounds of prescribed accuracy can be generated. In contrast, in the point QFT procedures, the designer arbitrarily selects the phase values at which the bounds are to be generated.

5. CONCLUDING REMARKS

It has been shown in this paper how IQFT lends rigor and reliability to Horowitz's basic QFT design technique. Several fundamental issues in QFT, such as selection of design frequencies, phases, and plant parameter combinations, are resolved in IQFT.

Much, however, remains to be added to the baggage of IQFT tools, even for the basic single input-output linear plant case. Preliminary work [34] on loop-shaping using global optimization tools of IA has met with some success, but needs to be made computationally more efficient and tested on more examples. The computational verification of existence of a solution to the basic QFT design problem seems to be possible, and is an exciting prospect in IQFT. Further, the cases of multivariable and nonlinear plants are as yet untouched in IQFT.

REFERENCES

[1] F. N. Bailey, D. Panzer, and G. Gu. Two algorithms for frequency domain design of robust control systems. *International Journal of Control*, 48(5):1787–1806, 1988.

[2] C. Borghesani, Y. Chait, and O. Yaniv. The Quantitative Feedback Theory Toolbox for MATLAB. 1995.

[3] Y. Chait, C. Borghesani, and Y. Zheng. Single loop QFT design for robust performance in the presence of non-parametric uncertainties. *Trans. of the ASME Journal of Dynamic Systems, Measurement and Control*, 117:420–424, 1995.

[4] Y. Chait and C. V. Hollot. A comparision between H-infinity methods and QFT for a SISO plant with both parametric uncertainty and performance specifications. In O. D. I. Nwokah, editor, *Recent Developments in quantitative feedback theory*, pages 33–40. 1990.

[5] 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.

[6] W. Chen and D. J. Ballance. Plant template generation for uncertain plants in QFT. *Trans. of the ASME Journal of Dynamic Systems, Measurement and Control*, 121:359–364, 1999.

[7] B. Cohen, M. Nordin, and P. O. Gutman. Recursive grid methods to compute value sets for transfer functions with parametric uncertainty. In *Proc. of ACC*, pages 3861–3865, 1995.

[8] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, New York, 1983.

[9] I. M. Horowitz. Survey of Quantitative Feedback Theory (QFT). *International Journal of Control*, 53(2):255–291, 1991.

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

[11] S. Jayasuriya. Frequency domain design for robust performance under parametric, unstructured, or mixed uncertainties. *Trans. of the ASME Journal of Dynamic Systems, Measurement and Control*, 115:439–451, 1993.

[12] R. B. Kearfott. Abstract generalized bisection and a cost bound. *Mathematics of Computation*, 49(179):187–202, 1987.

[13] R. B. Kearfott. Some tests of generalized bisection. *ACM Transactions on Mathematical Software*, 13(3):197–220, 1987.

[14] R. Klatte, U. Kulisch, M. Neaga, D. Ratz, and Ch. Ullrich. *PASCAL–XSC Language Reference with Examples*. Springer-Verlag, Berlin Hidelberg, 1993.

[15] R. E. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, 1979.

[16] R. E. Moore. Global optimization to prescribed accuracy. *Computers Math. Applic.*, 21(6/7):25–39, 1991.

[17] P. S. V. Nataraj. Computation of QFT bounds for robust tracking specifications. submitted.

[18] P. S. V. Nataraj and A. Prakash. On subdivision direction selection for QFT template generation. submitted.

[19] P. S. V. Nataraj and G. Sardar. Computation of QFT bounds for robust sensitivity and gain-phase margin specifications. *Trans. of the ASME Journal of Dynamic Systems, Measurement and Control*, 122:528–534, September 2000.

[20] P. S. V. Nataraj and G. Sardar. Template generation for continuous transfer functions using interval analysis. *Automatica*, 36:111–119, 2000.

[21] P. S. V. Nataraj and S. Sheela. An algorithm for extraction of boundary rectangles from interval QFT templates. submitted.

[22] P. S. V. Nataraj and S. Sheela. A template generation algorithm using parallel function evaluations and adaptive subdivisions. submitted.

[23] P. S. V. Nataraj and S. Sheela. A unified procedure for generation of QFT bounds to a prescribed accuracy. submitted.

[24] O. D. I. Nwokah, S. Jayasuriya, and Y. Chait. Parametric robust control by quantitative feedback theory. *AIAA journal of Guidance and Control*, 5:207–214, 1992.

[25] L. B. Rall. *Automatic Differentiation, Techniques and Applications*. Number 120 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1981.

[26] D. Ratz and T. Csendes. On the selection of subdivision directions in interval branch-and-bound methods for global optimization. *Journal of Global Optimization*, 7:183–207, 1995.

[27] J. M. Rodrigues, Y. Chait, and C. V. Hollot. An efficient algorithm for computing QFT bounds. *Trans. of the ASME Journal of Dynamic Systems, Measurement and Control*, 119(3):548–552, 1997.

[28] S. M. Rump. INTLAB - INTerval LABoratory. In T. Csendes, editor, *Developments in Reliable Computing*. Kluwer Academic Publishers, 1999.

[29] G. Sardar and P. S. V. Nataraj. A Template Generation Algorithm for Non-rational Transfer Functions in QFT Designs. In *Proc. 36th IEEE Conf. Decision and Control*, pages 2684–2689, San Diego, USA, 1997.

[30] S. Sheela and P. S. V. Nataraj. Fast template generation for non-rational transfer functions. submitted.

[31] S. Sheela and P. S. V. Nataraj. A reliable parallel-adaptive algorithm with back-tracking for QFT template generation. *IETE Jl. Technical Review*. to appear.

[32] M. Sidi. Feedback synthesis with plant ignorance, nonminimum phase, and time-domain tolerances. *Automatica*, 12, 1976.

[33] D. F. Thomspon and O. D. I. Nwokah. Analytical loop shaping methods in quantitative feedback theory. *Trans. of the ASME Journal of Dynamic Systems, Measurement and Control*, 116:169–177, 1994.

[34] P. Vamsikrishna. Design of controllers using global optimization techniques. Master's thesis, Systems and Control Engg. Group, IIT Bombay, India, 2000.

[35] O. Yaniv and I. Horowitz. Quantitative feedback theory - reply to criticisms. *International Journal of Control*, 40:945–962, 1987.

[36] Y. Zhao and S. Jayasuriya. An H-infinity formulation of quantitative feedback theory. *Trans. of the ASME Journal of Dynamic Systems, Measurement and Control*, 120(3):305–313, 1998.