Recursive Grid Methods to Compute Value Sets and Horowitz-Sidi Bounds *

Per-Olof Gutman † Mattias Nordin ‡ Bnayahu Cohen §

March 6, 2006

Abstract

In this paper, recursive extensions to the standard equidistant Grid method are proposed whereby the gridding is adapted locally such that a prescribed distance is achieved between neighbouring points in the computed value set (template). Also presented is the Prune algorithm which finds the outer border of a value set defined by a set of points whose nearest neighbour lies within a prescribed distance. The Prune algorithm is part of the recursive grid methods, but can also be used independently with other methods to compute value sets. As an alternative to analytical or search algorithms, a recursive grid algorithm is presented to compute Horowitz-Sidi bounds (QFT bounds, or boundaries). Isaac Horowitz's contribution to computational methods for QFT is outlined in the perspective of the presented algorithms.

Keywords: Quantitative Feedback Theory (QFT), Robust control, template generation, bound generation.

*Supported by NUTEK grant no 9156700, the Fund for the promotion of research at the Technion—Israel Institute of Technology, and the European Union Marie Curie Transfer of Knowledge Program.

†Faculty of Civil and Environmental Engineering, Technion — Israel Institute of Technology, Haifa 32000, Israel (address for correspondance), email: peo@techunix.technion.ac.il, telephone: +972 4 8292811, fax: +972 4 8228898, on sabbatical with Dipartimento di Ingegneria, Università del Sannio, Benevento, Italy.

‡Volvo Construction Equipment Components, 63185 Eskilstuna, Sweden, email: mattias.nordin@volvo.com

§M.E.S. - Medical Electronic Systems Ltd., P.O.B. 3017, Ceasarea 38900, Israel, email: Beni@mes-ltd.com

Introduction, with a historical note

When the first author (Gutman) arrived to Rehovot, Israel, in 1984 to take up a position as a control engineer at El-Op Electro-Optics Industries Ltd., Isaac Horowitz was an active and well established professor at the Department of Mathematics of the Weizmann Institute of Science. QFT was already well developed by Horowitz and his numerous graduate students, from the first QFT paper [22] to the recent Ph.D.-thesis by Yaniv, published as [39]. All Horowitz's students applied the new methods to various examples, and wrote appropriate software. The mathematician Dr. Linda Neumann was employed by Horowitz as a scientific programmer, a QFT design program had been written in FORTRAN, and many extremely challenging design problems were solved, e.g. the 5x5 MIMO flight control problem, [23].

When my previous Ph.D.-thesis advisors, Professor Per Hagander, and Professor Karl Johan Åström in Lund, Sweden, learnt that I was relocating to Rehovot, they strongly urged me to contact Prof. Horowitz and his group. Professor Horowitz graciously offered me to teach a course on adaptive control at the Weizmann institute, and Dr. Neumann and Dr. Yaniv showed me how to efficiently solve by QFT an uncertain servo control problem I had been assigned at El-Op. I understood the immense potential of QFT and we have since used it for many research and applied problems, e.g. [18, 38, 32, 30, 17].

Professor Horowitz seemed to enjoy discussing QFT with me, maybe because of perceiving my being a "convert" from another school of control. He

suggested the control of uncertain plants with "nuisance" non-linearities as a possible research area, with our discussions reflected on page 342 (point 17) and page 474 in his book [21]. In fact, much of the subsequent research by myself and my co-workers dealt with these topics, e.g. [32, 33, 6, 31].

Professor Horowitz was also very interested in the computerized implementation of QFT design and analysis tools. In his original design program, mentioned above, value sets (templates) of parametric rational functions, with each parameter belonging to an interval, were computed, for each chosen frequency, by the *Grid Method*, whereby each parameter interval is equidistantly gridded, and the values of the rational function are computed at the grid points in the parameter space. The Grid Method is also the only template computation method in the Horowitz inspired [19] which was also written in FORTRAN and made use of the user interaction modules of the control design and analysis programs SYNPAC, and IDPAC developed at Lund Institute of Technology, Sweden, [2], and in the "The QFT Toolbox for Matlab" [37].

It is however well known, see e.g. [16], that the Grid Method may fail to yield a set of template points suitable for control systems design. Therefore many alternative template computation methods have been proposed over the years, some of which are briefly mentioned below. For the successor of [19], written mainly by the authors of this paper, `Qsyn - the Toolbox for Robust Control Systems Design for use with Matlab`, [15], more

efficient template computation algorithms were developed: the Real Factored Form method, [16], the Recursive Edge Grid Method, [11], and the Recursive Grid Methods together with a "pruning" algorithm to find the outer border of a template, published here. The Recursive Grid and pruning algorithms were originally presented as the conference contribution [12].

Let us recall that a Horowitz-Sidi bound for a given specification and for a given frequency ω, [22], is the border in the complex plane between the feasible and infeasible sets of the nominal compensated open loop frequency function value $L_{nom}(j\omega) = G(j\omega)P_{nom}(j\omega)$ where G denotes the compensator, and P_{nom} the nominal plant case. If and only if $L_{nom}(j\omega)$ belongs to the feasible set is the specification satisfied for all plant cases at the given frequency ω.

In Horowitz's original software, see also the Matlab m-files on page 466ff in [21], a Horowitz-Sidi bound for a given frequency ω and a given specification is computed point-by-point as a line search along constant phase grid lines in the Nichols chart, conceptually as follows:

Algorithm 1:

1. Choose a phase ϕ;
2. Choose an initial very high gain, A;
3. Choose implicitly an appropriate $G(j\omega)$ such that $|L_{nom}(j\omega)| = A$, and

$\arg(L_{nom}(j\omega)) = \phi$;

4. Check if the specification is satisfied. If No, let $A = A + \delta A$, if Yes, let $A = A - \Delta A$. Return to point 3, until the *upper* Horowitz-Sidi bound point is found with required accuracy, or the desired gain range is searched;

5. If the upper Horowitz-Sidi bound point was found, choose an initial very low gain, a;

6. Choose implicitly an appropriate $G(j\omega)$ such that $|L_{nom}(j\omega)| = a$, and $\arg(L_{nom}(j\omega)) = \phi$;

7. Check if the specification is satisfied. If No, let $A = A - \delta A$, if Yes, let $A = A + \Delta A$. Return to point 5, until the *lower* Horowitz-Sidi bound point is found with required accuracy (note that the upper and lower bound points may co-incide);

8. Let $\phi = \phi + \Delta\phi$. Return to point 2, until all chosen phases have been investigated.

Graphically, Algorithm 1 is very elegant. It is performed by sliding a given template with its nominal along constant phase grid lines in the Nichols chart, while investigating if the given specification is satisfied. The bound points are marked on the Nichols chart. Clearly, Isaac Horowitz used his deep control theoretical understanding when proposing this algorithm.

Algorithm 1 was used in [19] and is the main algorithm in [37]. However, Algorithm 1 does not compute the correct Horowitz-Sidi bound, if the bound intersects a given phase grid line more than twice, since the complete phase grid line is not investigated. See Figure 7 below. [4] was the first work to investigate this phenomenon of *multi-valued* bounds.

In general, some search procedure is necessary to find the Horowitz-Sidi bounds. In certain cases, notably for sensitivity specifications, the Horowitz-Sidi bounds can be computed explicitly by quadratic inequalities, [36], which represents a clear improvement over Horowitz's original, and is also used in [37]. Here, the recursive grid method for the computation of Horowitz-Sidi bounds used in Qsyn since its inception [15] is presented, noting that it was independently discovered also in [27], and extended in [26]. Other contributions to the bound computation problem are e.g. [8], [13], [35], and [28].

We recognize with gratitude the contribution by Isaac Horowitz who together with his students developed the very practical and useful control design technique called QFT. His inspiration guided us and others to create efficient computational tools for QFT.

This paper is organized as follows: The problem of computing value sets is discussed in section 1, together with a brief literature survey. The Prune algorithm is presented in detail in section 2, together with an illustrative example. The Recursive Grid algorithms are outlined in sections 3 and 4,

together with a physical example. The recursive bound computation algorithm is found in section 5. A brief Conclusion and an Acknowledgement concludes the paper.

1 Computing value sets

Example Consider the real valued scalar function $f(x) = 1/(0.0001 + (x - 0.95552928514718)^2)$, of the real valued variable $x \in [0, 1]$. It is easy to analytically calculate that the value set (range) of f is $[1.09512686477738, 10000]$. It is however not trivial to find the upper boundary of the range by numerical methods. If you would use the Grid Method and divide the x-interval equidistantly into e.g. 1001 points, then you will get $f(0.955) = 9972.063984025603$, and $f(0.956) = 9977.891738555019$ whereby $f(0.956) - f(0.955) < 20$, but the found maximum value is removed more than 20 units from the true upper limit 10000. What would induce you to subdivide the x-interval further? How do you know that enough is enough? For comparison, the Matlab function `fplot('1/(0.0001 + (x-0.95552928514718)^2)',[0,1],1e-3)` gives the maximum as 9977.89173855495, in spite of the fact that adaptive gridding of the x-values is used, and a relative tolerance of 0.1%, i.e. 10 units, was demanded.

The example shows that computing value sets is non-trivial. Over the years a number of algorithms have been proposed to compute value sets of parametric rational functions, with each parameter belonging to an interval.

Some methods require a special structure of the parametric rational function. See e.g. [3], [14], [1], [16], [28], [34]. A survey is found in [5]. Some recent works investigate symbolic computation and composition properties, see e.g. [10], [9].

The most straightforward, but computationally intensive method is the Grid method, whereby each parameter interval is equidistantly gridded, and the values of the rational function are computed at the grid points in the parameter space. When the Edge Theorem holds, [14], it is sufficient to grid the edge in the parameter space, i.e. in turn, each parameter is gridded while all the others are kept at an interval end points. The Edge Grid method has been extended to the Recursive Edge Grid method in [11] where each edge is gridded adaptively, by e.g. bisection, to achieve a prescribed resolution of the value set, i.e. a prescribed maximum distance between the computed neighbouring points. The Recursive Edge Grid method compares favourably with the method of [14] from a computational point of view.

For the purpose of investigating robust stability, [1], or for robust control system design, [22], only the outer border of the value set is needed. Most of the above mentioned methods give interior points as well. This paper presents an algorithm that finds the outer border points, under the assumption that the value set was computed with a known resolution. Appropriately, this algorithm is called the Prune algorithm. The algorithm may substantially reduce the computational effort when computing value sets for tree structured

transfer functions, [1]. An algorithm akin to the Prune Algorithm in [12] was published several years later in [7].

One aim of this paper is the presentation of Recursive Grid algorithms that give a prescribed resolution of the outer border of the value set, for all types of parametric frequency functions for which the original Equidistant Grid method is applicable. It should be noted that also unstructured uncertainty of the type $P(j\omega) = P_{\rm nom}(j\omega) + M(j\omega)$ or $P(j\omega) = P_{\rm nom}(j\omega)(1 + M(j\omega))$, $r = |M(j\omega)| \leq m(\omega)$, $\phi = \arg(M(j\omega)) \in [0, 2\pi]$ can be parameterized in r and ϕ, see [15], [20]. In the first Recursive Grid method I, the parameter space is recursively gridded such that the prescribed resolutions is achieved. The Prune algorithm is applied during and after the recursion.

It should be noted that all grid methods compute underbounds of the true value set, unless the parametric rational function has some special structure [14], [16], or satisfies some regularity condition [24]. This restriction holds, by the way, also for function plotting routines in programs like Matlab, [25].

The second Recursive Grid method is applicable when a Lipschitz constant is known for the parametric rational function. Then interior points sufficiently far away from the hitherto computed interim outer border of the value set need not to be computed at all, resulting in substantial computational savings. Recursive gridding is interleaved with pruning.

A breakthrough in template computation was achieved in [29] and papers referenced therin, where the outer bounds of value sets of parametric

frequency functions are computed without knowing a Lipschitz constant, by the use of interval analysis. The transfer function is decomposed into elementary functions, and the parameter intervals subdivided, such that the gain and phase extent can be analytically computed on the subintervals. The proposed method is shown by example to be computationally superior to other methods, including the ones in this paper.

2 The Prune Algorithm

Without restriction, consider a discrete set of disjoint complex points $V = \{v_1, v_2, ..., v_n\}$, where identical points are considered as one. Introduce

Definition 1 *A set V is said to be ϵ-connected if for every pair of points v_i,$v_j \in V$, there is a sequence of points V_{ij}=$\{v_{ij}^1$,v_{ij}^2,...,$v_{ij}^k\} \subset V$ such that $v_{ij}^1 = v_i$, $v_{ij}^k = v_j$ and $|v_{ij}^m - v_{ij}^{m+1}| \leq \epsilon \; \forall m = 1, 2, 3..k-1$.*

Let now V be ϵ-connected and construct

$$C(V) = \bigcup_{i=1}^{n} \left\{ x : |x - v_i| \leq \frac{\epsilon}{2} \right\} \tag{1}$$

It follows from (1) and Definition 1 that $C(V)$ is a connected, but not necessarily simply connected set in the complex plane. Let $\partial C(V)$ denote the outer border of $C(V)$, i.e. the border encountered when approaching $C(V)$ from infinity. A point $v \in V$ is called an *interior* point of V if $\partial C(V) = \partial C(V \setminus \{v\})$. A point $v \in V$ is called a *border* point of V if it is not an *interior* point. Let $B(V) \subset V$ be the set of *border* points of V and $I(V) = V \setminus B(V)$ the set of

interior points of V. It is easily shown that $B(V)$ is also ϵ-connected and that $\partial C(B) = \partial C(V)$. Define the ϵ-circles $c_i = \{z : |z - v_i| = \epsilon/2\}$, $i = 1, 2, \ldots, n$. In figure 1 an example of V with its ϵ-circles are plotted. $B(V)$ is the set of all points with solid circles and $I(V)$ is the set of points with dotted circles. It is obvious from the construction of $C(V)$ that $\partial C(V)$ is a continous curve

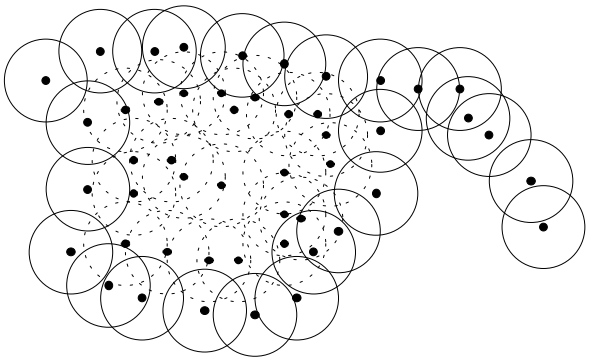

Figure 1: V, B and $\partial C(V) = \partial C(B)$

of circular arcs $A(V) = \{a_1, a_2, \ldots, a_k\}$. Each of these arcs is associated to some *border point* in a sequence $B^*(V) = \{b_1, b_2, \ldots, b_k\}$. It holds that

$$B(V) = \bigcup_{i=1}^{k} \{b_i\} \tag{2}$$

Note that the arcs are unique but two or more arcs can have the same center, and therefore not all points in $\bigcup_{i=1}^{k} \{b_i\}$ need to be disjoint, see figure 2.

We can now construct an algorithm, the Prune Algorithm, which finds $B(V)$ by recursively finding the points of $B^*(V)$. This can be done by finding these circular arcs and their associated centers in V. The idea is to track $\partial C(V)$ and recursively find the consecutive arcs in $A(V)$ The algorithm con-

sists of three parts: **(i)** Find a starting point b_1 and its sucessor b_2. **(ii)** Given to points b_i and b_{i-1} find the successor b_{i+1}. **(iii)** A terminating condition to know when $C(V)$ is encircled. Let us now proceed with the three steps:

(i) A starting point b_1 can be found by

$$b_1 = \arg\max_{v \in V} \text{real}(v) \tag{3}$$

i.e. one of the rightmost points in V. It is clear that the point $b_1 + \epsilon/2$ must belong to $\partial C(V)$. It is also clear that, remembering that the points in V are disjoint, no other point $v \in V$ gives a contribution to $\partial C(V)$ at $b_1 + \epsilon/2$. Hence b_1 is a *border* point and $b_1 + \epsilon/2 \in a_1$, see the example in figure 2. Then follow the arc a_1 in the positive direction until the first time it intersects another ϵ-circle round some point $v \in V$. All possible candidates $v \in V$ for intersection must satisfy $0 < |v - b_1| \leq \epsilon$. Since V is ϵ-connected there must be at least one such v. Define the angle

$$\phi_1(v) = \angle(v - b_1) - \arccos(|b_1 - v|/\epsilon) \tag{4}$$

The first circle that intersects a_1 is the one with the smallest angle $\phi_1(v)$, see $\phi_1(b_2)$ in figure 2. It now follows that

$$b_2 = \arg \min_{0<|v-b_1|\leq\epsilon} \phi_1(v) \tag{5}$$

If the minimizing argument in (5) is not unique, there are several circles that intersect in the same point of a_1. In that case choose the minimizing v of (5) farthest away from b_1, because the others only give a contribution to $\partial C(V)$ at the intersection point.

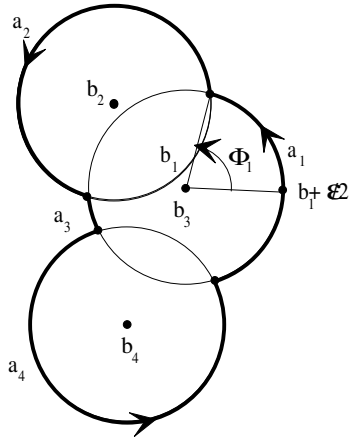

Figure 2: A three point exaple of how the Prune Algorithm works. Note that $B^*(V)$ includes four points although there are only three points in V

(ii) Consider two consecutive points $b_{i-1}, b_i \in B^*(V)$, see figure 3. The intersection point between a_{i-1} and a_i is denoted $x_{i-1,i}$. The next point $b_{i+1} \in B^*(V)$ is found by following the arc a_i from the intersection point $x_{i-1,i}$ until another intersection is encountered. All possible candidates $v \in V$

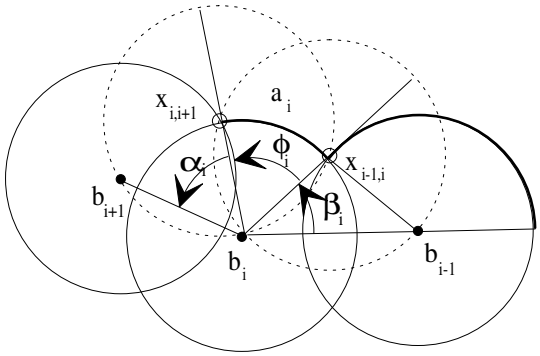

Figure 3: A step in the recursion (ii). Follow the arc a_i from $x_{i-1,i}$ until it intersects another ϵ-circle.

for intersection must satisfy $0 < |b_i - v| \leq \epsilon$. Since V is ϵ-connected there are

always possible candidates. Note that $b_{i-1} = b_{i+1}$ is possible. The point b_{i+1} can now be found by choosing the candidate with the smallest angle $\phi_i(v)$, see figure 3. Geometry gives:

$$\alpha_i(v) = \arccos(|b_i - v|/\epsilon) \tag{6}$$

$$\beta_i = \arccos(|b_{i-1} - b_i|/\epsilon) \tag{7}$$

$$\alpha_i(v) + \beta_i + \phi_i(v) = \angle\frac{v - b_i}{b_{i-1} - b_i} \tag{8}$$

from which

$$\phi_i(v) = \angle\frac{v - b_i}{b_{i-1} - b_i} - \arccos(|v - b_i|/\epsilon) - \arccos(|b_{i-1} - b_i|/\epsilon) \tag{9}$$

b_{i+1} is now given by

$$b_{i+1} = \arg\min_{0<|v-b_i|\leq 1} \phi_i(v) \tag{10}$$

If the minimizing argument in (10) is not unique choose the point farthest away from b_i, because the other minimizing arguments only gives a contribution to $\partial C(V)$ at the point $x_{i,i+1}$, which already is given by a_i.

(iii) Note that since (10) only depends on the two previos points in $B^*(V)$ we reach a closed orbit in $B^*(V)$ as soon as $b_1 = b_{k+1}$ and $b_2 = b_{k+2}$, because it follows then from induction that $b_m = b_{k+m}$ for all integers $m > 0$. We have then reached the starting arc a_1 and can terminate the recursion. Now the sequence $B^* = \{b_1, b_2, ..., b_k\} \subset V$ with the required conditions is found. In other words we have encircled the set V and found the *border* points.

Consider figure 3. An interpretation of one step in the algorithm works is that it takes the dotted circle of radius $\epsilon/2$ touching b_{i-1} and b_i and rolls it around b_i until it touches b_{i+1}. Hence all the consecutive points in $B^*(V)$ is given by rolling a circle with diameter ϵ around the points of V. The algorithm will catch concavity with a curvature less than $2/\epsilon$, which is evident from the rolling circle interpretation of the Prune Algorithm.

2.1 Example

Let us consider an example taken from [1, pages 158-159], with two value sets $A = \{x+i2 : x \in [-2, 4]\}$ and $B = \{x+iy : x \in [-3, 4], y \in [-1, 3]\}$ are given. We are interested in finding the border ∂C of the product $A \cdot B$. In [1] it is shown that $\partial C \subset \partial A \cdot \partial B$. In order to calculate ∂C the sets ∂A and ∂B are gridded with steplength 0.1 and then multiplied into a discrete set V. Note now that V is ϵ-connected with $\epsilon = 0.1 \min(\max|A|, \max|B|) = 0.2\sqrt{5} < 0.5$. We then compute an approximation of ∂C by applying the Prune Algorithm to V with $\epsilon = 0.5$. In figure 4 $V \subset \partial A \cdot \partial B$ and the approximation of ∂C is shown. Our implementation of the Prune Algorithm in MATLAB 6.5, [25], uses the efficient sparse matrix function to find the possible candidates, i.e. points within the distance ϵ. This example with 13200 points is pruned in 0.56 seconds on a 1.86 Ghz Pentium M computer, with a straightforward m-file implementation.

3 The Recursive Grid Algorithm I

Consider a continuous transfer function $P(j\omega, q)$, with q an uncertain parameter vector such that

$$q \in Q = \{q \mid \underline{q_i} \leq q_i \leq \overline{q}_i \; \forall \; i = 1, 2, .., l \; \} \tag{11}$$

We want to compute an accurate approximation of the value set $V(P, Q, \omega) = \{P(j\omega, q) \mid q \in Q\}$, for a fixed frequency ω. As pointed out in e.g. [1], dense gridding of the uncertain domain is often the only possibility to achieve this goal. But what is a "dense gridding"? It is obvious that this depends on the properties of $P(j\omega, q)$ and the required accuracy. A common "brute force" method is to grid the parameters equidistantly and then plot the values of $P(j\omega, q)$ for all these parameter combinations. Let us define a grid in the uncertainty set Q as follows: Let $\tilde{q}_i = \{\tilde{q}_i^1, ..., \tilde{q}_i^{n_i}\}$ be a vector such that $\tilde{q}_i^1 = \underline{q_i}$, $\tilde{q}_i^{n_i} = \overline{q_i}$, and $\tilde{q}_i^j - \tilde{q}_i^{j-1} = (\overline{q_i} - \underline{q_i})/(n_i - 1)$. With $q_{\min} = [\underline{q}_1, \underline{q}_2, ..., \underline{q}_l]$, $q_{\max} = [\overline{q}_1, \overline{q}_2, ..., \overline{q}_l]$ and $n = [n_1, n_2, ..., n_l]$ we now have the set of grid points

$$Q_{\mathrm{g}}(q_{\min}, q_{\max}, n) = \tilde{q}_1 \times \tilde{q}_2 \times ... \times \tilde{q}_l \subset Q \tag{12}$$

The size of Q_g is $N = \prod_{i=1}^{l} n_i$ and even for reasonably small l this might be prohibitively large, if a high accuracy is required. Therefore we construct a recursive algorithm that is adaptive in the sense that it makes necessary parts of the grid progressively finer, such that an ϵ-connected set of evaluations of the uncertain transfer function $P(j\omega, q)$ is achieved. The steps of the Recursive Grid I algorithm are:

1. Select an initial grid Q_g^0. The Q_g^0-gridding divides Q into $\prod_{i=1}^{l}(n_i - 1)$ smaller uncertainty sets, so called Q-boxes, [1]. Evaluate the transfer function at the N_0 gridpoints, and hence each Q-box has got the transfer function evaluated at all its 2^l vertices.

2. Select a Q-box. For each parameter, q_i, check the difference between the values of $P(j\omega, q)$ at the end points of the edges in the direction of q_i. If one of the differences is greater than ϵ, the grid is made finer in the q_i-direction *in this particular Q-box, not in the entire Q*. Evaluate the transfer function at the vertices of the new, smaller Q-boxes.

3. Repeat point 2 recursively for each of the smaller Q-boxes until all differences of adjacent vertex transfer function values are less than ϵ. The smaller Q-boxes have common points with the larger Q-boxes in a tree-like structure. Hence each "branch" and "sub-branch" are ϵ-connected, and whenever the end is reached of a sufficiently large branch, the Prune algorithm is used to reduce the number of memorized value set points.

4. Repeat points 2 and 3 for all original Q-boxes. When finished, prune a last time to get the outer border of the computed value set.

3.1 Example

Consider a simplified elastic two mass system. A motor with inertia $J_m = 0.4$ [kgm^2] is driving a load with inertia $J_l \in [5.6, 8]$ [kgm^2]. They are coupled

via a shaft with stiffness $k \in [5880, 5900]$ [Nm/rad] and damping $c \in [30, 300]$ [Nm/rad/s]. Introducing $q = [J_l, k, c]^T \in Q$, the uncertain transfer function from control torque to motor speed is given by

$$P(q, s) = \frac{J_l s^2 + ds + k}{J_l J_m s^3 + (J_l + J_m) d s^2 + (J_l + J_m) k s} \tag{13}$$

Let us compute the value set of (13) for $\omega = 10\pi$ [rad/s], with three different methods: (i) The Recursive Edge Grid Algorithm [11] (ii) The Recursive Grid Method and (iii) a "brute force" equidistant gridding, see figures 5 and 6. The required 2-norm resolution in the Nichols chart is 3.5 degrees and 0.5 dB, respectively.

(i): The Recursive Edge Grid Algorithm grids every edge of Q (11) finer and finer until its transfer function values are ϵ-connected. The total computed value set of the edges is then ϵ-connected and suitable for pruning. In this example 302 points were computed and the pruned border had 187 points.

(ii): The Recursive Grid Algorithm I computed 4328 points and the final border size was 241 points. Note that the Recursive Edge Grid Algorithm did not give the full value set, see the area below the S in figure 5.

(iii): To compare with equidistant gridding of Q we tried a grid with $17^3 = 4913$ points, somewhat more than in (ii). Note that this computed value set is suitable for pruning only with a much larger ϵ than required, and miss border segments with large concavity. Usually this feature is of the grid method is even more extreme, see e.g. the example in [3].

4 The Recursive Grid II Algorithm

A parametric frequency function $P(j\omega, q)$ satisfies a Lipschitz condition, if, for a given frequency ω and for any two parameter vectors $q_1, q_2 \in Q$, a positive constant $K(\omega)$ is known such that $|P(j\omega, q_1) - P(j\omega, q_2)| \leq K(\omega)|q_1 - q_2|$, where $||$ denote appropriate norms. In the initial Example in Section 1 $\max_{x\in[0,1]} f'(x)$ is the smallest possible Lipschitz constant. Clearly, if a Lipschitz constant is known, the required resolution of the gridding of the parameter set can be easily found as a function of the allowed error of the computed value set. In the initial Example in Section 1 it is easy to analytically compute $f'(s)$ and its maximum. For parametric transfer functions it is in general non-trivial to calculate a Lipschitz constant. A numerical computation would in general entail the same order of effort as computing the value set itself.

However, the recursive algorithm in the last section can be speeded up if a Lipschitz constant for the transfer function $P(j\omega, q)$ is known. Then, namely, the transfer function values at points *within* a Q-box whose vertex point values lie sufficiently far away from the hitherto computed interim outer border of the value set, also lie within that same interim border, and hence need not be computed. This means that some Q-box branches may not have to be computed to the last "leaf", resulting in computational savings. The modified algorithm has to start with an ϵ-connected set. The steps of the Recursive Grid II Algorithm are:

1. Use the Recursive Edge Grid method [11] along the edges of the original parameter domain Q, in order to achieve an ϵ-connected set whose border is found with the Prune Algorithm.

2. Select an initial grid Q_{g}^0. Evaluate the transfer function at its N_0 grid-points. Define the Q-boxes as in point 1 in section 3.

3. Select a Q-box of which at least one of its vertex values has been included in a previous prune operation. *If all its vertex values are sufficiently far away inside the interim border so that by the Lipschitz condition no interior point of the Q-box can assume a value outside the interim border, do not grid this Q-box further.* Otherwise proceed as in point 2 in section 3.

4. Repeat the above point recursively as in point 3 in section 3, with the difference that the prune operation is performed together with the interim border, when the number of new value points is larger than the number of interim border points. Thus, an updated interim border is computed.

5. Repeat the above until the whole of Q is covered.

5 A recursive Horowitz-Sidi bound algorithm

The bound computation method presented in this section was developed in 1995 and implemented in Qsyn, see [15]. Independently, the algorithm

was also discovered in [27] where, in addition, some computational problems around the "instability point" -1 are solved. The work [27] was recently updated and extended in [26].

Let $\{P_i(j\omega)\}$ denote the plant template, i.e. the value set of the plant frequency function, at the frequency ω, with i denoting a real valued vector index of the plant cases. Let $P_{\text{nom}}(j\omega)$ denote the nominal plant case. Let a *specification criterion function* $Z(G(j\omega), P_i(j\omega))$ be a function of the feedback controller frequency function value $G(j\omega) \in \mathbf{C}$ (to be assigned), P_i, and possibly other known frequency functions, for which a specification is specified. Let $z(\omega)$ be the specification value at frequency ω. Then a frequency domain specification may be defined as

$$f(G(j\omega)) \doteq Z(G(j\omega), P_i(j\omega)) - z(\omega) \;\; \text{OP} \;\; 0, \;\; \forall i \tag{14}$$

where OP $\in \{=, >, \geq, <, \leq\}$, and $f(G(j\omega))$ denotes the left hand side.[1]

Example The common *servo gain specification*, [22], for the standard two degrees-of-freedom feedback configuration is given by $a(\omega) \leq |FP_iG/(1+P_iG)| \leq b(\omega)$, $\forall i$, with F denoting the prefilter, $b > a > 0$, and the frequency argument of the frequency functions suppressed. From the servo gain specification emanates the *tolerance specification*, [21], which can be written in

[1]The specification definition in (14) may be generalized to several simultaneous inequalities, and for more than one frequency function value to be assigned in Z.

the form (14), with

$$Z = \frac{\max_i |P_i G/(1+P_i G)|}{\min_i |P_i G/(1+P_i G)|} \tag{15}$$

$$z(\omega) = \frac{b(\omega)}{a(\omega)} \tag{16}$$

$$\text{OP} = \leq \tag{17}$$

Example A *sensitivity gain specification* can be defined according to (14) by setting $Z = \max_i |1/(1+P_i G)|$, $z(\omega) > 0$, (and > 1 for some frequency range, according to Bode's integral theorem, ch. 10 in [21]), and OP=$\leq$.

The recursive bound computation algorithm can now be stated, for a specification given by equation 14:

1. Select a subset of the complex $L_{\text{nom}}(j\omega) = P_{\text{nom}}(j\omega)G(j\omega)$-plane for which the Horowitz-Sidi bound is to be computed, e.g. the subset whose gain $\in [-50, 50]$ dB, and whose phase $\in [-360, 0]$ degrees. Exclude from the subset the set $-P_{\text{nom}}(j\omega)\{P_i(j\omega)^{-1}\}$, where $\{P_i(j\omega)^{-1}\}$ denotes the template of plant inverses, since this set always belongs to the infeasible side of the Horowitz-Sidi bound, see Section 5.1;

2. Choose a grid resolution, e.g. 5 dB, and 5 degrees;

3. Grid the subset according to the chosen resolution;

4. Compute $f(G(j\omega))$ for the grid points, and collect the values into the matrix M;

5. Find the contour in M at value 0 which constitutes the desired Horowitz-Sidi bound, e.g. by using the Matlab command `contour(X,Y,M,[0 0])` where X and Y denote the grid coordinates;

6. Select a refined subset around the found bound, and a refined grid resolution. Repeat point 3 onwards, until the Horowitz-Sidi bound is computed with desired accuracy.

5.1 Example

Consider the uncertain plant $P(s) = k/s$, $k \in [1, 2]$, with the nominal $P_{\text{nom}}(s) = 1/s$, Consider the bound computation for 1 rad/s, i.e. $s = j$, for the tolerance specification (15,16,17) with $z(1) = 1.3497$ dB. The initial search set in the complex plane is defined by gain $\in [-50, 50]$ dB, and phase $\in [-360, 0]$, with the grid resolution 10 degrees, and 5 dB in the Nichols chart. The resulting bound, according to the algorithm above as implemented in `Qsyn` is marked by "A" in Figure 7. Smaller search sets are now found around the computed course bound, marked as "subset boxes" in Figure 7. Within these, the bound is computed with a resolution of 3 degrees and 1 dB. The resulting refined bound points are denoted by rings and the label "B".

Note also in Figure 7 the ringed straight-line template with the label $-P_{\text{nom}}(j)\{P_i(j)^{-1}\}$ which denotes the template of plant inverses multiplied with $-P_{\text{nom}}(j)$. Its nominal is obviously the point -1 (-180 degrees, 0 dB). The rings denote the computed template points. The significance of this tem-

plate is as follows: it is clearly seen in the tolerance specification (15,16,17) that $G = -P_i^{-1}$, $\forall i$, would not satisfy the specification. Hence, the points $-P_i^{-1}P_{\rm nom}$, $\forall i$, will belong to the infeasible part of the complex nominal open loop plane ($L_{\rm nom}$-plane), i.e. on the "forbidden" side of the bound. The template $-P_{\rm nom}(j)\{P_i(j)\}^{-1}$ thus indicates which side of the bound is infeasible.

The final Horowitz-Sidi bound is also shown in Figure 8. Note that Horowitz's original bound computation algorithm (Algorithm 1) would not have rendered correctly the "cusp"-like shape of the bound at phases [-150, -137] degrees, and hence might have resulted in a too conservative feedback regulator design.

5.2 Example

In [28] an efficient bound computation algorithm is presented, see also references quoted therein. For the 4th order aircraft example in section 5 in [28] a tracking bound for 0.1 rad/s is computed. Unfortunately, Table 2 in that paper is lacking, where the computational statistics were to be found. Our bound computation command `cbnd` in `Qsyn` needed 2.484 seconds on a 1.86 Ghz Pentium M computer to compute the same bound, based on a template consisting of 461 boundary points computed with the Recursive Grid I Algorithm above with a resolution of (1 degree, 1 dB). The resulting bound had also the resolution (1 degree, 1 dB). Note that the command `cbnd` also reads the template and specification files, presents graphical data as in Figure 7,

and writes the bound data into a file, in addition to computing the bound itself.

6 Conclusions

The main advantage of the proposed Recursive Grid methods is the fact that it is possible to determine the required resolutions *a priori*. We have tested the Recursive Grid method for template computation on numerous examples and found the computational savings relative to the "brute force" equidistant grid method to be substantial, typically between 50 and 95 %. The recursive Horowitz-Sidi bound computation method gives a correct bound within the desired resolution, for all types of templates and specifications.

Acknowledgement

Particularly warm thanks are due to Isaac Horowitz's former collaborator Dr. Linda Neumann of El-Op Electro-Optics Ind. Ltd., Rehovot, Israel who wholeheartedly supported our effort to realize the QFT design software `Qsyn - the Toolbox for Robust Control Systems Design for use with Matlab`, and who collaborated with us in many scientific, computational and application projects. The anonymous reviewers are acknowledged for their very useful inputs that made it possible to improve the paper considerably.

References

[1] J. Ackermann. *Robust Control: Systems with Uncertain Physical Parameters*. Springer Verlag, 1975.

[2] K. J. Åström. Computer aided modeling, analysis and design of control systems—A survey. *IEEE Control Systems Magazine*, 3(2):4–16, 1983.

[3] F. N. Bailey and C.-H. Hui. A fast algorithm for computing parametric rational functions. *IEEE Transactions on Automatic Control*, 34(11):1209–1212, 1989.

[4] F. N. Bailey, D. Panzer, and G. Gu. Two algorithms for frequency domain design of robust control systems. *International Journal of Control*, 48(48):17871806, 1988.

[5] D. Ballance and G. Hughes. Survey of template generation methods for quantitative feedback theory. In *Proceedings of the 1996 UKACC International Conference on Control. Part 1*, volume 427/1 of *IEE Conference Publication*, pages 172–174. IEE, IEE, Stevenage, England, 2-5 September 1996.

[6] C. Baril and P.-O. Gutman. Performance enhancing adaptive friction compensation for uncertain systems. *IEEE Trans. Control Systems Technology*, 5(5):466–479, 1997.

[7] E. Boje. Finding nonconvex hulls of QFT templates. *Transactions of the ASME. Journal of Dynamic Systems, Measurement and Control*, 122(1):230–232, March 2000.

[8] M. Brown and I. R. Petersen. Exact computation of the horowitz bound for interval plants. In *Proceedings of the 30th IEEE Conference on Decision and Control*, volume 3, pages 2268–2273, Brighton England, 11-13 December 1991. IEEE.

[9] J. Cervera, A. Baños, and I. M. Horowitz. Computation of siso general plant templates. In M. G. Sanz, editor, *Proc. 5th International Symposium on QFT and Robust Frequency Domain Methods*, pages 247–254, Universidad Pública de Navarra, Pamplona, Spain, 2001.

[10] W. Chen and D. J. Ballance. Plant template generation of uncertain plants in quantitative feedback theory. *ASME Journal of Dynamic Systems, Measurement and Control*, 121(2):358–364, 1999.

[11] B. Cohen. A program for computer aided design of robust control systems. Master's thesis, Faculty of Agricultural Engineering, Technion, Haifa, Israel, 1994.

[12] B. Cohen, M. Nordin, and P.-O. Gutman. Recursive grid methods to compute value sets for transfer functions with parametric uncertainty. In *Proc. 1995 American Control Conference*, volume 5, pages 3861–3865. IEEE, 21-23 June 1995.

[13] S. M. Fadali and L. E. LaForge. Algorithmic analysis of geometrically computed qft bounds. In *Proceedings of the 13th IFAC World Congress*, volume H, pages 297–302, San Fransisco, California, 1997. International Federation of Automatic Control.

[14] M. Fu. Computing the frequency response of linear systems with parametric perturbations. *Systems and Control Letters*, 15:45–52, 1990.

[15] P.-O. Gutman. *Qsyn - the Toolbox for Robust Control Systems Design for use with Matlab, User's Guide and Reference Guide.* Copyright: Per-Olof Gutman, 1996. manuals dowloadable from http://www.math.kth.se/optsyst/research/5B5782/index.html.

[16] P.-O. Gutman, C. Baril, and L. Neumann. An algorithm for computing value sets of uncertain transfer functions in factored real form. *IEEE Transactions on Automatic Control*, 39(6):1268–1273, June 1994.

[17] P.-O. Gutman, E. Horesh, R. Guetta, and M. Borshchevsky. Control of the aero-electric power station - an exciting qft application for the 21st century. *Int. J. of Robust and Nonlinear Control*, 13:619–636, June 2003.

[18] P.-O. Gutman, H. Levin, L. Neumann, T. Sprecher, and E. Venezia. Robust and adaptive control of a beam deflector. *IEEE Trans. Aut. Control*, 33(7):610–619, 1988.

[19] P.-O. Gutman and L. Neumann. Horpac — a program for robust control systems design. In *Proc. IEEE Control Systems Soc. 2nd Symposium on Computer Aided Control Systems Design*, Santa Barbara, California, 1985. IEEE.

[20] P.-O. Gutman, L. Neumann, and K. J. Åström. Incorporation of unstructured uncertainty into the horowitz robust design method. In *Proc. IEEE International Conference on Control and Applications, ICCON '89*, Jerusalem, Israel, 3-6 April 1989. IEEE.

[21] I. M. Horowitz. *Quantative Feedback Design Theory (QFT)*. QFT Publications, Boulder, Co., U.S.A., 1992.

[22] I. M. Horowitz and M. Sidi. Synthesis of feedback systems with large plant uncertainty for prescribed time domain tolerances. *International Journal of Control*, pages 287–309, 1972.

[23] I. M. Horowitz, O. Yaniv, and L.Neumann. Fligth control design with uncertain parameters. Technical Report AFWAL-TR-83-3036, Wrigth-Patterson Air Force Base, Ohio 45433, U.S.A., Sep. 1983.

[24] D. Kahaner, C. Moler, and S. Nash. *Numerical methods and software*. Prentice Hall, Englewood Cliffs, 1989.

[25] *MATLAB 6.5, High Performance Numeric Computation and Visualization Software*. The MathWorks inc., Cochituate Place, 24 Prime Park Way, Natick, Mass.

[26] J. C. Moreno and A. Baños. Improvements on the computation of boundaries in qft. *International Journal of Robust and Nonlinear Control*, 2006. to be published.

[27] J. C. Moreno, A. Baños, and F. J. Montoya. An algorithm for computing qft performance bounds. In L. Petropoulakis and W. E. Leithead, editors, *Proceedings of the Symposium on Quantitative Feedback Theory and other Frequency Domain Methods and Applications*, Brighton, England, 21-22 August 1997.

[28] P. Nataraj. Computation of qft bounds for robust tracking specifications. *Automatica*, 38(2):327–334, February 2002.

[29] P. Nataraj and S. Sheela. Template generation algorithm using vectorized function evaluations and adaptive subdivisions. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 124(4):585–588, December 2002.

[30] M. Nordin and P.-O. Gutman. Digital qft design for the flexible transmission benchmark problem. *European Journal of Control*, 1(2):97–103, 1995.

[31] M. Nordin and P.-O. Gutman. Controlling mechanical systems with backlash — a survey. *Automatica*, 38(10):1633—1649, October 2002.

[32] S. Oldak, C. Baril, and P.-O. Gutman. Quantative design of a class of nonlinear systems with parameter uncertainty. *Int J. of Robust and Nonlinear Control*, 4:101–117, 1994.

[33] S. Oldak and P.-O. Gutman. Self oscillating adaptive design of systems with dry friction and significant parameter uncertainty. *Int. J. Adap. Contr. & Signal Proc.*, 9:239–253, 1995.

[34] A. Rantzer and P. O. Gutman. An algorithm for addition and multiplication of value sets of uncertain transfer functions. In *Proc. of 30th CDC*, pages 2111–2115, Brighton, England, 1991. IEEE.

[35] J. Rodrigues, Y. Chait, and C. V. Hollot. A new algorithm for computing qft bounds. In *Proceedings of the 1995 American Control Conference*, volume 6, pages 3970 – 3974, Seattle, Washington, 21-23 June 1991. IEEE.

[36] O. Yaniv. *Quantitative Feedback Design of Linear and Nonlinear Control Systems*, volume 509 of *The International Series in Engineering and Computer Science*. Springer Verlag, 1999.

[37] O. Yaniv, Y. Chait, and C. Borgehesani. The Quantitative Feedback Toolbox for Matlab. In *Proc. of ROCOND 97*, Budapest, Hungary, 25-27 June 1991. IFAC.

[38] O. Yaniv, P.-O. Gutman, and L. Neumann. An algorithm for the adaptation of a robust controller to reduced plant uncertainty. *Automatica*, 26(4):709–720, 1990.

[39] O. Yaniv and I. M. Horowitz. A quantitative design method for mimo linear feedback-systems having uncertain plants. *Int. J. Control*, 43(2):401–421, 1986.

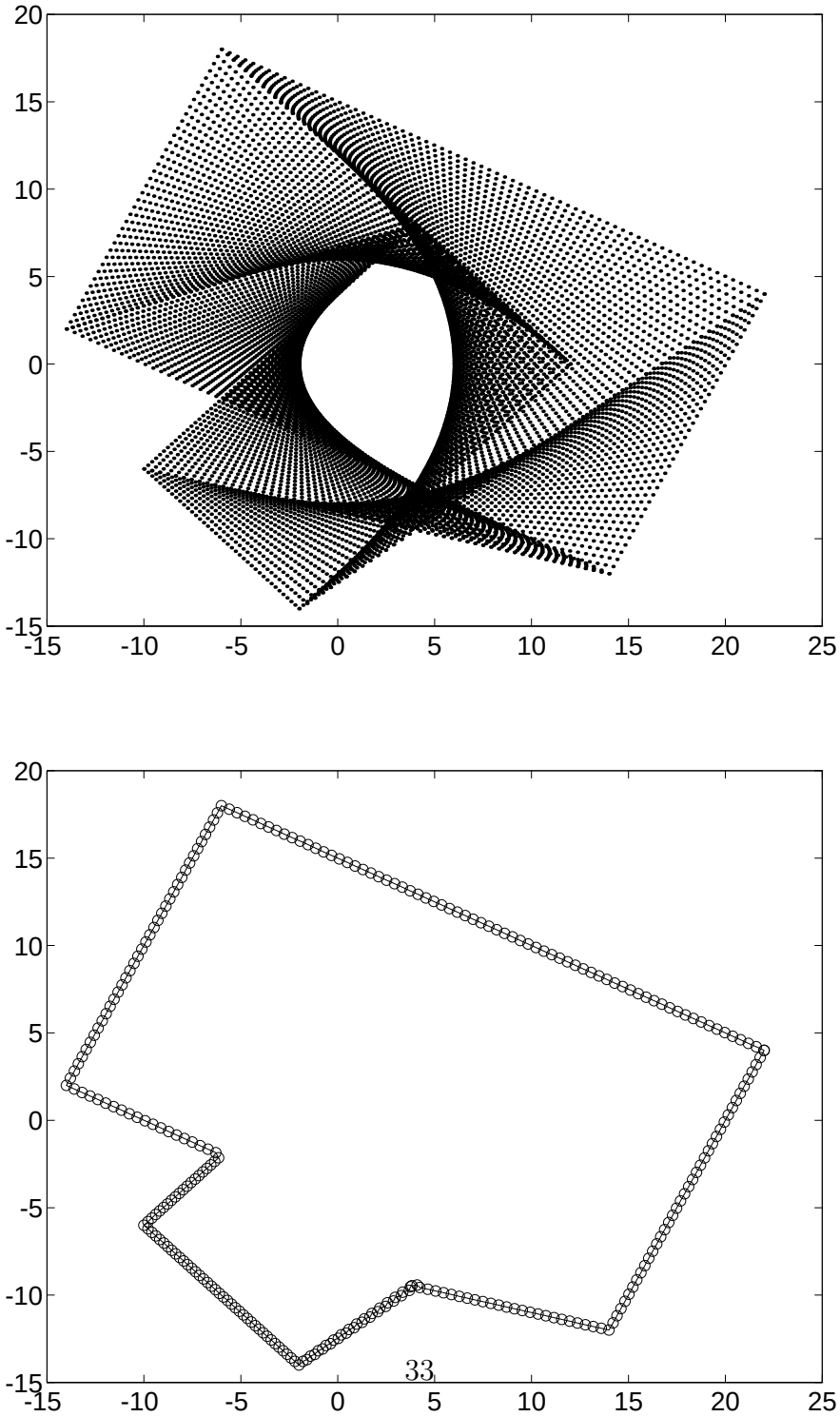

Figure 4: The value set V (upper plot) with 13200 points and the pruned set $B^*(V)$ (lower plot) with 286 points, including the ϵ-circles of $B^*(V)$

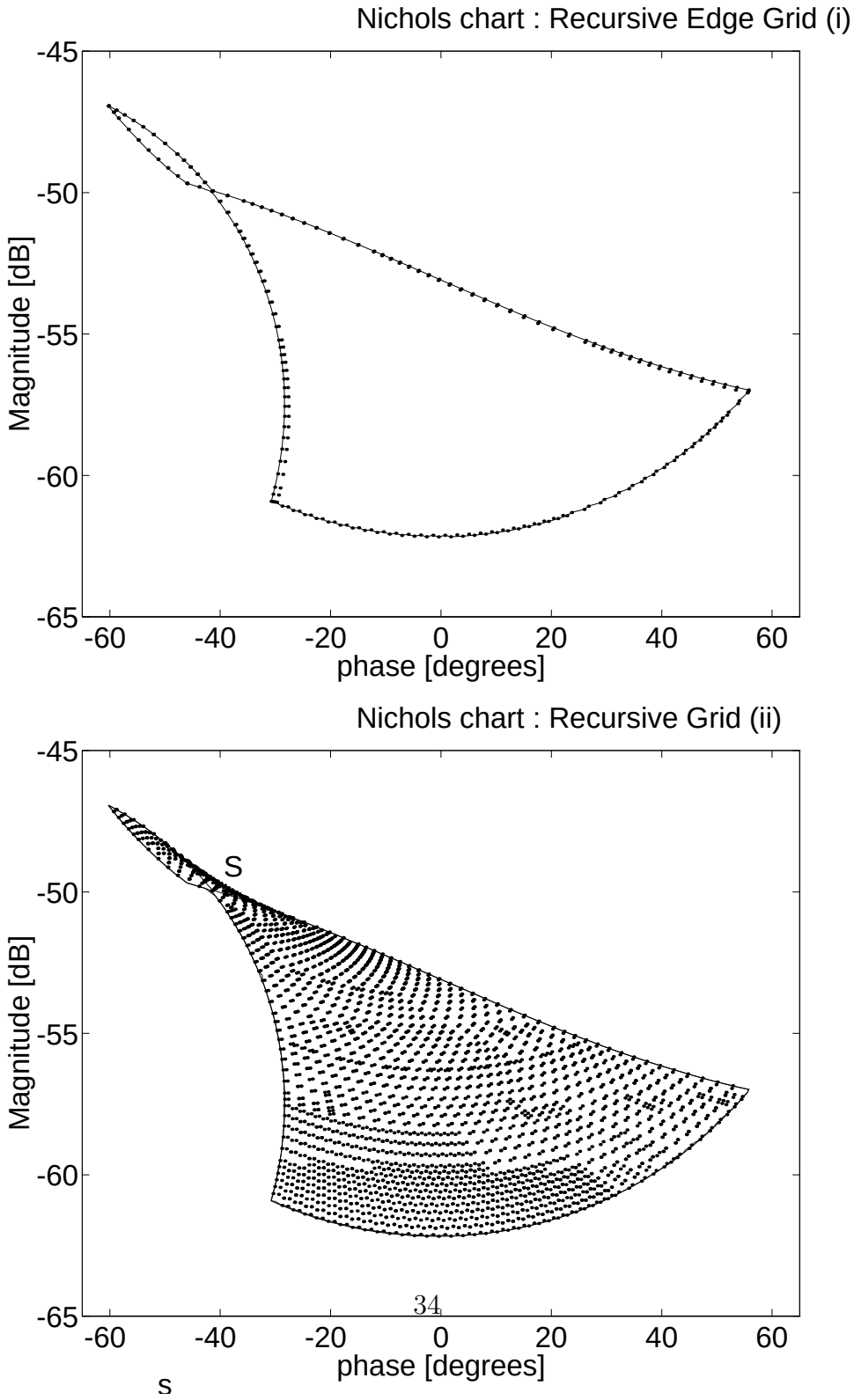

Figure 5: The value sets of $P(q, j10\pi)$, see the example

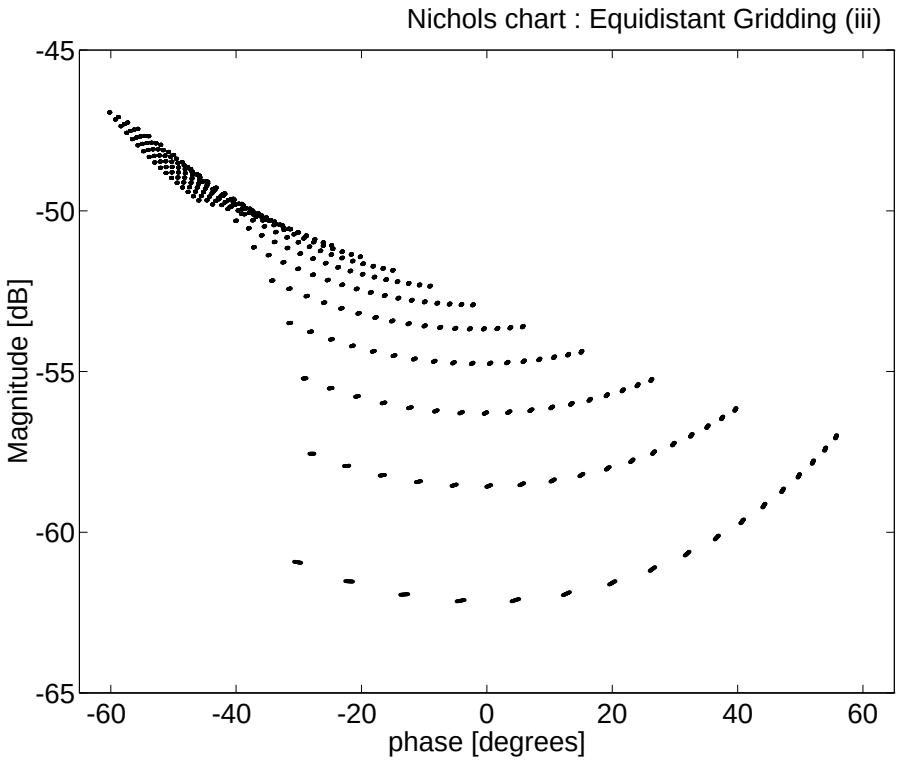

Figure 6: The value sets of $P(q, j10\pi)$, see the example

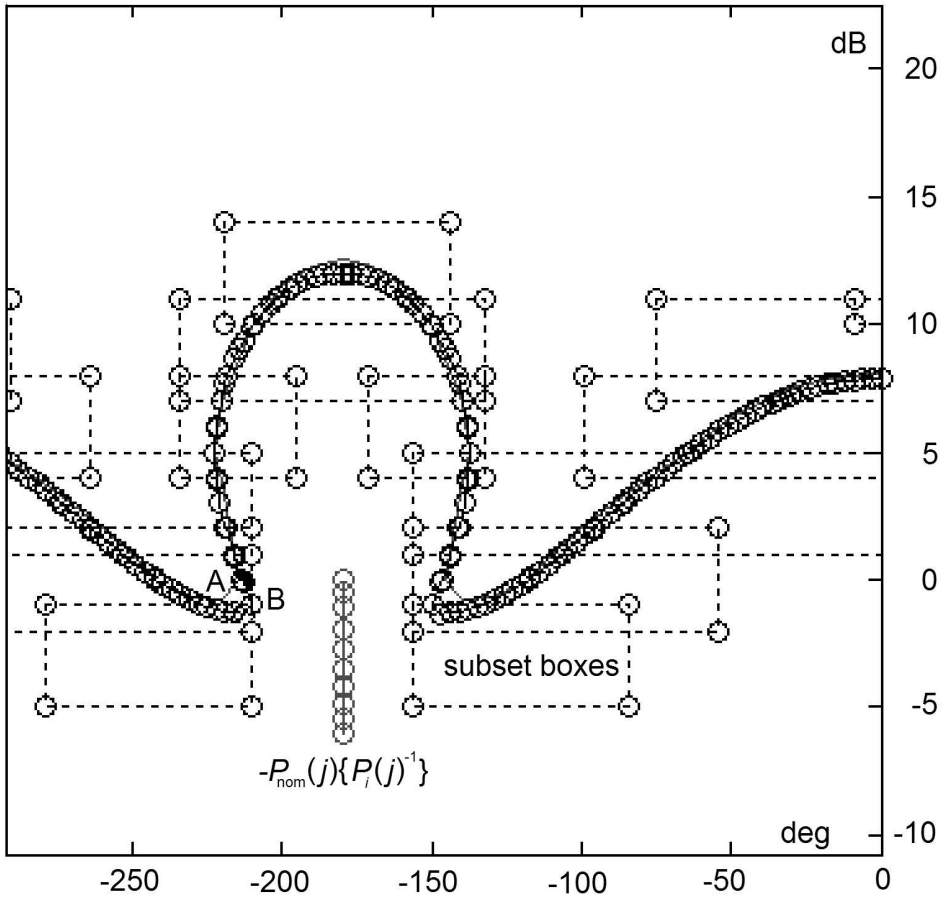

Figure 7: A snapshot from the monitor screen of a bound computation in `Qsyn`, displayed in the Nichols chart of the complex $L_{\rm nom}$-plane. "A" denotes the bound marked by — only, found with the coarse grid resolution (10 degrees, 5 dB). "B" denotes the refined bound, marked by — together with bold face rings **o**, found when the fine grid resolution (3 degrees, 1 dB) is used. For further explanations, see Example 5.1.

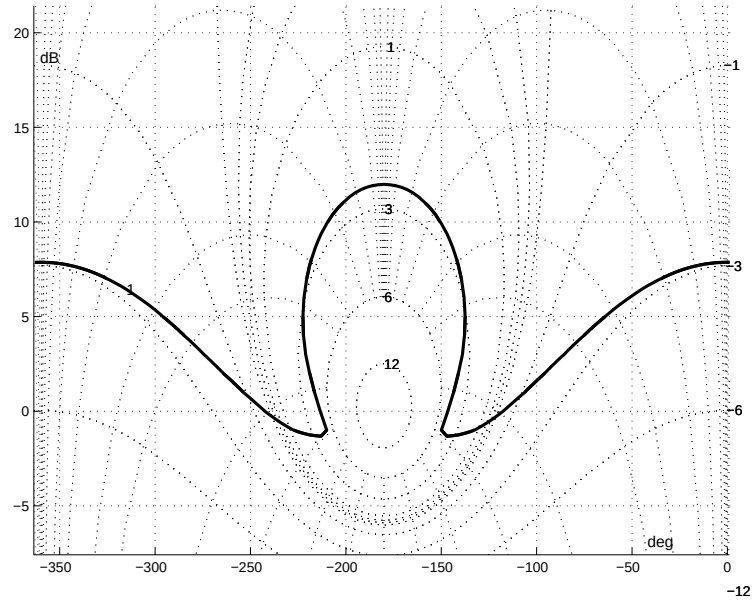

Figure 8: The final Horowitz-Sidi bound in Example 5.1 as computed in `Qsyn` with resolution (3 degrees, 1 dB), displayed in the Nichols chart of the complex L_{nom}-plane.