The QFT Frequency Domain Control Design Toolbox

For Use with MATLAB®

Craig Borghesani
Yossi Chait
Oded Yaniv

User's Guide

Quantitative Feedback Theory Toolbox User's Guide

Printing History

December 1994	First printing (The MathWorks, Inc.)
March 2001	Second printing (Terasoft, Inc.)
July 2003	Third printing (Terasoft, Inc.)

How to Contact Us

www.terasoft.com	Home page
support@terasoft.com	Technical support and product enhancement suggestions
sales@terasoft.com	Sales, pricing, and general information

Table of Contents

4 Using the Nichols Chart

5 Examples

6 Bounds and Loop Shaping

7 Reference

Preface

What's New

This major revision is the first of its kinds that is not backward compatible with any V1.x versions dating back to the original V1.0 release by The MathWorks, Inc., in 1993.

If you are a new user, then skip this part and proceed directly to the Introduction. However, we strongly recommend that previous users read this section first.

To begin, the use of LTI and FRD models and arrays from the Control Toolbox that began in V2.0 is now a Toolbox standard. All functions accept only these models as input arguments and compute output arguments in the same format. The exceptions to this rule are scalar arguments and matrix bounds. We list below all functions that have been removed from V2.5 and are no longer supported.

Owing to the ease by which algebraic manipulations available with LTI/FRD models and arrays, the following functions are no longer needed and have been removed from v2.5:

Conversions	
`cp2mp`	Complex matrix to magnitude/phase real matrices
`mp2cp`	Magnitude/phase real matrices to complex matrix

and

General Utility	
`freqcp`	Compute continuous-time frequency response sets
`dfreqcp`	Compute discrete-time frequency response sets
`qftdefs`	User-defined defaults

The above operations can be easily carried in the command line. For example, the Control Toolbox's `freqresp` now computes the frequency response of a plant set (i.e., LTI or FRD array).

However, certain algebraic manipulations with arrays having different numbers of elements are not supported by the Control Toolbox and will not produce correct results. Hence, the following functions have been removed

Arithmetic	
`addcp`	Addition of frequency response sets
`addnd`	Addition of transfer function num/den sets
`mulcp`	Multiplication of frequency response sets
`mulnd`	Multiplication of transfer function num/den sets
`clcp`	Compute closed-loop frequency response set
`clnd`	Compute closed-loop transfer function num/den set

and in their place we now have

Arithmetic	
`addtmpl`	Add LTI/FRD arrays
`cltmpl`	Closed-loop LTI/FRD arrays from open-loop arrays
`multmpl`	Multiply LTI/FRD arrays

Since LTI/FRD model objects now include sampling time for discrete-time systems, the following functions are no longer needed and have been removed

Interactive Design Environments	
`dlpshape`	Discrete-time controller design
`dpfshape`	Discrete-time pre-filter design

`lpshape` and `pfshape` now work in continuous-time and discrete-time settings.

The duplication of example files have been simplified. The following set of demo files has been removed

Examples	
`qftdemo`	Special demo facility for the examples in Chapter 5

Of course, the main set of M-files comprising of all examples is still available from `qftex*.m` and has been updated to comply with V2.5.

In addition, in many functions the number of input arguments is now smaller. For example, previous calls to `lpshape` had this form

```
lpshape(w,bdb,numP0,denP0,delay0,numC0,denC0,phs)
```

while in V2.5 the form is much simpler

```
lpshape(w,bdb,P0,C0,phs)
```

We recommend that you familiarize yourself with the new call formats and make use of our extensive set of updated example files to observe correct use of LTI/FRD arrays in this Toolbox.

Organization

This manual was written such that any user, from the practicing engineer to the researcher in academia, can quickly learn the basic concepts behind QFT. The only requirements are a working knowledge of classical frequency-domain concepts commonly taught at a junior/senior undergraduate course. Familiarity with discrete-time systems is required for discrete-time QFT design.

Chapter 2, The Feedback Problem, begins with two real-world examples, a compact disc mechanism and active vibration isolation in an engine to illustrate the need for feedback in general and the flexibility of QFT for a wide range of problems. It then describes how to formulate a QFT design problem for such systems: choose a feedback structure, model the process dynamics (with or without uncertainty) and finally define appropriate frequency-domain specifications.

Chapter 3, Feedback Design with QFT, leaps right into the QFT design procedure and leaves some of the theoretical details to Chapter 4. It begins with an introduction of the various steps in a typical design: generation of plant templates, computation of bounds, loop shaping and analysis. A detailed design is then developed for a generic robust performance problem to illustrate QFT design in general and use of the *Toolbox* functions in particular. The presentation in this chapter focuses first on continuous-time systems and then repeats the presentation for discrete-time systems.

Chapter 4, Using the Nichols Chart, provides theoretical background on stability analysis of feedback systems using Nichols charts (Nichols charts are the domain of choice for QFT designs). You will first

learn how the usual Nyquist plot in the complex plane is mapped into a similar plot in a Nichols chart. The stability criterion used with Nichols charts is then introduced as related to its counterpart, the Nyquist stability criterion in the complex plane. This criterion is illustrated with several examples. The notions of model uncertainty and plant templates are defined and are followed by the extension to a robust stability criterion. The section ends with a similar presentation of stability, uncertainty and robust stability concepts for discrete-time systems.

Chapter 5, Examples, includes fourteen examples illustrating a wide range of QFT designs. The examples cover continuous-time and discrete-time systems, plants with different types of uncertainties, single-loop, cascaded-loop and multi-loop systems, and some industrial problems.

Chapter 6, Bounds and Loop Shaping, consists of a detailed description of the bound computation functions and the special functions that create CAD environments for controller (loop-shaping) and pre-filter design.

Chapter 7, Reference, is the reference chapter that describes the *Toolbox* functions. Details of all the *Toolbox* functions follow in alphabetical order.

Two appendices are included: A Glossary and B Bibliography.

About the Authors

Craig Borghesani

Craig Borghesani received his B.S. degree in Interdisciplinary Engineering (Robotics) from Purdue University in 1990. He received his M.S. degree in Mechanical Engineering from the University of Massachusetts, Amherst, in 1993, where he developed the code for this Toolbox. Since then, he has started his own engineering software development company, Terasoft, Inc., which specializes in developing custom data visualization applications in MATLAB and Java.

Yossi Chait

Yossi Chait received his B.S. degree in Mechanical Engineering from Ohio State University in 1982, and his M.S. degree and Ph.D. degree in Mechanical Engineering from Michigan State University in 1984 and 1988, respectively. Currently he is an associate professor at the Mechanical Engineering Department, University of Massachusetts, Amherst.

Dr. Chait has numerous publications in the area of robust control design. He has been active in Quantitative Feedback Theory teaching and research for the past fifteen years. In recognition of this work, he was an Air Force Institute of Technology Distinguished Lecturer and was a Dutch Network Visiting Scholar at the Laboratory for Measurement and Control, Delft University, and Philips Research Laboratories, a visiting appointment at Tel Aviv University, an Academic Guest, Measurement and Control Laboratory, Swiss Federal Institute of Technology, ETH, Zürich, Switzerland and a Lady Davis Fellow, Department of Mechanical Engineering, the Technion, Haifa, Israel. Dr. Chait has consulted for industry in a broad range of applications, for example in automatic welding, real time particle analyzers and vibrations reduction. His recent research focuses on congestion control of the Internet and modeling of feedback in biological systems such as the Hypothalamus-Pituitary-Thyroid axis and Circadian rhythms. For more details visit http://www.ecs.umass.edu/mie/labs/dacs/. Dr. Chait resides in Western Massachusetts with his wife and two daughters.

Oded Yaniv

Oded Yaniv received his B.Sc. degree in Mathematics and Physics from Hebrew University, Israel, in 1974, and his M.Sc. degree in Physics and Ph.D. degree in Applied Mathematics from the Weizmann Institute of Science, Israel in 1976 and 1984, respectively. Currently he is on the faculty of the Electrical Engineering/Systems Department at Tel Aviv University, Israel.

Dr. Yaniv's research interests are in the areas of robust, linear and non-linear systems. He has been active in the Quantitative Feedback Theory area since starting his Ph.D. research under the supervision of Prof. Horowitz. Dr. Yaniv was a senior control engineer at the Systems Division, Tadiran, Israel, in 1983-1987. Since then he has been a consultant to major companies in the Israeli industry. Dr. Yaniv was invited several times to the US Air Force WPAF Labs under the Window of Science program, and presented numerous short courses on QFT around the world, for example, at NASA Goddard.

Acknowledgements

The authors would like to acknowledge Professor Isaac Horowitz, the original developer of the Quantitative Feedback Theory method.

The contributions of the following people are acknowledged: Ronen Bone, John Kresse, C.H. Houpis, Rob Osborne, Myoung Soo Park, Julie Rodrigues, Eyal Sapir, Richard Sating, Stuart Sheldon, Maarten Steinbuch, Yahali Theodor, Pepijn Wortelboer and Yuan Zheng.

We would like to thank the following people from The MathWorks: Liz Callanan, Roy Lurie, Andy Potvin and Jim Tung who assisted us at various stages of the project.

1 Introduction

About Quantitative Feedback Theory

In the 1960's, as a continuation of the pioneering work of Bode, Isaac Horowitz introduced a frequency-domain design methodology [1] that was refined in the 1970's to its present form, commonly referred to as the Quantitative Feedback Theory (QFT) [2,3]. The QFT is an engineering method devoted to practical design of feedback systems.[1]

Control design necessary to accomplish performance specifications in the presence of uncertainties (plant changes and/or external disturbances) is a key consideration in any real feedback design. In QFT, one of the main objectives is to design a simple, low-order controller with minimum bandwidth. Minimum bandwidth controllers are a natural requirement in practice in order to avoid problems with noise amplification, resonances and unmodeled high frequency dynamics. In most practical design situations iterations are inevitable, and QFT offers direct insight into the available trade-off between controller complexity and specifications during such iterations. QFT can be considered as a natural extension of classical frequency-domain design approaches.

The *foundation* of QFT is the fact that feedback is principally needed when the plant is uncertain and/or there are uncertain inputs (disturbances) acting on the plant. The *motivation* for QFT is feedback design in practice – an evolving process in which the designer must trade-off between complexity and specifications. The specific *characteristics* of QFT are:

- The amount of feedback is tuned to the amount of plant and disturbance uncertainty and to the performance specifications.
- Design trade-offs at each frequency are highly transparent between stability, performance, plant uncertainty, disturbance level and controller complexity and bandwidth.
- The method extends highly intuitive classical frequency-domain loop shaping concepts to cope with simultaneous specifications and plants with uncertainties.

The QFT philosophy for feedback design fits a wide range of applications:

- *Plant Uncertainty*. The controller should meet specifications in spite of variations in the parameters of the plant model. For example, the first mode's natural frequency in a compact disc drive may vary by ±5% from its nominal value due to manufacturing tolerances and a wide range of operating temperatures (if used in a car). QFT works directly with such uncertainties and does not require any particular representation.
- *Plant Models from Experiments*. Many systems have complex dynamics and are very difficult to model analytically. For example, the dynamics of the radial loop in a compact disc or a disk drive mechanism contain a large number of mechanical resonances – even a detailed finite-element analysis cannot generate a reasonable model for control design with tight specifications. A common approach is to run physical experiments and compute directly the frequency response of the plant. Frequency response uncertainty sets are then created and QFT works with such sets without requiring rational plant identification.

[1] Recent books on QFT are [30]-[32].

- *Linear Plants from Nonlinear Dynamics.* Unlike the conventional small signal linearization about an operating point, Horowitz's idea is to replace the nonlinear plant with a set of linear, time-invariant (LTI) plants using assumed input and output responses. The design (via standard QFT procedure) relies on Schauder fixed point theorem and Homotopic invariance to show stability of the nonlinear system (the mathematics is rather deep compared with that used in LTI systems). However, from the control engineer's viewpoint, the actual design procedure is as straightforward as in the LTI case.

- *Several Performance Specifications*. The design problem consists of several closed-loop performance specifications and the objective is to synthesize a controller to meet simultaneously all specifications (a robust performance problem). QFT reveals via QFT bounds the "toughness" of each specification relative to others. Moreover, real life specifications are often incomplete, e.g., in a noise control system noise reduction should be at least 24 *dB* in the range [100,500] *Hz*. QFT works with such incomplete form and does not require specifications to be defined at each frequency from zero to infinity.

- *Hardware Constraints*. In real life controllers are constrained by hardware. For example, the DSP board limits the locations of the controller's poles to be less than 100 *Hz*, limits the number of digits that can be used to represent the controller's coefficients or limits the controller's bandwidth. With QFT you quickly test if a particular controller (e.g., proportional) can solve the problem.

The above provided a brief background on QFT and presented some possible scenarios where this Toolbox can be useful in your feedback design. No prior familiarity with QFT is required with the exception of classical frequency-domain concepts. This manual is intended to provide you with the basic understanding of QFT as necessary to use the Toolbox effectively. The more you design with the Toolbox, the more you will learn about QFT. We recommend that you read the entire manual before beginning work with the Toolbox. For a detailed teaching reference of QFT please refer to [3].

Finally, a few words about the suitability of QFT to different classes of problems. The QFT method, originally developed for uncertain LTI systems single-loop systems, has been extended to cascaded loop systems and multi-loop systems using a sequential loop closure approach. *This Toolbox focuses on a class of feedback problems that involve uncertain, single-loop design (single-loop and decentralized systems)*. The *Toolbox* can also be used to solve multi-loop problems such as cascaded-loop and sequentially closed multi-loop systems but familiarity with the design algorithms is required (see Examples Example 7: Inner-Outer Cascaded Design, Example 8: Outer- Inner Cascaded Design and Example 15: Multi-Loop Design in Examples). QFT has also been successfully applied to time-varying and nonlinear systems (e.g., see [3]).

2 The Feedback Problem

Motivating Examples

The two examples in this chapter serve two purposes: to establish the need, in general, for feedback and to illustrate the suitability of QFT for a wide range of real-world problems. These examples belong to a particular class of problems from two major classes found in applications:

- The first class consists of "well defined" problems where the plant model (including uncertainty) is known with great accuracy and the performance specifications are defined from zero to infinite frequency.
- The second class includes plants for which available models are not sufficiently accurate for control design or may not even exist. And the specifications may be defined only over a finite bandwidth, e.g., in an engine vibration control we may require disturbance rejection (acceleration transmissibility) of -20 *dB* in the working frequency band on [100,200] *Hz*. In the second class, as in the two examples below, the best models for control design are obtained from measurements.

To execute a QFT design you are not required to identify a plant model from the data nor should you define specifications in any specific format over the whole frequency range from zero to infinity. QFT is equally suited to solve problems from either class; however, the ability to attack problems from the second class in a direct manner is precisely what renders QFT so powerful in applications. Given the fact that control design in applications is an iterative process, a control design/implementation/redesign iteration cycle can be performed efficiently using QFT, since QFT does not require a "well defined" problem after each implementation.

Compact Disc

A compact disc (CD) player (Fig. 1) is an optical decoding device that reproduces high-quality data from a digitally coded signal recorded as a spiral shaped track on a reflective disc.

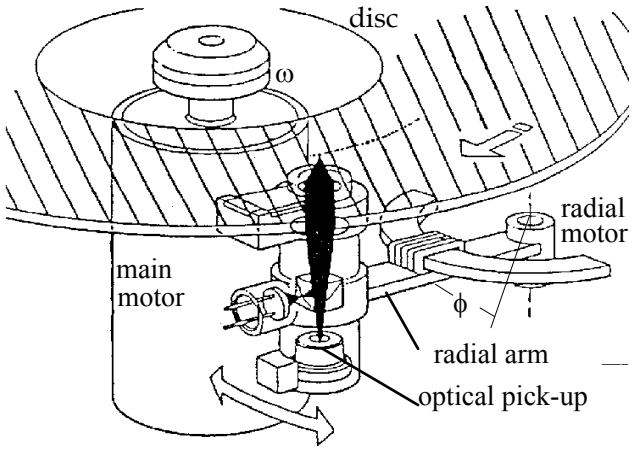


Figure 1: A schematic view of a Compact Disc mechanism

The difficulty in achieving good track following is due to disturbances and plant uncertainty. Disturbances are caused, for example, by external shocks when the CD is used in a car going over a bump

or in a portable CD used by a runner. Plant uncertainty is always a factor in mass production due to manufacturing tolerances. Feedback is clearly required in order to achieve good track following.

Figure 2 presents a block diagram of the radial control loop. The difference between the track position and the laser beam spot position on the disc is detected by the optical system; it generates a radial error e_R signal via a gain G_{opt}. A controller K feeds the radial motor with the current I_{rad}. This in turn generates a torque resulting in an angular acceleration. The transfer function from the current I_{rad} to the angular displacement ϕ of the arm is called $G_{act}(s)$ A (nonlinear) gain G_{arm} relates the angular displacement with the spot movement in the radial direction. Only the control-error signal e_R is available for measurement. Assuming constant radial velocity ω, the goal is to control the position of the spot on the disc.

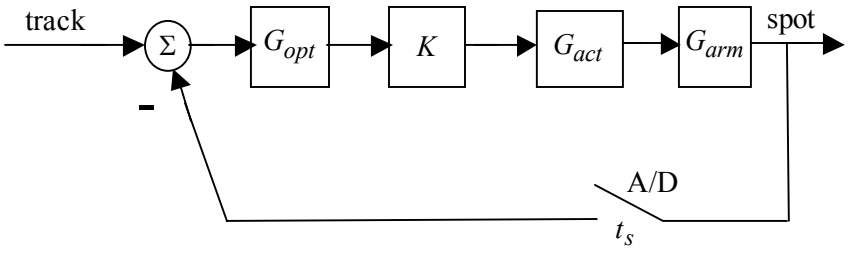


Figure 2: Block diagram of the radial loop

Now that the feedback structure is defined, the next step involves modeling of the radial loop dynamics. With the Toolbox you can define the model either as a rational transfer function or in terms of its frequency response (possibly from measurements). The CD dynamics, difficult to model analytically, are characterized by mechanical vibrations that fall within the controlled bandwidth. The nominal dynamics (Fig. 3) were found by averaging over several hundreds frequency response tests. At low frequencies the actuator transfer function from current input I_{rad} to position error output e_R is a critically stable system with a phase lag of 180° (rigid body mode). The erratic low frequency response is indicated by low coherence. At higher frequencies the measurement shows parasitic dynamics due to mechanical resonances of the radial arm and mounting plate (flexible bending and torsional modes).

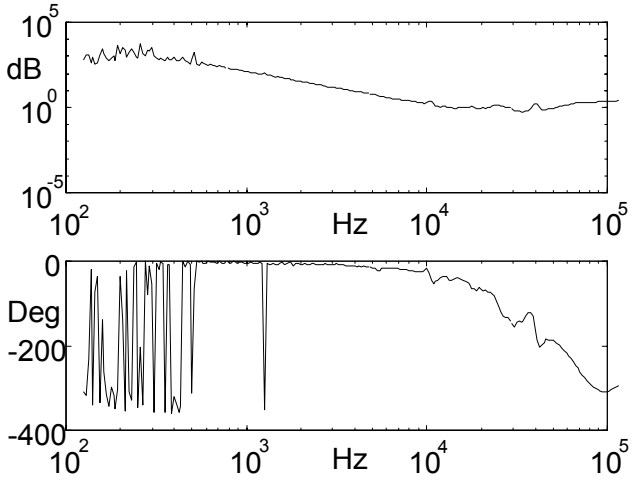


Figure 3: Measured nominal radial open-loop frequency response

Due to manufacturing variations, we are required to define uncertainty model. Important uncertain parameters in the dynamics are three undamped natural frequencies with nominal values of 0.8, 1.62 and 4.3 kHz. To quantify possible variations, we allow each natural frequency to vary independently by ±2.5% around its nominal value. The plant frequency response set can be computed from the measured

data (nominal case) and from the above parametric variations (see Example 14: CD Mechanism (Sampled-data)).

Now that the plant dynamics are defined, you can consider the feedback design objectives. The radial loop design must take into account several conflicting factors:

- Accommodation of mechanical shocks acting on the player,
- Achievement of the required disturbance attenuation at the rotational frequency of the disc, necessary to cope with significant disc eccentricity,
- Playability of discs containing faults,
- Audible noise generated by the actuator, and
- Power consumption.

In general, design objectives will be a combination of time-domain and frequency-domain criteria. The QFT, being a frequency-domain method, requires frequency-domain specifications. In many cases, it is possible to translate "soft" time-domain criteria, such as overshoot and settling time, into appropriate frequency-domain specifications. Although, satisfaction of the frequency-domain specifications cannot guarantee the original time-domain criteria, this approach was found to work in many design examples. An excellent description of the possible translation approaches is given in [3].

The above listed criteria can be formulated in the frequency domain. When using QFT, you need not define the specifications in any specific format such as rational functions or weighting matrices. The specifications are: (a) robust stability, (b) gain and phase with margins

$$\left| \frac{G_{arm} G_{act} K G_{opt}}{1 + G_{arm} G_{act} K G_{opt}} (j\omega) \right| \le 3, \text{ for all uncertainty, } \omega \ge 0$$

and (c) robust sensitivity such that the closed-loop sensitivity function meets the magnitude specification shown in Fig. 4.

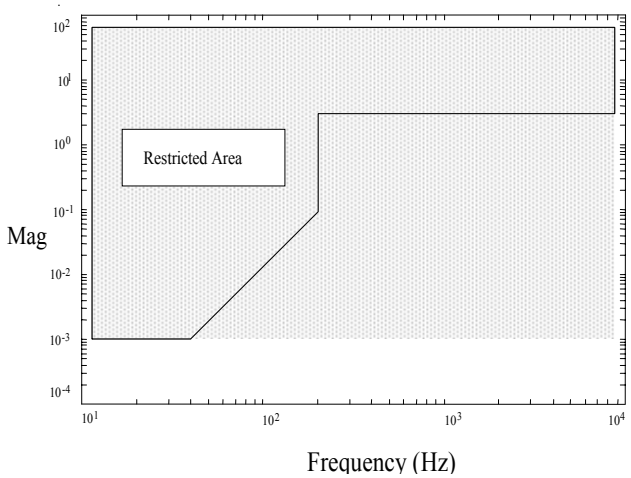


Figure 4: Robust sensitivity reduction specification

Finally, the feedback problem is to design the controller, *K*, such that the above specifications are met.

In classical frequency-domain designs, stability margins were related to the gain and phase distances between the open-loop plot and the critical point (-1,0). An alternative, yet equivalent way to specify such margins is via maximal amplitudes of certain closed-loop relations (see discussion in Robust

Stability (Margins) Bounds). In this problem, to ensure reasonable stability margins, there should be no large peaking in the sensitivity function (track to error) and the complementary sensitivity function (track to spot) at any frequency over all possible parameter variations. A "tough" performance specification is placed on the sensitivity function in the frequency band of [0,200] *Hz*.

This problem appears in Example 14: CD Mechanism (Sampled-data).

Engine Active Vibration Isolation

This example involves single-axis active vibration isolation (courtesy of LORD Corporation, Cary, NC). The experimental plant frequency response is from an accelerometer mounted on a structure and an active mount connecting the structure to a vibrating engine. The feedback system shown in Fig. 5 has the open-loop plant *P* consisting of the combined engine + structure + mount + amplifier dynamics.

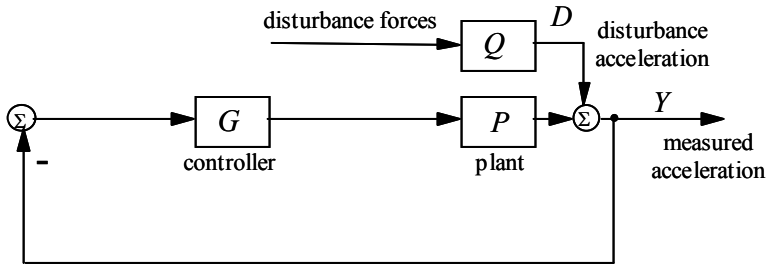


Figure 5: The active vibration isolation feedback system

The frequency response of the open-loop plant is shown Fig. 6.

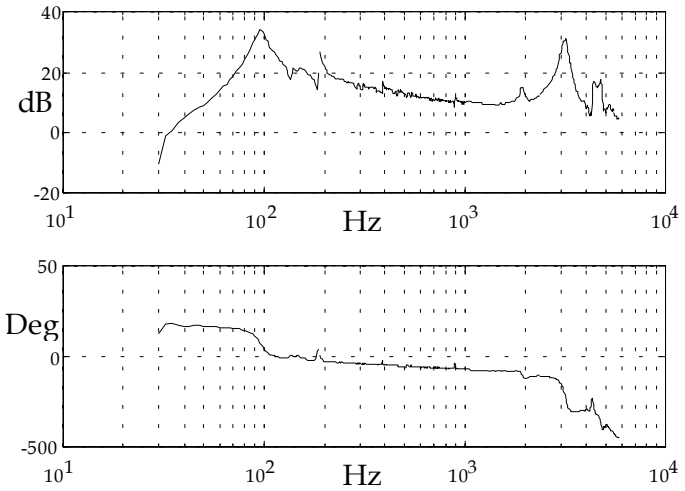


Figure 6: Measured open-loop frequency response

There are two primary control objectives. The first is stability with reasonable margins

$$\left|\frac{PG}{1+PG}(j\omega)\right| \le 1.2, \quad \omega \ge 0$$

and the second is disturbance rejection (transmissibility of disturbance acceleration to measured acceleration) of -20 *dB* in the working frequency band

$$\left|\frac{1}{1+PG}(j\omega)\right| \le 0.1,\ \omega \in [100,200]\text{Hz}.$$

The interaction between the controller and the plant dynamics outside this frequency range should be minimized. Due to hardware constraints, the controller cannot have more than five poles.

This problem appears in Example 11: Active Vibration Isolation.

Formulation of the QFT Design Problem

When the response of an open-loop process does not meet its desired behavior due to uncertainty in its dynamics, and/or uncertainty in the input signals (e.g., disturbances), you should consider using feedback. The Toolbox focuses on feedback problems described in Fig. 7 where the controller to be designed is single input-output. The structure shown in Fig. 7 covers many single-loop systems, cascaded-loop and multi-loop systems designed sequentially or decentralized. Note that Fig. 7 equally represents continuous-time or discrete-time systems (i.e., P can be $P(s)$ or $P(z)$).

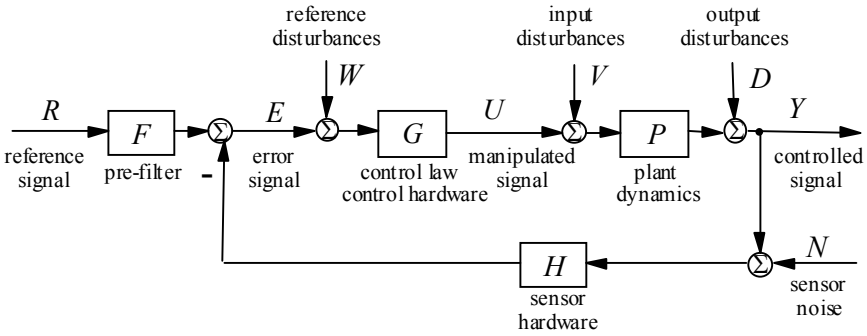


Figure 7: The single-loop feedback system

The feedback system shown in Fig. 7 consists of the plant (open-loop process dynamics), the controller to be designed (e.g., PID) and possibly another transfer function referred to in the manual as a *second known transfer function*. With respect to Fig. 7, if the controller to be designed is G (in the forward path), then H could be used to denote sensor dynamics, while if the controller is H (in the feedback path), G could be used to represent other dynamics.

Formulating Frequency-Domain Specifications

The QFT design, performed in the frequency domain, follows very closely classical designs using Bode plots. The model for the open-loop dynamics can either be fixed or include uncertainty. If the problem requires that the specifications be met with the uncertain dynamics, we call it a *robust performance* problem. That is, the performance specifications must be satisfied for all possible cases admitted by the specific uncertainty model. Various descriptions of uncertainty models in the Toolbox are the focus of the next section.

You can place performance specifications on any single-loop closed-loop relation as shown in Tables 1 and 2 ($F = 1$ when controller bounds are computed). Specifications in the Toolbox are entered in terms of frequency responses. Note that, when possible, the dependency on the Laplace variable, s, or the frequency, ω, is omitted for presentation convenience.

Table 1: Single-loop specification types

Specification	Example of application	Toolbox notation (ptype)
$\left\lvert F\frac{PGH}{1+PGH}\right\rvert \le Ws_1$	gain and phase margins (with sensor dynamics)	1
$\left\lvert F\frac{1}{1+PGH}\right\rvert \le Ws_2$	sensitivity reduction	2
$\left\lvert F\frac{P}{1+PGH}\right\rvert \le Ws_3$	disturbance rejection at plant input	3
$\left\lvert F\frac{G}{1+PGH}\right\rvert \le Ws_4$	control effort minimization	4
$\left\lvert F\frac{GH}{1+PGH}\right\rvert \le Ws_5$	control effort (with sensor dynamics)	5
$\left\lvert F\frac{PG}{1+PGH}\right\rvert \le Ws_6$	tracking bandwidth (with sensor dynamics)	6
$Ws_{7a} \le \left\lvert F\frac{PG}{1+PGH}\right\rvert \le Ws_{7b}$	classical 2-DOF QFT tracking problem	7
$\left\lvert F\frac{H}{1+PGH}\right\rvert \le Ws_8$	rejection of disturbance at plant output (with sensor dynamics)	8
$\left\lvert F\frac{PH}{1+PGH}\right\rvert \le Ws_9$	rejection of plant input disturbances (with sensor dynamics)	9

In this table, Ws_i denotes the specification placed on the magnitude of the transfer function, and where `ptype = i` is used as an input argument in many Toolbox functions to define the specification of interest.

To illustrate use of Table 1, in a continuous-time setting, the sensitivity reduction specification shown in Fig. 4 looks like

$$\left|\frac{1}{1+PGH}(j\omega)\right| \le Ws(\omega),\ \omega \in [0,200]\text{Hz}$$

where the real valued $Ws(\omega)$ takes on the frequency dependent values as in Fig. 4.

A similar situation exists in a discrete-time setting. A sensitivity reduction problem would look like (t_s denotes sampling time)

$$\left|\frac{1}{1+PGH}(z)\right| \le Ws(\omega), z = e^{j\omega t_s},\ \omega \in [0, \pi/t_s].$$

As mentioned above, the Toolbox can also be used in a sequential design of cascaded-loop and multiple-loop systems that involve single-loop design at each design step. Advanced QFT users with knowledge of relevant algorithms can use the following linear fractional transformations as general problem settings:

Table 2: Multiple-loop specification types

Specification	Example of application	Toolbox notation (ptype)
$\left\|\frac{A+BG}{C+DG}\right\| \le Ws_{10}$	inner-loop design of a cascaded system with two loops	10
$\left\|\frac{\|A\|+\|B\|\|G\|}{C+DG}\right\| \le Ws_{11}$	single-loop design in a multi input-output problem	11

Note that with `ptype=10`, `genbnds` can be used to solve any of the problems in Table 1 above except `ptype=7`. The input arguments, A, B, C and D are function of the various plants and controllers in cascaded-loop and multi-loop systems. For example, with `ptype=10` and $A = 0$, $B = H$, $C = 1$ and $D = PH$ is the same as the problem in Table 1 above with `ptype=5`.

Formulating Open-Loop Dynamics Description

Most functions require input arguments in terms of their frequency responses. The open-loop dynamics can be defined in two ways:

1. A model (e.g., transfer function) when it is known.
2. A frequency response when only the measured frequency response data is known.

There are two limitations imposed when frequency responses are used. Closed-loop stability cannot be analyzed by the algorithm and analysis of the closed-loop response is limited to the fixed frequency vector. In many cases you can easily analyze stability by counting crossings in the Nichols chart (see Using the Nichols Chart)

If the open-loop system description does not include uncertainty, then a transfer function model can be defined using a numerator and denominator pair of row vectors or a complex frequency response row vector. Both forms are standard MATLAB format. If you have a state-space model, then you can convert

it to a transfer function model or compute its frequency response. Due to numerical issues, you should avoid use of transfer functions for high-order systems.

A more interesting case occurs when the system description includes uncertainty. The frequency response of an uncertain dynamics is defined in this Toolbox by a complex frequency response matrix, where each row denotes the response of single case. The transfer function model of an uncertain system can be defined as follows. We first consider the continuous-time case, follow with the discrete-time case, and finally discuss their specific data formats within the Toolbox.

Continuous-Time

A continuous-time uncertain transfer function model can have parametric, non-parametric or mixed parametric and non-parametric structures. Parametric uncertainty implies specific knowledge of variations in parameters of the transfer function. For example, consider the set

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

Similarly, a parametric transfer function is also one whose numerator and denominator coefficients lie in intervals.

A non-parametric uncertainty is used in several cases: (1) when the exact nature of uncertainty cannot be correlated to the model's parameters, (2) in conjunction with measurements and robust identification, and (3) to simplify solving the feedback design problem.

One possibility for defining your uncertain dynamics is to directly measure the frequency response of the process using an experiment. You can end up with a set of responses if the measurements were made with several plants that are similar but are not exactly the same. For example, the frequency response of two similar disk drives should be expected to be different due to manufacturing tolerances. Also, if measurements were taken at different operating points (for nonlinear processes) you will end up with a response for each operating point. Hence, your uncertain dynamics will be described by

$$\mathcal{P} = \left\{ P_i(j\omega) :\ i = 1,\ldots n \right\}$$

where n denotes the number of separate measurements.

Another structure of a non-parametric transfer function considered here is

$$\mathcal{P} = \left\{ P(s) = P_0(s)\left(1 + \Delta_m(s)\right) :\ \left|\Delta_m(j\omega)\right| < R_m(\omega),\ \ \Delta_m(s) \text{ stable} \right\}.$$

In addition, in the Toolbox we allow the plant set, $\mathcal{P}$, to include mixed uncertainties (both parametric and non-parametric). In such a case, combining the above models suggests the following plant family $\mathcal{P}$ with mixed uncertainty:

$$\mathcal{P} = \left\{ Ps = \frac{ka}{s(s+a)}\left(1 + \Delta_m s\right) :\ k \in [1,10],\ a \in [1,10],\ \left|\Delta_m(j\omega)\right| < R_m(\omega),\ \Delta_m(s) \text{ stable} \right\}.$$

The difference between the two models, parametric and non-parametric, has a very important consequence in control design. Whenever possible use a parametric over a non-parametric model. The reason is that non-parametric representations ignore specific prior knowledge of the phase of the uncertain

plant. To see this point, consider the above parametric plant. Let the nominal plant be at the values $a = k = 10$. The frequency response sets of the parametric uncertain plant are shown in Fig. 8 at frequencies $\omega = 0.5,5,25,90$ rad/sec. Only the boundaries of each response set are shown (solid lines); however, each point within the solid lines is also part of the set. A non-parametric representation for this plant is obtained by selecting the radius function $R_m(\omega)$ such that at each frequency the non-parametric frequency response set forms a circle (in the complex-plane) that encloses the parametric response set. One such radius function is

$$R_m(\omega) = \left| 0.9 \frac{j\omega/.91+1}{j\omega/1.001+1} \right|.$$

On a Nichols chart, the circle becomes an ellipse, and the non-parametric frequency response sets for $\omega = 0.5,5,25,90$ rad/sec are shown in Fig. 8 with a dashed line superimposed over the parametric sets.

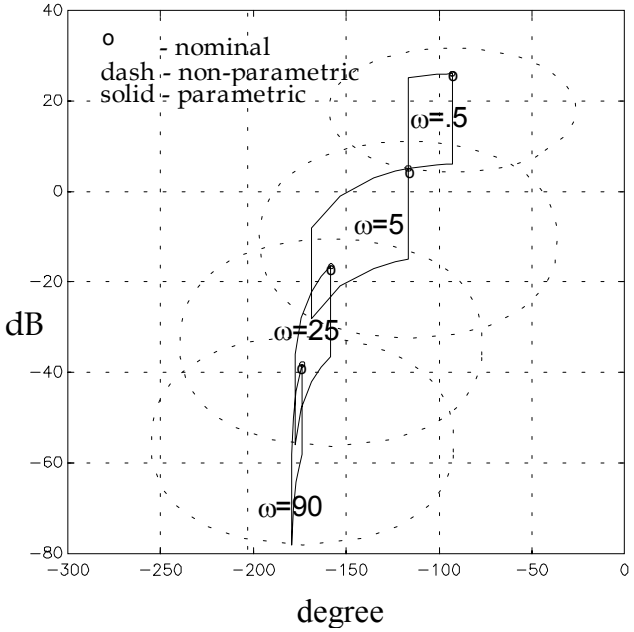


Figure 8: Parametric and non-parametric frequency response sets

Of course, the choice of the nominal plant affects the level of over-bounding.

At a fixed frequency, the plant's frequency response set (regardless of the uncertainty model) is called a *template*. Templates are often obtained directly from frequency response measurements.

Discrete-Time

One possibility for defining your uncertain dynamics is to directly measure the frequency response of the process using an experiment. You can end up with a set of responses if the measurements were made with several plants that are similar but are not exactly the same. For example, the frequency response of two similar disk drives should be expected to be different due to manufacturing tolerances. Also, if measurements were taken at different operating points (for nonlinear processes) you will end up with a response for each operating point. Hence, your uncertain dynamics will be described by

$$\mathcal{P} = \left\{ P_i(j\omega):\ i = 1,\ldots n,\ \ \omega \le \pi / t_s \right\}.$$

where *n* denotes the number of separate measurements and t_s denotes sampling time. Note that for a discrete-time design the sampling frequency used for measuring the responses must be chosen to match the design.

A discrete-time uncertain transfer function can have parametric, non-parametric or mixed parametric and non-parametric structures. Parametric uncertainty implies specific knowledge of variations in parameters of the transfer function. For example

$$\mathcal{P} = \left\{ P(z) = \frac{kz}{z-a} : k \in [1,10],\ a \in [0.8, 0.9] \right\}$$

Similarly, a parametric transfer function is also one whose numerator and denominator coefficients lie in intervals.

The structure of a non-parametric transfer function considered here can be a set of discrete-time frequency responses or

$$\mathcal{P} = \left\{ P(z) = P_0(z)\left(1 + \Delta_m(z)\right) : \left|\Delta_m\left(z = e^{j\omega t_s}\right)\right| < R_m(\omega),\quad \Delta_m(z)\ stable \right\}$$

In addition, in the Toolbox we allow the plant, *P*, to include mixed uncertainties (both parametric and non-parametric). In such a case, combining the above models suggests the following plant family $\mathcal{P}$ with mixed uncertainty

$$\mathcal{P} = \left\{ P(z) = \frac{kz}{z\text{-}a}\left(1 + \Delta_m(z)\right) :\ \ k \in [1,10],\ a \in [0.8, 0.9],\ \left|\Delta_m\left(z = e^{j\omega t_s}\right)\right| < R_m(\omega), \Delta_m(z)\ \text{stable} \right\}$$

Data Format

The data format in the Toolbox is now consistent with linear time-invariant (LTI) models in the *Control Toolbox*. Fixed models are defined as a transfer function (TF), a zero/pole/gain (ZPK), a state-space (SS) or a frequency response data (FRD).

Parametric uncertainty can be modeled using LTI arrays. An LTI array is a collection of TF, ZPK, SS or FRD objects.

Because the Toolbox now supports only LTI models for its input and output arguments, it is essential that you become familiar with these concepts. You can type `ltimodels` at the command line for a quick overview, however, you should read Chapters 1-4 in the *Control Toolbox* manual to become familiar with definitions, creation of such models, properties, supported math and logical operations and the concept of arrays.

3 Feedback Design with QFT

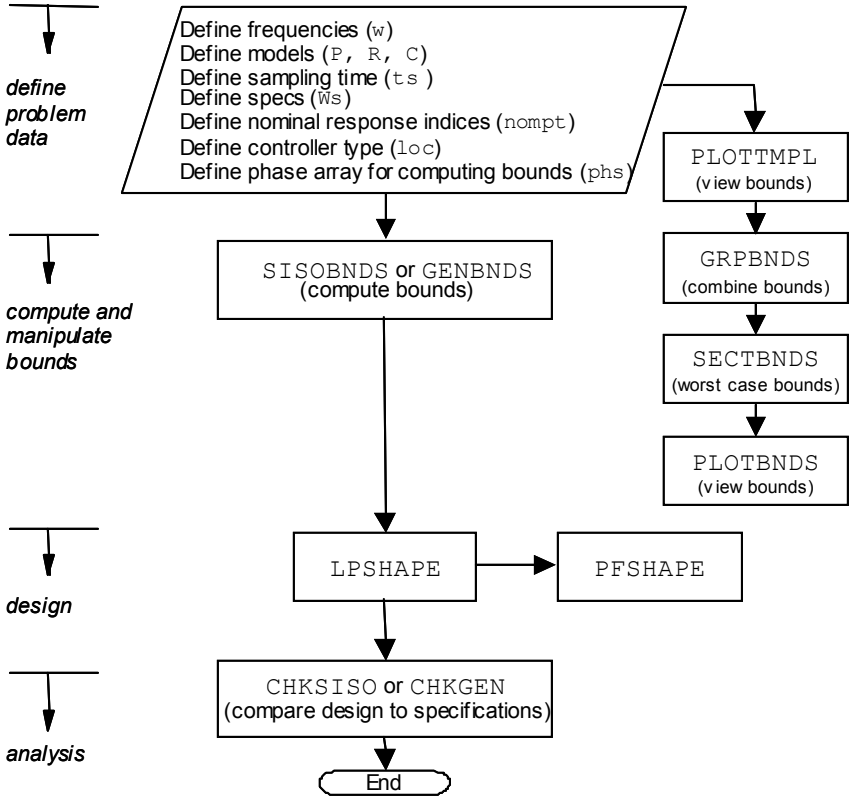
Getting Started

In this chapter we describe all relevant details of a single-loop QFT design. The design procedure is developed in a constructive manner using a simple example. A QFT design typically involves three basic steps:

1. Computation of QFT bounds,
2. Design of the controller (and possible pre-filter), and
3. Detailed analysis of the design.

In systems with parametric uncertainty models, you must first generate plant templates prior to step 1. At a fixed frequency, the plant's frequency response set is called a *template*. Given the plant templates, QFT converts closed-loop magnitude specifications into magnitude and phase constraints on a nominal open-loop function. These constraints are called *QFT bounds*. A nominal open-loop function is then designed to simultaneously satisfy its constraints (expressed as bounds) as well as to achieve nominal closed-loop stability. In a two degree-of-freedom design, a pre-filter will be designed after the loop is closed (i.e., a controller has been designed). Due to engineering approximations involved, an analysis step usually follows the design step.

The various steps in a QFT design procedure are shown in the following flow chart. Note that some steps are optional. For example, you need not define the sampling time if you are performing a continuous-time design. You can even jump right into loop design (`lpshape`), if robustness is not an issue. The functions all have many defaults values; for example, you do not have to pass bounds into the design function `lpshape`.



Flow chart showing basic steps in a QFT design

The QFT design procedure for continuous-time systems is now presented. An exposition for discrete-time systems then follows, although the two procedures are quite similar. The discussion for each class of systems is divided into several sections based on the conventional order of QFT design execution. These sections discuss topics such as templates, choosing frequencies, choosing the nominal plant, stability bounds, performance bounds and design. Where deemed necessary, each section is further divided into two parts: concept and practice. The concept part covers conceptual issues while the practice part reveals how the particular procedure is practiced in the Toolbox.

Continuous-Time

The basic steps in a QFT design procedure are presented in this section by first discussing each step conceptually and then showing how it is applied in practice. For this purpose we consider a generic robust design problem with simultaneous specifications and parametric plant model. Suppose the uncertain plant $P(s)$ in the system shown in Fig. 9

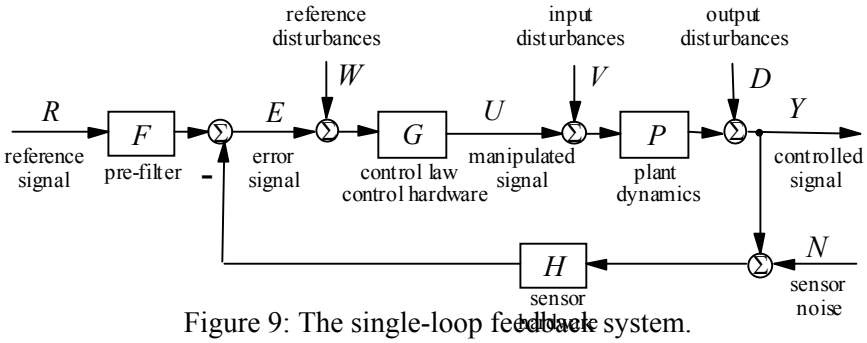


Figure 9: The single-loop feedback system.

is described by the parametric family $\mathcal{P}$

$$\mathcal{P}=\left\{P(s)=\frac{k}{(s+a)(s+b)}:\ \ k\in[1,10],\ a\in[1,5],\ b\in[20,30]\right\}.$$

We assume a sensor with unity gain $H(s) = 1$. The feedback problem is to design a controller $G(s)$ such that the closed-loop system is robust stable and has at least 50° phase margin for all $P(s)\in\mathcal{P}$. The specification

$$\left|\frac{PG}{1+PG}(j\omega)\right|\le W_s=1.2,\ \text{for all}\ P\in\mathcal{P},\ \omega\in[0,\infty)$$

implies at least 50° lower phase margin and at least 1.66 lower gain margin (not simultaneously). To compute in general these margins, use the following formulae [8]

$$\begin{aligned}\text{lower gain margin} &= 1+W_s^{-1}\\ \text{lower phase margin} &= 180^\circ-\theta,\theta=\cos\left(0.5W_s^{-1}-1\right)>0\,.\end{aligned}$$

The Robust Stability (Margins) Bounds Section includes more details about margins and bounds. Note that different margins bounds can be computed using the sensitivity function problem `sisobnds` with `ptype=2` (see [8] and [15] for details). In addition, there are two robust performance specifications: reject plant output disturbance according to

$$\left|\frac{Y}{D}(j\omega)\right|\le\left|0.02\frac{(j\omega)^3+64(j\omega)^2+748(j\omega)+2400}{(j\omega)^2+14.4(j\omega)+169}\right|,\quad\text{for all}\ P\in\mathcal{P},\ \omega\in[0,10]$$

(no specific reason for the transfer function on the right-hand side) and reject plant input disturbance according to

$$\left|\frac{Y}{V}(j\omega)\right| \le 0.01, \text{ for all } P \in \mathcal{P},\ \omega \in [0,50].$$

We first discuss conceptually the design procedure for robust stability and for robust performance: generating templates, computing bounds, loop shaping and analysis. We follow with a step-by-step description of how a QFT design is performed in this example. Note that in the discussion below, all the commands shown in a separate line such as

```
nompt = 21;
```

also appear in each example file such as the file `qftex1.m`.

Templates

One of the most important factors in control design is to use an accurate description for the plant dynamics. Because QFT involves frequency-domain arithmetic, its design procedure requires you to define the plant dynamics only in terms of its frequency response. The term template is used to denote the collection of an uncertain plant's frequency responses at a given frequency. The use of templates frees you from the need to have any particular plant model representation. In QFT, you can use frequency response measurements obtained from experiments to describe the dynamics. However, specific uncertainly models are often used, such as parametric and non-parametric models. The relation between such models and their templates is explained below.

Concept

The Toolbox allows mixed uncertainty model for the plant *P* (parametric and non-parametric), and it allows for one additional parametric uncertain transfer function in the loop (*G* or *H*). Because parametric uncertainty must be defined in the Toolbox in terms of a finite set of plants (most often obtained by forming a grid in the uncertain parameter space), you should always carefully study the resulting plant response set. A generic illustration of "good" and "bad" grid choices are illustrated in Fig. 10. In general, there are no rules for obtaining a reasonable approximation of the boundary from the structure of the parametric uncertain plant. However, for specific cases, such as transfer functions with coefficients belonging to known intervals or with coefficients related to the uncertain parameters in a linear or multi-linear fashion, you can find some useful results in [e.g., 9-11].

Figure 10: "good" and "bad" approximations of a plant template.

The algorithms for computing bounds require input data in terms of frequency responses (templates) rather than in terms of numerator/denominator transfer functions. For simply connected templates, it is necessary and sufficient to work only with the boundary of these templates [15]. (This is related to a celebrated result in complex variables, the maximum principle.) The possible difficulty with this

approach is that if the plant has a large number of independently uncertain parameters, the template will have to be described with hundreds or even thousands of cases. Even with powerful computers, a solution may require an unrealistic long time to derive. One option is to convert the parametric model (or part of it) into a non-parametric model. This conversion makes possible mathematical solutions to complex problems; however, it comes with the price of design conservatism [12]. The idea advocated here is to arrive at an approximation where the template's grid points are "close" to each other uniformly. A crude discretization can be obtained using a simple grid over each uncertain parameter. For example,

$$\mathcal{P} = \left\{ P(s) = \frac{k}{(s+a)(s+b)} :\ k = [1,2,5,8,10],\ a = [1,3,5],\ b = [20,25,30] \right\}.$$

Practice

In the first step of a QFT design procedure you define models of open-loop transfer functions (i.e., plant, actuators and sensors. Transfer functions models, fixed or uncertain, are defined as LTI models. The boundary of the corresponding template can be established using results from [9-11] in special cases, or in general (as done in this example) using a grid of the parameter space since the number of uncertain parameters is small. Such a procedure will most likely yield interior template points that are not necessary for a QFT design and results in an undue computational burden. A more careful study of the template can reduce this burden by eliminating interior points. Our study showed that 40 plant elements are sufficient to describe the template's boundary. Typically, you will investigate how each uncertain parameter affects the shape of the template (holding all others fixed), then include another uncertain parameter eventually building up the template boundary.

The following defines a numerator and denominator matrix pair for the plant set (40 elements) that corresponds to the template's boundary:

```
c = 1; k = 10; b = 20;
for a = linspace(1,5,10),
 P(1,1,c) = tf(k,[1,a+b,a*b]);  c = c + 1;
end
k = 1; b = 30;
for a = linspace(1,5,10),
 P(1,1,c) = tf(k,[1,a+b,a*b]);  c = c + 1;
end
b = 30; a = 5;
for k = linspace(1,10,10),
 P(1,1,c) = tf(k, [1,a+b,a*b]);  c = c + 1;
end
b = 20; a = 1;
for k = linspace(1,10,10),
 P(1,1,c) = tf(k, [1,a+b,a*b]);  c = c + 1;
end
```

The above uses the notion of LTI arrays. Note that the first two indices are used for input/output relations while the third is used for arrays (i.e., uncertainty in our context). Please refer to Chapters 1-4 in the *Control Toolbox* manual for more details.

Next, we must define a nominal plant element that will be used throughout the design. The choice of nominal element is arbitrary and has no effect on the design (except when it is pre-defined as in a non-parametric model). Let us arbitrarily choose the following element (an integer index)

```
nompt = 21;
```

The next step consists of computing the frequency response set of the uncertain plant in rad/sec. The frequency array must be chosen based on the performance bandwidth and shape of the templates. Margin

bounds should be computed up to the frequency where the shape of the plant template becomes invariant to frequency. Here, at approximately ω = 100 rad/sec, the template's shape becomes fixed, a vertical line. Our array includes several frequencies within the performance bandwidth of [0,50] and this particular frequency of ω = 100:

```
w = [0.1,5,10,100];
```

If you are uncertain how to select such a frequency array, simply start at a low frequency and advance with 2-3 octaves steps within the performance bandwidth and add frequencies above it as needed for margin bounds at higher frequencies. A more detailed discussion of how to select frequencies can be found in the next section.

It is very important that you view the plant templates before proceeding with the design. Viewing templates lets you verify that the template boundary approximation is reasonable and that you have selected an appropriate frequency array. To view the templates with a highlighted nominal case invoke

```
plottmpl(w,P,nompt);
```

which results in the plot shown in Fig. 11. The templates are shown based on the second input argument, a subset of the full frequency array.

Hint: You can zoom in to any region in the plot if several templates appear clustered together, or right mouse button to toggle on/off showing of a particular template. Please refer to the Reference Chapter for details.

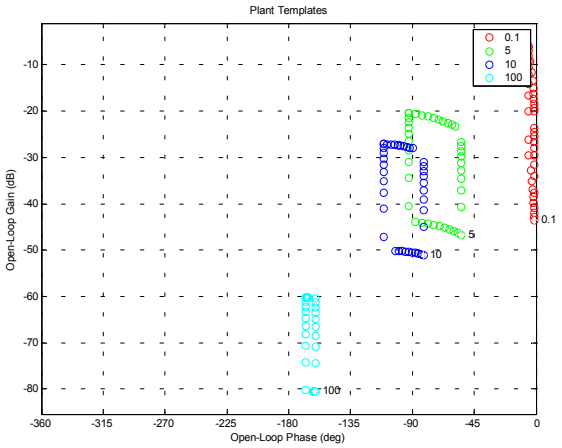


Figure 11: Plant templates at several frequencies.

We cannot over emphasize the importance of working with "smooth" approximation of templates (those whose boundaries are described by a sufficient number of points). If too few points are used, the computed bounds will not be relevant to your original plant description whose boundary is a continuous smooth curve. For example, let us use only two points to describe the template ω = 5 rad/sec.

```
P1 = P(:,:,[1,21]);
```

The template (Fig. 12) exhibits almost 30 *dB* spacing.

```
plottmpl(w(2),P1);
```

The fatal implication of such a template on the shape of the bounds is discussed in the Bounds section below.

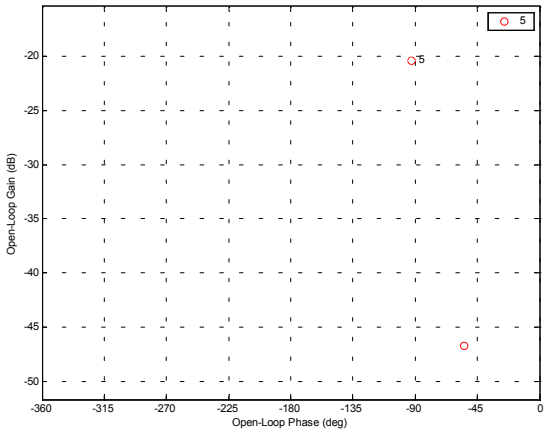


Figure 12: A two-element plant template

Choosing Frequencies

In any QFT design, you have to select a frequency array for computing templates and for computing bounds (as explained below). An important question, for which there is no definite global answer, is how to select this array from the possible range between zero and infinity. Fortunately, for engineering design we need only a small set that can be found with, at the most, a few iterations. The basic rule is that for the same specification, the bounds will change only with changes in the shape of the template. Therefore, you should look for frequencies where the shape of the template shows significant variations compared to those at other frequencies. How low should you select a frequency? Well, most plants will exhibit dynamics with monotonic behavior in terms of the template shape below a certain frequency, ω_1, or in the limit

$$P(s) \xrightarrow[\omega \to 0]{} \frac{k_\infty}{s^m}, \; m = 0,1,2,\ldots$$

where *m* denotes the free integrators. The specifications below ω_1 are most often monotonic too, either a constant or a linear function of the frequency. So you can start with ω_1 as the lowest frequency in the array. What about the largest frequency in the array? Parametric uncertain plants will exhibit dynamics with monotonic behavior above a certain frequency ω_2, or in the limit

$$\lim_{\omega \to \infty} P(s) = \frac{k_\infty}{s^n}$$

where *n* denotes the excess of plant poles over zeros. Considering that at high frequencies the only specification should be a constant robust stability margin (see next section), the corresponding bounds for $\omega \geq \omega_2$ will all be the same. So select ω_2 as your largest frequency in the array.

Next you should select a frequency grid between ω_1 and ω_2. The idea is to select a grid such that you compute bounds that capture variations in shapes and in specifications across that frequency band. As a first cut, start with a grid every one octave or a few octaves (it will include many more frequencies than actually needed). From the resulting bounds, you will obtain an insight into the nature of the bounds and their relation to the plant dynamics and specifications. This insight will be used to eliminate most of the redundant frequencies. By redundant we mean that having performance bounds within a few *dB* values from each other is not needed for design. Usually, for the same problem, since a performance bound at

ω_x lies above the bound at ω_y ($\omega_y > \omega_x$), we need only a few nicely (frequency) spaced bounds for control design. One exception is the case of plants with resonant dynamics with variations at the natural frequencies or damping ratios. In such cases, the plant template and the corresponding bounds are not monotonic around a natural frequency (you may need to select a few frequencies within the band of natural frequency uncertainty). Finally, to get a better feel for selecting the array, go over the example files and observe the frequency arrays chosen there relative to the problem data.

In certain problems, analysis of a completed design may indicate that you did not meet some specification over a small frequency range. This can happen only at a range for which you do not have a frequency in the array and obviously did not compute a bound there. This is what we mean by the need for iteration. In such a case, select a new frequency within this range, re-compute bounds and then augment the design as necessary.

Choosing the Nominal Plant

In order to compute bounds, you will have to designate one plant element from the uncertain set as the nominal plant (if there is no uncertainty the fixed plant is the nominal one). This is required in order to perform QFT design with a single nominal loop. If you described the plant with a non-parametric uncertainty model with disk uncertainty, the nominal plant is already determined. However, when the uncertain set corresponds to parametric uncertainty you have a choice. As long as the set satisfies the assumptions on the uncertainty model given in Continuous-Time, you may choose any plant case. Pick the one, which you think is most convenient for design. Note that the nominal plant index is an integer.

Bounds

Given the plant templates, QFT converts closed-loop magnitude specifications into magnitude and phase constraints on a nominal open-loop function. These constraints are called *QFT bounds*. The three most common types of bounds are illustrated in Fig. 13 using both complex plane and Nichols chart representations.

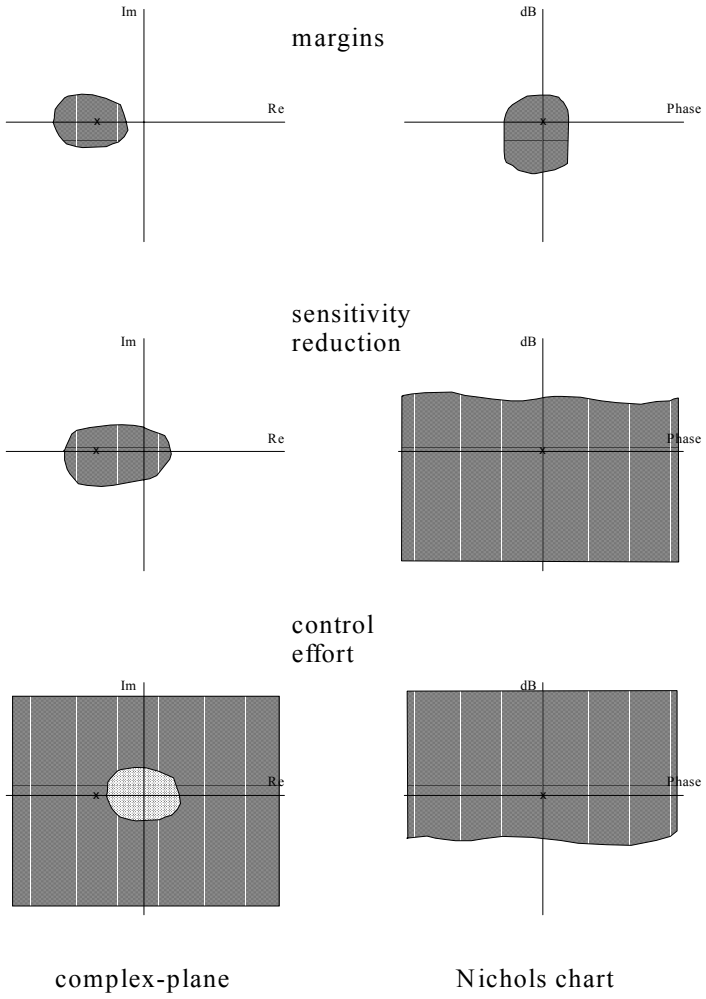


Figure 13: Common types of QFT bounds.

Margin-type problems result in bounds about the critical point (top pair) where the loop response must remain outside the bounds (the dark shaded region is the one to avoid). Sensitivity reduction type problems that require increased loop gain, result in bounds about the origin (middle pair) where the loop response must remain outside the bounds. Control effort-type problems which limit the amount of loop gain, result in bounds about the origin (bottom pair) where the loop response must remain inside the bounds.

Robust Stability (Margins) Bounds

In this section we discuss the consequences of the robust stability results in Robust Stability in terms of bounds on the nominal loop. We have not presented these results yet since familiarity with the theory is not required at this point for explaining the QFT design procedure. *Note that the terms stability bounds and margin bounds have historically the same meaning in the QFT context.*

Concept

The two conditions for robust stability (Robust Stability Criterion 2) are: (1) stability of the nominal system (corresponding to the nominal plant) and (2) the Nichols envelope does not intersect the critical point q (which is the (-180°,0 *dB*) point in a Nichols chart or the (-1,0) point in the complex plane) The

second condition is equivalent to placing a magnitude constraint on the complementary sensitivity function

$$\left|\frac{L}{1+L}(j\omega)\right| < \infty, \text{ for all } P \in \mathcal{P}, \ \omega \geq 0 .$$

By assumption, the templates are simply connected and $L(s)$ has a fixed number of unstable poles. Hence, if $1+L(j\omega) \neq 0$, the Maximum Principle implies that it is necessary and sufficient to check the above condition only over the boundary of the template. It follows that we can replace the above by

$$\left|\frac{L}{1+L}(j\omega)\right| < \infty, \text{ for all } P \in \partial\mathcal{P}, \ \omega \geq 0$$

where $\partial\mathcal{P}$ denotes the boundary of the template. Our numerical algorithms require the approximation of $\partial\mathcal{P}$ by a finite number of plant cases (except if it is a disk shape as in non-parametric uncertainty model). The approximation introduces the problem, however, that you can never be sure that the critical point q does not intersect $\partial\mathcal{P}$ at a point that was removed during the discretization process. Therefore, the above condition is typically replaced by the following margin condition

$$\left|\frac{L}{1+L}(j\omega)\right| < W_s > 1, \text{for all } P \in \partial\mathcal{P}, \ \omega \geq 0 .$$

Graphically speaking, for each μ there is a closed curve about the critical point q – the classical closed-loop constant magnitude circle [13]. The weight W_s is used as a safety factor in this context. Even if the critical point q actually lies on the template's boundary in between adjacent grid points, it will not escape (fit in between) a large enough constant magnitude circle. The smaller W_s is, the larger the spacing that can be tolerated (i.e., a more crude approximation).

A similar margin like specification is given by a constraint on the sensitivity function

$$\left|\frac{1}{1+L}(j\omega)\right| \leq W > 1, \text{ for all } P \in \partial\mathcal{P}, \ \omega \geq 0 .$$

The difference between this condition and the one in terms of complementary sensitivity is that this one enforces a smaller loop gain when the plot crosses the -180° line below 0 *dB*. In conditionally stable systems, the complementary sensitivity condition enforces a larger loop gain when the plot crosses the -180° line above 0 *dB*. The gain and phase margins relative to W can be easily computed as done earlier with respect to a complimentary sensitivity weight W_s.

Given the plant templates, QFT translates the robust margin (or stability margin only if $W_s = \infty$ or W = ∞) constraint into required conditions on the phase and magnitude of the controller. These constraints are referred to *as QFT bounds*. For example, the robust margin bound at ω = 1 is shown in Fig. 14.

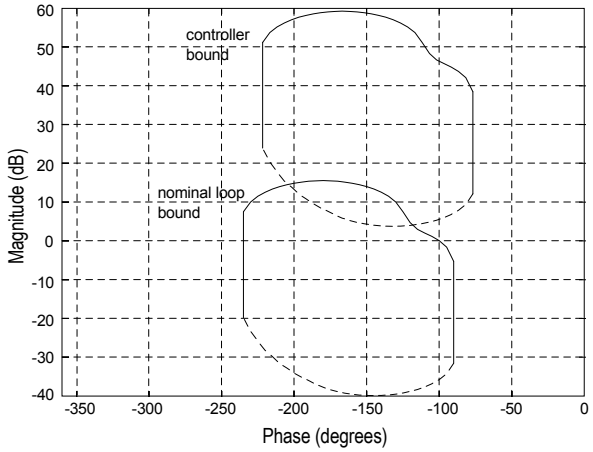


Figure 14: Controller and nominal loop robust margin bounds at ω =1.

To apply Criterion 2, these bounds are subsequently translated into bounds on the nominal loop, $L_0(j\omega)$, bounds. A bound for is simply equal to the bound for $G(j\omega)$ shifted vertically by the magnitude of $P_0(j\omega)$ and horizontally by the phase of $P_0(j\omega)$, where $P_0(s)\in\mathbf{P}$ is an arbitrarily selected nominal plant. In a given problem, the same nominal plant must be used throughout the design. The nominal loop bound for the robust margin at ω = 1 is shown above, where the nominal plant is (recall that we earlier selected it to be the 21st element)

$$P_0(s)=\frac{k_0}{(s+a_0)(s+b_0)},\ k_0=1,\ a_0=5,\ b_0=30\ .$$

A word is in order regarding our graphical notation: a bound plotted with a *solid line* implies that $L_0(j\omega)$ must lie above or on it in order to meet the particular specification. A bound plotted with a *dashed line* implies that $L_0(j\omega)$ must lie below or on it in order to meet the particular specification. If the closed-loop specification was strict inequality, the loop response cannot lie right on the bound.

If $L_0(j\omega)$ lies right on its margin bound, then we achieve an optimal design in the sense that at that frequency

$$\max_{P\in\mathcal{P}}\left|\frac{L}{1+L}(j\omega)\right|=W_s\ .$$

A question arises as to what frequencies should be chosen for computing bounds. In theory, one should compute bounds over the entire $j\omega$-axis. In practice, bounds can be computed only up to a finite frequency. This finite frequency is selected based on the high-frequency asymptotic behavior of the plant's frequency response. As ω increases, the change in the shape of the template decreases and eventually it approaches a fixed shape. In a mixed uncertainty plant, if $R_{m_i}(\omega) = R_m$ for some $\omega>\omega_{hf}$, we also have a fixed-shape template at high frequencies. In our example we have

$$P(j\omega)\approx\frac{k}{(j\omega)^2},\ \omega\geq\omega_{hf}\ .$$

For $\omega\geq\omega_{hf}$, the shape of the templates of this class of plants on a Nichols chart is a vertical line (see Fig. 11). This implies that the robust margin bounds for $\omega\geq\omega_{hf}$ are all the same. By plotting several templates at increasing frequencies, one can find the value of this frequency. The robust margin bound is

shown in Fig. 15 at $\omega_{hf} = 100$. From this point on, when we mention bounds we imply nominal loop bounds.

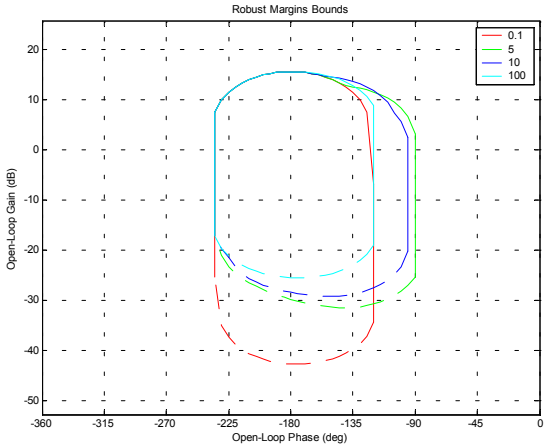


Figure 15: Robust margin bounds including at $\omega_{hf} = 100$.

The Toolbox uses colors to indicate bounds at different frequencies. The frequency legend can be found at the top left corner of the plot screen. We assign integers to each bound to relate it to a problem type `ptype` (e.g., margin=1 and output disturbance rejection=2; see Table 1: Single-loop specification types and Table 2: Multiple-loop specification types).

Practical Considerations

There are functions for computing bounds (sisobnds) in single-loop systems and general-purpose bounds (genbnds) for cascaded-loop and sequentially designed multiple-loop systems. One function (there may be more than one) that fits the robust margin specification is `sisobnds(1,...)` with the generic call

```
bdb = sisobnds(ptype,w,Ws,P,R,nom,C,loc,phs);
```

The second input argument defines the frequency array where margin bounds are to be computed (must be a subset of the frequency array in `P` if an FRD object). The specification here is fixed, so

```
Ws1 = 1.2;
```

The nominal plant element in `P` is `nompt = 21` was already defined earlier. The controller to be designed is in the forward loop ($G(s)$ in the Toolbox notation) with unity feedback ($H(s) = 1$). The robust margin bounds can be computed by invoking

```
bdb1 = sisobnds(1,w,W1,P,R,nompt);
```

You may be asking yourself why we used only 7 input arguments when above we show 10 input arguments. An important feature of the Toolbox is the use of default values. In most designs it is not necessary to define all input arguments; those not specified are assigned default values. For example, the argument loc is used to indicate whether the controller to be designed is in the forward path (G) or in the feedback path (H); the default is `loc = 1` indicating forward path. Using these defaults, the input arguments list can be significantly shortened. As a general rule, an empty matrix implies a default value. In our example, because the last four input arguments in invoking `sisobnds(1,...)` above were not defined, the program will automatically use their default values, specifically, `C = 1` (H(s) is 1), `loc = 1` ($G(s)$ is the controller to be designed), and `phs = [0,-5,-10,...,-360]°`. This default phase array is a reasonable choice for most problems.

You may recall that in the Templates section we discussed the issue of "smoothness" of templates. Bounds for a template whose boundary is defined with too few points will most likely end up non-connected. For example, taking the two-point template shown in Fig. 12 and a specification of `W1 = 3.5` results in the non-connected bound shown in Fig. 16.

```
sisobnds(1,w(2),3.5,P1);
```

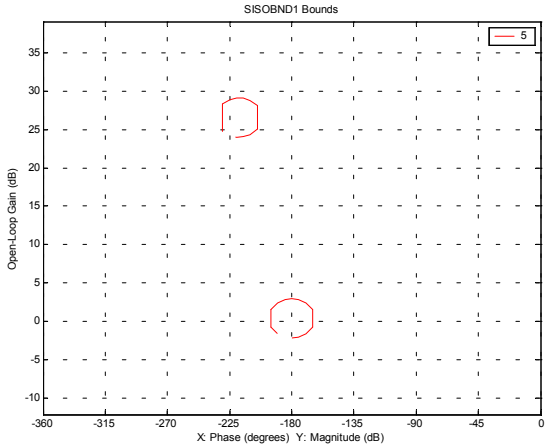


Figure 16: Non-connected bound of the two-point template.

If you detect non-connected bounds, take a closer look at your plant templates (look for too coarse a grid, i.e., large gaps between adjacent template points). Of course, if the plant is originally non-connected, then you have no choice. You must be extremely careful in evaluating robust stability for such non-connected plants.

Robust Performance Bounds

This section explains how different performance specifications, for example sensitivity reduction or tracking specifications, are converted into bounds on the nominal loop.

Concept

Performance specifications are typically defined within a finite frequency bandwidth that is related to closed-loop system bandwidth and spectrum of the disturbances. Except for rare cases, there is very little to be gained by specifying transfer function magnitudes up to frequency of infinity. The reason is that in physical systems open-loop transmission of signals are negligible beyond a certain finite frequency (real-life transfer functions are strictly proper). At the high frequency band, the magnitude of the (strictly proper) transfer function is very small, say less than 0.0001 (<-80 *dB*), and hence its contribution to the time response of the system is negligible (except at a small neighborhood of $t = 0$ sec). For this reason it make little sense to define the nominal open-loop dynamics or the uncertainty at high frequencies far above the system's bandwidth. Therefore, in a QFT design, performance is specified only up to a finite frequency whose value is always problem dependent.

Practical Considerations

One function that fits the output disturbance rejection transfer function constraint is `sisobnds(2,...)` with the generic call

```
bdb = sisobnds(2,w,Ws2,P,R,nom,C,loc,phs);
```

The performance weight is defined

```
Ws2 = tf(.02*[1,64,748,2400],[1,14.4,169]);
```

and the desired bandwidth [0,10], the bounds are computed from

```
bdb2 = sisobnds(2,w(1:3),Ws2,P,0,nompt);
```

One function that fits the input disturbance rejection transfer function constraint is `sisobnds(3,...)` with the generic call

```
bdb = sisobnds(3,w,Ws3,P,R,nom,C,loc,phs);
```

The specification is a constant

```
Ws3 = 0.01;
```

and the bounds (within the bandwidth of interest) are computed from

```
bdb3 = sisobnds(3,w(1:3),Ws3,P,0,nompt);
```

Note that `w = 50` is not included. To see why this frequency is not included, add `w = 50` to `w` (`w = sort([w,50])`), re-compute the template and compute the bounds with `wbd3`, then plot the bounds.

The nominal loop bounds for the robust output disturbance rejection and the robust input disturbance rejection are shown in Fig. 17-18 for several frequencies within the desired performance bandwidth.

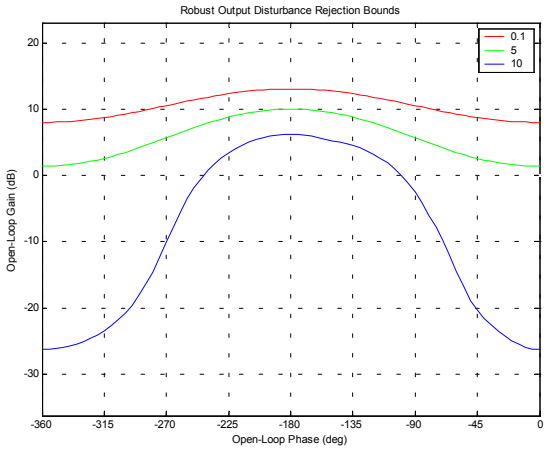


Figure 17: Robust output disturbance rejection bounds.

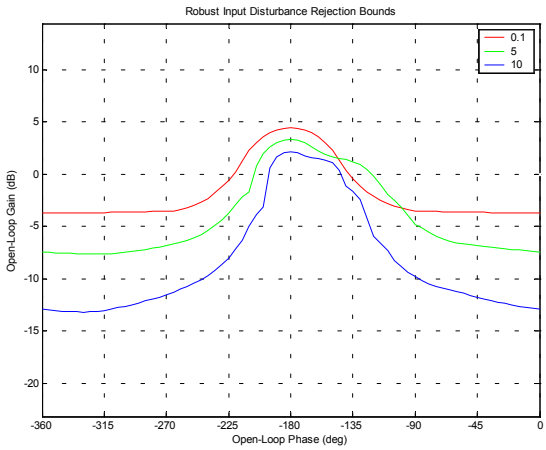


Figure 18: Robust input disturbance rejection bounds.

Working with Bounds

At this point we have computed bounds for all performance and margin problems. The next step is to combine them into a single variable by invoking

```
bdb = grpbnds(bdb1,bdb2,bdb3);
```

This is done solely for convenience, as it is always simpler to work with a single variable. Let us view the bounds (Fig. 19) using

```
plotbnds(bdb);
```

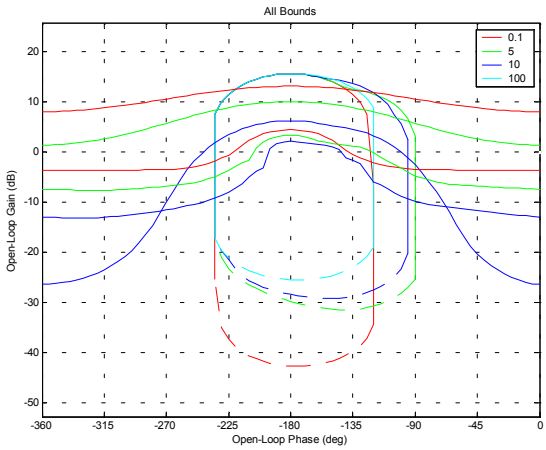


Figure 19: Superposition of all bounds.

Note the legend shown at the top left corner of your figure (specific frequencies can toggled on/off using right mouse button). Viewing the grouped bounds is often used to compare different specifications and quickly identify competing bounds that a nominal loop cannot satisfy simultaneously. In general, when the problem involves more than one set of bounds, one should compute the worst case bound of all sets. It is much simpler to work with a single, worst case bound (i.e., the intersection of all bounds) than with a collection of many bounds.

To compute the worst case bound invoke

```
ubdb = sectbnds(bdb);
```

The function `sectbnds` includes the most general algorithms for computing the intersection of any combinations of bounds. To view the worst case of all bounds, invoke again the same bound plotting function

```
plotbnds(ubdb);
```

The picture is much clearer now as shown below (Fig. 20). Take another look at Fig. 19 to visualize how Fig. 20 was arrived at.

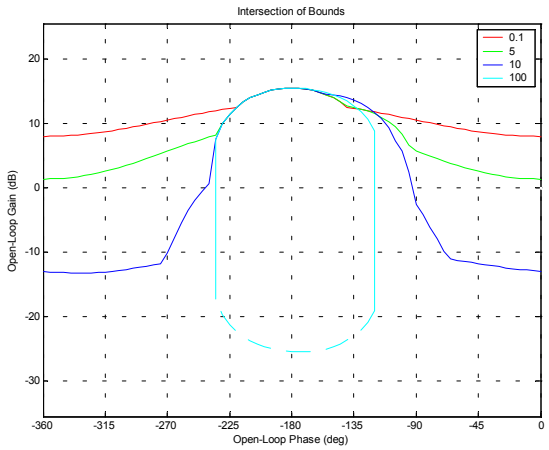


Figure 20: Worst-case (intersection) of all bounds.

We are now ready for loop shaping, that is, design of the controller G.

Design (Loop Shaping)

Having computed stability and performance bounds, the next step in a QFT design involves the design (loop shaping) of a nominal loop function that meets its bounds. The nominal loop is the product of the nominal plant and the controller (to be designed). The nominal loop has to satisfy the worst case of all bounds. The Toolbox includes an interactive design environment for loop shaping. A detailed description of the design environment can be found in The Interactive Design Environment (IDE). Nominal loop shaping is done using `lpshape` with the generic call

```
lpshape(wl,ubdb,P0,C0,phs);
```

The first input argument defines the frequency array for loop shaping. It should be the same one you would use in Bode plotting, that is, cover 3-5 decades in most cases. You can add or subtract decades to the frequency array inside the design environment, or even let the function select one for you by entering an empty matrix []. So let us start with

```
wl = logspace(-2,3,100);
```

Next, define the nominal plant. Control design is performed using the nominal loop is $L_0(s) = P_0(s)G(s)$ ($H = 1$ here), where $P_0(s)$ must be the same nominal plant used during bound computations, hence

```
L0=P(1,1,nompt);
```

```
L0.ioDelay = 0; % no delay
```

If $H(s)$ is also uncertain then $L_0(s) = P_0(s)G(s)H_0(s)$. Often, you know *a priori* of certain poles and zeros that the controller must have, e.g., an integrator, or have designed a controller using other methods. This is the purpose for the initial controller input argument `C0`.

All example M-files (`qftex1-qftex15`) include pre-designed controllers. In this example we include two pre-designed controllers, one proper and the other strictly proper. When you run this example (`qftex1.m`) you will prompted to select one at the design step. However, to accomplish our objective of teaching you how to loop shape, we use no initial controller that is

```
C0 = tf(1,1);
```

Hint: You may find that in certain cases, e.g., plants with unstable and non-minimum phase zeros, it is difficult to loop shape a stabilizing controller manually. A recommended approach in such cases is to *a priori* design a stabilizing control for the nominal plant using one of many available methods (e.g., see *Control System*, *Robust Control*, and μ-*Analysis and Synthesis* Toolboxes). Use this controller as your initial controller. You can import a linear time-invariant controller into `lpshape` using the function `putqft`.

You are now ready to enter the loop-shaping environment

```
lpshape(wl,ubdb,L0,C0);
```

where we have used the default setting `phs = [0,-5,-10,...,-360]` degrees.

Let us now learn how one typically performs loop shaping. *This aspect of the QFT design is usually the most difficult for novice users*. The more experience you gain, the easier it gets. Invoking the above function results in the plot shown in Fig. 21.

Generally speaking, loop shaping involves adding poles and zeros until the nominal loop lies near its bounds and results in nominal closed-loop stability. The following is one possible loop shaping sequence. There are many other sequences, equivalently acceptable, that are based on the particular experience of the user. This is really the beauty of QFT; it provides the user with the power to consider different controller complexity and values and weight possible trade-offs almost instantly. An excellent exposition of loop shaping can be found in [3] (as well as in [30]-[32]).

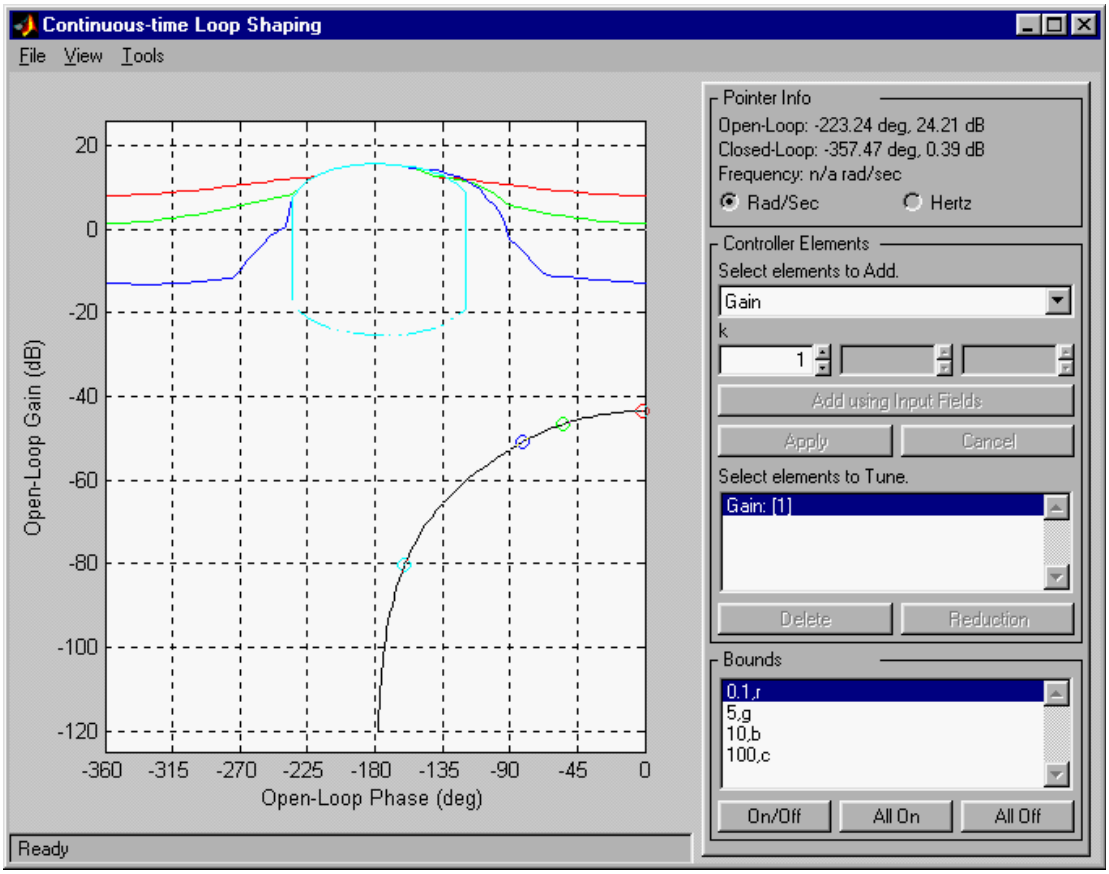


Figure 21: Initial nominal open-loop response and its bounds.

The nominal loop to be designed is $L_0(s) = P_0(s)G(s)$ where $P_0(s)$ is the nominal plant. Within the design environment, although you loop shape the plot of $L_0(s)$ during design you are in essence loop shaping the controller $G(s)$. *Therefore, in any design step, the zeros and the poles you are working with are your controller elements G(s).*

Let us first iterate on the value of the controller's gain to bring it closer to the performance bound at $\omega = 0.1$. This and other loop shaping options can be done by using either graphics protocol (via mouse) or text protocol (via keyboard). For more details, please refer to the Interactive Design Environment section in Chapter 6. The descriptions below show commands in boldface such as **Select Elements to Add** that imply the use user interface controls.

Using graphics protocol: Selecting the Gain element in the **Select Elements to Add** pull-down menu , select a point on the nominal loop by pressing and holding down the mouse button. A special marker appears when the mouse lies on top of any part of the loop and you will be able to drag the selected frequency. The Toolbox will compute the gain multiplication or division necessary to move the loop to its new location. The gain can be modified further by re-selecting the marker on the loop and dragging the loop up or down to a new location.

Using text protocol: Select the Gain element in **Select Elements to Add** pull-down menu and enter numerical value(s) in the box(s) below. Pressing **Add Using Input Fields** displays both the current response and the updated response. Pressing **Apply** accepts this new element. For the Gain, enter the new value of `379` and press <enter>. Press **Apply** to accept the new value (Fig. 22).

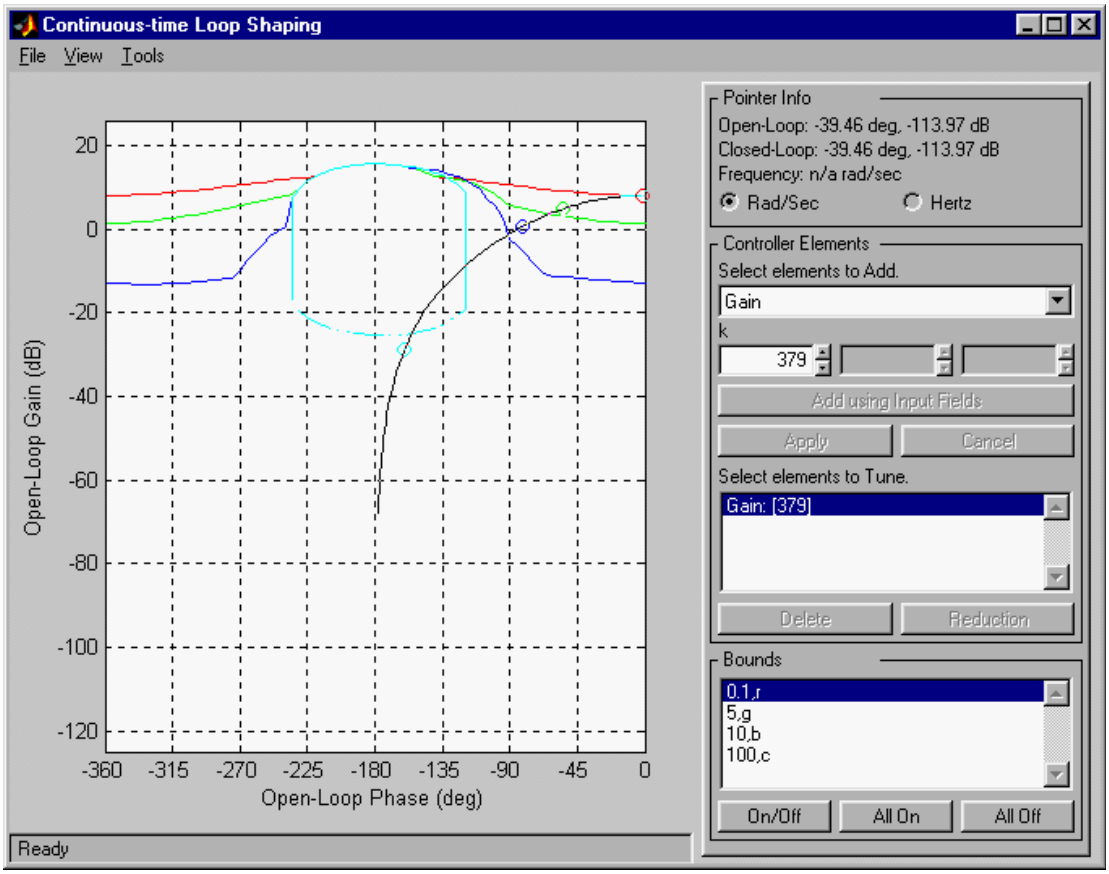


Figure 22: Open-loop plot with a controller gain of 379.

Next, you can observe that a phase lead is necessary because the nominal loop lies inside the margin bounds at higher frequencies. So let us add a real zero.

Using graphics protocol: Select real zero in **Select Elements to Add** pull-down menu, select the point on the response plot where the frequency is $\omega = 61$ (the program shows the corresponding nominal loop frequency in the upper-right hand corner of the screen). In our case, it is approximately where the loop crosses the -20 *dB* horizontal line. Once you select that point by pressing the left mouse button a (red) marker will show up at that point. Now, drag the selected point to the new location of (-100°,-20 *dB*) (approximately 50° to the right). We are adding more phase lead than may seem necessary because we anticipate adding more poles to make the controller at least proper. Again, after the zero is implemented graphically, you can modify the response by re-selecting and dragging the displayed (red) marker. Finally, you should end up with a zero value of $z = 42$ and the present loop as shown below. If this is not your value, perform this using the above text protocol. The result is shown in Fig. 23.

If you wish to extend the frequency response plot beyond the present frequency vector, select **Tools|Frequency...** and then enter values for lower and upper limits of frequency and the number of points. Take some time now to try this procedure.

Using text protocol: select Real Zero in **Select Elements to Add,** enter 42 at the **zero** box below, press **Add Using Input Fields** then press **Apply**.

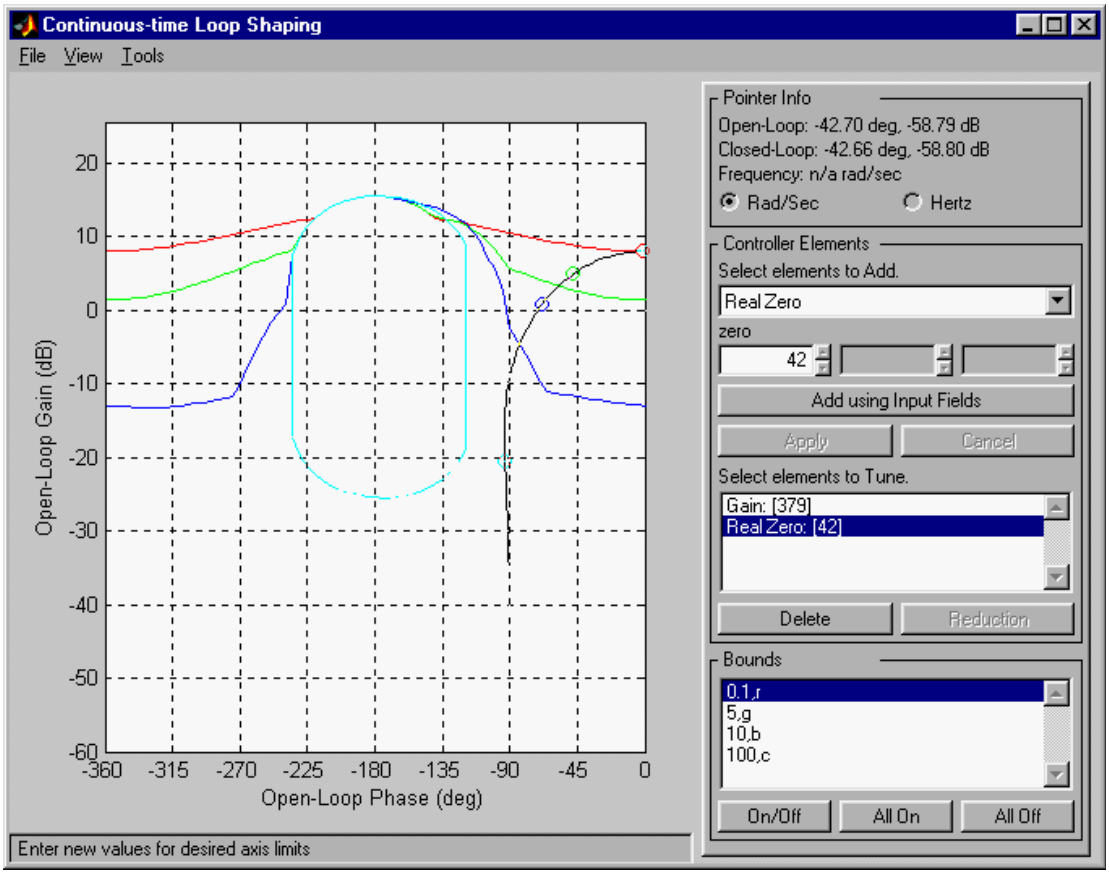


Figure 23: Open-loop plot with an added controller zero at $z = 42$.

The final step involves shaping the high-frequency response of the nominal loop (and of the controller) with the goal of dropping its magnitude as fast as possible to satisfy a roll-off constraint. For this step, you must first decide on the complexity of the controller based on physical constraints, e.g., speed of DSP board. Suppose you decided on a relative degree of 1 (i.e., degree of denominator minus degree of numerator). A second-order pole best suits this objective.

Using graphics protocol: Select Complex Pole in the **Select Elements to Add** pull-down menu, select a point on the nominal loop to be shifted to the left with the second-order term. The program will compute the required damping ratio and natural frequency. Note that when using a second-order term it may not always be possible to achieve both magnitude and phase changes as desired (only stable terms can be generated in a *graphics protocol*). Again, after the second-order term is implemented graphically, you can modify both the natural frequency and the damping ratio by re-selecting the (red) marker and dragging it to a new location.

Using text protocol: Select Complex Pole in the **Select Elements to Add** pull-down menu, enter a damping ratio of `0.5` at the **zeta** box and a natural frequency of `250` at the **wn** box below, press **Add Using Input Fields** and finally press **Apply**. A damping ratio of 0.5 is an optimal choice between minimal oscillations and maximal magnitude/phase slope. The result is shown in Fig. 24.

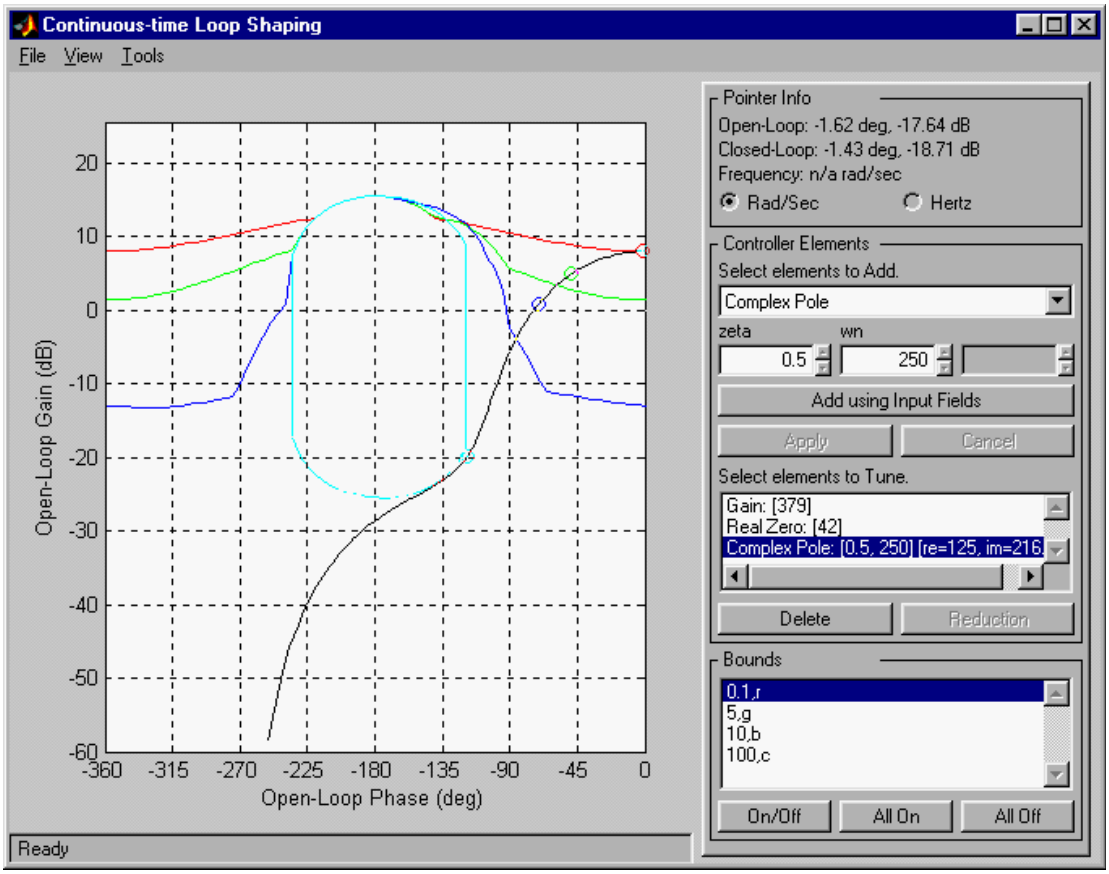


Figure 24: The plot with the new second order (w_n=250 and zeta=0.5).

Now, use the sliders to fine-tune the value of the natural frequency so that the nominal loop passes just below the robust margin's high frequency bound (at $\omega = 100$). This guarantees that it will not violate the margin bounds higher frequencies due to the invariant shape of the template. The final result is shown in Fig. 25 with $w_n = 247$ (use the edit option to enter this value if necessary).

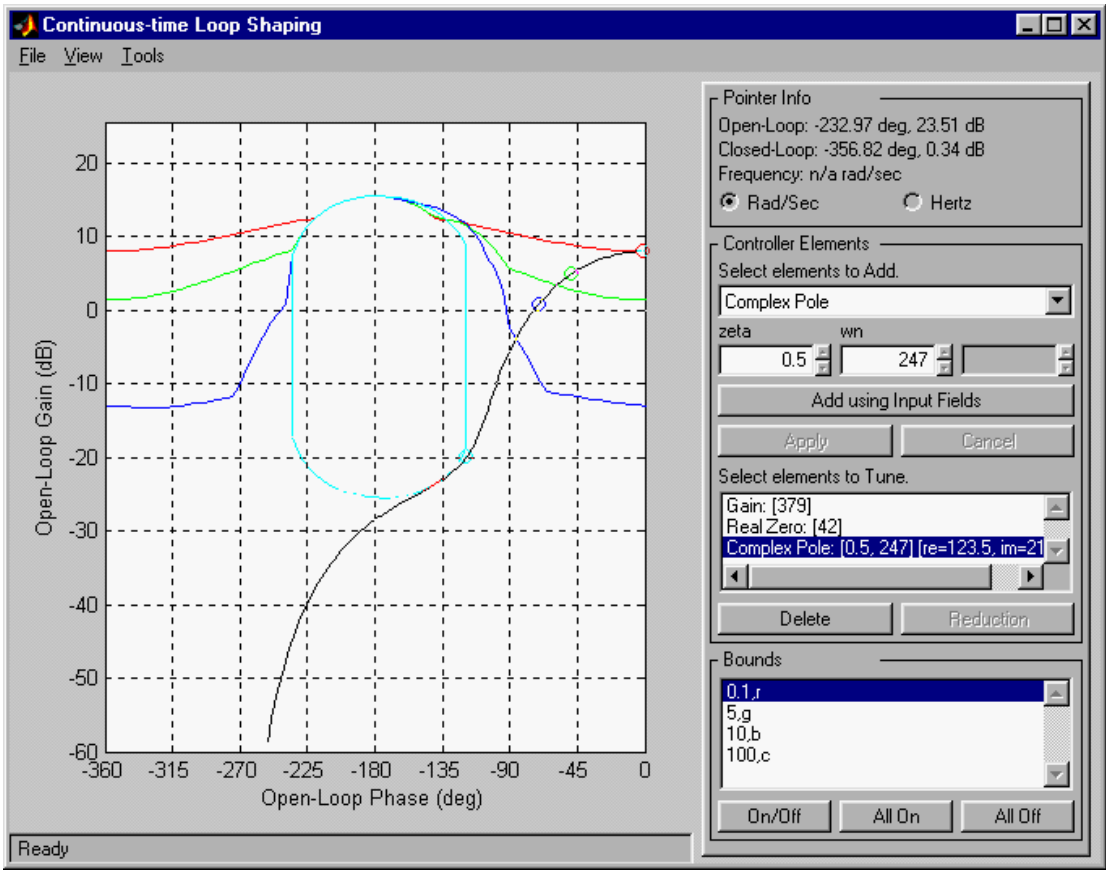


Figure 25: Final design with a tuned second order (w_n=247).

The final interactively designed controller is given by

$$G(s)=\frac{379\left(\frac{s}{42}+1\right)}{\frac{s^2}{247^2}+\frac{s}{247}+1}.$$

We recommend that you verify that the bottom part of the high frequency margin bound is not violated by the nominal loop. This is important because, in that region on a Nichols chart, relatively small open-loop *dB* differences result in rather large closed-loop *dB* variations. To check, zoom in to that region by selecting one corner and dragging the "rubber-band box" to define new axis limits. The result of this zoom is shown in Fig. 26.

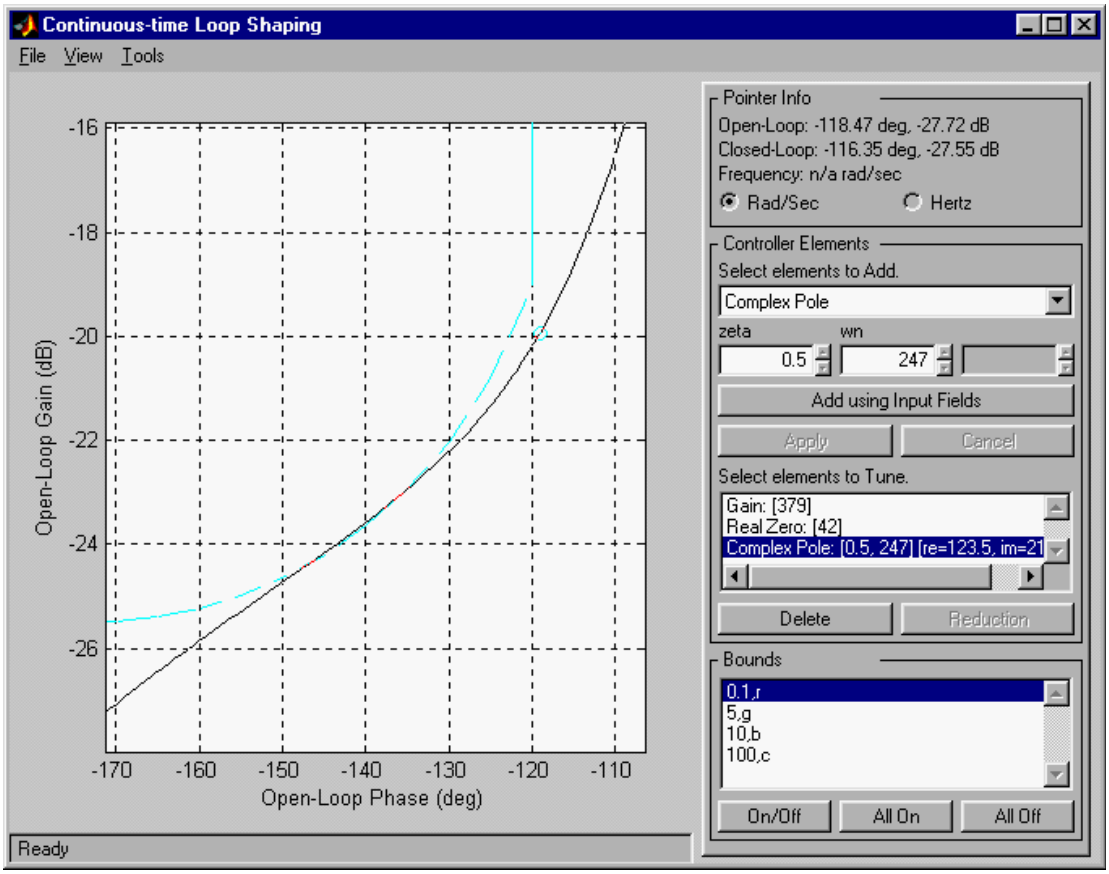


Figure 26: Zoomed-in view of the final design.

If you are not satisfied with the design, you can continue with loop shaping. You will discover that by adding more terms to the controller, you can place the nominal loop closer and closer to its bounds. However, this comes at a cost of increased complexity of the controller. In general, there are infinitely many nominal loops that can meet the bounds (if at least one solution exists). They differ in their complexity and bandwidth (the example file includes two such solutions). In addition to the above terms used during loop shaping, the program includes lead/lag, integrators, proper 2nd order (denoted in QFT Toolbox as super 2nd or 2/2), notch and complex lead terms (see Design Elements section in Chapter 6 for more details). The example file contains another controller that solves the feedback problem

$$G(s)=\frac{379\left(\frac{s}{42}+1\right)}{\frac{s}{165}+1}.$$

This controller is only proper and is far less attractive in practice when compared to the previous controller in terms of sensor noise amplification and robustness against high frequency unmodeled dynamics. The measure of optimality in QFT is to design a controller that meets its bounds and has the minimal high frequency gain for the given controller's complexity.

In the present design, robust stability (see Robust Stability) is achieved since the nominal loop does not violate the robust margin bounds and does not cross the ray $\mathbf{R_0}=\{(\phi,r):\ \phi=-180^\circ, r>0\ \text{dB}\}$. Therefore, we have completed the design.

Hint: If the designed controller appears to have too high an order (i.e., you were a bit too quick with the mouse...), you can perform on-line model order reduction. The reduction algorithms were specialized to exploit our specific knowledge of the controller's poles and zeros. A detailed description of the available model reduction options can be found in The Interactive Design Environment (IDE).

The IDE functions do not have any output arguments. Within the IDE functions, the design can be stored in a specialized MAT-file or even placed in the workspace. In the MATLAB® command line you can retrieve any design using the function `getqft` (see Reference Chapter for details).

Analysis

When you complete a QFT design, as we have just done, you should analyze the closed-loop response at frequencies other than those used for computing bounds. Typically, this is done at the same frequency array used in loop shaping by invoking the function

```
chksiso(ptype,w,wbd,Ws,P,R,G,H,F)
```

We recommend that you use a denser plant template than the one used in computing bounds. So let us increase the number of plant cases along the boundary from 40 to 100:

```
c = 1; k = 10; b = 20;
for a = linspace(1,5,25),
 P(1,1,c) = tf(k,[1,a+b,a*b]);  c = c + 1;
end
k = 1; b = 30;
for a = linspace(1,5,25),
 P(1,1,c) = tf(k,[1,a+b,a*b]);  c = c + 1;
end
b = 30; a = 5;
for k = linspace(1,10,25),
 P(1,1,c) = tf(k,[1,a+b,a*b]);  c = c + 1;
end
b = 20; a = 1;
for k = linspace(1,10,25),
 P(1,1,c) = tf(k,[1,a+b,a*b]);  c = c + 1;
end
```

In certain cases, e.g., plants with resonances, you may also want to increase the number of frequencies in `wl`. To analyze closed-loop margin (`ptype = 1`), invoke the above function that results in a plot of the worst (over the uncertainty) closed-loop response magnitude versus the specification. In particular, invoking

```
chksiso(1,wl,W1,P,R,G);
```

results in the plot shown in Fig. 27.

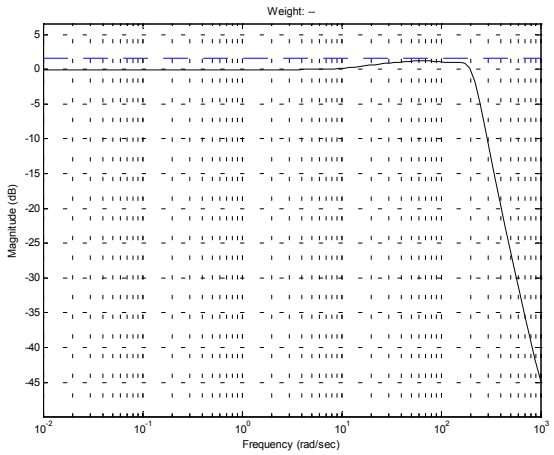


Figure 27: Analysis of robust margin problem.

Hint: You can easily zoom in to any frequency bands where the closed-loop transfer function appears to be very close to its specification.

To analyze the output disturbance rejection problem, invoke the following

```
ind=find(wl<=10);
chksiso(2,wl(ind),W2,P,R,G);
```

which results in the plot shown in Fig. 28.

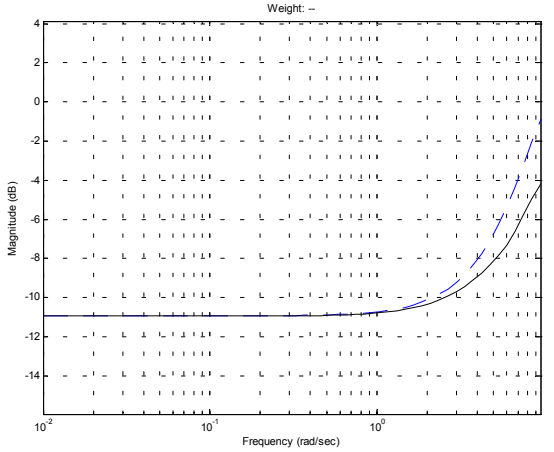


Figure 28: Analysis of robust output disturbance problem.

To analyze the input disturbance rejection problem, using `ptype = 3` invoke the following

```
chksiso(3,wl(ind),W3,P,R,G);
```

which results in the plot shown in Fig. 29.

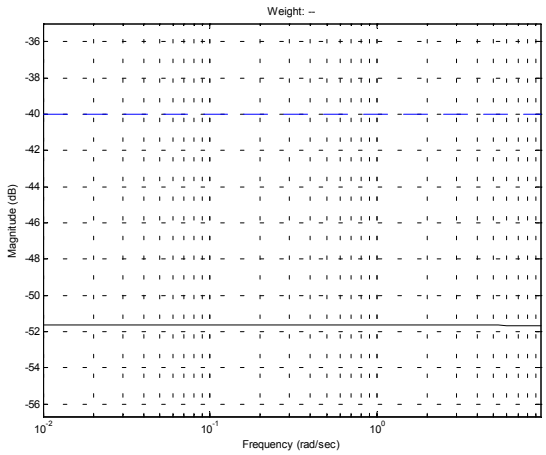


Figure 29: Analysis of robust input disturbance problem.

If the design fails at some frequency (or over a frequency band), you may decide to compute the corresponding bound for the specific problem at that frequency, then re-design the nominal loop. Alternatively, you can skip the computation of a new bound and directly go into loop shaping. For engineering purpose, adding some gain and/or phase to the loop at that frequency may be sufficient.

Design (Pre-Filter Shaping)

If the feedback system involves tracking of reference signals, then clearly your best choice would be to use a pre-filter $F(s)$ in addition to the controller $G(s)$ embedded within the closed-loop system. Examples Example 2: 2-DOF Design and Example 13: 2 DOF Design (Discrete-Time) include such a design (`qftex2.m` and `qftex13.m`).

The Toolbox includes an interactive design environment for pre-filter shaping. A detailed description of the interactive design environment can be found in Chapter 6.

Pre-filter shaping is done using pfshape with the generic call

```
pfshape(ptype,w,Ws,P,R,G,H,F0)
```

- The input arguments are the same as used in computing bounds. The argument `ptype` defines the particular closed-loop transfer function of interest, e.g., sensitivity or complementary sensitivity. The input arguments are:
- `w` is the frequency array where margin bounds are to be computed
- `Ws` is an upper and lower magnitude bound on the closed-loop transfer function
- `P` is the frequency response set of the plant
- `R` is the disk radius in a multiplicative uncertain plant
- `G` and `H` are the frequency responses of the other functions in the loop
- `F0` is an initial pre-filter.

To illustrate pre-filter design, suppose that for the problem considered in the last section there is an additional tracking specification of the form

$$\left| F \cdot \frac{PG}{1+PG} \right| \le 1.1, \text{ for all } P \in \mathcal{P} .$$

This is a two-degree-of -freedom design. In such cases which involve pre-filters, the first step would be to close the loop by designing the controller $G(s)$ and assuming $F(s) = 1$. Once $G(s)$ is give, we can focus on design of $F(s)$ to meet tracking specifications. For our example, the pre-filter design environment is initiated by invoking

```
pfshape(1,wl,1.1,P,0,G)
```

Invoking the above function results in the plot shown in Fig. 30.

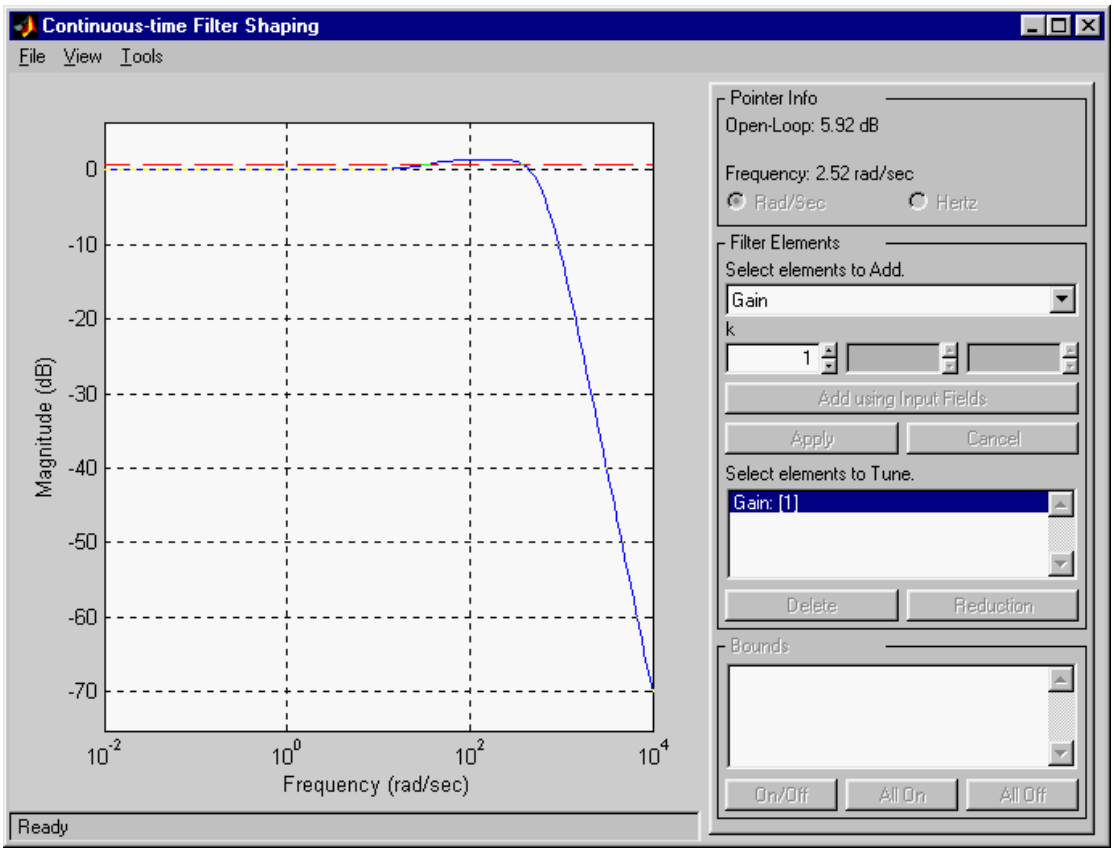


Figure 30: Closed-loop tracking response with $F(s)$=1.

Shown in Fig. 30 are the worst-case closed-loop response over plant uncertainty and the specification (dashed line). The pre-filter design environment uses the same commands as in loop shaping, so the earlier description on loop shaping holds here too. Design of the pre-filter is usually a trivial task. In this example, a pole at 140 will do the job: select Real Pole from the **Select Elements to Add** pull-down menu, enter 140, press **Add Using Input Fields**, and press **Apply**. It would be simpler to add the pole directly using the *graphics protocol*. This addition results in Fig. 31.

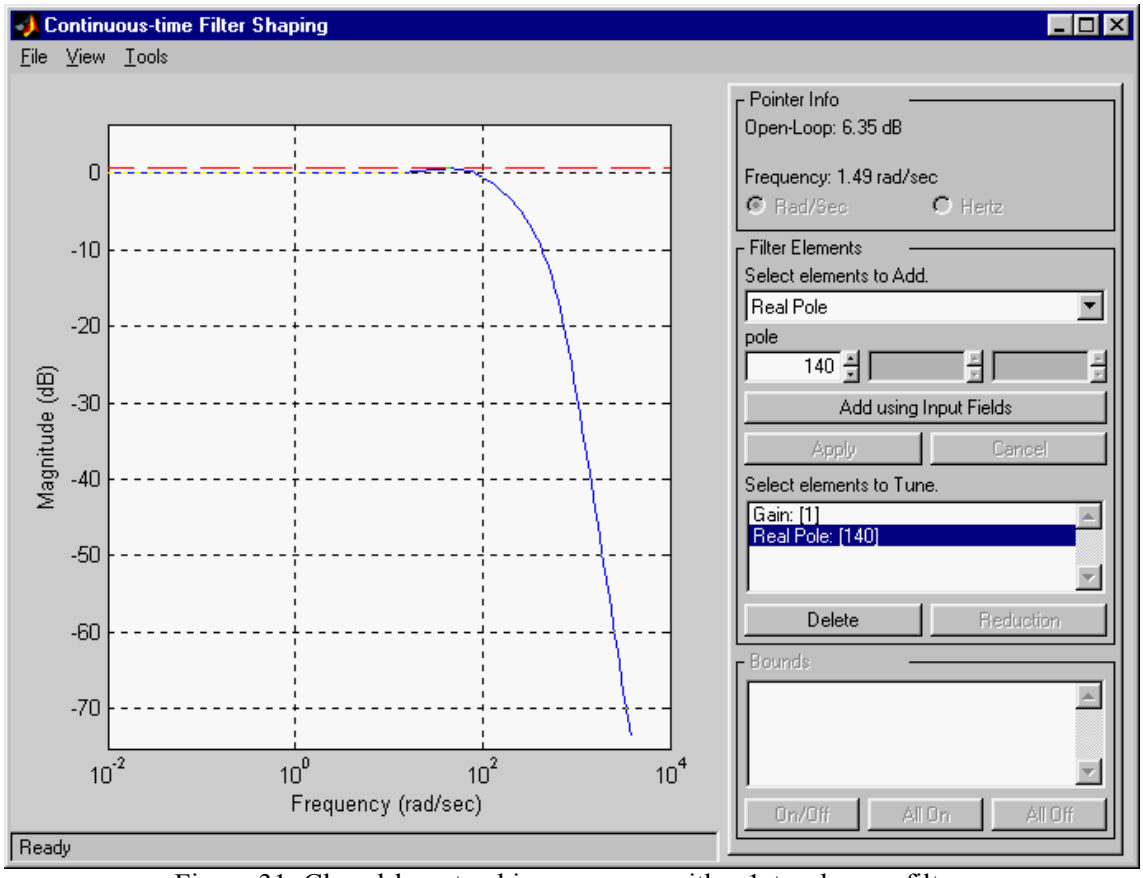


Figure 31: Closed-loop tracking response with a 1st order pre-filter.

Discrete-Time

The discussion on the use of the Toolbox for continuous-time systems extends naturally to discrete-time systems. (The Laplace variable *s* is replaced with $z = e^{j\omega t_s}$, t_s = sampling time, and the frequency band of interest is limited to $\omega \in \left[0, \pi/t_s\right]$ rad/sec). With the exception of the robust stability bounds and the loop shaping procedure, all other topics such as templates, bounds and analysis are only summarized here. You should first read the previous section on continuous-time systems before you proceed with this section.

The general block diagram is the same one used in continuous-time system and shown again in Fig. 32 below.

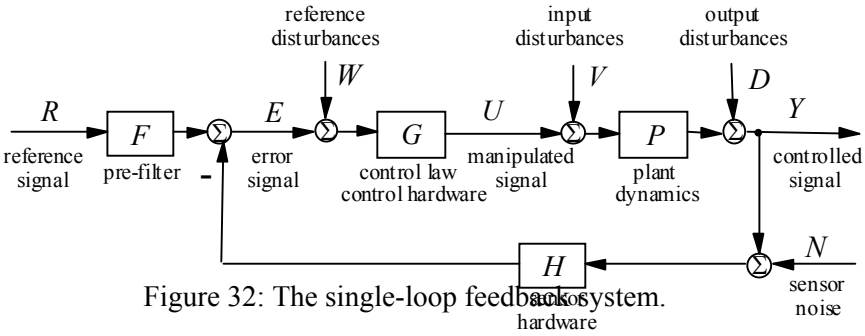


Figure 32: The single-loop feedback system.

Templates

The Toolbox allows mixed uncertainty model for the plant *P* (parametric and non-parametric), and it allows for one additional parametric uncertain transfer function in the loop (*G* or *H*). The problem of accurately defining the plant templates is even more difficult in sampled-data systems where the continuous-time plant is parametric uncertain.

Robust Stability (Margins) Bounds

Bounds are computed based on the condition for robust stability in the discrete-time case (see Robust Stability) that for the system shown in Fig. 32 is given by

$$\left|\frac{PGH}{1+PGH}(z)\right| < \infty, \text{ for all } P \in \partial\mathcal{P},\ z = e^{j\omega t_s},\ \omega \in \left[0, \frac{\pi}{t_s}\right]$$

where $\partial\mathcal{P}$ denotes the exterior boundary of the template of the discrete-time family $\mathcal{P}$. The robust margin condition is given by

$$\left|\frac{PGH}{1+PGH}(z)\right| < \mu > 1, \text{ for all } P \in \partial\mathcal{P},\ z = e^{j\omega t_s},\ \omega \in \left[0, \frac{\pi}{t_s}\right].$$

Because a discrete-time QFT design is similar to that in continuous-time, please refer to that section for a detailed discussion on the various aspects of the procedure.

Bounds are computed with the functions sisobnds and genbnds as done in continuous-time systems.

Robust Performance Bounds

The procedure for computing discrete-time bounds for robust performance problems is the same as the one explained previously for continuous-time systems.

Bounds are computed with the functions sisobnds and genbnds as done in continuous-time systems.

Design (Loop Shaping)

Once discrete-time bounds are computed, you are ready for next design of loop shaping a nominal loop function to satisfy its bounds. The nominal loop would have to satisfy the worst case of all bounds. This Toolbox includes The Interactive Design Environment (IDE) for loop shaping.

Nominal loop shaping is done using `lpshape` with the generic call (LTI models must be discrete-time)

```
lpshape(wl,ubdb,L0,C0,phs);
```

This function allows control design directly in the z-domain. That is, you can work with the equivalent of real poles and zeros, integrators and lead/lag terms in the variable z. It is well known that z-domain loop shaping is an unfamiliar topic for most engineers. However, in applications where the system bandwidth is relatively close to half the sampling frequency, there is no substitution to direct z-domain design due to the distortion introduced in the $s \rightarrow z$ and $w \rightarrow z$ mappings.

Design (Pre-Filter Shaping)

If the feedback system involves tracking signals, then clearly your best choice would be to use a pre-filter $F(z)$ in addition to the controller $G(z)$ embedded within the closed-loop system. Example 12 in Chapter 5 includes such a design.

The Toolbox includes The Interactive Design Environment (IDE) for pre-filter shaping.

Pre-filter shaping is done using pfshape with the generic call (LTI models must be discrete-time)

```
pfshape(ptype,w,Ws,P,R,G,H,F0)
```

4 Using the Nichols Chart

The Nichols chart

The Nichols chart is the domain of choice for QFT design. If you are used to designing with Bode plots, this chapter is aimed at demonstrating that Bode and Nichols plot designs are similar. Your insight and experience with classical loop shaping on Bode plots can be easily ported to the Nichols chart. It should be emphasized here that in contrast to the original intent of Nichols plots, in the QFT Toolbox we do not use closed-loop grids by default (they can be turned on however). The reason is that the Toolbox mathematically translates closed-loop specifications into open-loop bounds.

The Nichols chart represents complex numbers in terms of their magnitudes and phases. Each complex number, s, has a Cartesian representation (x,y) and a polar representation (r,ϕ). The coordinates of the Nichols chart are $(\phi, 20\log(r))$. The horizontal coordinate, ϕ, typically ranges between -360° and 0°, while the vertical coordinate, $20\log(r)$, ranges theoretically from $-\infty$ *dB* to $+\infty$ *dB*. The Nichols chart used in practice is naturally limited to a finite range of magnitudes.

The phase of a Nyquist plot usually extends outside the (-360°,0°] range. If one wishes to retain continuity of the Nichols plot, one has to extend it periodically in the phase coordinate. A Nyquist curve winding k times around the origin would be transformed this way into a continuous (but not closed!) curve drawn along a scroll of at least k Nichols sheets. This curve will be called the multiple-sheeted Nichols plot. Analysis and design of feedback systems can be equally performed using a single-sheeted plot or a multiple-sheeted plot (for connected templates only). The decision to use one or the other is a matter of convenience only. The Toolbox offers one, multiples and fractions of the Nichols chart. A typical single-sheeted Nichols chart, shown in Fig. 33 (*with closed-loop grid), is equally applicable to continuous-time and to discrete-time systems. The Control System* Toolbox function `ngrid` draws Nichols chart closed-loop grids.

The horizontal and vertical coordinates are used for the phase (degrees) and the magnitude (*dB*), of the open-loop function $L(s)$. The phase (degrees) and magnitude (*dB*) of the closed-loop transfer function, $L(s)/(1+L(s))$, are the curves shown inside the chart. As mentioned earlier, in this Toolbox we only work with the open-loop coordinates while the closed-loop coordinates are replaced by the use of bounds.

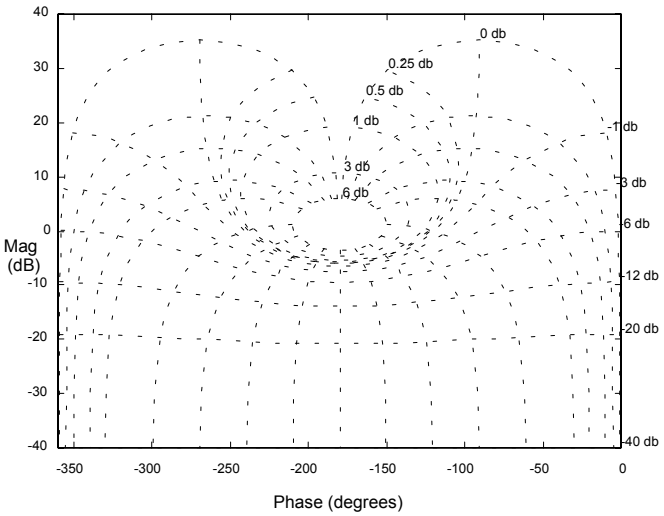


Figure 33: The Nichols chart.

Continuous-Time

Stability

Consider the linear, time-invariant, continuous-time, single-loop feedback system shown in Fig. 34.

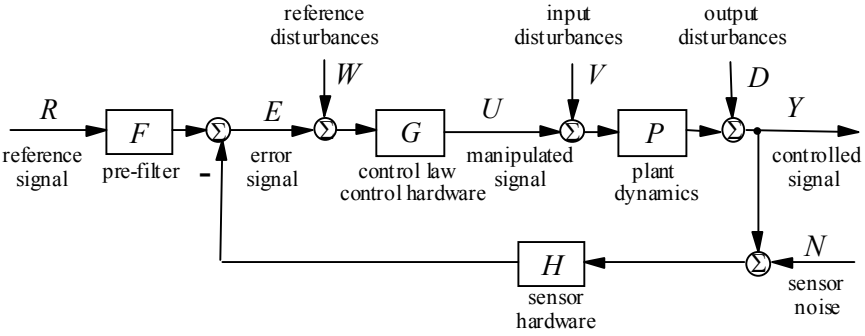


Figure 34: The single-loop feedback system.

The loop transmission (open-loop function), $L(s) = P(s)G(s)H(s)$, is assumed to be a product of a rational (proper or strictly proper) function and a pure time delay. We assume that no unstable pole/zero cancellations take place in $L(s)$. A standard Nyquist contour, with right jω-axis indentations as necessary to account for imaginary axis poles of $L(s)$ is shown in Fig. 35.

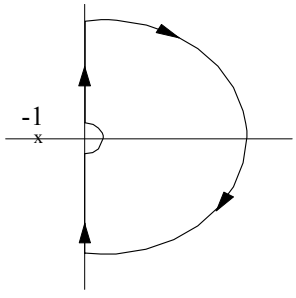


Figure 35: The continuous-time Nyquist contour.

Definition. The Nyquist plot of $L(s)$ is said to have a crossing if it intersects the negative part of the real axis, Re[$L(s)$]<-1. The sign of the crossing is either positive or negative, depending on the direction of the plot at the crossing point. Crossings and corresponding signs in both the complex plane and the Nichols chart are illustrated in Fig. 36.

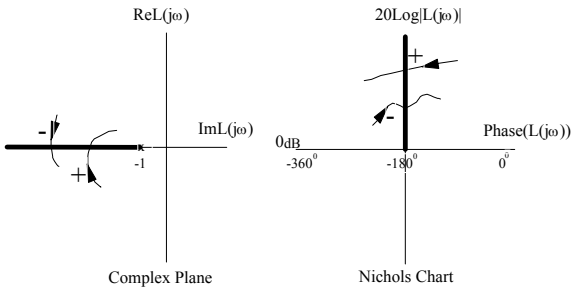


Figure 36: The notion of crossing.

The following is the Nichols chart stability criterion used in QFT fixed plants [4]. Let n denote the total number (counting multiplicity) of the unstable poles of $L(s)$ inside the Nyquist contour.

Criterion 1: The feedback system in Fig. 34 is stable if:

- The single-sheeted Nichols plot of $L(s)$ does not intersect the point $\mathbf{q}$:=(-180°, 0dB), and the net sum of its crossings of the ray $\mathbf{R_0}$:={(ϕ,r): ϕ=-180°, r>0dB} is equal to n; or
- The multiple-sheeted Nichols plot of $L(s)$ does not intersect any of the points $(2k+1)\mathbf{q}$, $k = 0,1,2,\ldots$, and the net sum of its crossings of the rays $\mathbf{R_0} + 2k\mathbf{q}$ is equal to n.

The number of crossings is equivalent to the number of encirclements of the critical point (-1,0) by the Nyquist plot of $L(s)$. Therefore, the key relation is $Z = N+n$, where Z denotes total number of closed-loop poles inside the Nyquist contour, N denotes number of crossings and n denotes total number of the poles of $L(s)$ inside the Nyquist contour. Two examples of Nichols plots and stability analysis are presented below [4].

Example 1: Consider a unity feedback system that has the following stable open-loop function

$$L(s) = \frac{k}{(s+1)(s+5)(s+10)},\ k > 0.$$

The Nichols plot is shown in Fig. 37 on a multiple-sheeted chart ($k = 3000$). Two positive crossings of the two rays, $\mathbf{R_0} := \{(\phi, r): \phi = -180^\circ,\ r > 0\text{dB}\}$ and $\mathbf{R}_1 := \{(\phi, r): \phi = 180^\circ,\ r > 0\text{dB}\}$, can be observed.

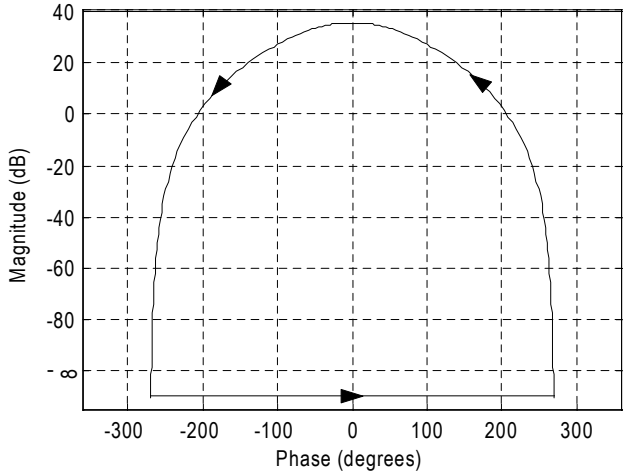


Figure 37: Nichols plot of Example 1 on a multiple-sheeted chart.

From Criterion 1, since the system is open-loop stable we must reduce the gain (i.e., shift the plot down vertically) to eliminate any crossings. If we reduce the gain by a factor of 3 (approximately 9.5 *dB*), the plot will be just below the rays $\mathbf{R_0}$ and $\mathbf{R_1}$. Hence, we conclude that the closed-loop system is stable if $k<1000$. Note that for strictly proper functions, $L(s) = 0$ at the semi-infinite circle portion of the Nyquist contour. On the Nichols chart, this is represented by a horizontal segment at $-\infty$ *dB* starting at $L(j\infty)$ and ending at $L(-j\infty)$. The width of the segment is equal to (no. of poles - no. of zeros) x 360°.

The same analysis can be done using the single-sheeted Nichols chart as shown in Fig. 38. Any segment of the plot can be shifted horizontally by $\pm j360°$, $j=0,\pm1,\pm2,....$.

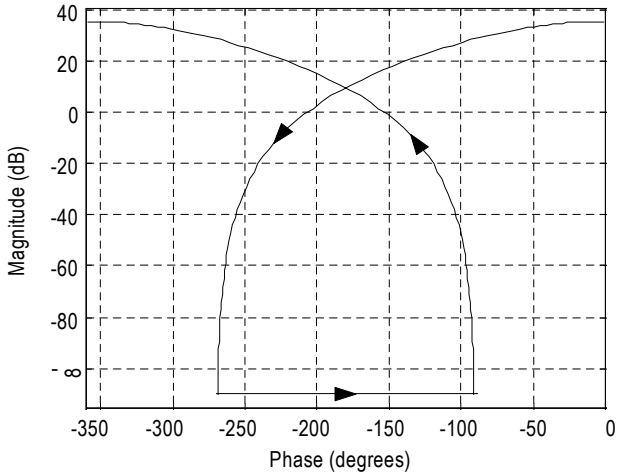


Figure 38: Nichols plot of Example 1 on a single-sheeted chart.

In control design, it is customary to plot only half Nyquist plots (i.e., the Bode plot), taking advantage of conjugancy of transfer functions with real coefficients. Conjugancy can also be exploited with Nichols plots. In this example, the half-plot shown in Fig. 39 indicates a single positive crossing or equivalently a total of two positive crossings for the full plot.

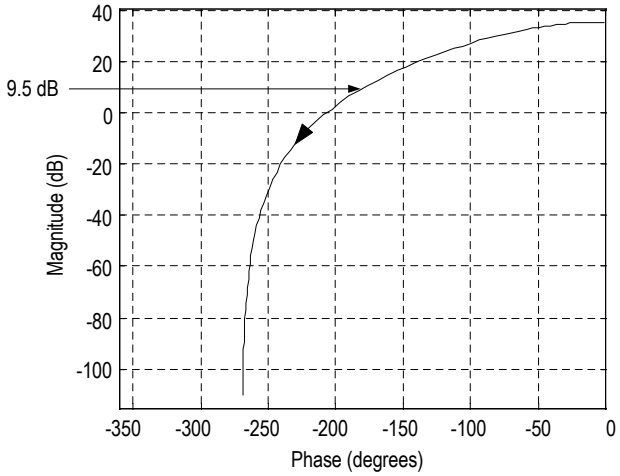


Figure 39: Half Nichols plot of Example 1.

Special care must be taken when the loop has integrators, as in the following example.

Example 2: Consider a unity feedback system that has the following open-loop function

$$L(s)=\frac{k}{s(s+1)(s+10)},\ k>0.$$

The single-sheeted Nichols plot is shown in Fig. 40 with $k = 1$.

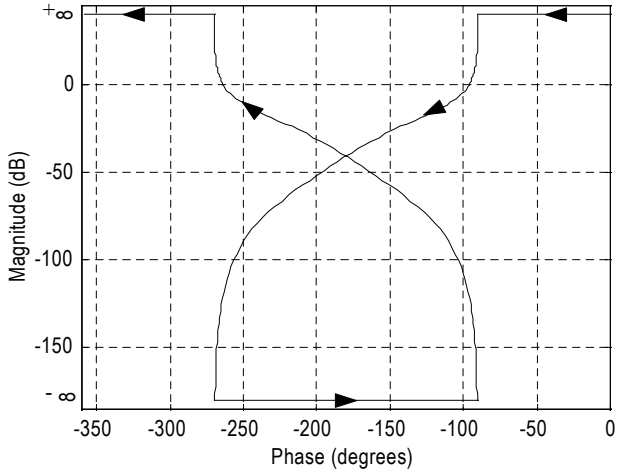


Figure 40: Nichols plot of Example 2 on a single-sheeted chart.

Note that unlike Nyquist plots, Nichols plots may not be closed. In this example, if the gain is increased, the plot will eventually cross the ray $\mathbf{R_0} := \{(\phi, r): \phi = -180^\circ,\ r > 0\text{dB}\}$ twice. This happens when $k = 100$ (40 *dB*). Hence, the system is closed-loop stable if $k<100$. For $k>100$, there are two positive crossings or two unstable closed-loop poles. You can also observe a segment of the plot at $+\infty$ *dB*. The Nyquist plot has a semi-infinite circle for each integrator (or other $j\omega$-axis poles) in $L(s)$ which translates into segments at $+\infty$ *dB* on a Nichols chart. Specifically, a Nichols plot will have a 180° horizontal segment at $+\infty$ *dB* for each $j\omega$-axis pole in $L(s)$. For practical reasons such segments are rarely shown as part of the plot, but must be considered in stability analysis. There is a rather simple rule for drawing (or visualizing) such segments on a Nichols chart: first draw the basic (Bode) plot from $\omega \to 0^+$ up to very large frequency, then

connect to it a 180°-wide horizontal segment (for each integrator) such that left edge of the segment ends at the point $L(j0^+)$. In this example we have a single integrator implying a segment 180°-wide which should be connected to $L(j\omega)$ at -90° with a very large magnitude (approaching $+\infty$ *dB*) at $\omega\to 0^+$. This segment should then start at +90° and end at -90°. However, in both full and half plots the phase axis does not include positive phases. Hence, we start the segment at -270° and continue toward -360°, then jump to 0° and continue to -90°, totaling 180° (one integrator). Note that you need not physically draw these segments; it suffices to attach *imaginary* segments to the actual plot when counting crossings for stability analysis (Fig. 41).

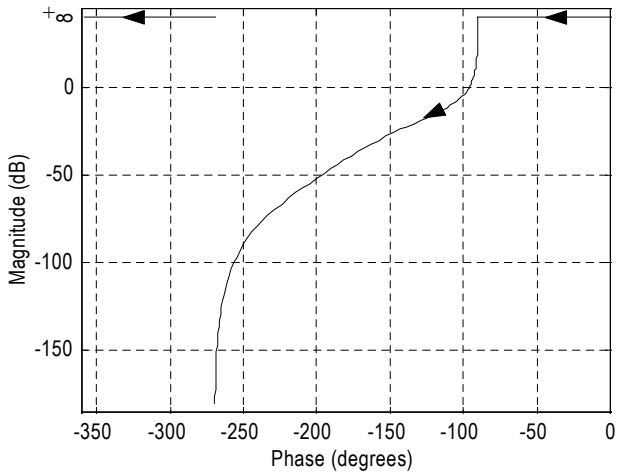


Figure 41: Half Nichols plot of Example 2.

In certain cases with poles on the jω-axis, the plot may appear to be tangential to the ray $\mathbf{R_0} := \{(\phi, r): \phi = -180^\circ,\ r > 0\text{dB}\}$ in which case it is not clear how to count crossings. For example, consider the open-loop function

$$L(s) = \frac{k}{s^2(s+1)},\ k > 0\,.$$

As $\omega\to 0$, the phase of $L(j\omega)$ approaches -180° and the magnitude approaches ∞. To correctly count any crossings, you need to realize that in fact $L(j\omega)$ does not lie on the ray $\mathbf{R_0}$ at infinity, it is only tangential to it. This can be observed by taking the partial fraction in the limit as $\omega\to 0$

$$L(j\omega) = -\frac{k}{\omega^2} + j\frac{k}{\omega} + \frac{k}{j\omega+1} \xrightarrow{\omega\to 0} -\frac{k}{\omega^2} + j\frac{k}{\omega}\,.$$

Although the real part is at $-\infty$, there is always a non-zero imaginary part as well (of course it is much smaller in magnitude compared to the real part). Hence, the plot does not lie on the ray $\mathbf{R_0}$ as $\omega\to 0$ and it is possible to count crossing. Another way to interpret the type of crossing is by figuring the phase at $\omega\to 0^+$.

Robust Stability

In many physical situations, the actual plant dynamics are known to belong to a set (family) of plants $\mathcal{P}$. The idea of robust stability in QFT amounts to checking stability using one nominal loop $L_0(s) = P_0(s)G_0(s)H_0(s)$, where $P_0(s) \in \mathcal{P}$ is termed the nominal plant, and then demonstrating stability

of the whole set P by some argument involving the nature of $\mathcal{P}$. This property is commonly referred to as *robust stability*.

At each point, jω, on the Nyquist contour, the responses of $L(j\omega)$ fill in a neighborhood of the nominal response $L_0(j\omega)$. The collection of all the responses of the plant $P(j\omega)$ is called a *template*. Assuming the controller to be fixed, the shape of the collection of all the responses of $L(j\omega)$ is the same as that of the template $P(j\omega)$. The shape of the template can range from a non-connected region to a convex region (see Fig. 42).

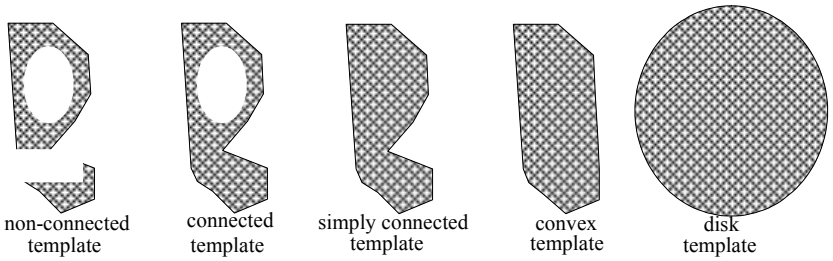


Figure 42: Various templates.

For design purposes, one typically enlarges the template into a simply connected region (roughly speaking, it is made of a single "piece" and has no holes). Another possibility is to define the template as the convex hull of the region (in a convex set any two points in the set can be connected via a line that lies entirely in the set). The most conservative approach would be to turn the region into a disk (non-parametric model).

As we traverse the Nyquist contour, the union of these templates is called the Nichols envelope. Note that templates unify the way QFT treats uncertainty since parametric, non-parametric or mixed uncertainty plant models all have a similar frequency response representation. If your template has holes, the Toolbox algorithms will automatically "fill in" and assume that no holes exist.

Naturally, you may ask yourself what models of uncertain plants generate the templates shown in Fig. 42. Because the focus in QFT is on engineering design, we will not attempt here to provide a complete answer that covers all possible pathological cases, nor is it claimed that QFT is applicable to all classes of systems. However, if the plant you are working with represents a realistic system then it most likely fits the following mold: (1) the plant is strictly proper with the possibility of a pure time delay, and (2) the plant model is parametric and/or non-parametric uncertainty where its numerator and denominator sets or pole and zero sets depend continuously on the uncertain parameters set. If your plant is non-rational or has some other exotic structure, QFT may still be applicable, but you must proceed with care. Essentially, what we want to avoid is a template composed of disjointed parts (i.e., non-connected), though with proper care it may be possible to design for robustness with QFT even with non-connected templates.

The following is the Nichols chart robust stability criterion [4-6, 23] used in QFT. The loop transfer function L is assumed to belong to a set L, which has the uncertainty form described in Chapter 2. In addition to the trivial assumption of no unstable (including $j\omega$-axis) pole/zero cancellation in any $L(s)$ in the set, the criterion requires either one of the following groups of conditions. The first group is: (1) $L(s)$ is strictly proper, (2) the uncertain parameters belong to a compact and simple connected set, (3) the coefficients of the numerator and denominator of $L(s)$ depend continuously on the uncertain parameters, and (4) the coefficients of the highest degree s terms in the numerator and denominator of $L(s)$ cannot vanish. The second group is: (1) at each fixed frequency, the responses of all $L(j\omega)$ form a convex set in the complex plane, and (2) the number of unstable poles in $L(s)$ is fixed.

Criterion 2: Consider the feedback system shown in Fig. 34. Assume that the uncertain set $\mathcal{L}$ satisfies one of the above groups of conditions. Let $L_0(s) \in \mathcal{L}$ denote the nominal plant. The feedback system is robust stable if:

- The nominal closed-loop system corresponding to $L_0(s)$ is stable and the single-sheeted Nichols envelope does not intersect the point **q**; or
- The nominal closed-loop system corresponding to $L_0(s)$ is stable and the multiple-sheeted Nichols envelope does not intersect any of the points $(2k+1)\mathbf{q}$, $k=0, 1, 2, ...$

The condition that the single-sheeted Nichols envelope does not intersect the point **q** is the same as requiring that $1+L(j\omega)\neq 0$ for all $L(s) \in \mathcal{L}$ and for all frequencies on the $j\omega$-axis.

Discrete-Time

Stability

Consider the linear, time-invariant, discrete-time, single-loop feedback system shown in Fig. 34. The loop transmission is denoted by $L(z) = P(z)G(z)H(z)$, and $1+L(z)$ is assumed to be proper. In addition, we assume that no unstable pole/zero cancellations take place in $L(z)$. Let $z = e^{j\omega t_s}$, $\omega \in \left[0, \pi/t_s\right]$ (t_s is the sampling time). The standard Nyquist contour, with unit circle indentations as necessary to account for poles of $L(z)$ on the unit circle, is shown in Fig. 43. Let n denote the total number (counting multiplicity) of the poles of $L(z)$ outside the unit circle.

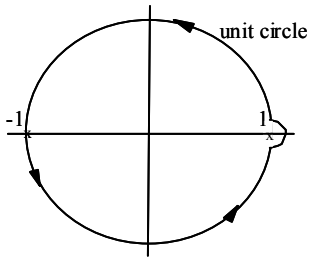


Figure 43: The discrete-time Nyquist contour.

Definition. The Nyquist plot of $L(z)$ is said to have a crossing if it intersects the negative real axis, $ReL(z)<-1$. The sign of the crossing is either positive or negative, depending on the plot's direction at the crossing point. Here, crossings and signs in both complex plane and Nichols chart have the same meaning as in the continuous-time case.

Criterion 3: The feedback system in Fig. 34 is stable if [4,7].
- The one-sheeted Nichols plot of $L(z)$ does not intersect the point $\mathbf{q}$:=(-180°, 0dB), and the net sum of its crossings of the ray $\mathbf{R_0}$:={(ϕ,r): ϕ=-180°, r>0dB} is equal to n; or
- The multiple-sheeted Nichols plot of $L(z)$ does not intersect any of the points $(2k+1)\mathbf{q}$, $k = 0,1,2,\ldots$, and the net sum of its crossings of the rays $\mathbf{R_0} + 2k\mathbf{q}$ is equal to n.

Two examples of nominal Nichols stability analysis are presented below.

Example 3: Consider a unity feedback system that has the following open-loop function

$$L(z) = \frac{k(z+0.9)}{(z-1)(z-0.7)},\ k>0.$$

The sampling time is t_s = 0.1 seconds. The single-sheeted Nichols plot is shown in Fig. 44 for k = 0.1. For this gain there are no crossings.

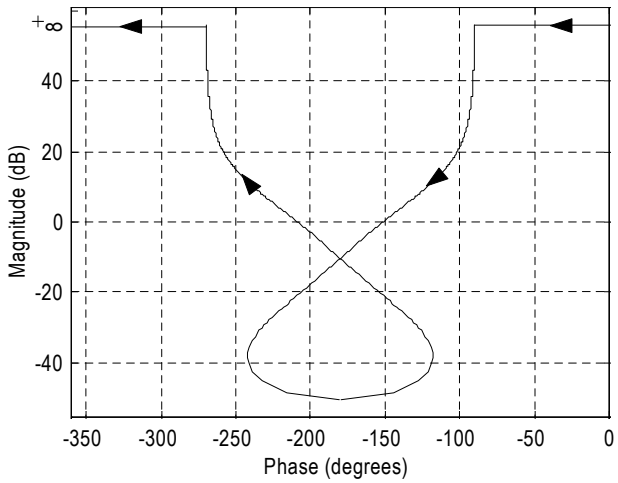


Figure 44: Nichols plot of Example 3 on a single-sheeted chart.

If the gain is kept below $k<0.33$ (an increase of approximately 10.4 *dB*), the closed-loop remains stable. Two positive crossings will occur if $0.33 \le k < 34.09$ indicating instability. A single positive crossing occurs for $k \ge 34.09$ (an increase of approximately 50.6 *dB*), also indicating instability. Note that the Nyquist plot will have a semi-infinite circle for each pole of in $L(z)$ lying on the unit circle. For a detailed discussion on such segments, see Example 2 above.

Example 4: Consider a unity non-minimum phase feedback system that has the following open-loop function

$$L(z) = \frac{k(z+2)(z+0.9)}{(z-1)(z-0.7)(z-0.3)},\ k>0\,.$$

The sampling time is t_s = 0.1 seconds. The multiple-sheeted Nichols plot is shown in Fig. 45 for k = 0.01, where no crossings can be observed.

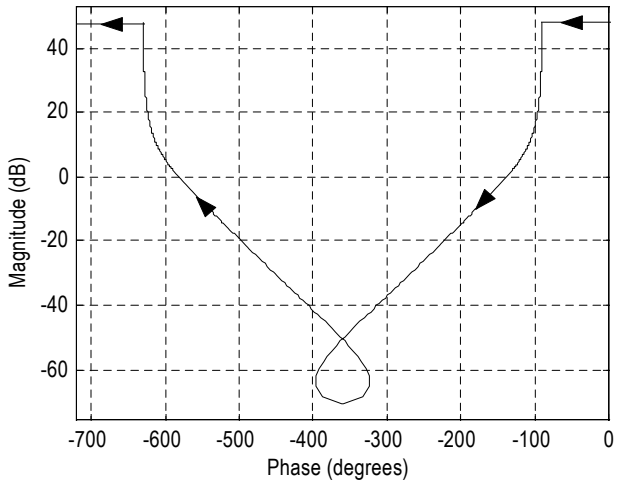


Figure 45: Nichols plot of Example 4 on a multiple-sheeted chart.

If the gain is kept below $k<0.027$ (an increase of approximately 9 *dB*) the closed-loop system remains stable. Two positive crossings will occur if $k \ge 0.027$ indicating two unstable closed-loop poles.

Robust Stability

The uncertain system for which this Toolbox is applicable includes the same class described in the previous section on continuous-time systems. The concept of templates can be extended to $L(z) = P(z)G(z)H(z)$ where $P(z) \in \mathcal{P}$.

The following is the Nichols chart robust stability criterion used in QFT. The open-loop transfer function $L(z)$ is assumed to belong to a set `L`, which has the form described in Chapter 2. In addition to the trivial assumption of no unstable (including $j\omega$-axis) pole/zero cancellation in any $L(z)$ in the set, the criterion requires either of the following groups of conditions. The first group (if one extends [23] to the discrete-time case) is: (1) $L(z)$ is strictly proper, (2) the uncertain parameters belong to a compact and simple connected set, (3) the coefficients of the numerator and denominator of $L(z)$ depend continuously on the uncertain parameters, and (4) the coefficients of the highest degree z terms in the numerator and denominator of $L(z)$ cannot vanish. The second set is: (1) at each fixed frequency, the responses of all $L(z)$ form a convex set in the complex plane, and (2) the number of unstable poles in $L(z)$ is fixed.

Criterion 4: Consider the single-loop system shown in Fig. 34. Assume that the uncertain set $\mathcal{L}$ satisfies one of the above sets of conditions. Let $L_0(z) \in \mathcal{L}$ denote the nominal plant. The feedback system is robust stable if:

- The nominal closed-loop system corresponding to $L_0(z)$ is stable and the single-sheeted Nichols envelope does not intersect the point **q**; or
- The nominal closed-loop system corresponding to $L_0(z)$ is stable and the multiple-sheeted Nichols envelope does not intersect any of the points $(2k+1)\mathbf{q}$, $k=0, 1, 2, ...$

The condition that the single-sheeted Nichols envelope does not intersect the point **q** is the same as requiring that $1 + L\left(z = e^{j\omega t_s}\right) \neq 0$ for all $L(z) \in \mathcal{L}$ and for all frequencies $\omega \in [0, \pi / t_s]$.

5 Examples

Introduction

There are 15 solved examples in this Toolbox. You can find them in the `qftdemo` directory with the `qftex#.m` file names (`#` denotes the example number from 1 to 15). We have also prepared a special QFT demo (to invoke type `qftdemo`). The difference between the `qftex#.m` files and the demo facility is that the `qftex#.m` are standard batch files that users are most comfortable with, while the files in the demo facility involve heavier use of *Handle Graphics*™ (for presentation purpose only and not needed for QFT design).

The examples can be divided into five groups. The first group, examples 1-6, is intended to expose the user to QFT in general, and to the use of this Toolbox for design of robust control systems with parametric and non-parametric uncertainties and simultaneous specifications. The first example is the one used to describe the QFT design procedure in Feedback Design with QFT. The second introduces a traditional QFT robust tracking problem, which is unlike similar robust tracking problem settings in other methods. The third describes design for a plant with non-parametric uncertainty. The fourth example focuses on a control design of a fixed plant and illustrates that the Toolbox is also useful for classical frequency-domain designs. The fifth is the ACC benchmark design problem consisting of a highly vibratory mechanical system. The sixth example is interesting in that it considers a missile with plants evaluated at several flight envelope locations with different non-parametric uncertainties and different performance specifications at each location.

The second group, examples 7 and 8, focuses on systems with more than one output, the cascaded design problem, illustrating how additional measurements can reduce control bandwidth significantly. The seventh presents a two-output problem where the two loops are closed sequentially in a simplistic approach: starting with the inner loop and ending with the outer loop. The eight examples present the natural approach: starting with the outer loop and ending with the inner loop.

The third group, examples 9-11 (and in some sense 14), illustrates application of QFT to practical systems such as those having significant mechanical vibration. The plant model in example 11 is described only in terms of its experimentally measured frequency response.

The fourth group, examples 12-14, focuses on design of discrete-time controllers with uncertain plants. Example 14 illustrates a continuous-time control design for a sampled-data system.

The fifth group includes a single example. Example 15 is a 2x2 multi-loop (MIMO) robust performance problem. Although this version of the Toolbox does not fully support MIMO design, its bound computation algorithms are general purpose and can be applied to any QFT problems including MIMO. However, in order to solve a cascaded-loop or a multi-loop QFT problem, the user must be familiar with the QFT's MIMO algorithms (see [30] and references wherein). No attempt is made here to explain these algorithms or how they were derived.

A few words are in order regarding the examples in this Toolbox. We see their role as solely to illustrate the application of QFT in feedback design. Practical feedback design involves many steps, e.g., modeling and identification, hardware selection, design and implementation. Therefore, except in few realistic cases, we did not attempt to relate examples to real physical problems. In general, since QFT is a linear, time-invariant design method, existence conditions for feedback solutions are the same as for any other linear, time-invariant design method.

The examples already include solutions. In some of the examples, you may initially find it difficult to follow our thought process used during loop shaping. Unfortunately, there is no easy recipe we can offer for loop shaping except experience. An excellent exposition of various loop-shaping issues can be found in [3]. We suggest that you first try our design for a given example, delete it, then start adding each term to the controller to see its effect.

A summary of the examples, divided into the five groups, is shown in Table 3.

Table 3: The Toolbox examples.

Example	Description
1	robust performance (main example)
2	robust performance (2 DOF QFT tracking)
3	non-parametric uncertainty
4	classical design (no uncertainty)
5	ACC benchmark
6	robust performance (mixed uncertainties types and mixed performance)
7	robust performance (cascaded-loop: inner → outer)
8	robust performance (cascaded-loop: outer → inner)
9	robust performance (flexible mechanical system)
10	robust performance (inverted pendulum)
11	robust performance (active vibration isolation)
12	discrete-time robust performance (main example)
13	discrete-time robust performance (2 DOF QFT tracking)
14	robust performance (compact disc drive)
15	robust performance (2x2 MIMO)

The following block diagram (Fig. 46) is applicable to most of the examples in this chapter.

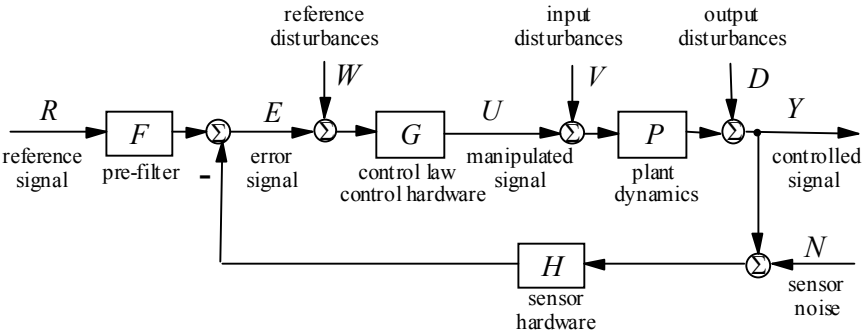


Figure 46: The single-loop feedback system.

Examples

Example 1: Main Example

The first example is the one used to describe the details of the QFT design procedure in Feedback Design with QFT.

Example 2: 2-DOF Design

This problem illustrates a traditional QFT tracking problem using a two degrees-of-freedom (2-DOF) design of $F(s)$ and $G(s)$ for an uncertain system [13, Ch. 21]. Consider a unity feedback control system (Fig. 46) with a parametric uncertain plant model described by

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

The closed-loop specifications are robust stability in terms of a margin specification

$$\left| \frac{PG}{1+PG}(j\omega) \right| \le 1.2, \text{ for all } P \in \mathcal{P}, \; \omega \ge 0$$

and a tracking specification

$$T_U(\omega) \le \left| F \frac{PG}{1+PG}(j\omega) \right| \le T_L(\omega), \quad \text{for all } P \in \mathcal{P}, \omega \in [0,10]$$

where

$$T_U(\omega) = \left| \frac{0.6854(j\omega+30)}{(j\omega)^2+4(j\omega)+19.752} \right| \quad \text{and} \quad T_L(\omega) = \left| \frac{120}{(j\omega)^3+17(j\omega)^2+828(j\omega)+120} \right|.$$

These two transfer functions were arrived at based on upper and lower bound tolerances on the step response of the system to reference signals (for details see [13]).

Let us review a two degrees-of-freedom design in QFT. The first design step involves computation of bounds for the robust margin (using `sisobnds(1, ...)`) and robust tracking (using `sisobnds(7,...)`) problems. In particular, `sisobnds(7,...)` computes bounds to guarantee that the variations in the tracking transfer function are less or equal to $T_U(\omega) - T_L(\omega)$ in *dB*. (This does not guarantee that the tracking specification is met.) In the second design step you shape a nominal loop ($L_0 = P_0 G(s)$) to meet its bounds. The third and final step focuses on shaping of the pre-filter, $F(s)$, so that the tracking transfer function lies within its bounds.

You will find out that computing robust tracking bounds (`sisobnds(7,...)`) takes significantly longer turn run compared with all other bounds, e.g., the margin bounds (`sisobnds(1,...)`). The reason is explained in the Limitations section in Chapter 6. The problem setup and its QFT solution using the Toolbox can be found in the file `qftex2.m`.

Example 3: Non-Parametric Uncertainty

This problem illustrates control design for a plant with a non-parametric uncertainty model (see [12]). Consider a control system (Fig. 46) with a non-parametric uncertain plant model described by

$$\mathcal{P} = \left\{ P(s) = \frac{10}{s(0.1s+1)}\left(1+\Delta_m(s)\right):\ \Delta_m(s) \text{ stable},\quad \left|\Delta_{\mathrm{m}}(\mathrm{s})\right| < \left| \frac{0.9\left(\frac{j\omega}{0.91}+1\right)}{\frac{j\omega}{1.001}+1} \right| \right\}.$$

The specifications are robust stability and robust sensitivity according to

$$\left|\frac{1}{1+PG}(j\omega)\right| \le 0.089\omega^2, \quad \text{for all } P \in \mathcal{P},\ \omega \le 5.$$

As discussed in Robust Stability, the associated QFT robust stability constraint is given by

$$\left|\frac{PG}{1+PG}(j\omega)\right| < \infty, \quad \text{for all } P \in \mathcal{P},\ \omega \ge 0.$$

Because some stability margin is always essential to guard against unmodelled high frequency dynamics, we use a more realistic robust stability problem in terms of a robust margin specification

$$\left|\frac{PG}{1+PG}(j\omega)\right| \le 1.2, \quad \text{for all } P \in \mathcal{P}, \omega \ge 0.$$

The problem setup and its QFT solution using the Toolbox can be found in the file `qftex3.m`.

Example 4: Classical Design for Fixed Plant

This problem illustrates that the Toolbox is equally effective in design of feedback systems that do not have uncertainty. In such cases, it offers an efficient platform for classical frequency response. Consider a fixed plant described by (Fig. 46)

$$P(s) = \frac{10}{s(s+)1}.$$

The specifications are stability, gain margin of at least 1.8, zero steady state error for velocity reference commands, and bandwidth limitation of

$$\left| \frac{PG}{1+PG}(j\omega) \right| \le 0.707, \ \omega \ge 10 .$$

The gain margin specification can be solved either `sisobnds(1, ...)` or `sisobnds(6, ...)` with `Ws = 1.2` (for which GM = 1.83). The steady state error specification can be met by including an integrator in the controller. Naturally, since the problem does not involve any uncertainty, it can be solved using feed-forward open-loop structure. The problem setup and its QFT solution using the Toolbox can be found in the file `qftex4.m`.

Example 5: ACC Benchmark

Consider the American Control Conference (ACC) benchmark control design problem [16,17]. The plant corresponds to a parametric uncertain flexible mechanical system (Fig. 46) described by

$$\mathcal{P}=\left\{P(s)=\frac{k}{m_1 s^2\left(m_2 s^2+\left(1+\frac{m_2}{m_1}\right)k\right)}:\ m_1=m_2=1, k\in[0.5,2]\right\}.$$

The specifications in [16] were given in terms of the time response of the closed-loop system. However, for a frequency response design, the following robust margin specification was found appropriate

$$\left|\frac{PG}{1+PG}(j\omega)\right|\le 2.25,\ \text{for all}\ P\in\mathcal{P},\ \omega\ge 0.$$

Note that the plant templates are the interesting part of this example. In the frequency band $\omega \in [1,2]$, each template consists of two non-connected parts, one with 0° phase and the other with -180° phase. This is due to that the 2nd order pole has no damping. One part of the template has an infinite length (when $\omega = \omega_n$). For example see the one at $\omega = 1.5$. In the band $\omega \in [1,2]$ all templates have the same shape, hence their corresponding robust stability bounds should be similar. However, due to discretization over k and over ω, it is unlikely that you will have a perfect match $\omega = \omega_n$ where the template's length is infinite (resonance). The templates you see in the range $\omega \in [1,2]$ have large magnitude ranges, but not quite all the way to infinity. When performing a design, you should be aware of this fact. To distinguish the bounds, you can click on a specific frequency legend button (top left corner) to alternate between show/hide of that bound. Note also that the templates are non-connected in the range for $\omega \in [1,2]$. The problem setup and its QFT solution using the Toolbox can be found in the file `qftex5.m`.

Example 6: Missile Stabilization

The example focuses on the performance of a missile along its vertical trajectory [26]. The missile is roll-stabilized and has a cruciform wing configuration. Due to the aim of this missile, only stabilization in the vertical plane will be considered using an autopilot that operates the control surfaces. The general form of the block diagram of the stabilized missile is shown in Fig. 47.

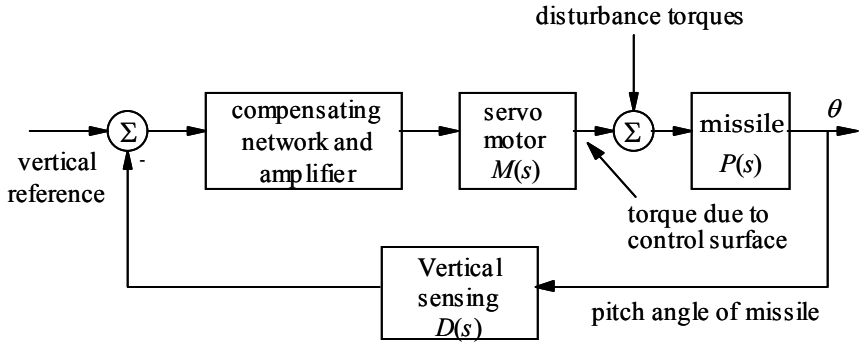


Figure 47: Missile control system.

Modeling of the dynamics to the missile involves aerodynamics, gravitational and propulsive forces. A simplified *unstable* open loop transfer function relating control-surface deflection to pitch angle relative to the vertical is

$$P(s) = \frac{a_1 s + a_2}{b_1 s^3 + b_2 s^2 + b_3 s + b_4}.$$

A servomotor, a 27-volt dc armature-controlled electric motor, is used to reduce the effect of disturbances. Its transfer function is

$$M(s) = \frac{1/107}{0.001 s^2 + 0.13 s + 1}.$$

The amplifier providing the necessary power is

$$A(s) = \frac{1}{0.01 s + 1}.$$

The rate gyro vertical sensor (voltage proportional to signal) measures pitch angle according to

$$D(s) = 27 \frac{40 s}{s^2 + 1.2 \times 40 s + 40^2}.$$

To reflect operation at different points in a flight envelope we consider three cases:

case 1: a_1=335, a_2=237, b_1=20.7, b_2=39, b_3=257, b_4=-9.5,
case 2: a_1=315, a_2=227, b_1=19.7, b_2=37, b_3=247, b_4=-9.0,
case 3: a_1=345, a_2=247, b_1=23.7, b_2=36, b_3=267, b_4=-10.5.

The plant set includes three cases $[P_1(s),P_2(s),P_3(s)]$ corresponding to the above three cases. Additional modeling error is modeled via the multiplicative form

$$\mathcal{P}=\left\{P_i(s)\left(1+\Delta_i(s)\right):\ \Delta_i(s)\ \text{ stable},\ \left|\Delta_i(j\omega)\right|<R_i,\ R_i=[0.1,0.05,0.075]\right\}.$$

That is, the plant model has both parametric and non-parametric uncertainties. Note that each case has a *different* multiplicative error model.

The specifications are robust margin

$$\left|\frac{PGMAD}{1+PGMAD}(j\omega)\right|\le W1_i,\ \text{for all}\ P\in\mathcal{P},\ \omega\ge 0$$

where

$$W1=\begin{bmatrix}W1_1\\W1_2\\W1_3\end{bmatrix}=\begin{bmatrix}1.3\\1.2\\1.25\end{bmatrix}$$

corresponding to each plant element. The robust input disturbance rejection is

$$\left|\frac{P}{1+PGMAD}(j\omega)\right|\le W2_i,\ \text{for all}\ P\in\mathcal{P},\ \omega\in[1,8]$$

where

$$W2=\begin{bmatrix}W2_1\\W2_2\\W2_3\end{bmatrix}=\begin{bmatrix}0.040\\0.036\\0.038\end{bmatrix}$$

corresponds to each plant element. In essence, the objective here is to find a single controller that meets all specifications at the three operation points instead of using the gain scheduling approach. Note that the interlacing property (one unstable open-loop pole trapped between two unstable zeros) dictates that the controller must be unstable. The problem setup and its QFT solution using the Toolbox can be found in the file `qftex6.m`.

Example 7: Inner-Outer Cascaded Design

Consider again the system in Example 1 but allow an additional measurement at the output of $P_2(s)$. The new block diagram is shown in Fig. 48.

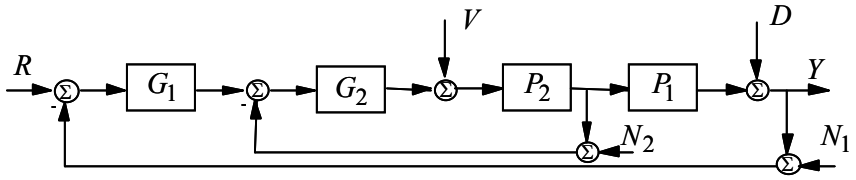


Figure 48: The cascaded feedback system.

where the two parametric uncertain plant models are

$$\mathcal{P}_1 = \left\{ P_1(s) = \frac{1}{(s+a)(s+b)} :\ [a \in 1,5],\ b \in [20,30] \right\} \quad \text{and} \quad \mathcal{P}_2 = \left\{ P_2(s) = k :\quad \mathrm{k} \in [1,10] \right\}.$$

This is a cascaded-loop system. For many reasons, such as plant uncertainty, sensor noise, nonlinearities, limited sensor reliability and forcing rate and/or amplitude saturation, we recommend you use a cascaded-loop feedback structure if feasible. This example describes how to execute such designs where the most inner loop is designed first. The control design problem is to find two forward controllers, $G_1(s)$ and $G_2(s)$, such that:

- The inner closed-loop system should be robust stable with at least 50° phase margin for all $P_2 \in \mathcal{P}_2$.
- The outer closed-loop system should be robust stable with at least 50° phase margin for all $P_1 \in \mathcal{P}_1$ and $P_2 \in \mathcal{P}_2$, and should satisfy plant output disturbance rejection to

$$\left| \frac{\mathrm{Y}}{\mathrm{D}}(j\omega) \right| \le \left| \frac{0.02(j\omega)^3 + 64(j\omega)^2 + 748(j\omega) + 2400}{(j\omega)^2 + 14.4(j\omega) + 169} \right|, \quad \text{for all } P_1 \in \mathcal{P}_1,\ P_2 \in \mathcal{P}_2,\ \omega \in [0,10]$$

and should reject plant input disturbance according to

$$\left| \frac{Y}{V}(j\omega) \right| \le 0.01, \quad \text{for all } P_1 \in \mathcal{P}_1,\ P_2 \in \mathcal{P}_2,\ \omega \in [0,50].$$

The design of the inner loop is straightforward. However, the design of the outer loop involves some additional computations. First, we have to compute the closed-loop transfer function of the inner loop T_2

$$T_2(s) = \left| \frac{P_2(s)G_2(s)}{1 + P_2(s)G_2(s)} \right|.$$

The effective open-loop plant for design of the outer loop is the product

$$P_{12}(s) = T_2(s)P_1(s).$$

Let us now derive the various closed-loop transfer functions for the margin and performance specifications.

The margin specification is given by

$$\left|\frac{P_{12}G_1}{1+P_{12}G_1}(j\omega)\right| \le \mu = 1.2, \text{ for all } P_1 \in \mathcal{P}_1,\ P_2 \in \mathcal{P}_2,\ \omega \ge 0\,.$$

whose constraint fits the format of the function `sisobnds(1,...)`. The output disturbance rejection specification is given by

$$\left|\frac{1}{1+P_{12}G_1}\right| \le \left|\frac{0.02(j\omega)^3+64(j\omega)^2+748(j\omega)+2400}{(j\omega)^2+14.4(j\omega)+169}\right|, \text{ for all } P_1 \in \mathcal{P}_1,\ P_2 \in \mathcal{P}_2,\ \omega \in [0,10]$$

whose constraint fits the format of the function `sisobnds(2,...)`. The input disturbance rejection specification is given by

$$\left|\frac{\frac{P_{12}}{G_2}}{1+P_{12}G_1}\right| \le 0.01, \text{ for all } P_1 \in \mathcal{P}_1,\ P_2 \in \mathcal{P}_2,\ \omega \in [0,50]$$

which *does not* readily fit any of the functions in the Toolbox. However, with the utility functions you can transform almost any constraint into a constraint that fits one of the functions `sisobnds(1-9,...)`. This is typically accomplished by matching the above transfer function into a similar one from `sisobnds(1-9,...)`. There may be more than one possible match. In this problem, for example, by replacing P_{12} with its equivalence we have

$$\frac{\frac{P_{12}}{G_2}}{1+P_{12}G_1} = \frac{\frac{P_{12}}{G_2}}{1+\frac{P_{12}}{G_2}G_2G_1} = \frac{P}{1+PGH}$$

where

$$P = \frac{P_{12}}{G_2}, \quad G = G_1, \quad H = G_2\,.$$

The transfer function on the right-hand side in the previous equation fits exactly the format of `sisobnds(3,...)`. More importantly, in effect the same loop used in the previous two specifications (margin and output disturbance rejection) is also used here since G_2 is a fixed function. Recall that you must use the same loop and same nominal loop in computing all bounds in one problem.

The problem setup and its QFT solution using the Toolbox can be found in the file `qftex7.m`. Also shown is the reduction in control bandwidth compared with the single-loop design of Example 1.

Example 8: Outer- Inner Cascaded Design

The setup here is the same as in Example 7. The difference will be that here we first close the outer loop, then follow with closure of the inner loop. It can be argued that this is the more natural approach for loop closure, since the inner loop is introduced mainly to reduce the bandwidth of the outer controller G_1. The outer controller should not be designed to achieve robustness against full variations in both P_1 and P_2, only those in P_1 with some extra margins. The inner controller G_1 is then designed to reduce this burden on G_2 (see [3] for details).

The question is then how best to design this G_2 to achieve this goal. Closing first the inner loop (as in Example 7) is done arbitrarily, since we cannot predict its effect on the outer loop. Therefore, we first close the outer loop by assuming $G_2 = \infty$. A nominal loop L_{10} is designed to meet these specifications. We then design G_2 such that, with the given L_{10}, all specifications will be met. The outer controller G_1 is then computed from L_{10} and G_2. The key idea here is of "free uncertainty" which, qualitatively speaking, says that G_2 can cope with large uncertainty in the main loop with relative low gains than would otherwise expected. For an excellent exposition of cascaded-loop designs and discussion of trade-off between the various loops please see [3, Ch. 12]. Note that in practice, the trade-off depends on known information on sensor spectrum at each loop. The salient details of the outer-inner design are now described.

In the first step, we close the outer loop. Assuming $G_2 = \infty$, the stability (margin) problem is simplified to

$$\left|\frac{L_1}{1+L_1}(j\omega)\right| \le 1.2, \text{ for all } P_1 \in \mathcal{P}_1,\ P_2 \in \mathcal{P}_2,\ \omega \ge 0$$

where

$$L_1 = P_1 G_1 T_2, \qquad T_2 = \frac{P_2 G_2}{1+P_2 G_2}.$$

Since $G_2 = \infty$, $T_2 = 1$, $L_1 = G_1 P_1$. The performance problem is similar to that in the single-loop example (Example 1)

$$\left|\frac{1}{1+\mathrm{L}_1}\right| \le \left|\frac{0.02(j\omega)^3 + 64(j\omega)^2 + 748(j\omega) + 2400}{(j\omega)^2 + 14.4(j\omega) + 169}\right|, \text{ for all } P_1 \in \mathcal{P}_1,\ P_2 \in \mathcal{P}_2,\ \omega \in [0,10].$$

Again, $G_2 = \infty$ is used. This approximation makes sense in that G_2 should not be designed to help with performance at low frequencies. Note that the actual nominal loop, L_{10},

$$L_{10} = G_1 P_1 T_{20}, \qquad T_{20} = \frac{G_2 P_{20}}{1+G_2 P_{20}}$$

must be designed to meet both specifications above. It should include anticipated dynamics (only in the sense of same numerator and denominator orders) of nominal inner loop. In this example we assume G_1 to be a simple second order, G_2 to have three zeros and four poles, and since both P_{10} is a simple second order and P_{20} is a gain, easy calculation shows that L_{10} should have three zeros and eight poles. For this purpose, it is sufficient to design a unique L_{10} with no zeros and five poles (since we can always add three pairs of similar zeros and poles).

This L_{10} should be designed with extra margins anticipating that the true loop has some additional uncertainty from P_2; this is best taken care of by staying away from the high frequency margin bound. Once such L_{10} is designed, we turn to design of the inner controller G_2 that must be designed against *all* specifications and *all* uncertainties in inner loop and outer loop alike. This is done as follows. The inner-loop margin problem

$$\left|\frac{P_2G_2}{1+P_2G_2}(j\omega)\right| \le 1.2, \quad \text{for all } P_2 \in \mathcal{P}_2,\ \omega \ge 0$$

can be computed as usual with `sisobnds(1,...)`. The transfer function for the outer loop margin specification looks like

$$\frac{L_1}{1+L_1} = \frac{P_1G_1T_2}{1+P_1G_1T_2}.$$

Using

$$G_1 = \frac{L_{10}\left(1+G_2P_{20}\right)}{P_{10}G_2P_{20}}$$

and the above defined T_2 to plug into the transfer function and further simplifying gives the problem

$$\left|\frac{L\left(_{10}P_1P_2\right)+\left(L_{10}P_{20}P_1P_2\right)G_2}{\left(P_{10}P_{20}+L_{10}P_1P\right)_2+\left(P_{10}P_{20}P_2+L_{10}P_{20}P_1P_2\right)G_2}\right| \le 1.2, \quad \text{for all } P_1 \in \mathcal{P}_1,\ P_2 \in \mathcal{P}_2,\ \omega \in [0,10].$$

One thing should become obvious at this point. The loop function in the inner-loop margin problem and the loop above are not the same. However, we have emphasized the fact that in order to design a single controller (G_2) to achieve simultaneously different specifications, we *must* translate all such specifications into bounds on the same nominal loop. For this purpose the Toolbox includes the function

```
bdb = genbnds(ptype,w,Ws,A,B,C,D,P0,phs)
```

that computes bounds for a general bilinear problem setting

$$\left|\frac{\mathrm{a}(j\omega)+\mathrm{b}(j\omega)\times G(j\omega)}{\mathrm{c}(j\omega)+\mathrm{d}(j\omega)\times G(j\omega)}\right| \le \mathrm{W}(\omega)$$

where *a*, *b*, *c* and *d* can be parametric uncertain transfer functions such as those shown above with G_2 being the controller. The function `genbnds` computes bounds on the controller G that are then multiplied by the nominal plant of the inner loop P_{20}.

The transfer function for the output disturbance rejection problem is

$$\frac{Y}{D} = \frac{1}{1+P_1G_1T_2}.$$

Using above derivations the problem can be simplified to

$$\left|\frac{(P_{10}P_{20})+(P_{10}P_{20}P_2)G_2}{(P_{10}P_{20}+L_{10}P_1P_2)+(P_{10}P_{20}P_2+L_{10}P_{20}P_1P_2)G_2}\right|$$

$$\le\left|\frac{0.02(j\omega)^3+64(j\omega)^2+748(j\omega)+2400}{(j\omega)^2+14.4(j\omega)+169}\right|,\quad \text{for all } P_1\in\mathcal{P}_1,\ P_2\in\mathcal{P}_2,\ \omega\in[0,10]$$

The function `sisobnds(10,...)` is again used here. The transfer function for the output disturbance rejection problem is

$$\frac{Y}{V}=\frac{P_1\dfrac{T_2}{G_2}}{1+P_1G_1T_2}.$$

Using above derivations the problem can be simplified to

$$\left|\frac{(P_{10}P_{20}P_1P_2)+(0)G_2}{(P_{10}P_{20}+L_{10}P_1P_2)+(P_{10}P_{10}P_1+L_{10}P_{20}P_1P_2)G_2}\right|\le 0.01,\quad \text{for all } P_1\in\mathcal{P}_1,\ P_2\in\mathcal{P}_2,\ \omega\in[0,50].$$

The function `sisobnds(10,...)` is again used here.

Once the above bounds are grouped and their intersection is found, we can loop shape the inner controller G_2 as usual. Having designed G_2, the outer controller can be extracted from

$$G_1=\frac{L_{10}(1+G_2P_{20})}{P_{10}G_2P_{20}}.$$

This G_1 will most likely have higher numerator and denominator orders than those assumed when the outer loop was first closed (only the relative degree is the same). This is not a problem since there are acceptable cancellations when forming L_{10} between T_{20}, P_{10} and G_2.

A typical outer-inner cascaded design may require several iterations. This is because the first outer-loop is not designed against the actual inner loop T_2. Two extreme cases are $T_2 = 1$ or $T_2 = P_2$ (as used in this example). The "optimal" design should place the low frequency gain of L_{10} somewhere in between the bound with $T_2 = 1$ and the more difficult ones for $T_2 = P_2$. It is conceivable that you use $T_2 = 1$, and the inner-loop design cannot supply enough gain to meet the low-frequency gain requirement. Outer-inner design should be attempted only if there is a significant uncertainty in both P_1 and P_2. It will require, most likely, a few iterations to decide how to best assign the bandwidth burden to G_2.

The problem setup and its QFT solution using the Toolbox can be found the file `qftex8.m`. Also included is a plot showing the reduction in control bandwidth compared with the single-loop design of Example 1: Main Example and the inner-outer cascaded design in Example 7: Inner-Outer Cascaded Design.

Example 9: Uncertain Flexible Mechanism

This problem was taken from an experimental project at Philips Research Laboratories [18]. Its objective was to study the possibilities of advanced control design methods leading to a high performance servomechanism design for products such as compact disc players. Reference [18] contains an excellent presentation of the various stages involved in a practical robust control design from the initial theoretical study to its experimental implementation. The control system is shown in Fig. 49

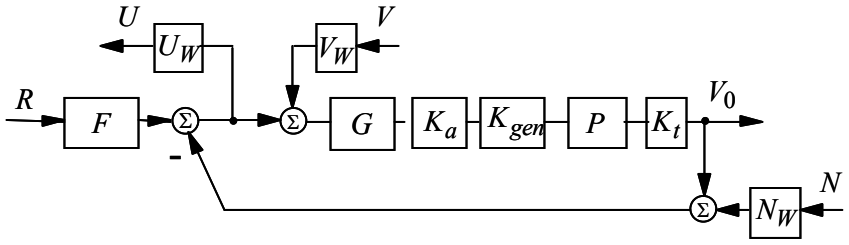


Figure 49: The servomechanism system.

where the parametric uncertain plant model involves significant mechanical flexibility and is described by

$$\mathcal{P} = \left\{ P(s) = \frac{K_{gen} K_t K_a K_m (d_s s + c_s)}{a_1 s^3 + a_2 s^2 + a_3 s + a_4} : \begin{array}{l} a_1 = j_1 j_2 \\ a_2 = j_1 (d_s + d_{m1}) + j_2 (d_s + d_{m2}) \\ a_3 = c_s (j_1 + j_2) + d_s + d_{m2} \\ a_4 = c_s (d_{m1} + d_{m2}) \end{array} \right\}$$

and where

K_{gen} = 0.05 *Amp/Volt*
K_t = 0.0133
K_a = 20 *Volt/Volt*
$j_1 = j_2$ = 1.46e^{-6} *kgm*^{2}
K_m = 34.5e^{-2} *Nm/A*
d_s=5e^{-6} *Nms/rad*
$d_{m1} = d_{m2}$ = 0.45e^{-6} *Nms/rad*

The uncertain stiffness is

$$C_s \in [0.0111, 0.0195]\ N/m.$$

In addition to robust stability, there are numerous performance specifications as follows.

Tracking:

$$\left| W \frac{R - V_0}{R}(j\omega) \right| \le 1, \text{ for all } P \in \mathcal{P},\ \omega \ge 0\,.$$

Noise Rejection:

$$\left| \frac{U}{N}(j\omega) \right| \le 1, \text{ for all } P \in \mathcal{P},\ \omega \ge 0\,.$$

Disturbance Rejection:

$$\left| W \frac{R - V_0}{V}(j\omega) \right| \le 1, \text{ for all } P \in \mathcal{P}, \ \omega \ge 0 .$$

Control Effort:

$$\left| \frac{U}{R}(j\omega) \right| \le 1, \text{ for all } P \in \mathcal{P}, \ \omega \ge 0 ,$$

$$\left| \frac{U}{V} \right| (j\omega) \le 1, \text{ for all } P \in \mathcal{P}, \ \omega \ge 0 .$$

where

$$W(s) = \frac{2\pi\omega_b}{s+1}, \ \omega_b = 10 \ Hz \ \text{(bandwidth)} .$$

For our QFT design, the following robust stability margin constraint is added.

Robust Margin:

$$\left| \frac{PG}{1+PG}(j\omega) \right| \le 1.1, \text{ for all } P \in \mathcal{P}, \ \omega \ge 0 .$$

The weights are: $n_w = 0.01$, $U_w = 0.33$ and $V_w = 0.1$. In addition, due to a DSP board limitation, the controller and pre-filter poles were limited to approximately 100 *Hz*.

In this problem, the plant templates are "fat" (i.e., stretched over a large phase range) over a frequency range due to the lightly damped mode. In this frequency band, the mode's natural frequency will significantly change with changes in the uncertain parameter c_s. Such changes typically result in a template with large phase variations (e.g., view the templates at $\omega = 155$ and $\omega = 180$ *rad/sec*). The corresponding robust margin (stability) bounds will also be "fat".

The strength of QFT is clearly highlighted in this problem: it is straightforward to compare between the relative "toughness" of different specifications simply by observing their bounds. Since we have already done so, you will find that the file `qftex9` includes only the toughest specifications from the set defined above. Certain bounds for the specifications were not considered here since they were found to lie below the ones that are shown.

A few words are in order regarding our approach here for the two-degrees-of-freedom feedback design. Some specifications involve transfer functions in terms of both $G(s)$ and $F(s)$. In our design, we first solved for $G(s)$ with $F(s) = 1$, in essence to reduce closed-loop sensitivity. After $G(s)$ has been designed, we have designed the pre-filter $F(s)$ to improve the tracking response relevant to the specifications. The problem setup and its QFT solution using the Toolbox can be found in the file `qftex9.m`.

Example 10: Inverted Pendulum

Control of an inverted pendulum is a classical problem found in many books. The example, a part of an experimental study [27], illustrates the interaction between mechanical resonances and closed-loop performance. The inverted pendulum with a vertically movable tip mass (to generate uncertainty) is connected via a flexible arm to a cart which can move horizontally (Fig. 50).

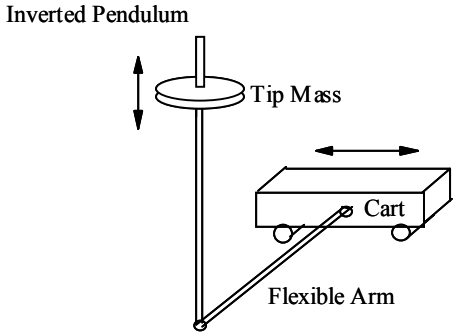


Figure 50: The inverted pendulum experiment.

The linearized model from the pendulum angle θ to the cart's motor current *I* is

$$P(s)=\frac{\theta(s)}{I(s)}=\frac{K\alpha}{s(s+\alpha)-k_{gf}K\alpha}\times\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}\times\frac{1/L\,s^2}{s^2-g/L}\times e^{-s\tau}$$

where

$L \in [0.3,0.45]$ *m*
$K \in [1.5,1.7]$ *m/volt/sec*
$\alpha \in [15,17]$ *1/sec*
$\omega_n \in [50,70]$ *rad/sec*
$\zeta \in [0.01,0.02]$
$\tau = 0.014$ *sec*
$g = 9.81$ *m/s²*
$k_{gf} = 0.1$ *volt/m*

The time delay is an engineering approximation for the zero-order hold and computational delay in the digitally implemented system. The specification is

$$\left|\frac{PG}{1+PG}(j\omega)\right|\le 2.1,\ \text{for all}\ P\in\mathcal{P},\ \omega\ge 0$$

which is often used as a rule of thumb for such systems. Note that in the frequency range $\sqrt{g/L}$ the template is non-connected including the bounds.

The problem setup and its QFT solution using the Toolbox can be found in the file `qftex10.m`. An interesting aspect of the design can be seen by a zoom-in around the (-180°,0dB)) point in the `lpshape` screen (the nominal loop wraps around). Though the template is non-connected due to the uncertainty in

the length L, with proper care we can execute a QFT design that guarantees robust stability (note the non-connected bounds at ω = 50).

Example 11: Active Vibration Isolation

This example involves single-axis active vibration isolation (courtesy of LORD Corporation, Cary NC). The experimental plant frequency response is between an accelerometer mounted on a structure and an active mount that connects the structure to a vibrating engine. The feedback system shown in Fig. 51 has the open-loop plant *P* consisting of the combined engine/structure/mount/amplifier.

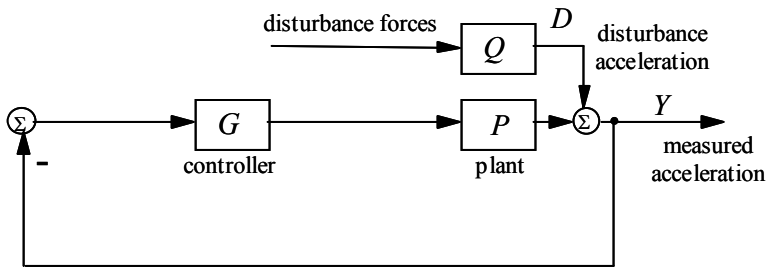


Figure 51: The active vibration isolation feedback system.

The frequency response of the open-loop plant is shown Fig. 52.

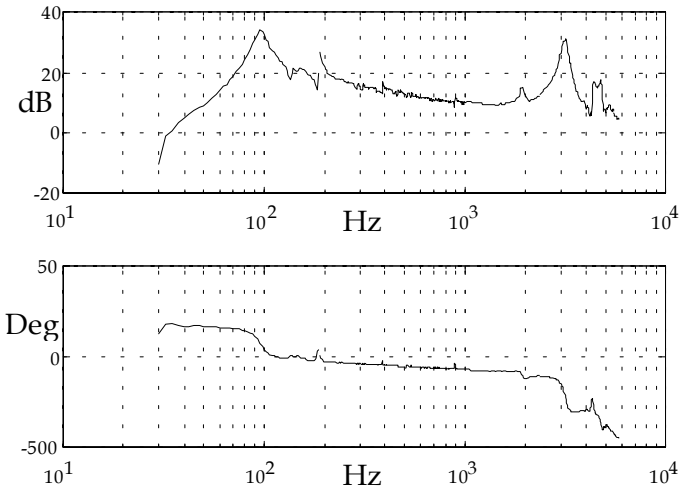


Figure 52: Measured open-loop frequency response.

There are two primary control objectives. The first is stability with reasonable margins

$$\left|\frac{PG}{1+PG}(j\omega)\right| \le 1.2,\ \omega \ge 0\,.$$

The second is disturbance rejection (transmissibility of disturbance acceleration to measured acceleration) of -20 *dB* in the working frequency band

$$\left|\frac{1}{1+PG}(j\omega)\right| \le 0.1,\ \omega \in [100,200]\ Hz\,.$$

Due to hardware constraints, the controller cannot have more than five poles.

It may be possible to identify a nominal rational transfer function whose frequency response is "close" to the experimental one; however, this step is not required in QFT design. Thus, the usual identify-design-implement-redesign cycle can be completed much more efficiently.

A few words are in order on our loop shaping here. While inside the lpshape window, view the designed elements. The low frequency real zero and 2nd order pole affect the loop response so that it does not cause encirclements. The high frequency 2nd order zero/pole pair around 2000 *rad/sec* is basically a notch element. Delete the 2nd order terms and you will see why we need it. The other terms are not as important and in fact can be eliminated by proper model order reduction. In fact, you can try it as follows.

The most important loop response is in the range of 1000-20000 *rad/sec*, since it is closest to its stability bounds there. When you attempt to reduce the order from five to four, the new loop response is unacceptable. However, if you try to use frequency weights, you may do better. For example, add a few weights with low magnitudes at $\omega<1000$ and $\omega>20000$ and large magnitude in the range $1000<\omega<20000$. You can even experiment with specifying the type of reduction: input, output or both. The HSV plot with weights will be different from that without weights. Note that the reduced order controller can be unstable, since stability cannot be guaranteed with weighted order reduction.

The problem setup and its QFT solution using the Toolbox can be found in the file `qftex11.m`.

Example 12: Main Example (Discrete-Time)

Consider a unity feedback sampled-data system shown in Fig. 53.

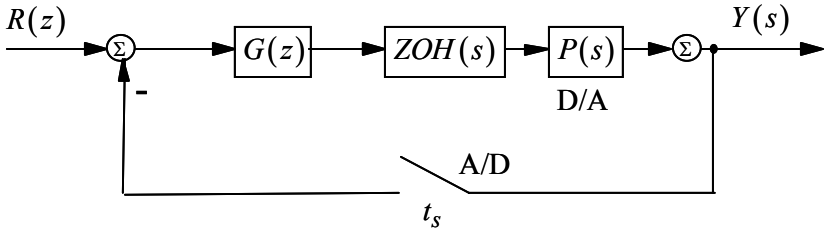


Figure 53: The sampled-data feedback system.

where t_s denotes sampling time. The discrete-time parametric uncertain plant model is given by

$$\mathcal{P} = \left\{ P(z) = \mathcal{Z}\left[\text{zoh}(s) P(s) \right] : \; P(s) \in \mathcal{P}s \right\}$$

where $\mathcal{Z}[\bullet]$ denotes the z-transform, $zoh(s)$ denotes a zero-order hold

$$zoh(s) = \frac{1 - e^{-st_s}}{s}$$

and where the continuous-time parametric plant model is

$$\mathcal{P}s = \left\{ \frac{k}{(s+a)(s+b)} : \; k \in [1,10], \; a \in [1,5], b \in [20,30] \right\}.$$

If the sampling time is sufficiently small relative to the system's bandwidth and there is no pole/zero cancellation of oscillatory modes in the open-loop functions, we can consider only the discrete-time system shown in Fig. 54.

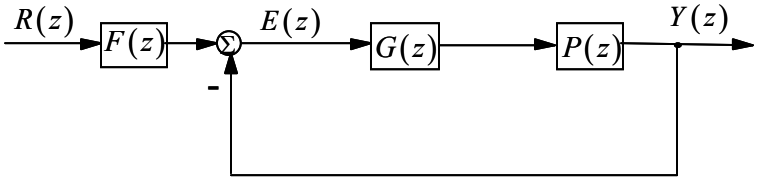


Figure 54: The discrete-time feedback system.

The control problem is to design the controller, $G(z)$ ($H(z) = 1$, $F(z) = 1$), such that the closed-loop system is robust stable and satisfies a margin constraint

$$\left| \frac{PG}{1+PG}(z) \right| \le 1.2, \text{ for all } P \in \mathcal{P}, \; z = e^{j\omega t_s}, \; \omega \in \left[0, \frac{\pi}{t_s} \right]$$

rejects plant output disturbance according to

$$\left|\frac{1}{1+PG}(z)\right| \le \left|0.02\frac{z^3+64z^2+748z+2400}{z^2+14.4z+169}\right|, \text{ for all } P\in\mathcal{P},\ z=e^{j\omega t_s},\ \omega\in[0,10]$$

and rejects plant input disturbance according to

$$\left|\frac{P}{1+PG}(z)\right| \le 0.01, \text{ for all } P\in\mathcal{P},\ z=e^{j\omega t_s},\ \omega\in[0,50]\ .$$

The problem setup and its QFT solution using this Toolbox can be found in the M-file `qftex12.m`. This example file includes three different solutions: for $t_s = 0.001$, $t_s = 0.003$, and $t_s = 0.01$ seconds. When you edit the file you will see that as t_s is increased, the controller becomes more "complex":

$$G(z) = 1950\frac{z-0.96}{z-0.8}, \qquad t_s = 0.001$$
$$G(z) = 4721\frac{z-0.9}{z-0.212}, \qquad t_s = 0.003$$
$$G(z) = 1998\frac{(z-0.3)(z-0.63)}{(z-0.2)(z+0.745)}, \qquad t_s = 0.01$$

As t_s increases, we may be forced to increase controller order and/or use poles with negative real parts in order to satisfy the demand for increasing phase lead. If t_s is increased further, it may be impossible to meet all specifications and robustly stabilize the system. The zero-order hold effect is similar to that of a non-minimum phase zero in limiting the achievable benefits of feedback. Note that in the limit $t_s \to 0$, this example reduces to Example 1: Main Example.

Example 13: 2 DOF Design (Discrete-Time)

This example illustrates is the discrete-time QFT tracking problem with a two-degree-of freedom (2 DOF) structure (Fig. 54). Consider a unity feedback sampled-data system shown above with $t_s = 0.001$ *seconds*. The discrete-time parametric uncertain plant model is given by

$$\mathcal{P} = \left\{ P(z) = \mathcal{Z}\left[\text{zoh}(s) P(s) \right] : \; P(s) \in \mathcal{P}s \right\}$$

where *zoh*(*s*) denotes a zero-order hold

$$zoh(s) = \frac{1 - e^{-st_s}}{s}$$

and where the continuous-time parametric uncertain plant model is given by

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

The specifications are: a margin constraint

$$\left| \frac{PG}{1+PG}(z) \right| \le 1.2, \text{ for all } P \in \mathcal{P}, \; z = e^{j\omega t_s}, \; \omega \in \left[0, \frac{\pi}{t_s} \right]$$

and a tracking constraint

$$T_L(\omega) \le \left| F \frac{PG}{1+PG}(z) \right| \le T_U(\omega), \text{ for all } P \in \mathcal{P}, \; z = e^{j\omega t_s}, \; \omega \in [0,10]$$

where

$$T_U(\omega) = \left| \frac{0.6854(j\omega + 30)}{(j\omega)^2 + 4j\omega + 19.752} \right| \quad \text{and} \quad T_L(\omega) = \left| \frac{120}{(j\omega)^3 + 17(j\omega)^2 + 828(j\omega) + 120} \right|.$$

The problem setup and its QFT solution using the Toolbox can be found in the file `qftex13.m`.

Example 14: CD Mechanism (Sampled-data)

This example considers robust control design for a single-loop compact disc mechanism. Although the feedback structure is sampled-data, owing to fast sampling, we perform the design in continuous-time. A compact disc player (Fig. 55) is an optical decoding device that reproduces high-quality data from a digitally coded signal recorded as a spiral shaped track on a reflective disc. The design problem is to achieve good track following in the presence of disturbances and parametric plant uncertainty, while using mainly measured frequency response data with limited identification (to define nominal natural frequencies). For a complete discussion of both SISO and MIMO QFT designs of this problem see [24,25].

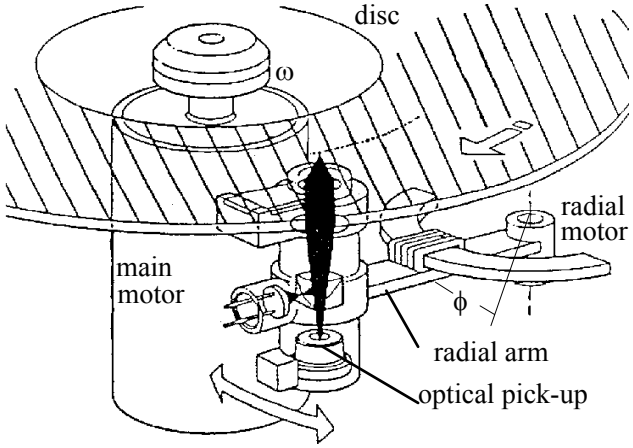


Figure 55: A schematic view of a Compact Disc mechanism.

The difficulty in achieving good track following is due to disturbances and plant uncertainty. Disturbances are caused, for example, by external shocks when the CD is used in a car going over a bump or in a portable CD used by a runner. Plant uncertainty is always a factor in mass production due to manufacturing tolerances. Feedback is clearly required in order to achieve good track following.

Figure 56 presents a block-diagram of the radial control loop. The difference between the track position and the laser beam spot position on the disc is detected by the optical system; it generates a radial error e_R signal via a gain G_{opt}. A controller K feeds the radial motor with the current I_{rad}. This in turn generates a torque resulting in an angular acceleration. The transfer function from the current I_{rad} to the angular displacement ϕ of the arm is called $G_{act}(s)$. A (nonlinear) gain G_{arm} relates the angular displacement with the spot movement in the radial direction. Only the control-error signal e_R is available for measurement. Assuming constant radial velocity ω, the goal is to control the position of the spot on the disk.

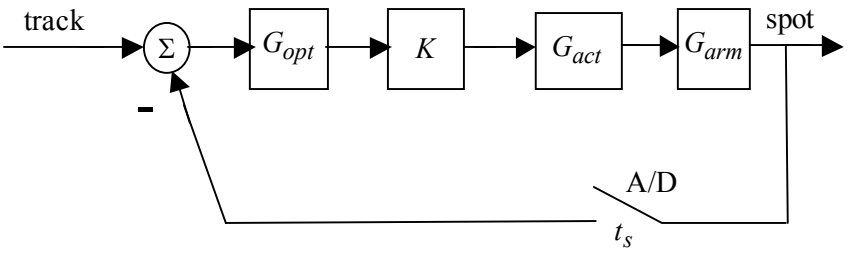


Figure 56: Block diagram of the radial loop.

Neither the true spot position, which can be interpreted as the system output, nor the track position is available as signals. In current systems K is a continuous-time PID controller. The radial servo system

has a design bandwidth of 500 *Hz*, a compromise value in which several conflicting factors are taken into account:

- accommodation of mechanical shocks acting on the player,
- achievement of the required disturbance attenuation at the rotational frequency of the disc, necessary to cope with significant disc eccentricity,
- playability of discs containing faults
- audible noise generated by the actuator, and
- power consumption.

The CD dynamics are characterized by mechanical vibrations that fall within the controlled bandwidth. Even with a reasonable identification of nominal transfer functions from frequency response measurements, experience has shown that relatively small identification errors may lead to significant reduction in closed-loop performance. The nominal dynamics (Fig. 57) were measured by averaging over several hundreds frequency response tests. At low frequencies the actuator transfer function from current input I_{rad} to position error output e_R is a critically stable system with a phase lag of 180° (rigid body mode). The erratic low frequency response is due to low coherence. At higher frequencies the measurement shows parasitic dynamics due to mechanical resonances of the radial arm and mounting plate (flexible bending and torsional modes). Based on practical experience, it is possible to define key CD quantities, those that vary from one player to another and from one track to another, which will have a significant effect on the dynamics. These are the three undamped natural frequencies with nominal values of 0.8, 1.62 and 4.3 *kHz*. To quantify possible variations, we allow each natural frequency to vary independently by ±2.5% around its nominal value. The frequency response set is then computed from the measured data (nominal case) and from the above parametric variations. Details of how it was done can be found in [24]. The M-file `qftex14` loads in the pre-computed uncertain plant information (125 elements are stored in `sisocd.mat`).

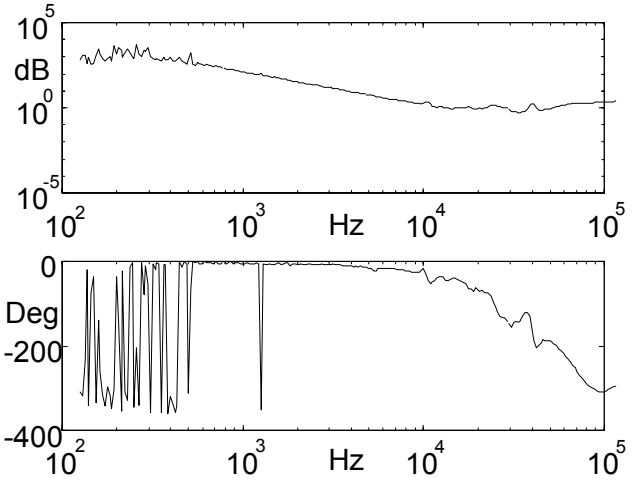


Figure 57: Measured nominal open-loop frequency response.

The specifications are: robust stability with margins

$$\left|\frac{PG}{1+PG}(j\omega)\right| \le 3, \text{ for all } P \in \mathcal{P}, \ \omega \ge 0$$

and robust sensitivity such that the closed-loop sensitivity function meets the magnitude specification shown in Fig. 58.

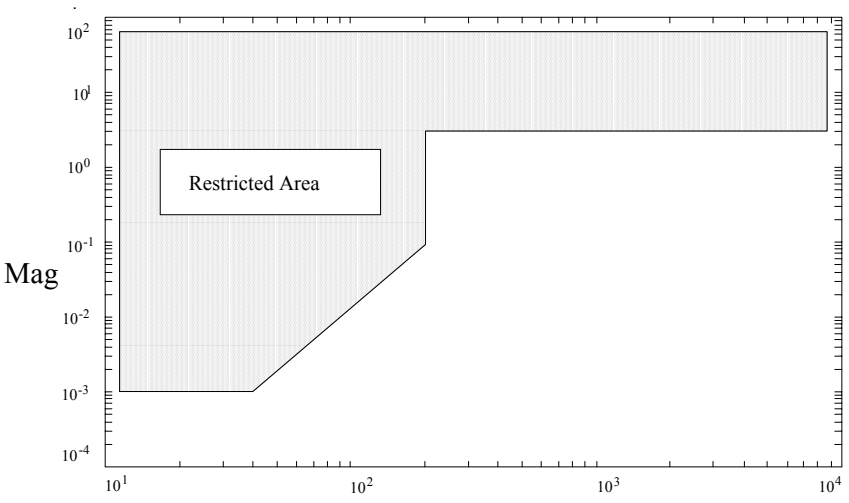


Figure 58: Robust sensitivity reduction specification.

The controller is to be implemented in a discrete-time form. That is, K is discrete-time and a zero-hold separates it from the dynamics $G_{act}(s)$. However, due to the fast sampling (relative to plant dynamics and closed-loop bandwidth), it is reasonable to measure the frequency response of a closed-loop sensitivity function S, then extract from it that of the plant P (the controller response G is known) using $P = (s^{-1}-1)/G$. The plant response is lumped together with a zero-order hold and digital control law computation delay (at a sampling rate of 17.5 *kHz*). Therefore, the effective block diagram for the design in the continuous-time structure is shown in Fig. 46. The controller can be designed in continuous-time and then be discretized. It is not a true sampled-data design but the approach is reasonable for engineering purposes. The key assumption is that, within the performance bandwidth and with fast sampling, the frequency response of the continuous-time dynamics is *similar* to that of the discretized dynamics. The implemented design uses a discretized version of $G(s)$ [24].

A few words are in order regarding templates and bounds here. For an obvious reason, the plant templates show negligible variations in frequencies other than those near the three uncertain natural frequencies. The template at the third natural frequency of 4.3 *kHz* exhibits the most phase and magnitude variations. In fact, the spacing between the template points is rather crude, which results in the non-smooth stability bound at this frequency (a combination of arcs). A closer spacing will smoothen the boundary, however, for design purposes if the loop is expected to lie away from the bound at that frequency, the present spacing suffices.

This example clearly illustrates the effects of the non-minimum-phase plant zero and of Bode sensitivity integral. To achieve the desired small sensitivity up to 200 *Hz*, we must sacrifice sensitivity at another frequency range nearby; specifically, we accept values larger than unity in the next immediate decade. In addition, it shows that sacrificing low-frequency phase margin can help in improving feedback properties at higher frequencies (without conditional stability one cannot get such a high low-frequency gain and minimize control bandwidth).

A few words are in order on our loop shaping here. While inside the `lpshape` screen, view the designed elements. There are two notches at around 5300 and 9500 *rad/sec* (shown by separate 2nd order zeros and poles due to model order reduction). Delete those elements and you will see why we need them. The high gain at low frequencies can be achieved only with a proper sacrifice of loop phase at that range (the 2nd order pole and subsequent zero). This example clearly illustrates the concept of "phase lag maximization" [3]. For more details on loop shaping for this problem see [29].

The problem setup and its QFT solution using the Toolbox can be found in the file `qftex14.m`.

Example 15: Multi-Loop Design

This problem illustrates a 2-input 2-output robust performance problem. The problem is rather simplistic and is not meant to reflect real-life problems, nor does it explore advantages and disadvantages of MIMO QFT design. It is meant to demonstrate use of the Toolbox in MIMO problems. Consider a unity feedback control system (Fig. 46) with a parametric uncertain plant model described by

$$P=\left\{P(s)=\begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}=\frac{1}{s^2+0.03as+10}\begin{bmatrix} a & 3+0.5a \\ 1 & 8 \end{bmatrix}:\ a\in[6,8]\right\}.$$

The closed-loop specifications are robust stability and robust margin in each channel

$$\left|1+P_{ii}^{e}g_i(j\omega)\right|\geq\frac{1}{1.8},\ i=1.2,\ \text{for all}\ P\in\mathcal{P},\ \omega\geq 0$$

where P_{ii}^{e} denotes the open-loop function at the *i*'th channel when all other channels (loops) are closed. Finally, the performance specification is given in terms of the sensitivity function

$$\left|s_{ij}(j\omega)\right|\leq\eta_{ij}(\omega),\ \ \omega\in[0,10],\ \ \ \eta_{ij}(\omega)=\begin{cases} 0.01\omega & i=j \\ 0.005\omega & i\neq j \end{cases}$$

where

$$S(s)=\begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}=(I+PG)^{-1}.$$

(*I* is a 2x2 identity matrix).

The MIMO QFT sequential design procedure considers diagonal controllers, hence

$$G(s)=\begin{bmatrix} g_1 & 0 \\ 0 & g_2 \end{bmatrix}.$$

The sequential procedure involves a sequential single-loop design of each (channel) in the system. In this example there are two channels (loops). Under mild assumptions (related to unstable pole/zero cancellations and fixed decentralized modes), robust stability of the MIMO system is related to the stability of MIMO characteristic equation det(*I*+*PG*). It can be expanded as follows

$$\det(I+PG)=(1+p_{11}g_1)\left(1+p_{22}^{e}g_2\right).$$

That is, the MIMO system is robust stable if each of the two functions on the right-hand side of the above equality is robust stable. The relations

$$p_{11}^{e}=p_{11}^{e}=p_{11}-\frac{p_{12}p_{21}g_2}{1+p_{22}g_2}$$

$$p_{22}^{e}=p_{22}-\frac{p_{12}p_{21}g_1}{1+p_{11}g_1}$$

describe the equivalent open-loop transfer function of each channel assuming the other has been closed. They can be used to establish stability margins as done in the single-loop case.

The inequalities corresponding to the robust margin problem in the first channel are

$$\left|1+p_{11}g_1(j\omega)\right| \geq \frac{1}{1.8}, \text{ for all } P\in\mathcal{P},\ \omega\geq 0$$

$$\left|1+\frac{\det P}{p_{22}}g_1(j\omega)\right| \geq \frac{1}{1.8}, \text{ for all } P\in\mathcal{P},\ \omega\geq 0$$

while the inequalities corresponding to robust sensitivity problems in the first channel are

$$\left|s_{11}(j\omega)\right| \leq \left|\frac{|p_{22}|+|p_{21}|\eta_{21}}{\det P\, g_1+p_{22}}(j\omega)\right| \leq \eta_{11}, \text{ for all } P\in\mathcal{P},\ \omega\in[0,10]$$

$$\left|s_{12}(j\omega)\right| \leq \left|\frac{|p_{12}|+|p_{22}|\eta_{22}}{\det P\, g_1+p_{22}}(j\omega)\right| \leq \eta_{12}, \text{ for all } P\in\mathcal{P},\ \omega\in[0,10]$$

Note that all closed-loop transfer functions depend on both $g_1(s)$ and $g_2(s)$, yet $g_2(s)$ is unknown. The above inequality reflects some conservatism due to this fact. The bounds for each of the above four constraints can be solved using the function `genbnds`.

Based on the characteristic equation, the nominal plant in the 1st channel is some plant $p_{11,nom}(s)$ from the family `P`, and the nominal loop function is $L_{11,nom}(s) = p_{11,nom}(s)g_1(s)$.

After the controller $g_1(s)$ is designed such that $1+L_{11,nom}(s)$ is stable and the above four constraint are satisfied, we can turn our attention to the next channel. In this example, design of $g_2(s)$ is also the last step in the sequential design, hence, it is the only remaining unknown controller to be designed.

The inequalities corresponding to the robust margin problem in the 1st and 2nd channels are

$$\left|1+p_{11}^{e}g_1(j\omega)\right| \geq \frac{1}{1.8}, \text{ for all } P\in\mathcal{P},\ \omega\geq 0$$

$$\left|1+p_{22}^{e}g_2(j\omega)\right| \geq \frac{1}{1.8}, \text{ for all } P\in\mathcal{P},\ \omega\geq 0$$

The inequalities corresponding to robust sensitivity problems in the 2nd channel are

$$\left|s_{21}(j\omega)\right| = \left|\frac{-\dfrac{p_{21}g_1}{1+p_{21}g_1}}{1+p_{22}^{e}g_2}(j\omega)\right| \leq \eta_{21}, \text{ for all } P\in\mathcal{P},\ \omega\in[0,10]$$

$$\left|s_{22}(j\omega)\right| = \left|\frac{1}{1+p_{22}^{e}g_2}(j\omega)\right| \leq \eta_{22}, \text{ for all } P\in\mathcal{P},\ \omega\in[0,10]$$

Again, the bounds for each of the above four constraints can be solved using the function `genbnds`. The nominal plant in the 2nd channel is some plant $p^e_{22,nom}(s)$ (with same $p_{11,nom}$) from the family $\mathcal{P}$, and the nominal loop function is $L_{22,nom}(s) = p^e_{22,nom}(s)g_2(s)$.

The problem setup and its QFT solution using the Toolbox can be found in the file `qftex15.m`.

6 Bounds and Loop Shaping

Introduction

This chapter describes the available functions for performing the two most important aspects in a QFT design: bound computation and loop shaping.

The first section introduces the basic usage of the two bound computation managers `sisobnds` and `genbnds`.

The second section covers the general loop-shaping functions available within the Interactive Design Environments `lpshape`, and `pfshape`.

Finally, the last section focuses on the notation and format used for continuous-time and discrete-time elements during loop shaping.

The Bound Computation Managers

This section introduces the basic usage of the two bound computation managers `sisobnds` and `genbnds`. `sisobnds` is applicable to single loop systems and `genbnds` is applicable to cascaded-loop and (sequentially-closed) multi-loop systems.

The algorithms used to compute the single-loop bounds (`sisobnds`) require computer memory space linearly related to the number of plant cases, frequencies and phases (except problem `ptype = 7`). If MATLAB returns the message: "out of memory...," consider reducing the number of cases and/or frequencies and/or phases. In general, if the number of plant cases is *n* and the length of the phase array is *m*, the bound solving manager `sisobnds` (excluding problem `ptype = 7`) utilizes approximately three (*n*x*m*) real matrices, while problem `ptype = 7` utilizes approximately three `(n!/(n-2)!xm)` real matrices (thus, problem `ptype = 7` takes the longest time to compute bounds).

Single Loop Bound Manager

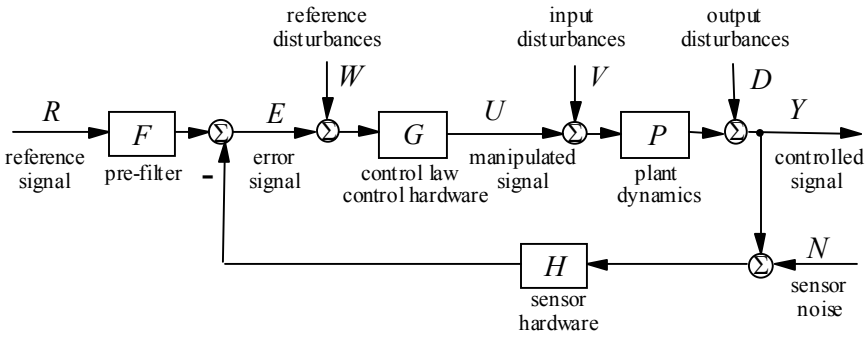
A generic call to the function `sisobnds` is as follows

```
bdb = sisobnds(ptype,w,Ws,P,R,nom,C,loc,phs)
```

The function requires a myriad of inputs, yet not all need be specified. Those that are not specified or entered as an empty matrix [ ] will automatically revert to their default values. The following table describes default values.

Arguments	Defaults
P,G,H	1
R	0
nom	1
loc	1
phs	[0°:-5°:-360°]

Below you will find explanations for the input and output variables with respect to the block diagram below.



bdb

Contains the bounds (in *dB*) on the nominal loop $L_0 = L_0 G_0 H_0$. The following is the general structure of a bound vector:

[upper bound; lower bound; frequency; problem type]

where upper bound and lower bound are vectors denoting where the nominal loop should lie above and below, respectively.

A bound can take on four different types at a fixed frequency and a fixed phase:

- The above bound is a real number and the below bound is a real number, or only one of the two exists. In such cases, the loop response must lie above the above bound *or* below the below bound.
- The above or below bounds (or both) can be any positive number, in which case the above bound is set to 20*log10(`myeps`) *dB* and the below bound is set to 20*log10(1/`myeps`) *dB* (`myeps=1e-16`). That is, for example, if the above bound is 20*log10(`myeps`), there is no minimum gain necessary for the loop magnitude. For convenience, the function `plotbnds` does not show such portions of a bound.
- There is no real positive controller gain that can solve the problem (referred to as "no LTI solution..."), in which case the above bound is set to 248 *dB* and the below bound is set to -248 *dB*.
- The bound is non-connected. This situation typically occurs with poor template boundary grid and in `genbnds(...)`. Another possibility is due to intersection, in which case you should loop shape with an un-intersected set of bounds. The above bound is set to 302 *dB* and the below bound is set to -302 *dB*.

Note: The bound computation algorithms can produce unrealistic outputs (for the most part, you should not encounter such cases). For instance, take a look at the input disturbance bounds in Example 9: Uncertain Flexible Mechanism. The isolated points representing bounds should have not been there and are due to numerical inaccuracy. A bound, however, should always prohibit the nominal loop from the critical point (with the exception of `genbnds(...)` with $c(j\omega) \neq 1$). If you zoom in around the critical point (-180°,0dB) in the input disturbance bound plot, you will see that there is a bound prohibiting the nominal loop from that region.

ptype

The integer argument `ptype` defines the particular closed-loop problem of interest as shown below.

`ptype`	I/O Problem
1	$\left\|\frac{PGH}{1+PGH}\right\| \leq Ws_1$

2	$\left\|\frac{1}{1+PGH}\right\| \le Ws_2$
3	$\left\|\frac{P}{1+PGH}\right\| \le Ws_3$
4	$\left\|\frac{G}{1+PGH}\right\| \le Ws_4$
5	$\left\|\frac{GH}{1+PGH}\right\| \le Ws_5$
6	$\left\|\frac{PG}{1+PGH}\right\| \le Ws_6$
7	$Ws_{7a} \le \left\|F\frac{PG}{1+PGH}\right\| £ Ws_{7b}$
8	$\left\|\frac{H}{1+PGH}\right\| \le Ws_8$
9	$\left\|\frac{PH}{1+PGH}\right\| \le Ws_9$

w

A frequency vector (rad/sec; must be a subset of the frequencies in an FRD model).

Ws

A performance weight. Can be a single number, vector or an LTI model AN LTI model can be used to specify different specifications for each plant case at each frequency.

In `sisobnds(7,...)`, the specification must consist of upper and lower values (see Example 2);

R

A disk radius in a multiplicative uncertain plant model. Specific values can be assigned to each case in a mixed parametric/non-parametric uncertain plant. It is represented by an LTI/FRD object.

nom

An integer corresponding to the nominal plant index in the LTI/FRD model. If there is another parametric uncertain transfer function in the loop (i.e., `C`) then `nom` should be a two-number vector specifying both nominal cases for `P` and for `C`.

loc

An integer (1 or 2) indicating location of the controller to be designed. `loc = 1` (default) implies $G(s)$ is the controller and hence the input variable `c` is the known $H(s)$. `loc = 2` implies $H(s)$ is the controller and `c` is the known $G(s)$.

phs

A vector defining the resolution of the computed bounds along the phase axis. The default is [0°:-5°:-360°].

Note: If the phase used for computing bounds is not the default one, it must also be used in all other functions. The phase -180° should always be part of the phase vector `phs`.

General Bound Manager

The special function `genbnds` was written with the advanced user in mind. It can be used in cascaded-loop designs (Example 8) and sequentially closed multi-loop designs (Example 15). With `ptype=10`, `genbnds` can be used to solve all the problems in `sisobnds` except for `ptype = 7`.

The general call to the function `genbnds` is as follows

```
bdb = genbnds(ptype,w,Ws,A,B,C,D,Pnom,phs);
```

Note that if `C`$\neq$`1` the resulting bounds may not include the critical point (-1,0) or (-180°, 0dB).

ptype

The argument `ptype` defines the particular closed-loop transfer function of interest as shown in the table below

`ptype`	**I/O Problem**
10	$\left\|\frac{A+BG}{C+DG}\right\| \le Ws_{10}$
11	$\left\|\frac{\|A\|+\|B\|\|G\|}{C+DG}\right\| \le Ws_{11}$

A, B, C, and D

`A`, `B`, `C`, `D`, and `P0` can be constants or LTI/FRD models. They are functions of the various plants and controllers in cascaded-loop and multi-loop systems.

Pnom

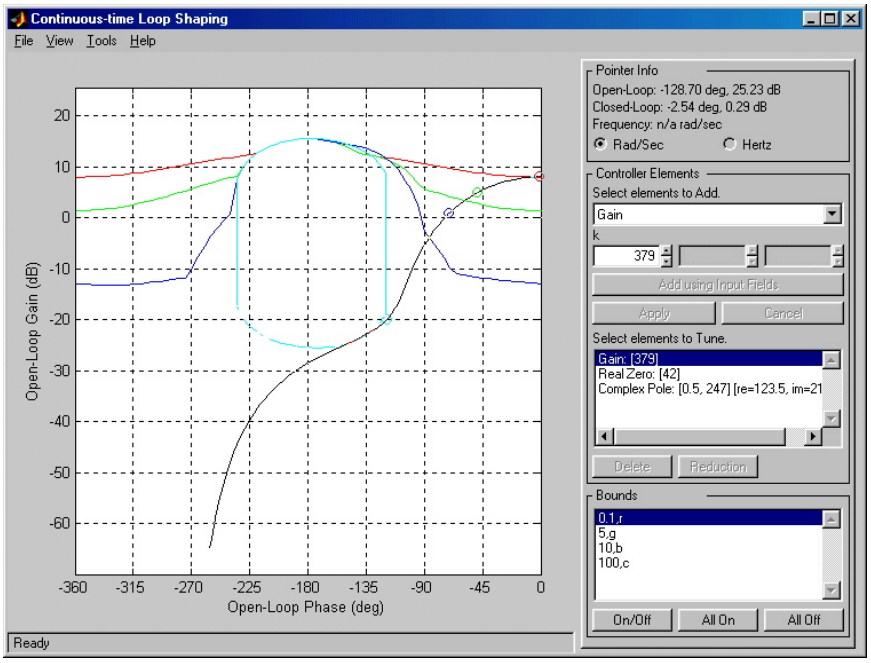
The input argument `Pnom` denotes the nominal plant such that the bounds are defined for the open loop function $L = G*Pnom$. Note that `Pnom` is not an index (as in `sisobnds`), rather it is an LTI/FRD object of the nominal plant model.

The Interactive Design Environment (IDE)

This section covers the general options available within the Interactive Design Environments `lpshape` and `pfshape`. The screen captures shown here are of an `lpshape` session running on a PC. Note that no matter what platform is being used the menus will only differ in a cosmetic sense.

IDE Menus

The Interactive Design Environment (IDE) functions provide access to the specific commands used in a QFT design [19]. The commands are now shown for the continuous-time loop-shaping function `lpshape`. A typical screen is shown below.



File

The **File** menu contains items related to opening and saving IDE created files and sending information to the workspace.

Open... displays a file selection dialog box that asks you for the name of the MAT-file created from within IDE using the **Save** option or created from the command line using the `getqft` function.

Save... displays a file selection dialog box that asks you for the name of a MAT-type file in which to store the present elements.

Each IDE function will save into a user specified file with a default extension specified in the following table. (these choices are not unique, any other extension can be used).

IDE Function	File Name Extension
`lpshape`	`*.shp` (continuous time) `*.dsh` (discrete time)
`pfshape`	`*.fsh` (continuous time) `*.dfs` (discrete time)

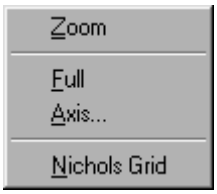
Import... opens a dialog box that allows you to enter a variable name for an LTI model to be transferred from the workspace.

Export... opens a dialog box that allows you to enter a variable name for the LTI model to be transferred to the workspace.

Print to Figure sends the graphical contents to a new figure for custom editing and exporting. See MATLAB's Reference Guide for more details.

Exit prompts you to exit and save the current design, exit without saving the current design, or cancel the exit.

View



Zoom toggles the zoom mode between on or off.

Full sets the axis limits to the FULL setting. The FULL setting is defined initially by the environment and can be changed using the **Axis...** option described later.

Axis... opens a dialog box that allows you to manually specify axis limits.

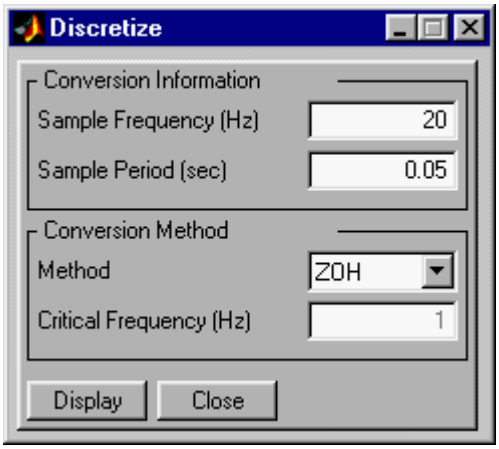
Nichols Grid toggles the display of the Nichols grid over the open-loop grid lines.

Tools

The **Tools** menu contains general commands such as viewing the plant elements, controller discretization, stability analysis, altering the working frequency array, external bode plots, and storing and recalling elements while within the design environment.

Plant displays the plant elements (unless entered as frequency response).

Discretize… provides a dialog to compare the continuous-time controller with a discretized controller using various user-selected discretization methods and sampling time.



Stability analyzes the nominal closed-loop stability (this is done by computing the eigenvalues of the closed-loop state-space matrix).

Frequency... opens a dialog box that allows you to change the first, last, and number of points in the working frequency array. The **Pad Frequency Vector** option in this mode adds additional points to improve smoothness of the frequency response plot when underdamped second-order elements are present. The **Pad** option may add a large number of frequencies and hence slow down the interactive design process. As values of damping and natural frequencies tend to be modified during a design process, try to periodically turn off then on again the **Pad** option. This will clean up the frequency vector.

Warning: It is possible that between two consecutive frequency points the phase of the response plot is discontinuous with jumps of more than 180° but less than 360°. This may occur when the resolution of frequency array is too crude at that band and the program will have a hard time figuring out how to connect a line between these two response points. In general, be careful when you see a straight line plotted with a near 180° span. Whenever possible first use

$$\frac{\omega_n^2}{s^2+\varepsilon s+\omega_n^2}, 0<\varepsilon<<1$$

instead of a pure oscillator

$$\frac{\omega_n^2}{s^2+\omega_n^2}$$

The significance of such a change is negligible from a design view point.

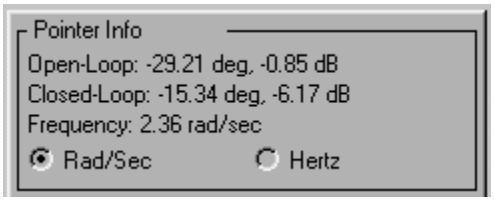
Bode Plots… plots the current design in a bode plot format. The open loop, closed-loop sensitivity and closed-loop complimentary functions are shown.

Store saves the present set of elements within the IDE (useful only within IDE for quick recall of a previous design).

Recall retrieves the last saved set of elements from within the IDE. If none have been saved, then this option returns the initial elements.

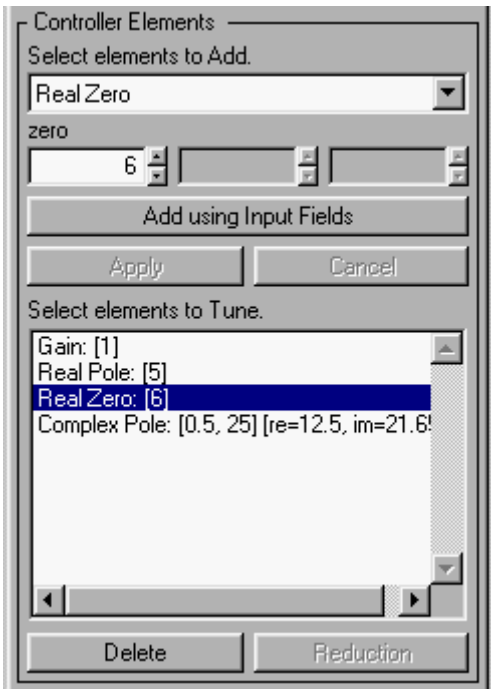
Design Control Panel

Pointer Info



The **Pointer Info** section provides mouse movement feedback whether the mouse is over the loop response or not. If the mouse is over the loop response, the nearest frequency is displayed in the units specified by the radio buttons.

Controller Elements



The **Controller Elements** section provides both numerical and graphical addition of elements. For mode details, please see the Design Elements section later in this chapter.

Adding Elements Numerically is accomplished by selecting a desired element from the **Element Popup**, entering the required values in the enabled input fields, and pressing **Add Using Input Fields**. The new loop response is displayed with the original for immediate comparison. The sliders to the right of each input field can be used to fine tune individual parameters of the element. Pressing **Apply** or selecting a new element to add or edit permanently accepts the element. The element can be deleted by pressing the **Delete** button.

Adding Elements Graphically is accomplished by selecting a desired element from the Element Popup and floating the mouse over the loop response.

Gain takes the difference between the initial frequency location and the current pointer location to modify the DC gain.

First Order adds a real, stable pole (zero) based on a negative (positive) phase difference at the selected frequency (red dot) between the initial location and the current pointer location. The necessary pole (zero) value is computed to result in the desired phase change. If the desired term is a pole, the new loop frequency response will have a magnitude reduction at the frequency. A zero will result in a magnitude increase at that frequency.

Limitation: |phase difference| < 88 degrees.

Exception: In `pfshape, the` implementation is based on magnitude difference.

Second Order adds a stable complex pole (zero) based on a negative (positive) phase difference and magnitude difference at the selected frequency (red dot) between the initial location and the current pointer location. The necessary natural frequency and damping ratio are computed to matched desired phase and magnitude change. Due to the nonlinear relation between natural frequency and damping ration and the associated phase and magnitudes, the feasible domain is limited (this limitations is removed in the **Super 2nd** element below).

Limitation: |phase difference| < 176 degrees.

Exception: In `pfshape, the` implementation is based on magnitude difference.

Lead/Lag adds a stable lead (lag) based on a negative (positive) phase difference at the selected frequency (red dot) between the initial location and the current pointer location. The necessary zero and pole pair are computed such that the element achieves its maximal phase at the selected frequency. If the desired term is a lag, the new loop frequency response will have a magnitude reduction at the frequency. A lead will result in a magnitude increase at that frequency.

Limitation: |phase difference| < 88 degrees.

Notch adds a notch based on the magnitude difference at the selected frequency (red dot) between the initial location and the current pointer location. If a magnitude reduction is desired, the pole's damping ration is set to 0.5 and the other is computed to achieve the magnitude change at the selected frequency.

Super 2nd adds a stable 2nd order zero over a stable 2nd order pole based on the phase difference and magnitude difference at the selected frequency (red dot) between the initial location and the current pointer location. The four free parameters allow for any match of desired magnitude and phase change.

Limitation: |phase difference| < 176 degrees.

Complex Lead/Lag adds a stable complex zero and a stable complex pole based on the phase difference and magnitude difference at the selected frequency (red dot) between the initial location and the current pointer location. Both terms have damping ratios of 0.45. This term can actually provide either lead or lag dynamics.

Limitation: |phase difference| < 176 degrees.

Note that the mouse pointer changes to a directional pointer representing the type of movement that will be registered when it is located on any point on the response plot (but not on the straight line connecting any two points). Upon selection of a point, a (red) marker appears on the plot at that frequency. Only after this point has been found, can the response plot be grabbed and moved to a new location using the selected element. To continue moving the plot you can edit the value of the new element by re-grabbing the marker. The process of dragging may appear slow on low-end computers (especially if the frequency vector is large).

In general, in order to relocate the frequency response at the chosen frequency to a new location on the plot (with different magnitude and phase), for an element type select the following:

Element Type	Match Magnitude Change	Match Phase Change
real pole/zero	no	yes
complex pole/zero	yes	yes
lead/lag	no	yes
notch	yes	no
super 2nd	yes	yes
complex lead/lag	no	yes

Note: The super 2nd element offers the best chance of successfully matching both magnitude and phase (with realizable elements), and can be added only via a mouse operation; upon completion, the super element is stored as separate zero and pole elements (i.e., it cannot be edited, deleted or iterated on as a super 2nd).

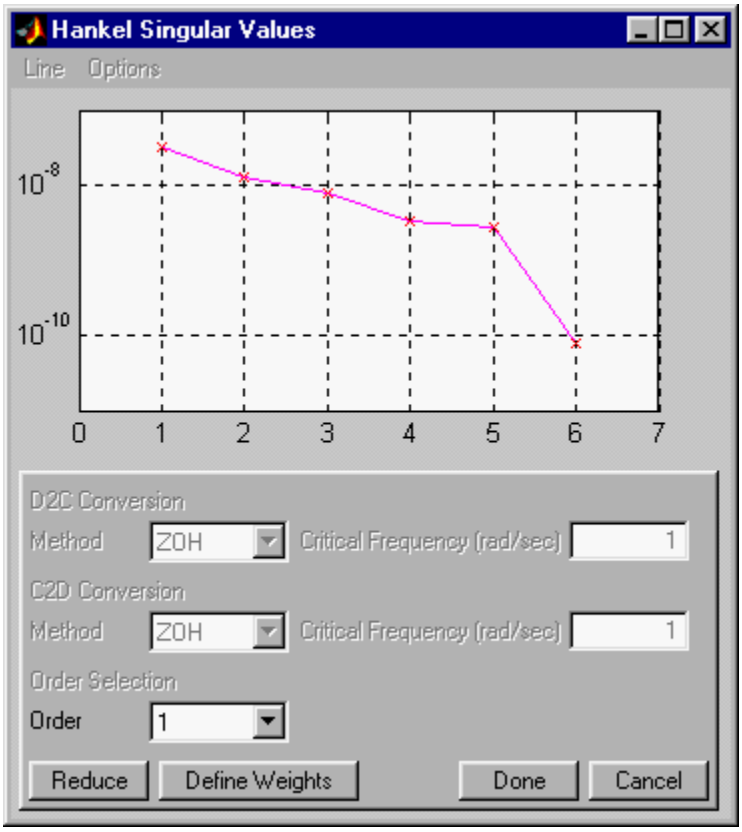
Note: Only stable and minimum-phase elements can be added via mouse operations.

If the mouse operation on a certain element calls for complex coefficients, unstable, or non-minimum phase, you will be prompted for such a situation and the element will not be implemented (see Table 4: Standard continuous-time elements in this Toolbox and Table 5: Standard discrete-time elements in this Toolbox for specific formats).

Editing is accomplished by selecting the desired element and either editing the parameters in the input fields or using the sliders to tune the parameters. *You can continue to edit the particular element* a*s long as the red dot is visible.*

Deleting is accomplished by selecting the desired elements (the gain cannot be deleted) and pressing the Delete button. Multiple elements can be selected by holding down the <ctrl> or <shift> keys while selecting.

Reduction is accomplished by selecting more than one element and pressing the **Reduction** button. A typical screen is shown below.

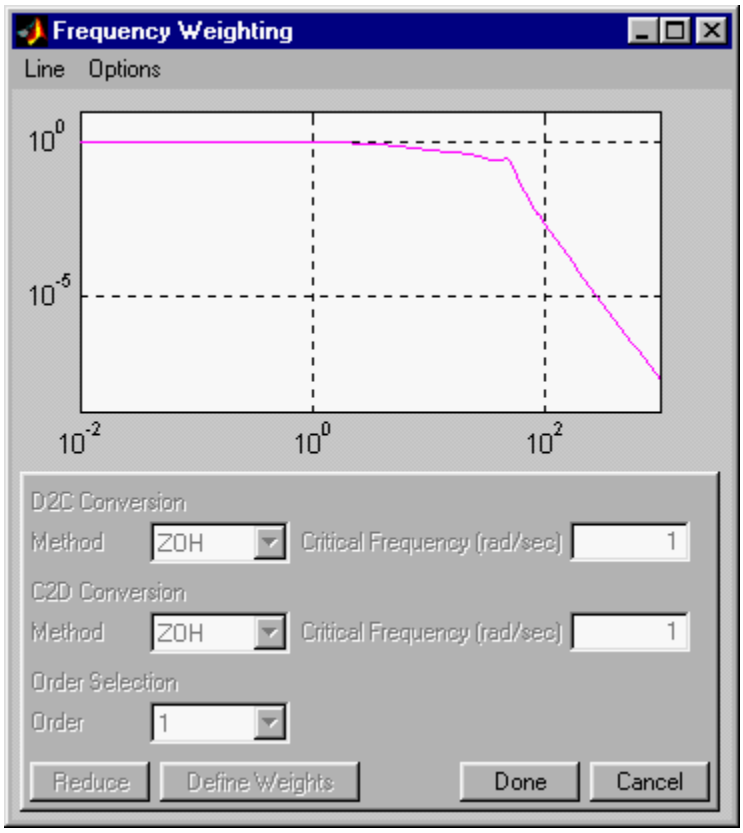


The algorithms were adapted from [20,28] (for an open-loop and closed-loop Model Reduction Toolbox please contact the author of [20,28] or us). The reduction is not applicable to discrete-time systems. In this mode, you first select terms from the displayed elements. Only a proper or strictly proper controller with stable poles (those with negative real part) can be selected for reduction. Pressing the **Reduction** button results in a new dialog box showing a plot of the Hankel Singular Values. At this stage you can do the following:

Reduce - performs reduction to the user specified order. The result will be the reduced-order response plot (dashed line) superimposed over that of the full-order plot and a list of the reduced-order elements.

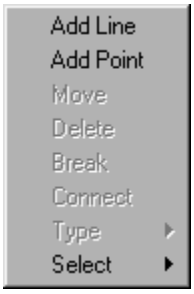
Define Weights - replaces the HSV plot with the magnitude plot of the frequency response that allows you to place affine frequency weights for reduction. These weights can be used to allow "trade-off" of errors between full-order and reduced order frequency responses. A typical screen is shown below.

Cancel - ignore the present reduction and close HSV dialog box.



The **Line** menu contains all the operations that can be performed on line segments. Each option is enabled or disabled depending upon the selection of line segments.

Line



Add Line allows addition of new line segments. In the process of adding a new line, if any line segments are either above or below they are eliminated. All new lines are the default type of **Both** (Red).

Add Point allows addition of new points.

Move, **Delete**, **Break**, **Connect**, and **Type** are only enabled when line segments are selected. Line segments are selected by either placing the mouse pointer over the desired segment or using the **Select** option.

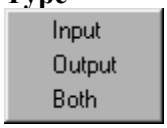
Move changes the pointer into a fleur (four-headed arrow) and upon holding down the mouse button over the specific segment(s) allows the user to move the selected segments to the desired location. **Move** can only be used with a single segment or segments that are connected.

Delete removes selected line segments.

Break separates segments on two ends of a selected line.

Connect links any two selected segments.

Type



Type allows for the changing of a line segment designation. The possible designations are: **Input**, **Output** and **Input-Output** (**Both**), each signifying the type of frequency weighted model order reduction in effect over that frequency range.

Select

Select allows the user to select either **All** or **None** of the present line segments.

The **Options** menu includes a number of miscellaneous operations.

Options

Full returns the axis to its original limits.

Zoom allows the user to change the present axis limits by defining new axis limits with a bounding box.

Clear removes all present line segments.

Open... opens a file dialog box that allows for the retrieval of saved line segments.

Save... opens a file dialog box that allows for the saving of present line segments.

Select - cancels the present reduction and return to the element selection mode.

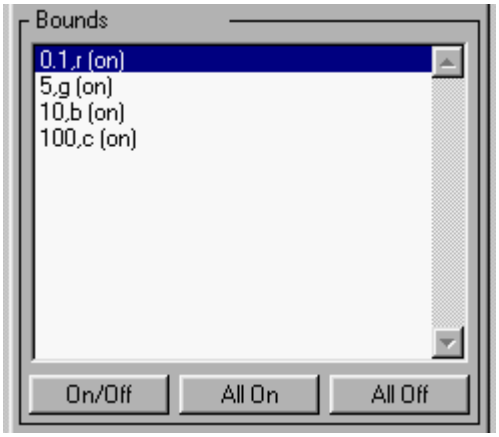
Done - accepts the present reduction and close the HSV dialog box.

In a discrete-time setting, the reduction algorithms transform the controller into a continuous-time version using any of the methods in the **D2C Conversion** pull down menu, followed by the reduction and then transformed back to the discrete-time setting using the method selected from the **C2D Conversion** pull down menu. You may need to experiment with different combinations of these methods to achieve best reduction.

Hints:

- Though you are reducing the controller's order with an open-loop measure, in effect you will be considering its effect on the closed-loop response. This is because the reduction effects are shown together with closed-loop specifications, i.e., the bounds.
- Stability of the reduced order model that was obtained using with frequency weights is not guaranteed.
- The selected order of the reduced-order model should be based on the relative values of Hankel Singular Values (HSV) shown in the plot. Look for a sharp drop in HSV value from one order to the next.
- After reduction, the relative degree of the controller may not be the same as before reduction (usually the returned relative degree is one).
- The reduction is done by first deriving a minimal-order balanced model. In some cases, you will have almost non-minimal modes removed even before reduction is done.
- If you are working with a large-order controller (say 20), or there is a large magnitude difference between the largest pole and smallest pole, you should perform reduction in several steps. At each step you will select a subset from all possible elements, preferably those with "close" break frequencies.
- Try to experiment with weight line types to improve quality of model fit.
- Try to avoid repeated poles in model reduction function. When repeated poles are present, the reduction algorithms must use `logm`; a slow and numerically suspect function. To avoid this situation, for example, in a repeated pair we suggest that you first modify the value of one pole by a small number (which will not affect loop response).
- Reduction may result in a new zero very far from the origin (in either left or right half planes). Such a zero, is often much faster than the rest of the poles and zeros, and can be deleted without affecting the response.

Bounds



The **Bounds** section provides the ability to selectively turn bounds on and off. The first value is the frequency at which the bound was computed, the second is its color, and the third is its display state of on or off. Double-clicking a selected bound toggles its display state.

Design Elements

The following notation and format is used for continuous continuous-time transfer function elements:

Table 4: Standard continuous-time elements in this Toolbox

Element	Mathematical Form
Real pole	$\frac{1}{s/p+1}$
Real zero	$s/z+1$
Complex pole	$\frac{1}{s^2/\omega_n^2+2\zeta s/\omega_n+1}$
Complex zero	$s^2/\omega_n^2+2\zeta s/\omega_n+1$
Super 2nd (2/2)	$\frac{a_1 s^2+a_2 s+1}{b_1 s^2+b_2 s+1}$
Integrator/Differentiator	$\frac{1}{s^n}$ or s^n
Lead or Lag	$\frac{s/z+1}{s/p+1}$
Notch	$\frac{s^2/\omega_n^2+2\zeta_1 s/\omega_n+1}{s^2/\omega_n^2+2\zeta_2 s/\omega_n+1}$ $(\zeta_1=0.5 \text{ or } \zeta_2=0.5)$
Complex lead	$\frac{s^2+2ads+a^2}{s^2+2bds+b^2}\frac{b^2}{a^2}$ $(d=0.45)$

Within the continuous-time IDE, the elements are always visible within the **Element Listbox**, for example, as shown below.

Complex poles and zeros are shown with their [zeta, wn] values followed by the root location (one from the complex-conjugate pair). Lead/Lag elements are shown with their [phase, frequency] values followed by the values of the zero and pole. Notch elements are shown with their [zeta1,zeta2,wn] values. The transfer function corresponding to the above elements (Table 4) is

$$1\times\frac{1}{s}\times\frac{1}{s+1}\times\frac{1}{\frac{s}{1111}+1}\times\frac{\frac{s}{10}+1}{1}\times\frac{1}{\frac{s^2}{44^2}+\frac{2\cdot 0.5}{44}s+1}\times\frac{\frac{s^2}{123.5^2}+\frac{2\cdot 0.1}{123.5}s+1}{1}\times\frac{\frac{s}{5.973}+1}{\frac{s}{20.26}+1}\times\frac{\frac{s^2}{77^2}+\frac{2\cdot 0.1}{77}s+1}{\frac{s^2}{77^2}+\frac{2\cdot 0.5}{77}s+1}$$

Treatment of elements in discrete-time is quite different in the way in how you define them. Although we offer similar elements as used in continuous-time types, such as 1st or 2nd orders, you have two choices to define a discrete-time controller (or pre-filter). With IDE functions, discrete-time elements are manipulated in terms of their continuous-time equivalence (using *z*-domain transform) as shown in table below (t_s = sampling time in seconds). That is, to add a first-order pole, you will enter the continuous-time value, e.g., $p = 20$, and the program will convert it to its discrete-time value as shown below.

Table 5: Standard discrete-time elements in this Toolbox

Element	Mathematical Form
Real pole	$\frac{\lvert 1-a\rvert\cdot z}{z-a}$ $a=e^{-pt_s}$ (p = equivalent *s*-plane pole location)
Real zero	$\frac{z-b}{\lvert 1-b\rvert\cdot z}$ $b=e^{-zt_s}$ (z = equivalent *s*-plane zero location)
Complex Pole	$\left\lvert\frac{1-2\cos(bt_s)e^{-at_s}+e^{-2at_s}}{\cos(bt_s)e^{-at_s}}\right\rvert\cdot\frac{\cos(bt_s)e^{-at_s}z^2}{z^2-2\cos(bt_s)e^{-at_s}z+e}$ $a=\zeta\omega_n, b=\omega_n\sqrt{1-\zeta^2}$ (*s*-plane equivalence)
Complex Zero	$\left\lvert\frac{\cos(bt_s)e^{-at_s}}{1-2\cos(bt_s)e^{-at_s}+e^{-2at_s}}\right\rvert\cdot\frac{z^2-2\cos(bt_s)e^{-at_s}z+e}{\cos(bt_s)e^{-at_s}z^2}$ $a=\zeta\omega_n, b=\omega_n\sqrt{1-\zeta^2}$ (*s*-plane equivalence)
Super 2nd (2/2)	$\left\lvert\frac{b_1+b_2+1}{a_1+a_2+1}\right\rvert\cdot\frac{a_1z^2+a_2z+1}{b_1z^2+b_2z+1}$
Predict/Delay	z^n or z^{-n}

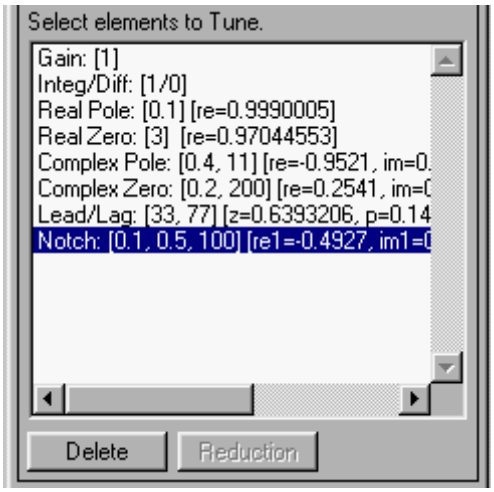
Table 5 (Cont.): Standard discrete-time elements in this Toolbox

Element	**Mathematical Form**		
Integrator or Differentiator	#	Integ	Diff
	1	$\frac{t_s z}{z-1}$	$\frac{z-1}{t_s z}$
	2	$\frac{t_s^2 z}{(z-1)^2}$	$\frac{(z-1)^2}{t_s^2 z}$
	3	$\frac{t_s^3 z(z+1)}{(z-1)^3}$	$\frac{(z-1)^3}{t_s^3 z(z+1)}$
Lead or Lag	$\frac{\lvert 1-p\rvert}{\lvert 1-r\rvert}\frac{z-r}{z-p}$		
Notch	$\left\lvert\frac{1-2cos(dt_s)e^{-ct_s}z+e^{-2ct_s}}{1-2cos(bt_s)e^{-at_s}z+e^{-2at_s}}\right\rvert \cdot \frac{z^2-2cos(bt_s)e^{-at_s}z+e^{-2at_s}}{z^2-2cos(dt_s)e^{-cT}z+e^{-2ct_s}}$ $a=\zeta_1\omega_n,\ c=\zeta_2\omega_n,\ b=\omega_n\sqrt{1-\zeta_1^2},\ d=\omega_n\sqrt{1-\zeta_2^2}$		
Complex Lead	$\left\lvert\frac{1-2\cos(b_1t_s)e^{-a_1t_s}+e^{-2a_1t_s}}{1-2\cos(b_2t_s)e^{-a_2t_s}+e^{-2a_2t_s}}\right\rvert \cdot \frac{z^2-2\cos(b_1t_s)e^{-a_1t_s}z+e^{-2a_1t_s}}{z^2-2\cos(b_2t_s)e^{-a_2t_s}z+e^{-2a_2t_s}}$ $a=\zeta\omega_n, b=\omega_n\sqrt{1-\zeta^2}, \zeta=0.45$ (s-domain equivalence)		

One choice to define elements is to do so within any IDE function. In that mode you enter values in continuous-time and the program uses the z-transform to convert them into discrete-time (as shown above). It is easier to predict the resulting frequency response of continuous-time over discrete-time elements within IDE. A good discussion on discrete-time frequency response of various elements can be found in [21].

Note that as done in the continuous-time IDE, all elements have unity DC gain. The only difference is in integrator/differentiator elements. A continuous-time integrator has a unity gain at ω = 1, while a discrete-time integrator from *z*-transform tables requires an additional t_S gain to have unity gain at that frequency.

Within the discrete-time IDE, the elements are always visible within the **Element Listbox**, for example, as shown below. $t_s = 0.01\,\text{sec}$.



Real poles and zeros are shown with their continuous-time location (used only for ease of manipulation within IDE) followed by the *z*-transformed discrete-time root location. Complex poles and zeros are shown with their continuous-time [zeta, wn] values followed by the *z*-transformed discrete-time root location. Lead/Lag elements are shown with their continuous-time [phase, frequency] values followed by the values of the *z*-transformed discrete-time zero and pole. Notch elements are shown only with their continuous-time [zeta1,zeta2,wn] values. The transfer function corresponding to the above elements in Table 5 is therefore (using `format short e`)

$$1\times\frac{z}{z-1}\times\frac{9.995e-4\,z}{z-9.99e-1}\times\frac{z-9.7045e-1}{2.9554e-2\,z}\times\frac{7.3712e-1\,z}{z^2-6.7436e-1\,z+4.1148e-1}$$

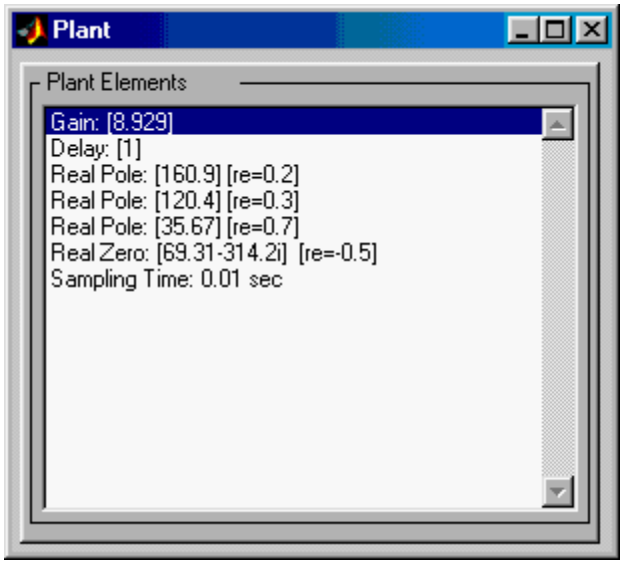
$$\times\frac{z^2+0.5082\;z+0.4493}{1.9575\,z}\times\frac{2.3698(z-0.6393)}{z-0.1453}\times\frac{0.6984(z^2-0.9854\;z+0.8187)}{z^2-0.7859\;z+0.3679}$$

The second choice to define elements is to pass them as input arguments into an IDE function. In that mode you are passing true discrete-time LTI model. This may lead to some interesting results. The inverse *z*-transform of a discrete-time pole (or zero) located between [-1,0) is a single complex-valued continuous-time pole (zero).

For example, suppose you do the following

```
P = tf(conv([1,0],[1,.5]),conv([1,-.3],conv([1,-.2],[1,-.7])));
P.Ts = 0.01;
lpshape([],[],P)
```

Within the discrete-time IDE, if you select **Tools**|**Plant**, the plant elements are displayed in the QFT Toolbox standard format



Two elements above may appear unusual. These are the delay element and the real zero with a *complex-valued* continuous-time zero. To understand why we have such elements, let us re-write the passed numerator and denominator in a form that uses only standard QFT Toolbox elements (Table 5: Standard discrete-time elements in this Toolbox)

$$\frac{z(z+0.5)}{(z-0.2)(z-0.3)(z-0.7)}$$
$$= \frac{1.5}{0.8\times 0.7\times 0.3}\times\frac{1}{z^2}\frac{z}{1}\times\frac{z+0.5}{0.5z}\times\frac{0.8z}{z-0.2}\times\frac{0.7z}{z-0.3}\times\frac{0.3z}{z-0.7}$$
$$= 8.929\times\frac{1}{z}\times\frac{z+0.5}{0.5z}\times\frac{0.8z}{z-0.2}\times\frac{0.7z}{z-0.3}\times\frac{0.3z}{z-0.7}$$

The non-unity gain element is required to compensate for forcing each standard element to have unity steady-state gain at $z = 1$. The z and z^{-1} elements are often used for padding purposes. The real discrete-time zero at $z = 0.5$ corresponds (via z-transform pair) to a single continuous-time complex pole $p = 69-314i$.

7 Reference

Functions by Class

This section contains detailed descriptions of all QFT Toolbox functions. It begins with a list of the functions grouped by subject area and continues with the reference entries in alphabetical order. Information is also available through the online help facility.

Interactive Design Environments (IDE)	
`lpshape`	Controller design
`pfshape`	Pre-filter design

Specialized X-Y Graphs	
`plotbnds`	Nichols plot of bounds
`plottmpl`	Nichols plot of templates

Arithmetic	
`addtmpl`	Add LTI/FRD arrays
`cltmpl`	Closed-loop LTI/FRD arrays from open-loop arrays
`multmpl`	Multiply LTI/FRD arrays

Bound Computation	
`sisobnds`	Single-Input/Single-Output setting bounds
`genbnds`	General setting bounds

Bound Utility	
`grpbnds`	Group several bounds into a single variable
`sectbnds`	Intersection of bounds

Analysis	
`chksiso`	Analysis of a SISO closed-loop configuration given open-loop LTI/FRD models
`chkgen`	Analysis of a general closed-loop configuration given open-loop LTI/FRD models

File Operation	
`putqft`	Import a design into an IDE file
`getqft`	Export a design from an IDE file

Examples	
`qftex#`	Solutions to the examples in Chapter 5

addtmpl

Purpose

Add LTI and/or FRD arrays.

Synopsis

```
P = addtmpl(P1,P2,utype)
```

Description

`addtmpl` produces an addition of two SISO objects or arrays (LTI and/or FRD models). If one is an FRD, the result is an FRD model.

`utype = 1` indicates correlated uncertainties (default) and `utype = 2` indicates uncorrelated uncertainties. In an uncorrelated case, each element in one array is matched with all the elements in the other array. When the dimensions of the arrays are the same, say n (note: array dimensions are not I/O dimensions), `P1+P2` produces the same result as `addtmpl(P1,P2,1)` — an object of array dimension n. `addtmpl(P1,P2,2)` produces an object of array dimension n^2. If the array dimensions are different, an uncorrelated case is assumed.

`addtmpl` works with both continuous and discrete systems (both systems must have the same sampling time).

Examples

Consider addition of two transfer function sets given by

$$P_1(s)=\frac{1}{s+a},\quad a\in[1,10] \qquad P_2(s)=\frac{b}{s+2},\quad b\in[0.1,0.5]$$

We first form LTI arrays to represent the above models using linear parameter space grids

```
c = 1;
for a = linspace(1,10,10),
 P1(1,1,c) = tf(1,[1,a]);  c = c + 1;
end

c = 1;
for b = linspace(0.1,0.5,10),
 P2(1,1,c) = tf(1,[1,2]);  c = c + 1;
end
```

The addition is computed from

```
P = addtmpl(P1,P2,2);
```

Due to uncorrelated uncertainties, the array dimension of the sum is

```
>> size(P1)
10x1 array of transfer functions
Each model has 1 output and 1 input.
```

```
>> size(P2)
10x1 array of transfer functions
Each model has 1 output and 1 input.

>> size(P)
100x1 array of transfer functions
Each model has 1 output and 1 input.
```

Note that using either

```
P = P1+P2;
```

or

```
P = addtmpl(P1,P2);
```

results in an erroneous addition since both assume correlated uncertainties (thought the results are different in that different elements are paired in the summation).

See Also

`cltmpl`, `multmpl`

chkgen

Purpose

Analysis of a general closed-loop configuration given open-loop LTI and/or FRD models

Synopsis

```
chkgen(ptype,w,Ws,A,B,C,D,G)
err = chkgen(ptype,w,Ws,A,B,C,D,G)
```

Description

`chkgen` plots at each frequency the maximal magnitude of a specified closed-loop design with respect to uncertainties. This function is used in multivariable control design settings.

`A`, `B`, `C`, `D`, and `G` are LTI/FRD objects or arrays; `Ws`, a weight, can be a single number, vector or an LTI or FRD model; `w` is a frequency vector (rad/sec; must be a subset of the frequencies in an FRD model).

The argument `ptype` defines the particular closed-loop I/O problem of interest as shown in the table below

ptype	I/O Problem
10	$\left\|\frac{A+BG}{C+DG}\right\| \le W_s$
11	$\left\|\frac{\|A\|+\|B\|\|G\|}{C+DG}\right\| \le W_s$

`err = chkgen(ptype,w,Ws,P,R,G,H,F)` returns the difference between the closed-loop specification `Ws`, and the worst case (maximum) closed-loop configuration designated by `ptype`. In particular, the error is given by

$$err = W_s - \max_{all\ T} |T|$$

where T denoted the I/O system, defined by `ptype`.

Upon invoking `chkgen` without an output argument, the result is displayed in a standard MATLAB figure window.

`chkgen` works with both continuous and discrete systems.

Limitations

1. The function does not analyze (robust) stability. It simply computes closed-loop magnitudes at the boundary of the template (hence, it is possible that $1+L = 0$ for some interior point of the template at some frequency). That is, only an *algebraic* test is performed.
2. It is possible that during nominal loop shaping you located the loop right on the bound at a certain frequency, yet `chkgen` shows that you did not satisfy the specification at that frequency. The reason is that the bound between any two adjacent phases is interpolated using a straight line. If the resolution of the phase vector used to compute the bound was too "crude," you cannot achieve a

reasonable approximation of the actual continuous bound curve. To resolve this problem you must increase the resolution of the phase vector.

See Also

`chksiso`

chksiso

Purpose

Analysis of a single-input/single-output closed-loop design given open-loop LTI and/or FRD models

Synopsis

```
chksiso(ptype,w,Ws,P,R,G,H,F)
err = chksiso(ptype,w,Ws,P)
err = chksiso(ptype,w,Ws,P,R,G,H,F)
```

Description

`chksiso` plots at each frequency the maximal magnitude of a specified closed-loop design with respect to uncertainties. `P`, `G`, `H` and `F` are LTI/FRD objects, `Ws`, a weight, can be a single number, vector or an LTI/FRD model; `w` is a frequency vector (rad/sec; must be a subset of the frequencies in an FRD model). `R`, a magnitude vector or an LTI/FRD model, denotes multiplicative uncertainty disk radius with respect to the plant `P`.

The argument `ptype` defines the particular closed-loop I/O problem of interest as shown in the table below

ptype	I/O Problem
1	$\left\lvert F\frac{PGH}{1+PGH}\right\rvert \le Ws_1$
2	$\left\lvert F\frac{1}{1+PGH}\right\rvert \le Ws_2$
3	$\left\lvert F\frac{P}{1+PGH}\right\rvert \le Ws_3$
4	$\left\lvert F\frac{G}{1+PGH}\right\rvert \le Ws_4$
5	$\left\lvert F\frac{GH}{1+PGH}\right\rvert \le Ws_5$
6	$\left\lvert F\frac{PG}{1+PGH}\right\rvert \le Ws_6$
7	$Ws_{7a} \le \left\lvert F\frac{PG}{1+PGH}\right\rvert \le Ws_{7b}$
8	$\left\lvert F\frac{H}{1+PGH}\right\rvert \le Ws_8$
9	$\left\lvert F\frac{PH}{1+PGH}\right\rvert \le Ws_9$

Arguments	Defaults
`P,G,H,F`	`1`
`R`	`0`

`err = chksiso(ptype,w,Ws,P,R,G,H,F)` returns the difference between the closed-loop specification, `Ws`, and the worst case (maximum) closed-loop configuration designated by `ptype` and uses the necessary default values. In particular, the error is computed at each frequency from

$$err(\omega) = \min_{all\ T}(Ws(\omega) - |T(\omega)|)$$

where *T* is the (uncertain) closed-loop transfer function (defined by ptype) and `Ws` is the specification.

If the problem involves different performance specification for each plant in the uncertain set, the above *err* is computed for each such plant-spec pair, and the plot shows the minimum (at each frequency) over all such *err* values. See Example 6: Missile Stabilization for an instance of such performance problem.

Upon invoking `chksiso` without an output argument, the result is displayed in a standard MATLAB figure window.

`chksiso` works with both continuous and discrete systems.

Limitations

1. The function does not check whether the family of closed-loop systems is (robustly) stable. It simply computes closed-loop magnitudes at the boundary of the template (hence, it is possible that $1+L = 0$ (L is the open-loop function) for some interior point of the template at some frequency). That is, only an *algebraic* test is performed.

2. It is possible that during nominal loop shaping you located the loop right on the bound at a certain frequency, yet the output of `chksiso` shows that you did not satisfy the specification at that frequency. The reason is that the bound between any two adjacent phases is interpolated using a straight line. If the resolution of the phase vector used to compute the bound was too "crude," you cannot achieve a reasonable approximation of the actual continuous bound curve. To resolve this problem you must increase the resolution of the phase vector.

Examples

Suppose you wish to analyze a feedback design where the uncertain plant is

$$\mathcal{P} = \left\{ P(s) = \frac{k}{(s+5)(s+30)} :\ k = [1,10] \right\}$$

the controller is

$$G(s) = \frac{379\left(\frac{s}{42}+1\right)}{\left(\frac{s}{165}+1\right)}$$

and the performance specification is

$$\left| \frac{PG}{1+PG}(j\omega) \right| \le 1.2, \omega \ge 0, \text{ for all } P \in \mathcal{P}$$

Define the open-loop data

```
c = 1;
for k = linspace(1,10,10),
 P(1,1,c) = tf(k,conv([1,5],[1,30]));  c = c + 1;
end
G = tf(379*[1/42,1],[1/165,1]);
```

and a frequency vector

```
w = logspace(-1,3,100);
```

The desired analysis is obtained by invoking

```
chksiso(1,w,1.2,P,0,G);
```

Alternatively, we can compute the maximal magnitude response at each frequency using the following

```
Trw = abs(freqresp(Tr,w));
Trmag = abs(squeeze(freqresp(Tr,w)));
maxT = max(Trmag,[],2);
```

A similar procedure applies in a discrete-time setting. Suppose we want to check the performance of the above system in a discrete-time implementation with a 0.01 second sampling time. The discretized open-loop data can be computed from

```
Ts = 0.01;
Pz = c2d(P,Ts,'foh');
Gz = c2d(G,Ts,'foh');
```

The frequency vector is defined up to the Nyquist frequency

```
wz = logspace(-1,log10(pi/Ts),100);
```

Analysis is obtained from

```
chksiso(1,wz,1.2,Pz,0,Gz);
```

See Also

`chkgen`

cltmpl

Purpose

Closed-loop arrays from open-loop arrays

Synopsis

```
cl = cltmpl(ptype,P,G,H,F,sgn,utype)
```

Description

`clcp` forms the correlated or uncorrelated closed-loop system designated by `ptype` that defines the particular closed-loop relation of interest as shown in the table below

ptype	I/O relation	ptype	I/O relation	ptype	I/O relation
1	$F \cdot \frac{PGH}{1+PGH}$	4	$F \cdot \frac{G}{1+PGH}$	7	$F \cdot \frac{PG}{1+PGH}$
2	$F \cdot \frac{1}{1+PGH}$	5	$F \cdot \frac{GH}{1+PGH}$	8	$F \cdot \frac{H}{1+PGH}$
3	$F \cdot \frac{P}{1+PGH}$	6	$F \cdot \frac{PG}{1+PGH}$	9	$F \cdot \frac{PH}{1+PGH}$

`sgn = 1` specifies positive feedback and `sgn = -1` specifies negative feedback (default).

`utype = 1` indicates correlated uncertainties (default) and `utype = 1` indicates uncorrelated uncertainties. In an uncorrelated case, each element in one array is matched with all the elements in the other array. If array dimensions are different, an uncorrelated case is assumed.

Arguments	Default Values
`P,G,H,F`	`1+0i`
`sgn`	`-1`
`utype`	`1`

`P, G, H` and `F` are LTI and/or FRD models. If mixed models are used, the result is an FRD model.

`cltmpl` works with both continuous and discrete systems (all systems must have the same sampling time).

`cl = cltmpl(ptype,P,G,[],[],[],utype)` computes the closed-loop LTI/FRD model designated by `ptype` from `P` and `G data`.

Examples

Compute the closed-loop tracking frequency response set

$$\left| \frac{P(s)G(s)}{1+P(s)G(s)} \right|$$

corresponding to:

$$P(s)=\frac{1}{s+a},\quad G(s)=\frac{a}{s+2},\quad a\in[1,10]$$

We first form LTI arrays to represent the above models using linear parameter space grids

```
c = 1;
for a = linspace(1,10,10),
 P(1,1,c) = tf(1,[1,a]);
 G(1,1,c) = tf(a,[1,2]);
 c = c + 1;
end
```

The result is computed from

```
T = cltmpl(1,P,G);
```

Due to uncorrelated uncertainties, the array dimension of the sum is

```
>> size(P)
10x1 array of transfer functions
Each model has 1 output and 1 input.

>> size(P)
10x1 array of transfer functions
Each model has 1 output and 1 input.

>> size(T)
10x1 array of transfer functions
Each model has 1 output and 1 input.
```

Note that if we let the uncertainties be uncorrelated (clearly not the case here)

```
T = cltmpl(1,P,G,[],[],[],2);
```

resulting in a 10x10 array dimension

```
>> size(T)
100x1 array of transfer functions
Each model has 1 output and 1 input.
```

The frequency response of this array is computed using a Control Toolbox command

```
w = logspace(-1,1);
Tfr = freqresp(T,w);
```

Note that the result is NOT an LTI model, rather a multi-dimensional matrix

```
>> size(Tfr)
ans =
     1     1    50   100.
```

In SISO cases (the first two indices correspond to input and output dimensions), it is convenient to eliminate the singleton dimensions

```
Tfr = squeeze(freqresp(T,w));
>> size(Tfr)
ans =
     50   100.
```

The result can be made a FRD model using

```
Tfr = frd(Tfr,w);
>> get(Tfr)
       Frequency: [50x1 double]
   ResponseData: [1x1x50x100 double]
          Units: 'rad/s'
             Ts: 0
        ioDelay: 0
     InputDelay: 0
    OutputDelay: 0
      InputName: {''}
     OutputName: {''}
     InputGroup: {0x2 cell}
    OutputGroup: {0x2 cell}
          Notes: {}
       UserData: []
```

See Also

`addtmpl`, `multmpl`

genbnds

Purpose

Compute QFT bounds

Synopsis

```
bdb = genbnds(ptype,w,Ws,A,B,C,D,P0,phs)
```

Description

`genbnds` computes QFT bounds on the nominal loop, $L_0 = P_0 G$, for the generic problem specified by `ptype` (G is the controller, P_0 is the nominal plant) as shown in the table below.

ptype	I/O relation
10	$\left\|\frac{A+BG}{C+DG}\right\| \le W_{s_{10}}$
11	$\left\|\frac{\|A\|+\|B\|\|G\|}{C+DG}\right\| \le W_{s_{11}}$

`A`, `B`, `C`, `D`, and `P0` can be constants or LTI/FRD models. `Ws`, a weight, can be a single number, vector or an LTI model; `w` is a frequency vector (rad/sec; must be a subset of the frequencies in an FRD model). The only default here is `phs` = [0°:-5°:-360°].

`genbnds` works for both continuous and discrete systems.

When invoked without a left-hand argument, `genbnds` displays the computed bounds.

For further details, refer to the Using the Bound Computation Manager section in Chapter 6.

Examples

Consider a unity feedback system with the uncertain plant described by

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

The desired closed-loop stability margin is given by

$$\left| \frac{PG}{1+PG}(j\omega) \right| \le 1.2, \omega \ge 0, \text{ for all } P \in \mathcal{P}$$

First, define problem data

```
c = 1;
for a = linspace(1,10,25),
 P(1,1,c) = tf(1,[1,a]);
 c = c + 1;
```

```
    end

    A = 0;
    B = P;
    C = 1;
    D= = P;
    inom = 1 % nominal case
```

finally, invoke

```
    w = [0.1,1,10,100];  % bounds will be computed at these freqs
    P0 = P(1,1,inom);
    bdb = genbnds(10,w,1.2,A,B,C,D,P0)
```

Use of this function is illustrated in Examples 7 and 8 (cascaded-loop) and Example 15 (multi-loop), all in Chapter 5.

See Also

`grpbnds`, `plotbnds`, `sectbnds`, `sisobnds`

getqft

Purpose

Export a QFT design as an LTI model.

Synopsis

```
C = getqft
C = getqft('filename')
```

Description

`getqft` opens a file selection dialog box that allows selection of a binary file created using the **File|Save...** option within any of the interactive design environment (IDE) functions.

`C = getqft` opens a file selection dialog box and returns the contents of the selected file with `C` being an LTI model.

`C = getqft('filename')` directly opens the file specified by `filename`.

The default IDE extensions are shown in the following table. (these choices are not unique, any other extension can be used).

Design Environment	File Extension
`lpshape`	`*.shp` (continuous-time)
`lpshape`	`*.dsh` (discrete-time)
`pfshape`	`*.fsh` (continuous-time)
`pfshape`	`*.dfs` (discrete-time)

`getqft` works for both continuous and discrete systems.

Algorithm

The returned model is in a balanced state-space form implementation of the algorithm described in [20].

Limitations

The balanced state-space form is not available in discrete-time systems or if there are imaginary axis or unstable poles. Repeated poles may cause numerical difficulties.

See Also

`lpshape`, `pfshape`, `putqft`

grpbnds

Purpose

Group several bounds into a single variable

Synopsis

```
bdb = grpbnds(var1,var2,...)
```

Description

`grpbnds` assigns the passed bounds to a single matrix. Its purpose is to reduce the number of input variables in functions requiring bounds.

Examples

Suppose you have computed the following bounds

```
bdb1 = sisobnds(1,w,Ws1,P);
bdb5 = sisobnds(5,w,Ws5,P);
```

then to group them, invoke

```
bdb = grpbnds(bdb1,bdb5);
```

See Also

`plotbnds`, `sectbnds`

lpshape

Purpose

Interactive environment continuous-time controller design in a Nichols chart format

Synopsis

```
lpshape()
lpshape(C0)
lpshape(w,bdb,P0,C0,phs)
```

Description

`lpshape` creates within MATLAB an interactive design environment (IDE) that allows use of either the mouse or keyboard to add specific controller elements in order to manipulate the frequency response.

Depending on the input arguments, the IDE is initiated in a continuous-time or a discrete-time setting.

`w` is a frequency vector (rad/sec), `bdb` denoted a QFT bound matrix, `P0` (the nominal loop) and `G0` (initial controller) are LTI/FRD models, `w` is a frequency vector (rad/sec; must be a subset of the frequencies in an FRD model).and `phs` is the phase used to compute `bdb`.

Upon entry, the nominal loop, L_0, is the product of the nominal plant and initial controller

$$L_0 = P_0 G_0$$

Arguments	Default Values
`w*`	`logspace(-2,3,100)` in continuous-time setting `logspace(-2,log10(π/Ts),100)` in discrete-time setting
`P0,C0`	`1`
`phs*`	`[0:-5:-360]`

`lpshape(w,[],P0)` initiates an IDE with user-specified nominal loop transfer function and frequency vector. No bounds are passed and the remaining inputs are set to their respective defaults as outlined in the above table.

For details on the interactive design environment, refer to The Interactive Design Environment section in Chapter 6.

Examples

Suppose you wish to loop-shape a controller for a continuous-time feedback system with the uncertain plant

$$\mathcal{P} = \left\{ P(s) = \frac{s}{s+a}, \quad a \in [.1,.8] \right\}$$

and a desired closed-loop stability margin is given by

$$\left| \frac{PG}{1+PG}(j\omega) \right| \le 1.2, \omega \ge 0, \text{ for all } P \in \mathcal{P}$$

First, define problem data

```
Ts = 0.01;
c = 1;
for a = linspace(0.1,0.8,25),
 P(1,1,c) = tf(1,[1,-a]);
 c = c + 1;
end
P.Ts = Ts;
```

then compute bounds

```
wbd = [0.1,1,10,100];
Ws = 1.2;
bdb1 = sisobnds(1,wbd,Ws,P);
```

and finally, initiate loop-shaping environment

```
nom = 1;
w = logspace(-1,log10(pi/Ts));
lpshape(wl,bdb1,P0(1,1,nom))
```

In addition, see Examples 1-10 in Chapter 5.

Suppose you wish to loop-shape a controller for a discrete-time feedback system (sampling time Ts=0.01 sec) with the uncertain plant

$$\mathcal{P} = \left\{ P(z) = \frac{z}{z-a}, \quad a \in [.1,.8] \right\}$$

and a desired closed-loop stability margin is given by

$$\left| \frac{P(z)G(z)}{1+P(z)G(z)} \right| \le 1.2 \, , z = e^{j\omega T_s} \, , \omega \in \left[0, \tfrac{\pi}{T_s} \right], \text{for all } P \in \mathcal{P} \, .$$

First, define problem data

```
Ts = 0.01;
c = 1;
for a = linspace(0.1,0.8,25),
 P(1,1,c) = tf(1,[1,-a]);
 c = c + 1;
end
P.Ts = Ts;
```

then compute bounds

```
w = [0.1,1,10,100];
Ws = 1.2;
bdb1 = sisobnds(1,w,Ws,P);
```

and finally, initiate loop-shaping environment

```
nom = 1;
w = logspace(-1,log10(pi/Ts));
lpshape(wl,bdb1,P0(1,1,nom))
```

See example files `qftex12.m` and `qftex13.m`.

See Also

`pfshape`

multmpl

Purpose

Multiply LTI and/or FRD arrays.

Synopsis

```
P = multmpl(P1,P2,utype)
```

Description

`multmpl` produces the product of two SISO objects or arrays (LTI and/or FRD). If one is an FRD, the result is an FRD model.

`utype = 1` indicates correlated uncertainties (default) and `utype = 2` indicates uncorrelated uncertainties. In an uncorrelated case, each element in one array is matched with all the elements in the other array. When the dimensions of the arrays are the same, say n (note: array dimensions are not I/O dimensions), `P1*P2` produces the same result as `multmpl(P1,P2,1)` — an object of array dimension n. `multmpl(P1,P2,2)` produces an object of array dimension n^2. If the array dimensions are different, an uncorrelated case is assumed.

`multmpl` works with both continuous and discrete systems (both systems must have the same sampling time).

Examples

Consider product of two transfer function sets given by

$$P_1(s)=\frac{1}{s+a},\quad a\in[1,10] \qquad P_2(s)=\frac{b}{s+2},\quad b\in[0.1,0.5]$$

We first form LTI arrays to represent the above models using linear parameter space grids

```
c = 1;
for a = linspace(1,10,10),    % use a 10-point grid
 P1(1,1,c) = tf(1,[1,a]);  c = c + 1;
end

c = 1;
for b = linspace(0.1,0.5,10), % use a 10-point grid
 P2(1,1,c) = tf(1,[1,2]);  c = c + 1;
end
```

The addition is computed from

```
P = multmpl(P1,P2,2);
```

Due to uncorrelated uncertainties, the array dimension of the sum is

```
>> size(P1)
10x1 array of transfer functions
Each model has 1 output and 1 input.
```

```
>> size(P2)
10x1 array of transfer functions
Each model has 1 output and 1 input.

>> size(P)
100x1 array of transfer functions
Each model has 1 output and 1 input.
```

Note that using either

```
P = P1*P2;
```

or

```
P = multmpl(P1,P2);
```

results in an erroneous addition since both assume correlated uncertainties (thought the results are different in that different elements are paired in the summation).

See Also

addtmpl, cltmpl

pfshape

Purpose

Interactive design environment (IDE) for design of a pre-filter for a specified closed-loop I/O configuration.

Synopsis

```
pfshape(ptype,w,Ws,P,R,G,H,F0)
```

Description

`pfshape` creates within MATLAB a pre-filter interactive design environment that allows use of either the mouse or keyboard to add specific elements.

Depending on the input arguments, the IDE is initiated in a continuous-time or a discrete-time setting.

The argument `ptype` defines the particular closed-loop I/O relation of interest as shown in the table below

ptype	I/O relation	ptype	I/O relation	ptype	I/O relation
1	$F\cdot\frac{PGH}{1+PGH}$	4	$F\cdot\frac{G}{1+PGH}$	7	$F\cdot\frac{PG}{1+PGH}$
2	$F\cdot\frac{1}{1+PGH}$	5	$F\cdot\frac{GH}{1+PGH}$	8	$F\cdot\frac{H}{1+PGH}$
3	$F\cdot\frac{P}{1+PGH}$	6	$F\cdot\frac{PG}{1+PGH}$	9	$F\cdot\frac{PH}{1+PGH}$

The systems `P`, `G`, `H` and `R` are LTI/FRD models or constants. `w` is a frequency vector to be used for displaying the responses. `Ws`, a weight, can be a single number, vector or an LTI/FRD model; `w` is a frequency vector (rad/sec; must be a subset of the frequencies in an FRD model). `R` denotes multiplicative uncertainty disk radius with respect to the plant `P`.

Arguments	Default Values
`P,G,H,F0`	`1`
`R`	`0`

`pfshape` opens graph window showing the Bode magnitude (*dB*) vs. frequency plot. The maximum magnitude of the closed-loop I/O relation is drawn with a solid line and |`Ws`| is drawn with a dashed line. For `ptype` = 7, both minimum and maximum magnitudes of the I/O relation are shown as well as both upper and lower weights of `Ws` as in `sisobnds(7,...)` (`Ws` is a 2-row magnitude matrix or a 2-model LTI/FRD object). For more details on this interactive design environment, refer to The Interactive Design Environment section in Chapter 6.

Examples

Suppose you wish to design pre-filter for a tracking problem with an uncertain plant

$$\mathcal{P}=\left\{P(s)=\frac{k}{(s+5)(s+30)}:\ k=[1,10]\right\}$$

and the controller

$$G(s)=\frac{379\left(\frac{s}{42}+1\right)}{\left(\frac{s}{152}+1\right)}$$

such that

$$\left|F\cdot\frac{PG}{1+PG}\right|\le 1.2,\ \text{for all}\ P\in\mathcal{P}$$

Define input data

```
c = 1;
for k = linspace(1,10,15),
 P(1,1,c) = tf(k,conv([1,5],[1,30]));
 c = c + 1;
end

G = tf(379*[1/42,1],[1/152,1]);
```

Then initiate the pre-filter design environment by invoking

```
pfshape(1,w,1.2,P,0,G)
```

In addition, see Example 2 and 9 in Chapter 5.

Suppose you wish to design s pre-filter for a tracking problem with sampling time of 1 second and an uncertain plant

$$\mathcal{P}=\left\{P(z)=\frac{k(z+0.9672)}{(z-1)(z-0.9048)}:\ k\in[0.01,0.05]\right\}$$

and the controller

$$\mathrm{G}(z)=\frac{12.8(z-0.883)}{z+0.5}$$

such that

$$\left|F\cdot\frac{PG}{1+PG}\right|\le 1.2,\ z=e^{j\omega T_s},\ \omega\in\left[0,\frac{\pi}{T_s}\right],\ \text{for all}\ P\in\mathcal{P}$$

Define input data

```
Ts = 1;
c = 1;
for k = linspace(0.01,0.05,15),
 P(1,1,c) = tf(k*[1,0.9672],conv([1,-1],[1,-0.9048]));
 c = c + 1;
end
P.Ts = Ts;

G = tf(12.8*[1,-0.883],[1,0.5]);
```

```
G.Ts = Ts;
```

Then initiate the pre-filter design environment by invoking

```
pfshape(1,w,1.2,P,0,G)
```

Another example can be found in Example 13 in Chapter 5.

See Also

`lpshape`, `getqft`

plotbnds

Purpose

Plot QFT bounds

Synopsis

```
plotbnds(bdb)
plotbnds(bdb,problem,phs)
```

Description

`plotbnds` plots the bound vector returned by `sisobnds` and `genbnds` with a legend in the upper left-hand corner of the figure window designating which bound was computed at which frequency.

`plotbnds(bdb)` plots the bounds in `bdb` using the defaults as shown in the above table.

`plotbnds(bdb,problem)` plots only the bounds associated with the passed types in `problem`.

`plotbnds(bdb,[],phs)` all the bounds in `bdb` with their corresponding phase vector, `phs`. This phase vector is the same that was used to compute the bounds using `sisobnds` or `genbnds`. Default value is `phs` = [0:-5:-360]

You can on/off toggle showing bounds by right-clicking the mouse and selecting options on the displayed window.

See Also

`grpbnds`, `sectbnds`

plottmpl

Purpose

Plot plant templates

Synopsis

```
plottmpl(w,P,nom)
```

Description

`plottmpl` plots the frequency-response templates and labels the nominal plant with an (*). `P` is an LTI/FRD array. `w` is a frequency vector (rad/sec; must be a subset of the frequencies in an FRD model).

`plottmpl(w,wbd,P)` plots the frequency response templates of `P` at the frequencies designated by `w`. The nominal plant index, `nom`, defaults to 1.

`plottmpl(w,[],P,10)` plots the frequency-response templates at all the frequencies with the 10th plant labeled as the nominal plant.

`plottmpl` works for both continuous and discrete systems.

The displayed bounds can be toggled on/off right-clicking the mouse and selecting options on the displayed window.

putqft

Purpose

Import controllers into the interactive design environment binary file format

Synopsis

```
putqft(C)
putqft('filename'C)
```

Description

`putqft(C)` opens a file selection dialog box and places the contents of the LTI model into the chosen binary file that can then opened by an interactive design environment (IDE). `Ts` denotes sampling time (seconds) and `C` denotes the controller.

`putqft('filename',C)` directly places the contents of the specific format into the `filename` which can then be opened by an interactive design environment.

An IDE file is essentially a zero/pole/gain description. For large order numerator/denominator or state-space forms, the conversion to zero/pole/gain format is suspect to numerical inaccuracies.

The interactive design environments are configured to search for files with the following extensions (though you can specify any file name):

Design Environment	File Extension
`lpshape`	`*.shp` (continuous-time)
`lpshape`	`*.dsh` (discrete-time)
`pfshape`	`*.fsh` (continuous-time)
`pfshape`	`*.dfs` (discrete-time)

See Also

`getqft`, `lpshape`, `pfshape`

qftex#

Purpose

Batch files for the Toolbox demo examples

Synopsis

```
qftex#
```

Description

can be any number from 1 to 15, each corresponds to an example # from chapter 5. These files are standard batch M-files.

sectbnds

Purpose

Intersect QFT bounds

Synopsis

```
ubdb = sectbnds(bdb)
```

Description

`sectbnds` performs set intersection on the bounds computed from `sisobnds` and `genbnds`. It also determines when the result is empty or a non connected set.

`ubdb = sectbnds(bdb)` returns the intersection of the bounds contained in `bdb`.

For a complete discussion on *bounds* please refer to The Bound Computation Managers.

Examples

Suppose you wish to design a controller for an uncertain plant

$$\mathcal{P} = \left\{ P(s) = \frac{1}{(s+a)(s+30)} : k = [1,10] \right\}$$

with peaking constraints

$$\left| \frac{PG}{1+PG} \right| \le 1.2, \text{ for all } P \in \mathcal{P}$$

and

$$\left| \frac{1}{1+PG} \right| \le 1.2, \text{ for all } P \in \mathcal{P}.$$

We first form an LTI array to represent the above model using a linear parameter space grid

```
c = 1;
for a = linspace(1,10,15),
 P(1,1,c) = tf(1,conv([1,a],[1,30]));
 c = c + 1;
end
```

To compute the corresponding bounds at low and high frequencies run

```
w = [1,100];
bdb1 = sisobnds(1,w,1.2,P);
bdb2 = sisobnds(2,w,1.2,P);
```

and grouped them

```
bdb = grpbnds(bdb1,bdb2);
```

then the intersection is computed by invoking

```
ubdb = sectbnds(bdb);
```

To evaluate the result we plot the bounds before and after the intersection

```
plotbnds(bdb)
plotbnds(ubdb)
```

See Also

`genbnds`, `grpbnds`, `plotbnds`, `sisobnds`

sisobnds

Purpose

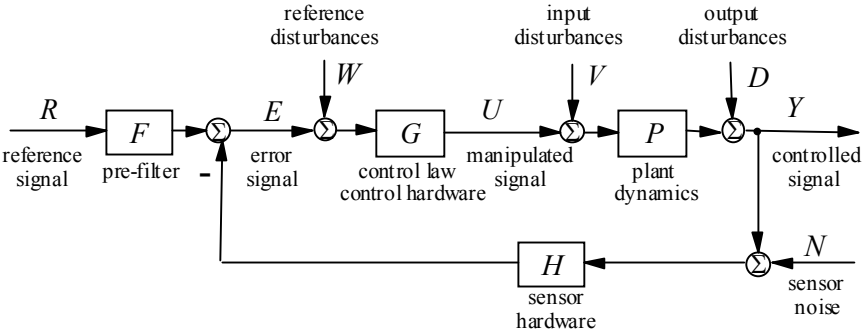
Compute single-input/single-output bounds

Synopsis

```
bdb = sisobnds(ptype,w,Ws,P)
bdb = sisobnds(ptype,w,Ws,P,R,nom,C,loc,phs)
```

Description

`sisobnds` computes single-input/single-output QFT bounds for the feedback system shown below



In terms of the nominal loop, $L_0 = L_0 G_0 H_0$. The performance problem is specified in `ptype` (see below).

ptype	I/O Problem
1	$\left\|\frac{PGH}{1+PGH}\right\| \le Ws_1$
2	$\left\|\frac{1}{1+PGH}\right\| \le Ws_2$
3	$\left\|\frac{P}{1+PGH}\right\| \le Ws_3$
4	$\left\|\frac{G}{1+PGH}\right\| \le Ws_4$
5	$\left\|\frac{GH}{1+PGH}\right\| \le Ws_5$
6	$\left\|\frac{PG}{1+PGH}\right\| \le Ws_6$
7	$Ws_{7a} \le \left\|F\frac{PG}{1+PGH}\right\| \le Ws_{7b}$
8	$\left\|\frac{H}{1+PGH}\right\| \le Ws_8$
9	$\left\|\frac{PH}{1+PGH}\right\| \le Ws_9$

Arguments	Defaults
P,G,H	1
R	0
nom	1
loc	1
phs	[0°:-5°:-360°]

P and C and R can also be represented by LTI/FRD models. Ws, a weight, can be a single number, vector or an LTI/FRD model; w is a frequency vector (rad/sec; must be a subset of the frequencies in an FRD model), the integer nom contains nominal plant index in an LTI/FRD array, the integer loc defines the controller location in the loop and w is a vector defining the resolution of the computed bounds along the phase axis.

For more details, please see Single Loop Bound Manager.

`loc` defines the controller location: `loc = 1` implies it is at $G(s)$ (i.e., forward path) while `loc = 2` implies it is at $H(s)$ (i.e., feedback path).

When invoked without a left-hand argument, `sisobnds` displays the computed bounds.

`sisobnds` works for both continuous and discrete systems.

For further details, refer to The Bound Computation Managers.

Examples

Consider a unity feedback system with the uncertain plant described by

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

The desired closed-loop stability margin is given by

$$\left| \frac{PG}{1+PG}(j\omega) \right| \le 1.2, \omega \ge 0, \text{ for all } P \in \mathcal{P} .$$

The above peaking constraint corresponds a phase margin of 50^{o} and a gain margin of 1.83.

We first form an LTI array to represent the above model using a linear parameter space grid

```
c = 1;
for a = linspace(1,10,15),
 P(1,1,c) = tf(1,[1,a]);
 c = c + 1;
end
```

The desired QFT bounds corresponding to the above constraint are computed from

```
Ws = 1.2;
w = [0,01,1,100];
bdb1 = sisobnds(1,w,Ws,P);
```

and are displayed using

```
plotbnds(bdb1);
```

In a discrete-time system, suppose the plant is described by (sampling time Ts=0.01 sec)

$$\mathcal{P} = \left\{ P(z) = \frac{z}{z-a}, \quad a \in [.1,.8] \right\}$$

and a desired closed-loop stability margin is given by

$$\left| \frac{P(z)G(z)}{1+P(z)G(z)} \right| \le 1.2 , z = e^{j\omega T_s} , \omega \in \left[0, \frac{\pi}{T_s} \right], \text{ for all } \; P \in \mathcal{P} .$$

We first form an LTI array to represent the above model using a linear parameter space grid

```
Ts = 0.01;
c = 1;
for a = linspace(0.1,0.8,25),
 P(1,1,c) = tf(1,[1,-a]);
 c = c + 1;
end
P.Ts = Ts;
```

compute bounds using

```
w = [0.1,1,10,100];
Ws = 1.2;
bdb1 = sisobnds(1,w,Ws,P);
```

which can be displayed using

```
plotbnds(bdb1);
```

An interesting problem is the traditional QFT tracking setting: a unity feedback system with the uncertain plant described by

$$\mathcal{P} = \left\{ P(s) = \frac{ka}{s(s+a)} : \quad k \in [1,10], a \in [1,15] \right\}$$

with desired tracking (i.e., complimentary sensitivity function) from $R(s)$ to $Y(s)$ given by

$$\left| W_{s_1}(j\omega) \right| \le \left| F \frac{PG}{1+PG}(j\omega) \right| \le \left| W_{s_2}(j\omega) \right|, \quad \omega \le 10, \text{ for all } P \in \mathcal{P}$$

where

$$W_{s_1}(s) = \frac{0.6584(s+30)}{s^2+4s+19.752} \quad \text{and} \quad . W_{s_2}(s) = \frac{120}{s^3+17s^2+82s+120}$$

Solving this problem involves two steps. In the first step we compute bounds for the controller $G(s)$, then loop-shape it. In the second step we shape a pre-filter, $F(s)$. Let us first design the controller.

We first form an LTI array to represent the above model using a linear parameter space grid

```
c = 1;
for a = linspace(1,15,15),
for k = linspace(1,10,10),
 P(1,1,c) = tf(k*a,[1,a,0]);
 c = c + 1;
end
```

Define the weight

```
Ws1 = tf(0.6584*[1,30],[1,4,19.752]);
Ws2 = tf(120,[1,17,82,120]);
Ws = [Ws1,Ws2];
```

then compute tracking bounds

```
w = [0.1,0.5,1,15];
```

```
bdb7 = sisobnds(7,w,,Ws,P);
```

Once a controller $G(s)$ is designed using the loop shaping environment `lpshape`, say it is given by `G`, the feedback design is completed with the design of the pre-filter

```
pfshape(7,w,Ws,P,G);
```

A similar discrete-time design problem is described in Example 12.

Algorithm

An implementation of the algorithms described in [8,14,15,22].

See Also

`genbnds`, `grpbnds`, `plotbnds`, `sectbnds`

A Glossary

above bound — the minimum value of the gain of the nominal open-loop transfer function, at some fixed phase, such that a specific magnitude constraint on a closed-loop transfer function is algebraically satisfied.

below bound — the maximum value of the gain of the nominal open-loop transfer function, at some fixed phase, such that a specific magnitude constraint on a closed-loop transfer function is algebraically satisfied.

bound — the allowable range of the gain of the nominal open-loop transfer function, at some fixed phase, such that a specific magnitude constraint on a closed-loop transfer function is algebraically satisfied.

loop shaping — the process of designing a nominal open-loop transfer function.

Nichols chart — a frequency response plot with phase (degrees) and magnitude (*dB*) of the open-loop transfer function as its coordinates.

Nominal plant — the designated plant for bound computation and open-loop shaping. It is either: (1) the fixed plant when there are no uncertainties, (2) an arbitrarily selected plant from a family of parametric uncertain plant model, or (3) the central plant of a family of nonparametric uncertain plant model.

robust stability — indicates that a closed-loop system is stable for any plant within the specified uncertainty model.

robust performance — indicates that a closed-loop system satisfies its performance specification(s) for any plant within the specified uncertainty model.

QFT — the Quantitative Feedback Theory method.

stability margins — the amount of gain and phase variations in the open-loop transfer function that can be tolerated (not simultaneously) before a stable closed-loop system becomes unstable.

templates — the collection, at a fixed frequency, of all frequency responses of an uncertain plant model.

B Bibliography

[1] Horowitz, I.M., 1963, *Synthesis of Feedback Systems*, Academic Press, New York.

[2] Horowitz, I.M, and Sidi, M., 1972, "Synthesis of feedback systems with large plant ignorance for prescribed time-domain tolerances," *Int. J. Control*, 16(2), pp. 287-309.

[3] Horowitz, I.M., 1992, *Quantitative Feedback Theory (QFT)*, QFT Publications, 4470 Grinnell Ave., Boulder, Colorado, 80303.

[4] Cohen, N., Chait, Y., Yaniv, O., and Borghesani, C., 1994, "Stability analysis using Nichols charts," *Int. J. Robust and Nonlinear Control*, Vol. 4, pp. 3-20.

[5] Jayasuriya, S., 1993, "Frequency domain design for robust performance under parametric, unstructured, or mixed uncertainties," *ASME J. of Dynamic Systems, Measurement, and Control*, Vol. 115, pp. 439-451.

[6] Lehtomaki, N.A., Sandell, N.R., and Athans, M., 1981, "Robustness results in linear-quadratic gaussian based multivariable control designs," *IEEE Trans Automatic Control*, Vol. AC-26, pp. 75-93.

[7] Philips, C.L., and Nagle, H.T., 1990, *Digital Control Systems Analysis and Design*," Prentice Hall, New York, pp. 241.

[8] Chait, Y., and Yaniv, O., 1993, "Multi-input/single-output computer aided control design using the Quantitative Feedback Theory, " *Int. J. Robust and Nonlinear Control*, Vol. 3, pp. 47-54.

[9] Gutman, P-O, Baril, C., and Neumann, L., 1990, "An image processing approach for computing value sets of uncertain transfer functions," *Procs. CDC*, pp. 1224-1229.

[10] Fu, M., 1990, "Computing the frequency response of linear systems with parametric perturbation," *Systems & Control Letters*, Vol. 15, pp. 45-52.

[11] Bartlett, A.C., 1993, "Computation of the frequency response of systems with uncertain parameters: a simplification," *Int. J. Control*, Vol. 57, 1293-1309.

[12] Chait, Y., and Hollot, C.V., 1990, "A comparison between H-infinity methods and QFT for a siso plant with both parametric uncertainty and performance specifications," *ASME Pub. Recent Development in Quantitative Feedback Theory*, O.D.I. Nwokah, Ed., pp. 33-40.

[13] D'Azzo, J.J., and Houpis, C.H., 1988, *Linear Control Design Analysis & Design*, McGraw-Hill.

[14] Yaniv, O., and Chait, Y., 1993, "Direct control design in sampled-data uncertain systems," *Automatica*, Vol. 29, pp. 365-372.

[15] Chait, Y., Borghesani, C., and Zheng, Y., "Single-loop QFT Design for Robust Performance in the Presence of Non-Parametric Uncertainties," *J. Dynamic Systems, Measurements, and Control*, to appear.

[16] Wie, B., and Bernstein, D.S., 1991, "Benchmark problems for robust control design," *Procs. American Control Conference*, pp. 1929-1930.

[17] Jayasuriya, S., Yaniv, O., Nwokah, O.D.I., and Chait, Y., 1992, "The benchmark problem solution by the Quantitative Feedback Theory, " *AIAA J. Guidance and Control*, Vol. 15, pp. 1087-1093.

[18] Smit, S.G., 1990, "H_∞ robust servo control theory and application to a flexible mechanism," *Nat. Lab. Technical Report*, Nr. 057/90, Philips Research, Eindhoven, The Netherlands.

[19] Borghesani, C., 1993, *Computer Aided-Design of Robust Control Systems Using the Quantitative Feedback Theory*, M.S. Thesis, Mechanical Engineering Department, University of Massachusetts, Amherst, MA.

[20] Wortelboer, P.M.R., and Bosgra O.H., 1992, "Generalized frequency weighted balanced reduction," *Procs. 31st IEEE CDC Conf.*, pp. 2848-2849.

[21] Knowles, J.B., 1978, "A comprehensive, yet computationally simple, direct digital control-system design technique," *Proc. IEE*, Vol. 125(12), pp. 1383-1395.

[22] Wang, G.G., Chen, C.W., and Wang, S.H., 1991, "Equation for loop bound in Quantitative Feedback Theory," *Procs. CDC*, pp. 2968-2969.

[23] Zhao, Y., and Jayasuriya, S., 1993, "Robust stabilization of linear time invariant systems with parametric uncertainties," *Advances in Robust and Nonlinear Control Systems*, DSC-Vol. 53, ASME WAM, New Orleans, pp. 79-86.

[24] Chait, Y., Park, M.S., and Steinbuch, M., 1994, "Design and implementation of a QFT controller for a compact disc player," *J. Systems Engineerin*g, Vol 4, pp. 107-117.

[25] Park, M.S., Chait, Y., and Steinbuch, M., "A new approach to multivariable QFT: theoretical and experimental results, " *1994 ACC Conference*, pp. 340-344.

[26] Gille, J-C, Pelegrin, M.J., and Decaulne, 1959, *Feedback Control Systmes*, McCgraw-Hill, Inc., New York, pp. 710-717.

[27] Kidron, O., 1993, *Robust Control Of Uncertain Resonant Systems*, M.Sc. Thesis, Electrical Engineering Department, Tel-Aviv University, Tel-Aviv, Israel.

[28] Wortelboer, P.M.R., 1994, *Frequency-Weighted Balanced Reduction of Closed-Loop Mechanical Servo-Systems: Theory and Tool*, Ph.D. Thesis, Delft University, The Netherlands.

[29] Chait Y., and Steinbuch, M., "Design trade-off issues in robust control of a compact disk player," *Symp. on Advances in Information Storage Systems*, 1994 ASME Winter Annual Meeting, in press.

[30] Yaniv, O., 1999, *Quantitative Feedback Design of Linear and Nonlinear Control Systems*, Kluwer Academic Publishers, Massachusetts, USA, Kluwer Academic Publishers, Massachusetts, USA.

[31] Houpis, C.H., and Rasmussen, S.J.*, 1999, Quantitative Feedback Theory Fundumentals and Applications*, Marcel Dekker, Inc., New York.

[32] Sidi, M., 1999, *DESIGN OF ROBUST CONTROL SYSTEMS: From Classical to Modern Practical Approaches*, Krieger Publishing.