Interval Methods for Analysis of Linear and Nonlinear Control Systems

Submitted in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

by

Jayesh Jayanarayan Barve

Rollno: 97423701

Systems and Control Engineering

Department of Electrical Engineering

Indial Institute of Technology

Bombay

ABSTRACT Existing tools of control system analysis have certain limitations in terms of their applicability, reliability, and accuracy, particularly for the *nonrational* class of linear and nonlinear systems. The main objective of this work is to develop tools that help overcome these limitations for a large class of linear and nonlinear, fixed and uncertain parameter systems.

The proposed tools are in the form of algorithms, and address the following problems: (a) computation of the well known Bode, Nyquist and Nichols frequency response plots for nonrational transfer functions, (b) computation of robust gain margins, phase margins, and crossover frequencies for nonrational transfer functions with general nonlinear parametric dependencies, (c) computation of spectral set for uncertain polynomials with polytopic as well as general nonlinear parametric dependencies, and (d) computation of limit cycle sets for seperable uncertain nonlinear systems covering a large class of nonlinear elements represented by nonrational describing functions, nonrational linear elements, and generic nonlinear parametric dependencies. Further, in the spirit of the well known root locus for linear systems, a new tool called *limit cycle locus* is proposed for uncertain nonlinear systems.

The proposed algorithms are developed in the framework of Moore's interval analysis [48]. The salient features of the proposed algorithms are their provision of several important guarantees: the computed results are reliable and accurate, *all* solutions are found, and the solutions are computed in finite number of iterations for a prescribed accuracy. An upper bound on the number of iterations required to achieve a prescribed accuracy is also given.

Theoretical results concerning key properties of the algorithms, such as convergence, termination, reliability, and accuracy are derived. The proposed algorithms are also demonstrated on several difficult practical examples, including those that cannot be readily solved with existing tools.

KEYWORDS: Control System Analysis, Describing Function, Frequency Response, Gain Margin, Interval Analysis, Limit Cycle, Nonlinear Control, Phase Margin, Robust Control, Spectral Set.

Contents

List of Figures

List of Tables

Glossary

Abbreviations:

Box: A parallelepiped having sides parallel to the coordinate axes.

DF: Describing function

GM: Gain margin

IA : Interval analysis

LTI: Linear time invariant

PM: Phase margin

VA: Vectorized adaptive

Generic symbols:

C : Real nonsingular $l \times l$ matrix (here $l = 2$), used as preconditioning matrix.

$d(.,.)$: Hausdorff distance between two nonempty compact sets in $\Re^n$.

$f(.)$: Arbitrary function.

$\bar{f}(.)$: The exact range of f .

$f'_y(.)$: the Jacobian of function f w.r.t. $\mathbf{y}$.

$F(.)$: Natural inclusion function for f.

$F'_y(.)$: A natural inclusion function for f'_y.

g : Transfer function of the linear element.

g_{mag} : Magnitude function in decibels.

g_{phase} : Phase function in degrees.

G_{mag} : Natural inclusion function for the magnitude function

G_{phase} : Natural inclusion function for the phase function

$\mathcal{G}_{cf}$: Set of all gain crossover frequencies.

$\mathcal{G}_m$: Set of all gain margins.

h : Describing function of the nonlinear element.

$I(\mathbf{X})$: Set of all boxes contained in $\mathbf{X}$.

$I(\Re^n)$: Set of all boxes contained in $\Re^n$.

$\mathcal{K}(\mathbf{X})$: Generalized Krawczyk operator applied to $\mathbf{X}$.

k : Iteration number.

$\mathcal{L}_C$: Set of limit cycles.

$\mathcal{L}_C^{a\,lg}$: Set of limit cycles as computed by the algorithm.

$\mathcal{L}_{C,\,mia}^{a\,lg}$: Set of limit cycles as computed by a MIA version of the algorithm.

$m(.)$: Midpoint of a box.

p : The polynomial.

$\bar{p}$: Exact range of polynomial.

P : Natural inclusion function for p.

$\mathcal{P}_{cf}$: Set of all phase crossover frequencies.

$\mathcal{P}_{cf}^{a\,lg}$: Computed set of all phase crossover frequencies.

$\mathcal{P}_{cf,mia}^{a\,lg}$: Computed set of all phase crossover frequencies, using MIA.

$\mathcal{P}_m$: Set of all phase margins.

$\mathbf{q}$: Vector of all system parameters, $\mathbf{q} = (q_1, \ldots, q_n)$.

$\mathbf{q}_h$: Vector of parameters of the nonlinear element.

$\mathbf{q}_g$: Vector of parameters of the linear element.

r : Radius or magnitude of a complex number in the polar representation.

$\mathbf{Q}$: Box of uncertain system parameters.

$\mathbf{Q}^0$: Initial box of uncertain system parameters.

s : Laplace variable.

$\mathcal{S}$: Spectral set of p.

$\mathcal{S}^{a\,lg}$: Spectral set of p as computed by the algorithm.

$\mathcal{S}_{mia}^{a\,lg}$: Spectral set of p as computed by the algorithm using machine interval arithmetic.

$w(.)$: Width of a box.

$\hat{\mathbf{y}}$: Point in $\mathbf{Y}$, usually taken as the midpoint of $\mathbf{Y}$.

$\mathbf{Y}$: Box of unknowns.

$\mathbf{Y}^0$: Initial search box of unknowns.

Greek:

α : Constant appearing in definition of convergence order.

ε_{dB} : Accuracy tolerance on magnitude (or gain margin as applicable), in dB.

$\varepsilon_{\deg}$: Accuracy tolerance on phase (or phase margin as applicable), in degrees.

ε_p : Polynomial range accuracy tolerance.

ε_x : Accuracy tolerance for search domain x.

ε_ω : Accuracy tolerance on crossover frequency.

ε_z : search domain (a plane of complex numbers) accuracy tolerance.

γ : Maximum number of iterations required by the algorithm.

θ : Angle of a complex number in the polar representation.

$\mathbf{\Omega}^0$: Frequency range over which the frequency response plots are desired (or the initial search frequency range for the margins).

ω : frequency

Symbols specific to different chapters:

In Chapter 2:

$\mathcal{L}$: List of boxes containing frequency subintervals $\mathbf{\Omega}$

$\mathcal{L}^{sol}$: List of frequency versus magnitude or frequency versus phase solution boxes satisfying the specified tolerance.

In Chapter 3:

f : function defined in (3.2).

$\mathcal{L}$: List of boxes of pairs of form $(\mathbf{X}, G_{mag}(\mathbf{X}))$

$\mathcal{L}^{sol}$: List of solution boxes, that satisfy both domain and range tolerance conditions.

$\mathbf{x}$: Vector containing unknown and parameters, defined as $(\omega, \mathbf{q})$.

$\mathbf{X}$: Box containing unknown and parameters, defined as $(\mathbf{\Omega}, \mathbf{Q})$.

$\mathbf{X}^0$: Initial $\mathbf{X}$ i.e. $\left(\mathbf{\Omega}^0, \mathbf{Q}^0\right)$.

In Chapter 4:

$\mathcal{L}$: List of boxes of unknowns $\mathbf{Z}$.

$\mathcal{L}^{sol}$: List of solution boxes.

z : Point in the rectangular coordinate complex plane

$\mathbf{Z}$: Box of unknowns in the complex plane.

$\mathbf{Z}^0$: Initial box of unknowns in the complex plane.

In Chapter 5:

$f(.)$: Vector function defined in (5.2).

$\mathcal{L}$: List of boxes of unknowns $\mathbf{X}$.

$\mathcal{L}^{sol}$: List of solution boxes.

$\mathbf{X}$: Box containing unknowns boxes in search domain and parameters, defined as $(\mathbf{Y}, \mathbf{Q})$.

$\mathbf{X}^0$: Initial box $\left(\mathbf{Y}^0, \mathbf{Q}^0\right)$.

$\mathbf{y}$: Vector of unknowns in polar complex plane of (r, θ)

In Chapter 6:

a : Amplitude of periodic input signal to nonlinear element.

$f(.)$: Vector function defined in (6.1).

$\mathcal{L}$: List of boxes of unknowns $\mathbf{X}$.

$\mathcal{L}^{sol}$: List of solution boxes.

$\mathbf{X}$: Box containing both unknowns and parameters, and defined as $(\mathbf{Y}, \mathbf{Q})$.

$\mathbf{X}^0$: Initial box $\left(\mathbf{Y}^0, \mathbf{Q}^0\right)$.

$\mathbf{y}$: Vector of unknowns (a, ω)

ω : Frequency of periodic input signal to the nonlinear element.

1

Introduction

Real life systems by themselves are generally not capable of achieving the specified performance while fulfilling the desired objectives. Therefore, the use of additional components in the form of controlling and feedback elements usually becomes mandatory [31]. A real life system along with the necessary additional components is called a *control* system, and the branch of engineering that deals with the analysis and synthesis of such systems is known as *control system* engineering. Before actually synthesizing and implementing a control system, it is usually analyzed and synthesized with the help of a mathematical model. The type and complexity of the mathematical model to be used for a particular control system depends upon various factors, such as complexity and characteristics of the system, the desired accuracy and performance, and the operating conditions.

Mathematical models used in control systems analysis may be classified as linear or nonlinear models, rational or nonrational transfer function models, fixed or uncertain parameter models, single variable or multivariable models, deterministic or stochastic models, continuous or discrete time models, etc., [3], [8], [55]. Naturally, different analysis and synthesis tools are required, depending upon the type of the mathematical model chosen. However, there are several common issues pertaining to all the control system analysis and synthesis tools, such as scope of applicability, reliability, and accuracy.

Let us examine some of the basic analysis tools used in control systems and their related issues.

1.1 Linear control systems

Linear control system theory is widely used for the analysis and synthesis of feedback control systems. The block diagram of a typical linear control system is shown in Fig. 1.1. In the

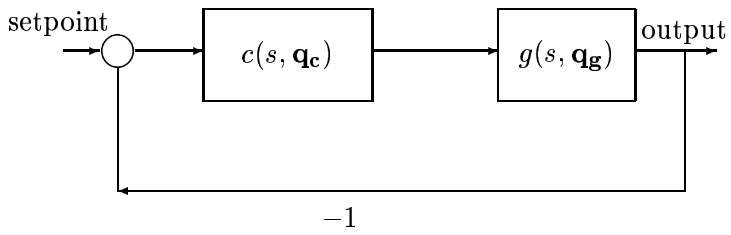

FIGURE 1.1. Block diagram of a typical linear control system.

figure, $g(s, \mathbf{q}_g)$ represents the transfer function of the system to be controlled (i.e., the plant), and $c(s, \mathbf{q}_c)$ represents the transfer function of the controller, where $\mathbf{q}_g$ and $\mathbf{q}_c$ represent the plant and controller parameter vectors. For a comprehensive treatment of classical linear control system theory, see [30], [55], and [50].

1.1.1 Frequency response plots

For over five decades, the classical Bode, Nyquist, and Nichols frequency response plots have been of immense use in frequency domain analysis and synthesis of linear control systems, see, [8], [10], [30] and [54] for an exposition of these tools.

For transfer functions that have a *rational* form, these frequency response plots can be computed to a fair degree of accuracy with the routines such as *bode*, *nyquist*, and *nichols*, which are based on the automatic frequency selection procedure available in the control systems toolbox of MATLAB [27],[46].

On the other hand, many important applications found in practice involve *nonrational* transfer functions. For instance, systems with time delay abound in chemical processes [56], hydraulic servomechanisms [20], [45], and nuclear reactor systems [14]. However, the above mentioned automatic frequency selection procedure is not readily applicable to nonrational transfer functions. To use the procedure for such functions, we first need to derive a satisfactory rational approximation to each nonrational term, and then apply the said procedure to the resulting rational transfer function.

For nonrational transfer functions involving only time delay as the nonrational term, the well known Pade approximation [16] has been widely used to effect a rational approximation.

However, the error introduced through the Pade approximation itself is difficult to estimate, and in certain applications it may turn out to be so appreciable as to produce frequency response plots of poor accuracy. Further, for nonrational transfer functions that involve other transcendental terms, such as trigonometric, hyperbolic, and inverse hyperbolic terms, it is quite difficult to find satisfactory rational approximations, and the only alternative then is to resort to the conventional gridding method for computing the frequency response plots.

The conventional gridding method consists of arbitrarily gridding the frequency range of interest, computing the frequency responses at the obtained grid points, and using interpolation to obtain the responses over the entire frequency range. However, this method has two significant limitations: (a) the number of grid points required to achieve a prescribed accuracy is in general unknown, and (b) for a given plot, the amount of error present is unknown, i.e., no error estimates are available. These limitations are particularly severe when the frequency response plots exhibit single or multiple sharp peaks or dips. Despite the severe limitations of the conventional gridding method, surprisingly little is found in the literature to overcome them, especially for nonrational transfer functions.

1.1.2 Gain and phase margins

Gain and phase margins [30] are popularly used as stability specifications in classical methods of analysis and synthesis of linear control systems. These specifications basically relate to the maximum allowable variation in the open loop gain or phase of the system to conserve closed loop stability.

For linear systems represented by *rational* transfer functions, a sophisticated method is available to compute the margins, in the form of the *allmargin* routine of MATLAB's control system toolbox [27]. However, many systems found in practice are represented by *nonrational* transfer functions, and unless some rational approximation to the function is used, no existing method can be readily applied to compute the margins for nonrational transfer functions.

Robust gain and phase margins

The system models found in classical approaches to linear control systems usually have fixed parameter values. However, for most real life systems, some uncertainty is present in the model parameters due to our ignorance of the system. The uncertainty can be modelled as one of these types: parametric, nonparametric, or mixed. In case of the parametric type, the parameter vector $\mathbf{q}_g$ in Fig.1.1 is taken to vary usually over a box like region[1]. Then, stability analysis, such as gain and phase margin analysis, must be carried out by considering every point in the parameter box, leading to the notion of robust stability of the system.

The problem of computing the robust stability margin for linear systems in the presence of parametric uncertainty has recently been the focus of some attention. To calculate the

[1] However, it is not necessary that all the parameters of a given transfer function have uncertainty.

so-called maximal parametric stability box, several approaches are available depending on the parametric uncertainty structure involved. For interval systems, Fadali *et al.* [23] present a method based on the reciprocal Nyquist value set to compute the combined gain and phase margins. For affine or multilinear parametric dependencies, the domain splitting technique of deGaston and Safonov [17], the mapping theorem based algorithm of Sideris and Pena [62], the Generalized Kharitonov theorem based method of Keel and Bhattacharyya [37] , and the critical direction procedure of Mahon *et al.* [43] are available. For special classes of nonlinear parametric dependencies, such as polynomial dependency, branch and bound algorithms based on geometrical programming are available [65], [25]. For fairly general nonlinear parametric dependencies, the interval arithmetic based bisection methods in [40] and [44] are applicable.

On the other hand, for the design of many control systems, the conventional robust gain and phase margins *remain* as important specifications, see, for instance, [31] and [37]. However, relatively fewer methods are available in the literature for directly calculating the robust gain and phase margins as well as the crossover frequencies for systems with parametric uncertainty. Bhattacharyya *et al.* [8] present a method based on the Generalized Kharitonov theorem for rational transfer functions with multilinear parametric dependency. Wilson *et al.* [66] address the case of rational transfer functions with nonlinear (rational) parametric dependencies using specially constructed minimized plants.

Thus, we see that there is a lack of direct methods to compute the robust margins and crossover frequencies for systems represented by *nonrational* transfer functions with general *nonlinear* parametric dependencies. As is well known, nonrational transfer functions involving time delays and transcendental terms (such as inverse hyperbolic terms) form a very important problem class and are found in key application areas, for instance, in chemical processes , hydraulic servomechanisms, and nuclear reactor systems.

1.1.3 Spectral sets

The coefficients of the characteristic polynomial of an uncertain linear system depend upon the system parameters. The parametric dependency can be categorized as independent, affine linear, multilinear, and nonlinear. The spectral set for the uncertain polynomial is defined as the set of all roots of the polynomials belonging to the polynomial family. The spectral set is of considerable interest in stability analysis of linear control systems, see [8], [4].

For polynomials whose coefficients are *affine linear* functions of the uncertain system parameters, i.e., for a polytope of polynomials, Barmish and Tempo [4] and Cerone [12] proposed techniques to compute the spectral set. A nice feature of these techniques is that they involve only a $2-$dim gridding of a bounded subset of the complex plane rather than a gridding of the parameter box. The feature is attractive as it circumvents a potential combinatoric explosion in computations with an increase in the number of parameters. For polynomials

whose coefficients are *multilinear* functions of the uncertain system parameters, Yang and Chen [67] present an algorithm to compute the spectral set.

Let us examine the limitations of the existing techniques for computation of the spectral set:

- First, an important requirement from a stability analysis viewpoint is that no actual points should be missing from the computed spectral set. The above mentioned techniques are based on gridding, and a well-known drawback of all gridding based techniques is that they compute underbounds of the actual sets, because of the very nature of the grid process. Therefore, potentially critical points could be missing from the computed spectral sets with the existing methods.
- Second, the existing techniques for affine and multilinear parameter dependencies lack the ability to compute the spectral sets to a prescribed accuracy. In particular,
  - it is *a priori* unknown how fine the parameter space or complex plane grid must be in order to achieve a prescribed accuracy, and
  - it is *a posteriori* unknown how accurate is the computed set for a selected fineness of the grid.
- Third, existing techniques do not provide any guarantee on the reliability, i.e., trustworthiness of the computed results in the face of various computational errors.
- Lastly, for uncertain polynomials having *nonlinear* parametric dependencies, techniques that can readily compute the spectral set are *not* available in the literature.

1.2 Nonlinear control systems

The block diagram of a typical separable nonlinear control system is shown in Fig. 1.2, where $h(a, \omega, \mathbf{q}_h)$ represents the describing function [3] of the nonlinear element, and $g(s, \mathbf{q}_g)$ represents the overall transfer function of all the linear elements in the control loop. The variables a and ω denote the amplitude and frequency of the input signal to the nonlinear element, while $\mathbf{q}_g$ and $\mathbf{q}_h$ represent the parameter vectors of the linear and nonlinear elements.

Nonlinear systems may exhibit autonomous constant amplitude closed loop oscillations, known as *limit cycles* [3], [26]. The popular describing function approach [3], [26] is mainly employed to predict the existence of the limit cycles. If limit cycles are predicted, then it is also of interest to know the number of limit cycles, their frequencies and amplitudes, and key characteristics such as stability or instability. Describing function analysis occasionally fails to predict the limit cycles, particularly when the system under consideration does not satisfy the assumption of filtering out the higher order harmonics. It is also possible for describing

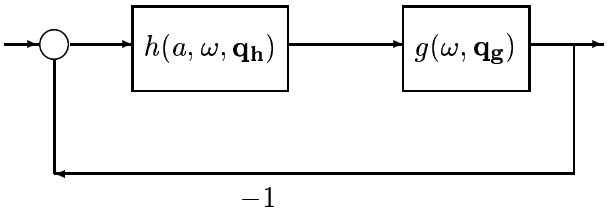

FIGURE 1.2. Block diagram of a typical separable nonlinear control system.

function analysis to predict no limit cycles, even when a limit cycle actually exists. Despite these limitations, describing function analysis has been successfully used in many practical applications, for example, see [2], [13], [53]. For a comprehensive treatment of the describing function approach, see, for instance, [3], [26].

1.2.1 Limit cycle locus

The root locus [21] has established itself as a very useful tool for the analysis and synthesis of linear systems. The root locus readily provides information about the stability and performance of the control system for variations in a given system parameter.

In the context of nonlinear systems, a similar tool that can be thought of is the limit cycle *locus*. The limit cycle locus for a given system parameter is the locus of the limit cycle points as the parameter varies over a given range. The chosen parameter may belong to the linear or nonlinear element of the system. The limit cycle locus can be useful in a variety of situations in nonlinear system analysis and design, for instance, in obtaining graphical insight into the system behavior, or in tuning the parameters of a controller. However, to our knowledge the direct limit cycle locus tool does not as yet exist in the control literature, and may be introduced. Though, somewhat similar concept of M-locus exists in the parametric approach of control system analysis literature [63], it does not use more useful and direct concept of limit cycle locus as introduced in our work. The M-locus approach is based on the Mikhailov stability criterion [63], and is quite tedious with the restricted applicability.

1.2.2 Limit cycle set

In real life, often there are uncertainties in the parameters of the nonlinear system. The conventional graphical technique [26] for finding the set of limit cycle points becomes quite tedious when uncertainties are present in the system parameters, because it requires plotting a number of polar plots and/or describing function curves depending on the parameters and their ranges. Indeed, the describing function approach for computing the limit cycle points of uncertain nonlinear systems has itself only recently attracted the attention of researchers. Fadali and Chachavalvoong [22] overbound the unknown coefficients of the system characteristic equation and use Kharitonov's theorem to do the overall stability test. The method gives non-conservative results only when the numerator of the linear element is constant. Tierno [64] fits a rational approximation to the describing function of the nonlinear element, and incorporates describing function analysis into a generalized structured singular value (μ) framework of robustness analysis. His method requires a good rational approximation to the describing function of the nonlinear element. Ferreres and Fromion [24] propose a μ based method for limit cycle analysis, but the method assumes uncertainties only in the linear element. Impram and Munro [33] pose the describing function analysis problem in a generalized interval polynomial framework. This method is non-conservative only when the coefficients of the linear element have interval or affine linear uncertainty structure.

Thus, there is a lack of methods which can readily compute the set of limit cycle points for the large and important class of uncertain nonlinear systems having

- a linear part represented by a *nonrational* transfer function with coefficients having a *nonlinear* parametric uncertainty structure, and
- a nonlinear part represented by a *nonrational* describing function with uncertain parameters.

Such nonlinear systems are commonly found, for instance, in chemical processes - in heat exchanger systems [11] and distillation columns [58], and in nuclear reactor systems [32].

1.3 Interval analysis

It is seen from the foregoing that nearly all existing methods of control system analysis, especially those involving parametric uncertainty, require some form of grid to be setup during the computations, such as a frequency space grid, a parameter space grid, and in general, a domain grid. As mentioned previously, the use of gridding leads to certain difficulties concerning the reliability and accuracy of the computed results. These difficulties essentially arise because existing methods are founded on the use of *point* methods and "*point* methods and computations with ordinary *floating-point* numbers have no direct way of dealing with

sets containing infinitely many or uncountably many points" [49, section 5], and further, "using *point* methods, there may be no indication, let alone guarantee, of the accuracy or completeness of the results" [18].

On the other hand, we see that in the current context of control system analysis, we deal with information or data that naturally occurs in the form of *intervals*, for e.g., computing frequency response plots over a frequency *interval*, computing stability margins for parameters varying over *intervals*, computing limit cycles over *interval* variations in parameter, amplitude and frequency variables, etc. Therefore, it seems to be more appropriate to adopt the *interval* analysis framework for the development of control system analysis methods.

The idea behind the interval analysis (IA) methods [47] is to design algorithms, which in a single computation, do the approximation and a rigorous error analysis. With interval methods, one can directly deal with interval sets containing infinitely many points, and perform set operations such as subdivisions, unions, intersections, finding convex hulls, testing for set inclusion, testing for disjointedness of sets, etc. The basis for the systematic approach of IA methods is to combine computer arithmetic with order relations. For details of IA and its methods, see [1], [28], [36], [48], [52], [60].

1.4 Objectives

The primary objective of this work is to develop reliable and accurate tools for the analysis of single-input single-output control systems involving linear systems, nonlinear systems, and nonlinear parametric uncertainty structures.

In particular, the following four main objectives are set for the present work:

1. *Frequency response plots:* To develop algorithms for reliably computing the well-known Bode, Nyquist, and Nichols frequency response plots to a prescribed accuracy, for a large class of linear systems. In particular, for nonrational transfer functions the algorithms should be readily applicable without requiring any rational approximations.

2. *Gain and phase margins:* To develop algorithms for reliably computing *all* gain margins, phase margins, and crossover frequencies to a prescribed accuracy, for a large class of uncertain linear systems. In particular, the algorithm should be readily applicable to a large class of nonrational transfer functions with nonlinear parametric dependencies, without requiring any rational approximations.

3. *Spectral set:* To develop algorithms for reliably computing the spectral set of uncertain polynomials to a prescribed accuracy, for a large class of nonlinear parametric dependencies.

4. *Limit cycle analysis:* To develop algorithms for reliably computing the limit cycle behavior of uncertain nonlinear systems to a prescribed accuracy, for a large class of nonrational transfer functions and describing functions with nonlinear parametric dependencies. In particular, the algorithm should be readily applicable to nonrational type of transfer and describing functions without requiring any rational approximations.

In addition, the desired properties of the above algorithms are specified as follows.

1. *Wide applicability:* The algorithms should apply to the following classes of linear systems, nonlinear systems, and parametric uncertainty structures, as the case may be:

   (a) Linear systems: The algorithms should apply to very generic forms of nonrational transfer functions consisting of time delay, trigonometric, and transcendental (such as inverse hyperbolic) terms.

   (b) Nonlinear systems: The algorithms should apply to a large class of separable nonlinearities represented by nonrational describing functions, including memory less, memory type, frequency independent, and frequency dependent nonlinearities.

   (c) Parametric uncertainty structures: The algorithms should apply to a large class of parametric uncertainty structures associated with the linear and nonlinear system, including interval, affine linear, multilinear, and general nonlinear types.

2. *Computational properties:* Each algorithm must offer several guarantees concerning reliability, enclosure, convergence, and finite termination of the computations. To elaborate,

   - *Reliability:* The reliability of the algorithm guarantees that the computed results are trustworthy despite the various computational errors, such as roundoff, approximation, and truncation.
   - *Enclosure:* The enclosure property of the algorithm guarantees that the computed solution set always contains all points of the actual solution set - that is, no point of the actual solution set is ever missed out. This means that in the frequency response plots, *none* of the magnitude or phase peaks and dips are missed out; similarly, in computing the gain and phase margins, spectral sets, or limit cycles, this property guarantees that *none* of the actual values of margins and cross over frequencies, characteristic roots, or limit cycles are missed out.
   - *Prescribed accuracy:* This property guarantees that the algorithm can generate results to a prescribed accuracy.
   - *Convergence and finite termination:* These properties guarantee that the algorithm indeed converges to the solution of a prescribed accuracy, in a finite number of iterations.

1.5 Contributions

Broadly the thesis pursues a novel approach to linear and nonlinear control system analysis in the framework of interval analysis. The main contributions of the thesis are summarized as follows:

1. *Frequency response plots for nonrational transfer functions:* We propose algorithms to compute the well known Bode, Nyquist, and Nichols frequency response plots for nonrational transfer functions. The proposed algorithms are very widely applicable - the magnitude and phase functions need to be only bounded and continuous in frequency. No rational approximation of any nonrational term in the transfer function is required by the algorithms. The proposed algorithms guarantee that the magnitude and phase plots are reliably computed to a prescribed accuracy, and in a finite number of iterations. Through several practical nonrational examples, we demonstrate the superior performance of the proposed algorithms over the widely used *bode*,*nyquist*, and *nichols* routines in MATLAB's control system toolbox [46], [27] and over the conventional gridding method.

2. *Robust gain and phase margins for nonrational transfer functions with nonlinear parametric dependencies:* We propose an algorithm to compute robust gain and phase margins and crossover frequencies for nonrational transfer functions with nonlinear parametric dependencies. The proposed algorithm is very widely applicable - the magnitude and phase functions need to be only bounded and continuous in frequency and system parameters, and the crossover frequencies be bounded. No rational approximation of the nonrational terms is required in the algorithm. The proposed algorithm guarantees that *all* robust margins and crossover frequencies are reliably found to a prescribed accuracy, and in a finite number of iterations. Moreover, an upper bound for the maximum number of iterations is *a priori* computable. Through several examples, we demonstrate the superiority of the proposed algorithm over the widely used *allmargin* routine in MATLAB's control system toolbox [46], [27] and the conventional gridding method.

3. *Spectral set for uncertain polynomials with affine linear parametric dependencies:* We propose an algorithm to compute the spectral set of a polytope of polynomials. The proposed algorithm guarantees that the spectral set is reliably computed to a prescribed accuracy, and that it contains all the actual spectral set points. Moreover, for a given accuracy, it computes the spectral set in a finite number of iterations, and an upper bound for this number is given. A further merit is that the computational complexity of the proposed algorithm is $O(n)$ in contrast to $O(n^2)$ for existing techniques, where n is the degree of the polynomial. We demonstrate the algorithm on a few examples taken from the literature.

4. *Spectral set for uncertain polynomials with nonlinear parametric dependencies:* We propose an algorithm to compute the spectral set of uncertain polynomials with nonlinear parametric dependencies. The proposed algorithm is very widely applicable - it can compute the spectral set for a very general class of uncertain polynomials where the polynomial coefficients need to be only continuous in the parameters. Further, the proposed algorithm guarantees that the spectral set is reliably computed to a prescribed accuracy, and that it contains all the actual spectral set points. Moreover, for a given accuracy, it computes the spectral set in a finite number of iterations, and an upper bound for this number is given. We demonstrate the algorithm on an example that cannot be solved using existing techniques.

5. *Limit cycle locus:* We introduce a novel and powerful tool called the limit cycle *locus* for the analysis of nonlinear systems. The limit cycle locus for a given uncertain parameter is the locus of the limit cycle points as the parameter varies over its range. The concept is similar to that of the root locus in linear systems, and can be as useful for nonlinear control systems. We then propose a reliable and accurate algorithm to compute the limit cycle locus for separable nonlinear systems with nonlinear parametric dependencies. The uncertainty may be in the parameter of the linear or nonlinear element in the system. The proposed algorithm makes use of the popular describing function technique and tools of interval analysis. We demonstrate the capability of the proposed tool on an example involving nonrational transfer and describing functions with nonlinear parametric dependencies, that cannot be readily solved using existing methods. An additional example is given to demonstrate how the proposed algorithm can be applied to tune a controller for achieving a prescribed limit cycle behavior.

6. *Limit cycle set:* We apply the above algorithm to compute the limit cycle set for uncertain nonrational nonlinear systems with nonlinear parametric dependencies. The proposed algorithm computes the limit cycles for a wide class of uncertain nonlinear systems where the transfer and describing functions need to be only continuous in the parameters and continuously differentiable in the amplitude and frequency of the periodic input signal. The proposed algorithm guarantees that the limit cycles are reliably computed to a prescribed accuracy, and that all (if any) limit cycles are indeed found. Moreover, for a prescribed accuracy, the proposed algorithm computes all the limit cycles in a finite number of iterations, and an upper bound for this number is given. We demonstrate the algorithm on the above challenging uncertain nonlinear system example that cannot be readily solved using existing methods.

1.6 Thesis organization

The rest of the thesis is organized as follows.

In chapter 2, we present algorithms for computing the Bode, Nyquist and Nichols frequency response plots. Through several practical nonrational examples, we demonstrate the superior performance of the proposed algorithms over the widely used routines in MATLAB's control system toolbox [46], [27] and over the conventional gridding method.

In chapter 3, we present an algorithm for computing robust gain margins, phase margins, and corresponding crossover frequencies for a very general class of nonrational transfer functions with nonlinear parametric dependencies. We prove the various properties of the proposed algorithm concerning enclosure, convergence, accuracy, reliability, and finite termination. Through several examples, we demonstrate the superiority of the proposed algorithm over the widely used *allmargin* routine in MATLAB's control system toolbox [46], [27] and the conventional gridding method.

In chapter 4, we present an algorithm for computing the spectral set for a polytope of polynomials. We prove the various properties of the proposed algorithm concerning enclosure, convergence, accuracy, reliability, computational complexity, and finite termination. We test and compare the algorithm on a few examples taken from the literature.

In chapter 5, we present an algorithm for computing the spectral set for polynomials with nonlinear parametric dependencies. We prove the various properties of the proposed algorithm concerning enclosure, convergence, accuracy, reliability, and finite termination. We demonstrate the algorithm on an example that cannot be solved using existing techniques.

In chapter 6, we present a similar algorithm to the above one for computing the limit cycle set for uncertain nonrational nonlinear systems with general nonlinear parametric dependency. We also introduce a new tool called the limit cycle locus. We demonstrate the capability of the proposed limit cycle locus tool and the algorithm on an example that cannot be readily solved using the existing methods. An additional example is given to demonstrate how the proposed tool of *limit cycle locus* and the algorithm can be applied to tune a controller for achieving a prescribed limit cycle behavior.

In Chapter 7, the overall conclusions of the work are given along with some suggestions for further work.

Some of the preliminaries of interval analysis required in this work are given in Appendix I.

2

Frequency responses

2.1 Introduction

In section 1.1.1, we described the limitations of existing tools for computing the Bode, Nyquist, and Nichols frequency response plots for *nonrational* transfer functions. In this chapter, we address these limitations and present algorithms that can *reliably* and *accurately* compute the frequency response plots for nonrational transfer functions, without requiring any rational approximations. The proposed algorithms, called as the vectorized-adaptive (VA) algorithms, are based on tools of interval analysis [48]. The main features of the proposed VA algorithms are:

1. VA algorithms are applicable to nonrational transfer functions whose magnitude and phase functions are bounded and continuous in frequency. Subject to these assumptions, there is no restriction on the structure or form of the transfer function. Thus, the transfer function can be described by *any* sequence of arithmetic expressions involving the frequency, using the built -in functions of a programming language. On the other hand, frequency response tools based on the automatic frequency selection procedure in MATLAB's control systems toolbox are restricted to rational transfer functions.

2. VA algorithms are *readily* applicable to nonrational transfer functions, that is, they do not need rational approximations to the transfer functions. Thus, multiple time delays and transcendental terms can be handled with equal ease, without the need for any rational approximation.

3. VA algorithms automatically compute magnitude and phase values that are guaranteed to have a prescribed accuracy. Moreover, error estimates are readily available from the computed plots. Thus, VA algorithms overcome the difficulties associated with

the conventional gridding method, such as proper grid size selection and lack of error estimates.

4. VA algorithms guarantee that the computed magnitude and phase values are *reliable* in face of all kinds of computational errors, such as roundoff, truncation, and approximation. In contrast, existing methods do not account for such computational errors.

5. VA algorithms compute the required plots in a *finite* number of algorithmic iterations, for a prescribed accuracy.

The rest of this chapter is organized as follows. In section 2.2, we present the proposed algorithm for computing Bode plots. In section 2.3, we give the proposed algorithms for computing Nyquist and Nichols plots, and in Section 2.4, the mathematical properties of the proposed algorithms. We test and compare the performance of the various methods on several nonrational examples in section 2.5, and give the conclusions of the chapter in section 2.6.

2.2 Proposed algorithm

Consider a linear system represented by the transfer function $g(s, \mathbf{q})$, where s is the Laplace variable and $\mathbf{q} \in \Re^n$ is a given vector of system parameters. Let ω denote the frequency. Define the magnitude and phase functions of $g(s, \mathbf{q})$ as

$$g_{mag}(\omega, \mathbf{q}) := 20 \log_{10} |g(s = j\omega, \mathbf{q})| \;\; ; \quad g_{phase}(\omega, \mathbf{q}) := \angle g(s = j\omega, \mathbf{q}) \tag{2.1}$$

Note that the magnitude is expressed in decibels (dB) and phase in degrees. We assume that the magnitude and phase functions are bounded and continuous in ω. For functions satisfying these assumptions, we propose algorithms that compute the Bode, Nyquist, and Nichols plots to a prescribed accuracy.

2.2.1 Algorithm for Bode magnitude plot

We first present the algorithm for computing Bode plots to a prescribed accuracy. The proposed algorithm uses natural inclusion functions [47] for interval evaluation of the magnitude and phase functions. We can find a natural inclusion function of the magnitude function as follows, see also Appendix I. In the expression for g_{mag}, replace each occurrence of ω by $\mathbf{\Omega}$, each occurrence of a pre-declared real function (like $\sin, \cos, \exp$, etc.) by the corresponding pre-declared interval function, and all real arithmetic operations by the corresponding interval arithmetic operations. The natural inclusion function of $g_{mag}(\omega, \mathbf{q})$ is denoted $G_{mag}(\mathbf{\Omega}, \mathbf{q})$. On identical lines, we can construct the natural inclusion function $G_{phase}(\mathbf{\Omega}, \mathbf{q})$ for the phase function.

Let $\mathbf{\Omega}^0$ denote the frequency range over which the Bode plots are required. With a single evaluation of $G_{mag}(\mathbf{\Omega}^0, \mathbf{q})$, we can obtain a magnitude plot comprising of a single frequency - magnitude box[1], or simply, a magnitude box. By inclusion property of natural inclusion functions [48, Theorem 3.1], $G_{mag}(\mathbf{\Omega}^0, \mathbf{q})$ encloses the actual Bode plot over entire $\mathbf{\Omega}^0$. However, this magnitude box $G_{mag}(\mathbf{\Omega}^0, \mathbf{q})$ usually has a width that considerably exceeds the prescribed magnitude accuracy ε_{dB}.

We may therefore repeatedly subdivide the given frequency range $\mathbf{\Omega}^0$, find the evaluations of G_{mag} over the frequency subintervals using interval arithmetic, and take the union of the results to get Bode magnitude plots comprising of smaller and smaller magnitude boxes which give increasingly accurate information about the actual magnitude values. By a fundamental result of interval analysis [48, Theorem 4.1], these magnitude plots will converge to the actual magnitude plot as we refine the partition of $\mathbf{\Omega}^0$. We may stop the subdivision process when the widths of all the magnitude boxes are less than the prescribed accuracy ε_{dB}. A similar method can be followed for $G_{phase}(\mathbf{\Omega}^0, \mathbf{q})$ to obtain a phase plot of a prescribed accuracy ε_{deg}. Thus, the proposed algorithm computes a collection of frequency - magnitude and frequency - phase boxes covering the actual Bode magnitude and phase plots. The magnitude or phase side of each box in this collection has a width not exceeding the prescribed accuracy tolerance ε_{dB} or ε_{deg}, respectively.

There are two existing approaches to the subdivision processes in interval analysis, namely, the uniform subdivision process and the adaptive subdivision process. The references for these two subdivision processes are discussed in [35], and [48, sec. 4.1], respectively. The attractive feature of the former approach is vectorized evaluation of G_{mag} and G_{phase} over all frequency subintervals of a partition, while that of the latter is *adaptive* subdivision of $\mathbf{\Omega}$.

The proposed algorithm is based on a novel subdivision process that combines the two advantageous features of existing subdivision processes mentioned above, i.e., vectorized functional evaluation and adaptive subdivision. We call the new subdivision process as the vectorized - adaptive process, and the proposed algorithm based on it as the *vectorized - adaptive* (VA) algorithm. For details on the use of vectorized interval operations for function evaluations, subdivisions, width checks, etc., see [51].

We first present the proposed VA algorithm for finding the Bode magnitude plot.

Algorithm (Computation of the Bode magnitude plot)

Input : An expression for the magnitude function $g_{mag}(\omega, \mathbf{q})$, the frequency range $\mathbf{\Omega}^0$ over which the response is to be computed, and a prescribed accuracy tolerance on the magnitude ε_{dB}.

Output: A plot of magnitude boxes enclosing the actual Bode magnitude plot. The width of the magnitude side of each box in the plot does not exceed ε_{dB}.

[1] In the sequel, we prefer to use the generic term *box* while referring to an *interval* or a *rectangle*.

Note: The algorithm is to be executed in the order given below, except when otherwise indicated.

BEGIN Algorithm

1. (**Initialization part**) From the expression for the magnitude function, construct a natural inclusion function G_{mag}.

2. Construct initial box and lists:

   (a) Set $\mathbf{\Omega} \leftarrow \mathbf{\Omega}^0$.

   (b) Set $k \leftarrow 0$ and initialize lists $\mathcal{L}^{sol} \leftarrow \{\}$, $\mathcal{L} \leftarrow \{\mathbf{\Omega}\}$.

3. (**Iterative part**) Start a new iteration:

   (a) Set $k \leftarrow k+1$ and $l_r \leftarrow$ length of $\mathcal{L}$.

   (b) Pick all boxes $\mathbf{\Omega}_{(i)}$, $i = 1, 2, ..., l_r$ from $\mathcal{L}$ and delete their entries from $\mathcal{L}$.

   (c) Evaluate $G_{mag}\left(\mathbf{\Omega}_{(i)}, \mathbf{q}\right)$, for $i = 1, 2, ..., l_r$.

4. Solution set:

   (a) IF $w\left(G_{mag}\left(\mathbf{\Omega}_{(i)}, \mathbf{q}\right)\right) < \varepsilon_{dB}$ THEN enter the pair $\left(\mathbf{\Omega}_{(i)}, G_{mag}\left(\mathbf{\Omega}_{(i)}, \mathbf{q}\right)\right)$ in $\mathcal{L}^{sol}$ and discard $\mathbf{\Omega}_{(i)}$ from further processing, for $i = 1, 2, ..., l_r$.

   (b) If no more $\mathbf{\Omega}_{(i)}$ remain, go to step 7.

5. Subdivision phase:

   For each remaining box $\mathbf{\Omega}_{(i)}$, subdivide $\mathbf{\Omega}_{(i)}$ to get subboxes $\mathbf{\Omega}^1_{(i)}$ and $\mathbf{\Omega}^2_{(i)}$ such that $\mathbf{\Omega}_{(i)} = \mathbf{\Omega}^1_{(i)} \cup \mathbf{\Omega}^2_{(i)}$. Enter the subboxes $\mathbf{\Omega}^1_{(i)} \cup \mathbf{\Omega}^2_{(i)}$ in $\mathcal{L}$.

6. End current iteration: Return to step 3.

7. (**Termination part**) For each pair of items in $\mathcal{L}^{sol}$ plot $G_{mag}\left(\mathbf{\Omega}_{(i)}, \mathbf{q}\right)$ versus $\mathbf{\Omega}_{(i)}$, and EXIT.

END Algorithm.

Remark 2.1 *If $G_{mag}(\mathbf{\Omega}, \mathbf{q})$ is used as an enclosure of the range of g_{mag} over $\mathbf{\Omega}$, then the error can be no greater than the width of $G_{mag}(\mathbf{\Omega}, \mathbf{q})$ itself, see [48, Theorem 3.1]. Therefore, we use the accuracy tolerance condition*

$$w\left(G_{mag}\left(\mathbf{\Omega}, \mathbf{q}\right)\right) < \varepsilon_{dB} \tag{2.2}$$

where, ε_{dB} is the prescribed magnitude accuracy tolerance.

Remark 2.2 *The continuity of the magnitude and phase functions in ω is required in order that the natural inclusion function G_{mag} and G_{phase} be continuous. The continuity property ensures that the widths of the intervals evaluations G_{mag} and G_{phase} tend to zero as the width of $\mathbf{\Omega}$ tends to zero, so that in turn, convergence and arbitrary accuracy can be achieved.*

2.2.2 Algorithm for Bode phase plot

The VA algorithm for computation of Bode phase plot is immediately obtained from the VA algorithm for Bode magnitude plot, by using the phase function g_{phase} instead of the magnitude function g_{mag}, and a prescribed accuracy tolerance ε_{deg} instead of ε_{dB}.

2.3 Algorithms for Nyquist and Nichols plots

The VA algorithm for computing the Nyquist plot is similar to that given above for Bode plots, except for the following changes:

- In step 3c: evaluate both $G_{mag}(\mathbf{\Omega}_{(i)}, \mathbf{q})$ and $G_{phase}(\mathbf{\Omega}_{(i)}, \mathbf{q})$ over $\mathbf{\Omega}_{(i)}$, for $i = 1, 2, ..., l_r$.
- In step 4a: IF

  $$w\left(G_{mag}(\mathbf{\Omega}_{(i)}, \mathbf{q})\right) < \varepsilon_{dB} \text{ and } w\left(G_{phase}(\mathbf{\Omega}_{(i)}, \mathbf{q})\right) < \varepsilon_{\text{deg}}$$

  THEN enter the triple $\left(\mathbf{\Omega}_{(i)}, G_{mag}\left(\mathbf{\Omega}_{(i)}, \mathbf{q}\right), G_{phase}(\mathbf{\Omega}_{(i)}, \mathbf{q})\right)$ in $\mathcal{L}^{sol}$, and discard $\mathbf{\Omega}_{(i)}$ from further processing, for $i = 1, 2, ..., l_r$.
- In step 7: For each pair of items in $\mathcal{L}^{sol}$ with the magnitude interval in absolute units, draw the polar plot of $\left(G_{phase}(\mathbf{\Omega}_{(i)}, \mathbf{q}), G_{mag}\left(\mathbf{\Omega}_{(i)}, \mathbf{q}\right)\right)$, annotate $\mathbf{\Omega}_{(i)}$, and EXIT.

The VA algorithm for computing the Nichols plot is similar to that for Nyquist plots, except for the following change:

- In step 7: For each pair of items in $\mathcal{L}^{sol}$ plot $G_{mag}\left(\mathbf{\Omega}_{(i)}, \mathbf{q}\right)$ versus $G_{phase}(\mathbf{\Omega}_{(i)}, \mathbf{q})$, annotate $\mathbf{\Omega}_{(i)}$, and EXIT.

2.4 Properties

The mathematical and computational properties of the VA algorithms readily follow from well-known theorems in interval analysis:

- Guaranteed enclosure property: it follows immediately from the inclusion property of natural inclusion functions [48, Theorem 3.1] that for any $\varepsilon_{dB}, \varepsilon_{\text{deg}} > 0$, the computed frequency response plots indeed enclose the corresponding exact ones.

- Convergence property: it is a fundamental result in interval analysis [48, Theorem 4.1] that as we refine the partition, the enclosures will converge to the actual range of the values over the given set. From this property, it follows that in the limiting case of $\varepsilon_{dB}, \varepsilon_{\text{deg}} = 0$, the VA algorithms give frequency response plots that converge to the exact ones.

- Error estimates: from Remark 2.1, the possible error at each frequency can be readily computed as the width of the corresponding box in the plot. Then, the maximum possible error over the entire frequency interval is the maximum width over all the boxes in the plots.

- Finite termination property: for any $\varepsilon_{dB}, \varepsilon_{\text{deg}} > 0$, the VA algorithms compute frequency response plots of a prescribed accuracy in a *finite* number of steps. This can be shown by proceeding on similar lines to [34, Theorem 2.10].

- Computational reliability of the VA algorithms means that the algorithms are stable when implemented on floating-point systems. We can make the VA algorithms computationally reliable by implementing them in an interval arithmetic compiler which uses machine interval arithmetic in all computations [38].

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

- Range of application of the VA algorithms is very vast, as the algorithms are applicable to any transfer function that is bounded and continuous in frequency.

2.5 Illustrative examples

We now test and compare the performance of the proposed VA algorithm for Bode plots, with those of the *bode* routine of MATLAB's control systems toolbox [27], and the conventional gridding method. We consider four practical nonrational transfer function examples for this testing. In all examples, we set the prescribed accuracy for Bode plots to $\varepsilon_{dB} = 1$ dB and $\varepsilon_{\text{deg}} = 1\,\text{deg}$, and carry out the computations on a PC Pentium-III 550 MHz machine. The examples considered are:

Example 2.1 *The heat exchanger system [56]:*

$$g(s, \mathbf{q}) = \frac{q_1(1 - q_2 e^{-q_3 s})}{(q_4 s + 1)(q_5 s + 1)}$$

where, $q_1 = 5000$, $q_2 = 0.5, q_3 = 10, q_4 = 40$ *and* $q_5 = 15$. *The frequency range of interest is* $\mathbf{\Omega}^0 = [0.01, 100]$ *rad/sec.*

Example 2.2 *The integrating system with measurement delay [45]:*

$$g(s, \mathbf{q}) = \frac{q_1}{s + q_1 e^{-q_2 s}}$$

where, $q_1 = 10$ *and* $q_2 = 10$. *The frequency range is* $\mathbf{\Omega}^0 = [0.01, 10]$ *rad/sec.*

Example 2.3 *The flexible and long hose pipe system connecting a servo-valve with the actuator in a hydraulic servo system [20]:*

$$g(s, \mathbf{q}) = \frac{1}{\cosh(q_1 s) + q_2 \sinh(q_1 s)}$$

where, $q_1 = 10$ *and* $q_2 = 0.1$. *The frequency range is* $\mathbf{\Omega}^0 = [0.01, 10]$ *rad/sec.*

Example 2.4 *The system for heating a* $1 - \dim$ *metal rod by the steam chest [56]:*

$$g(s, \mathbf{q}) = \frac{1}{1 + \frac{\sqrt{s}}{q_1}\sqrt{s}\tanh(\sqrt{s})}$$

where, $q_1 = 10$. *The frequency range is* $\mathbf{\Omega}^0 = [0.01, 1000]$ *rad/sec.*

We first implement the VA algorithm using the interval analysis toolbox INTLAB [61] in the MATLAB environment, and apply it to compute the Bode plots in all examples.

Next, we compute the Bode plots using the *bode* routine of control systems toolbox of MATLAB [46], [27]. As stated earlier, the *bode* routine has an automatic frequency grid selection procedure. However, this routine is applicable only to rational transfer functions. To handle nonrational transfer functions, some rational approximation of the transfer function has to be supplied.

Accordingly, in the first two examples, we approximate the time delay term with the well known first order and second order Pade approximations [16], and apply the *bode* routine to the resulting rational transfer functions. In the last two examples, however, we are unable to apply the *bode* routine, as satisfactory rational approximations to the involved inverse hyperbolic terms are difficult to find (the Pade approximation is applicable only to time delay terms).

Lastly, we compute the Bode plots using conventional gridding of the frequency interval. Two different number of grid points are used for this purpose: 100 and 1000 points.

2.5.1 Results

The Bode plots computed using the various methods are shown in Figs. 2.1 to 2.6. To benchmark and compare the obtained results, the Bode plots are computed using a very dense grid of 5×10^5 points in all examples. The errors in the Bode plots computed with *bode* routine and gridding are calculated with respect to these very dense grid plots. However, with the

VA algorithm the maximum possible error in the plots can be computed immediately as the maximum width of all the boxes in the plots, see Remark 2.1.

Accordingly, the maximum error is found for the various methods and shown in Tables 2.1 and 2.2. For more clarity, some of the Bode plots are also zoomed around the frequencies where the maximum errors occur, and shown in the figures.

2.5.2 Discussion

From the results in Tables 2.1 and 2.2, we observe the following:

Conventional Gridding method: In all examples except the last, the results computed with the conventional gridding method have *significant* errors, even for a grid of 1000 points. The maximum magnitude error is as much as -15 dB in Example 2.2, while the maximum phase error is as much as 90 deg in the same example. Over all examples, on the average the maximum magnitude error is about -9 dB, and the maximum phase error is about 47 deg.

bode **routine of MATLAB**: As mentioned earlier, the *bode* routine could be applied only to the first two examples. In these examples, the results are found to have *considerable* errors in the magnitude, and *very large* errors in the phase plots. The difference in errors with first and second order Pade approximations is relatively little. The maximum magnitude error is as much as $+30$ dB and occurs in Example 2.2, while the maximum phase error is as much as 5474 deg and occurs in the same example.

Proposed VA algorithm: In all examples, the results computed with the VA algorithm have errors within the prescribed tolerances. Further, the VA algorithm is able to capture *all* the magnitude and phase peaks that occur in the given frequency range, whereas some of these peaks are missed by the conventional gridding method. For instance, in Example 2.3, all 32 magnitude peaks are captured by the VA algorithm in the given frequency range, whereas only 15 peaks could be found by the gridding method.

Table 2.3 reports the number of iterations required, computational time taken, and the number of magnitude and phase boxes computed by the VA algorithm. It is interesting to note that in nearly all examples, the CPU time taken by the VA algorithm is just a few seconds. Thus, the VA algorithm is evidently computationally efficient.

2.5.3 Nichols and Nyquist plots

We also test and compare the performance of the proposed VA algorithms for Nichols and Nyquist plots, with those of the *nichols* and *nyquist* routines of MATLAB's control systems toolbox [27] and the conventional gridding method. We consider the same nonrational transfer function examples given above for this testing.

A sample Nichols plot is shown in Fig. 2.7 for Example 2.1, while Table 2.4 reports the number of iterations required, computational time taken, and the number of result boxes

computed by the VA algorithm for computing the Nyquist plots. Similar remarks to the Bode plots hold for these plots.

2.6 Conclusions

The results of the examples show that the application of Pade approximations in conjunction with the *bode* routine of the control systems toolbox [27] may lead to Bode plots of *poor* accuracy, for nonrational transfer functions. Further, the amount of error present in the plots computed with this method remains unknown.

Moreover, the widely applicable conventional gridding method to compute frequency response plots is also liable to yield large errors in the plots. The main difficulty with this method is that, in general, the user is unaware of what the grid size should be so that plots of prescribed accuracy can be obtained. For instance, in our examples, it was found that grids even with 1000 points yielded plots of poor accuracy. Without a good estimate of the grid size to be used and without a systematic way of estimating the error present in the computed plots, there is every risk that one may be lead to erroneous analysis and synthesis results using the plots computed with the conventional gridding method.

In contrast, with the proposed VA algorithm we obtain *guarantees* that the computed frequency response plots are *reliable* and *accurate* throughout a given frequency range. The VA algorithm can be used for a very wide spectrum of nonrational transfer functions, and it does not require rational approximation of any nonrational terms. The VA algorithm thus relieves the user of the difficulties associated with proper grid size selection, finding good rational approximations, and the lack of error estimates associated with existing frequency response computation methods.

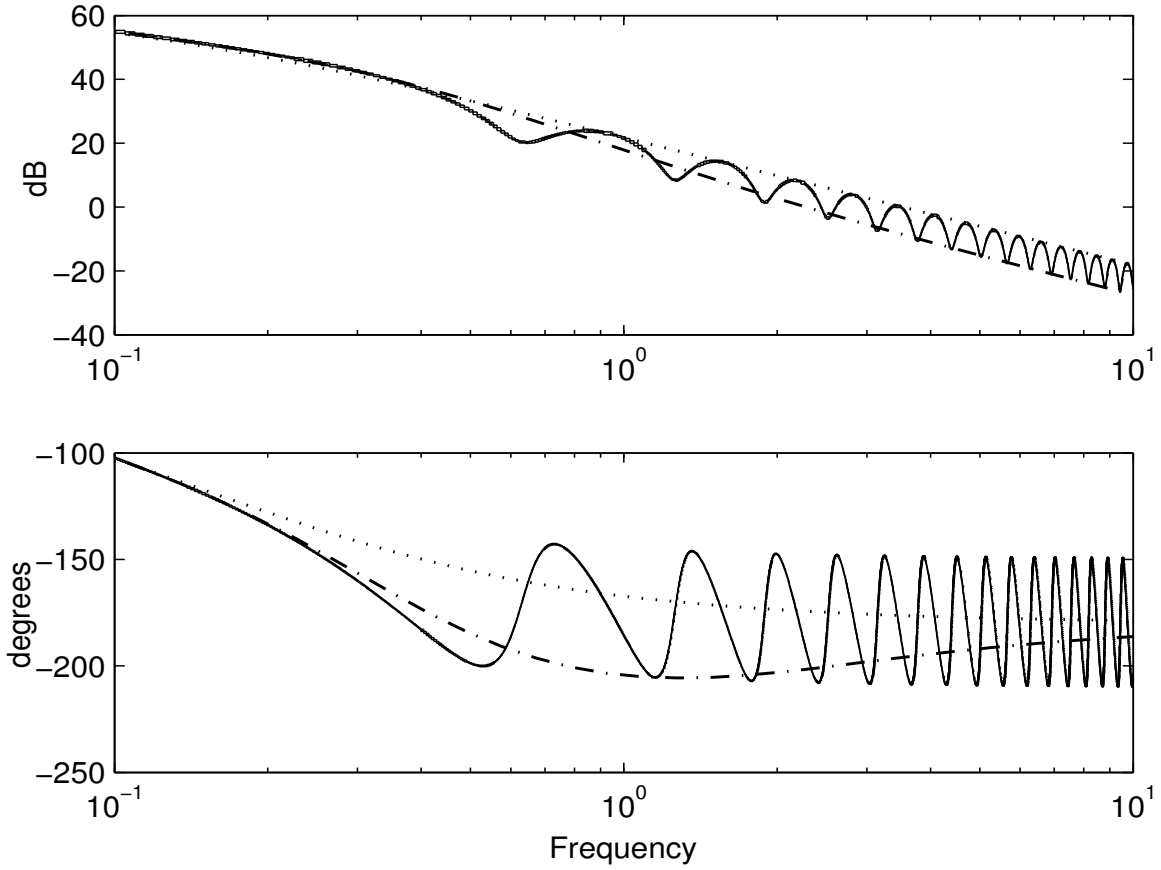

FIGURE 2.1. Comparison of Bode plots obtained using different methods for the heat exchanger system in Example 2.1. For better clarity of the errors, the plots are displayed here only over the frequency range $[0.1, 10]$. (dotted: Pade 1st order, dash-dot: Pade 2nd order, solid boxes: proposed algorithm).

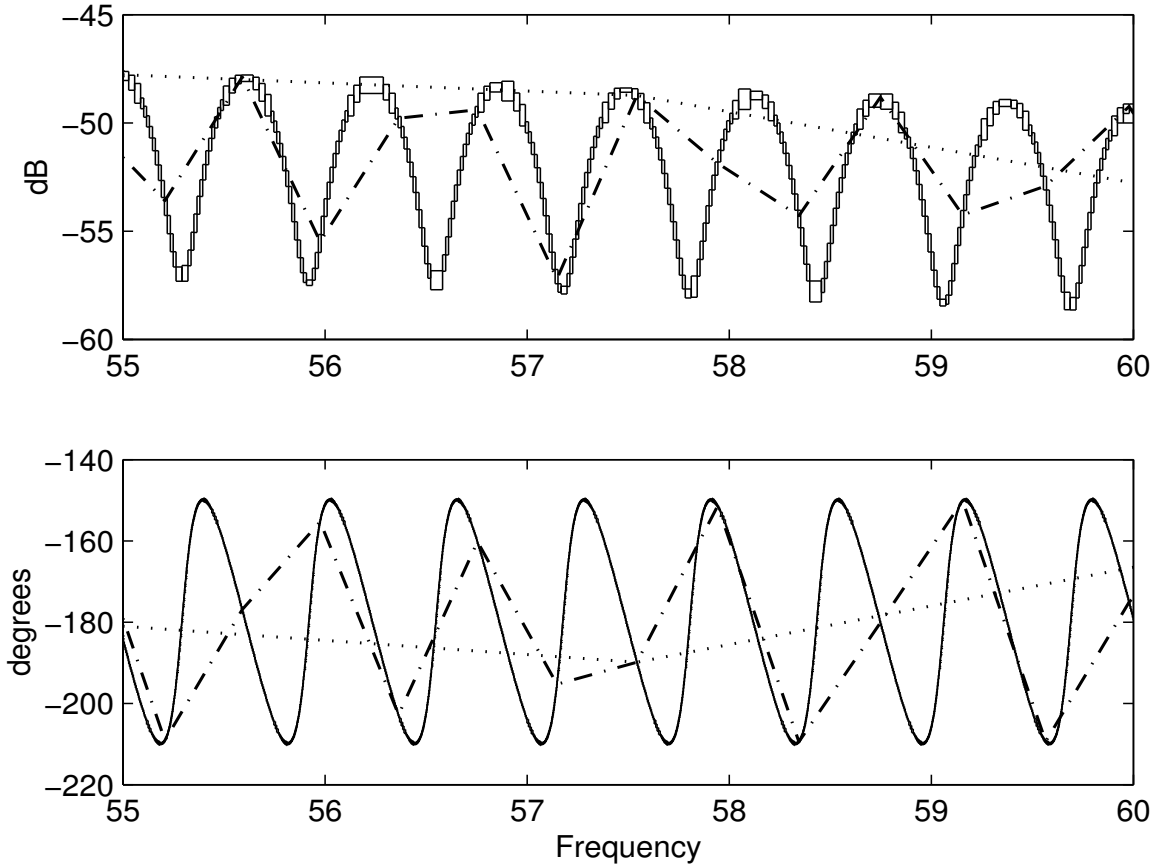

FIGURE 2.2. Bode plots for the heat exchanger system in Example 2.1, zoomed around the frequencies where large errors occur. (dotted: grid size 100, dash-dot: grid size 1000, solid boxes: proposed algorithm).

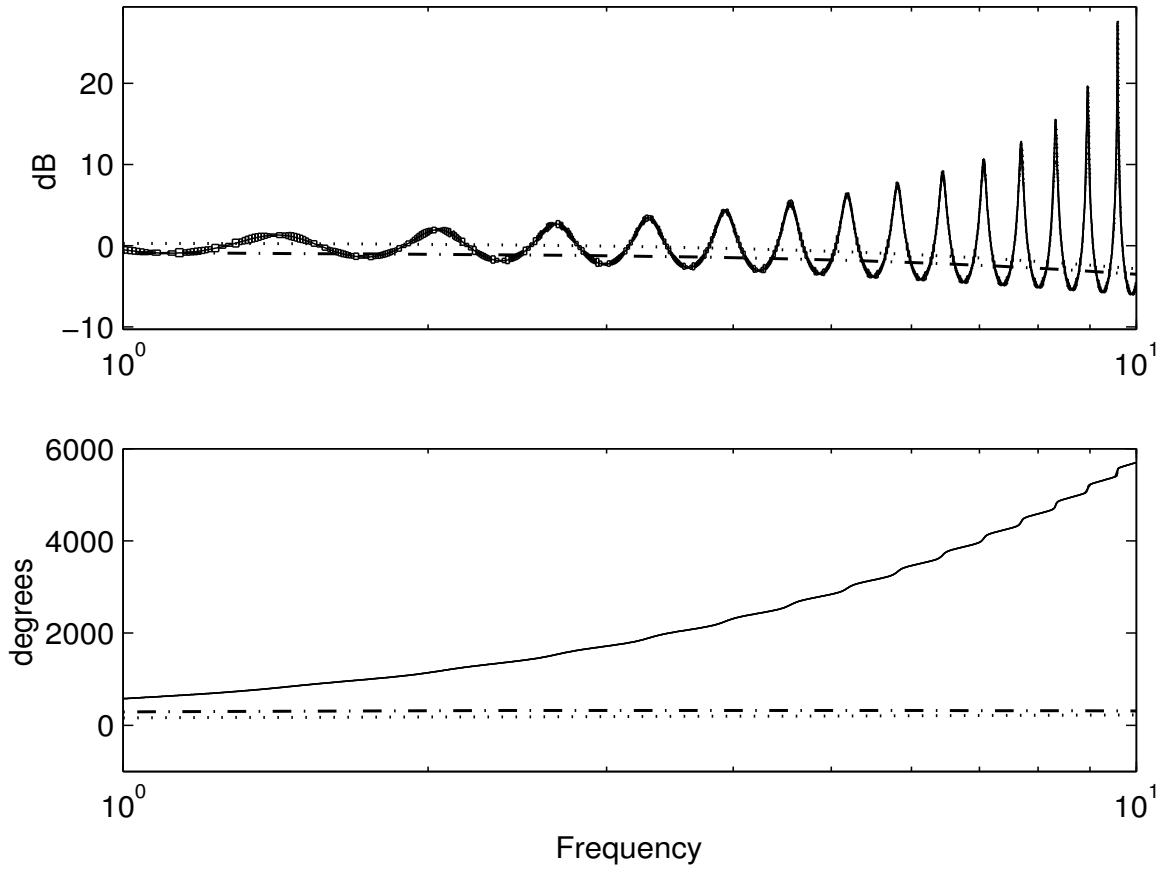

FIGURE 2.3. Comparison of Bode plots obtained using different methods for the integrating system with measurement delay in Example 2.2. For better clarity, the plots are displayed here only over the frequency range $[1, 10]$. (dotted: Pade 1st order, dash-dot: Pade 2nd order, solid boxes: proposed algorithm).

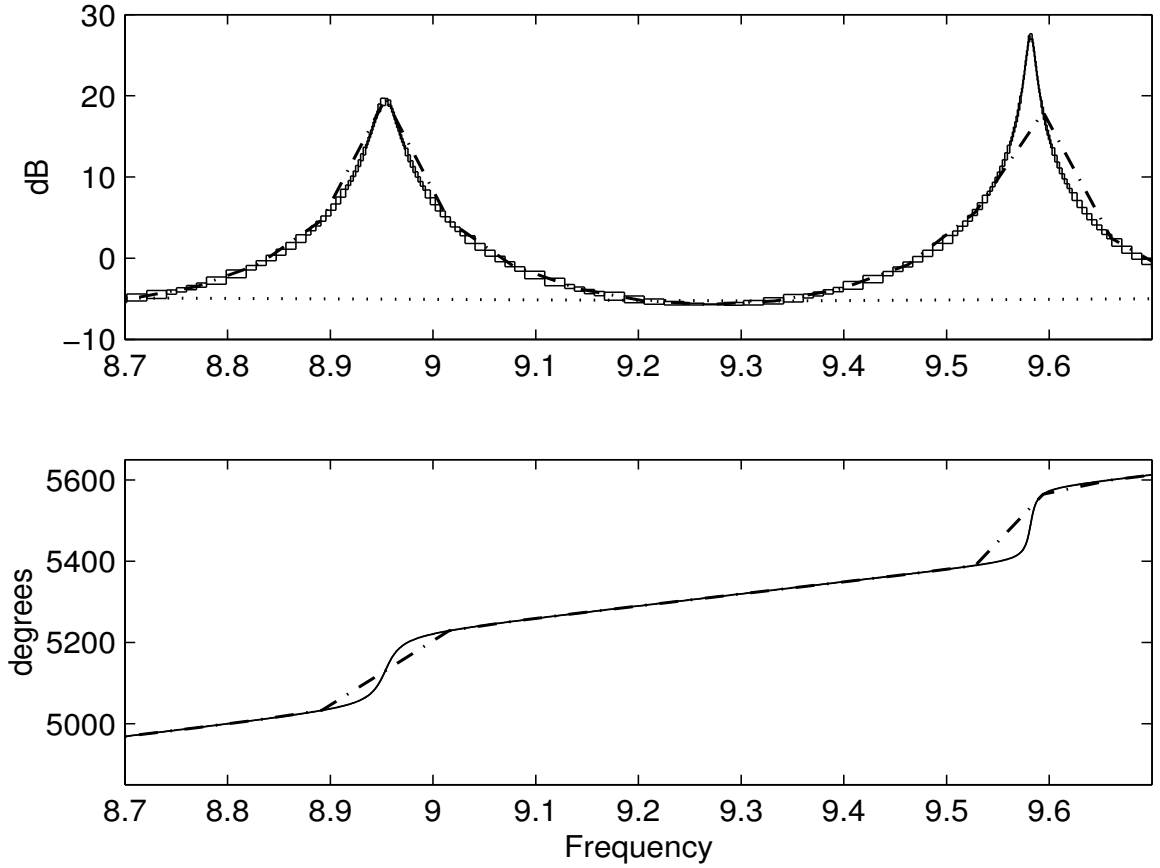

FIGURE 2.4. Bode plots for the integrating system with measurement delay in Example 2.2, zoomed around the frequencies where large errors occur. (dotted: grid size 100, dash-dot: grid size 1000, solid boxes: proposed algorithm). Note: The phase plot with grid size 100 is out of range in the shown plot.

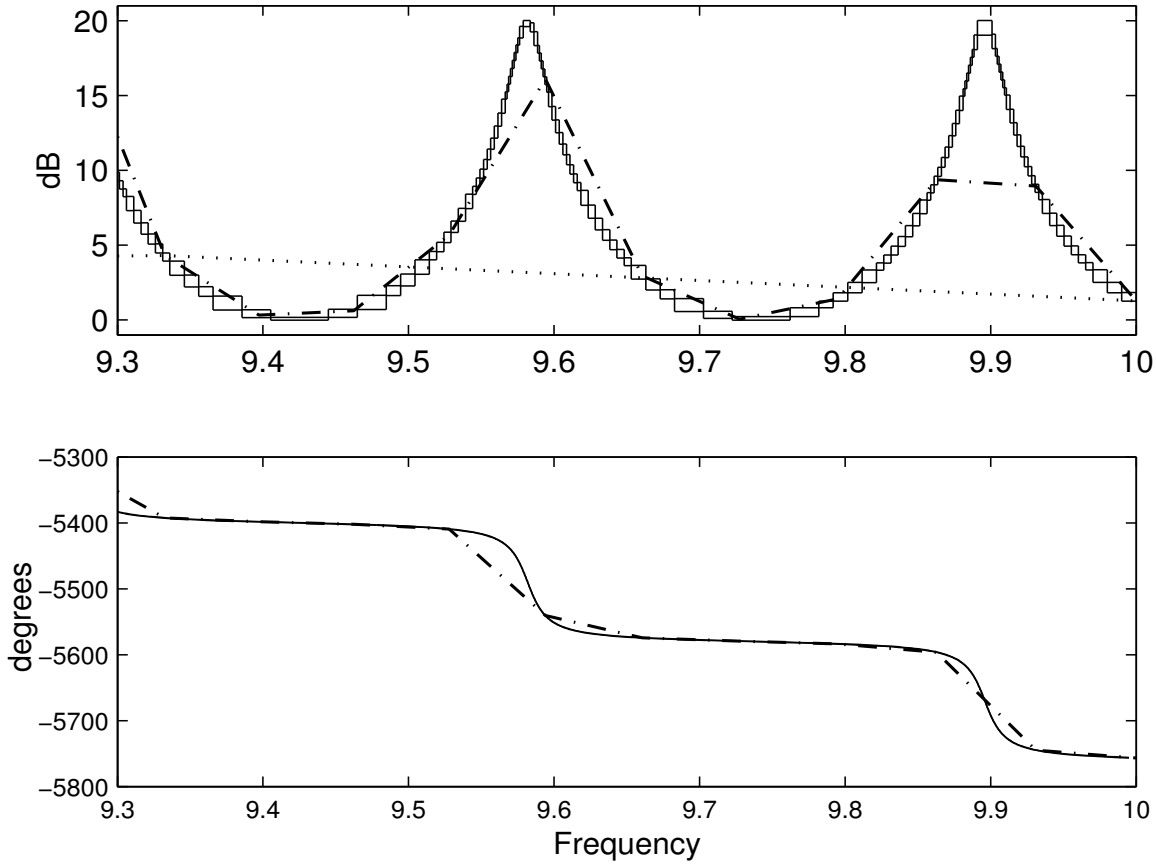

FIGURE 2.5. Bode plots for the hydraulic servo system with long tube in Example 2.3, zoomed around the frequencies where large errors occur. (dotted: grid size 100, dash-dot: grid size 1000, solid boxes: proposed algorithm). Note: The phase plot with grid size 100 is out of range in the shown plot.

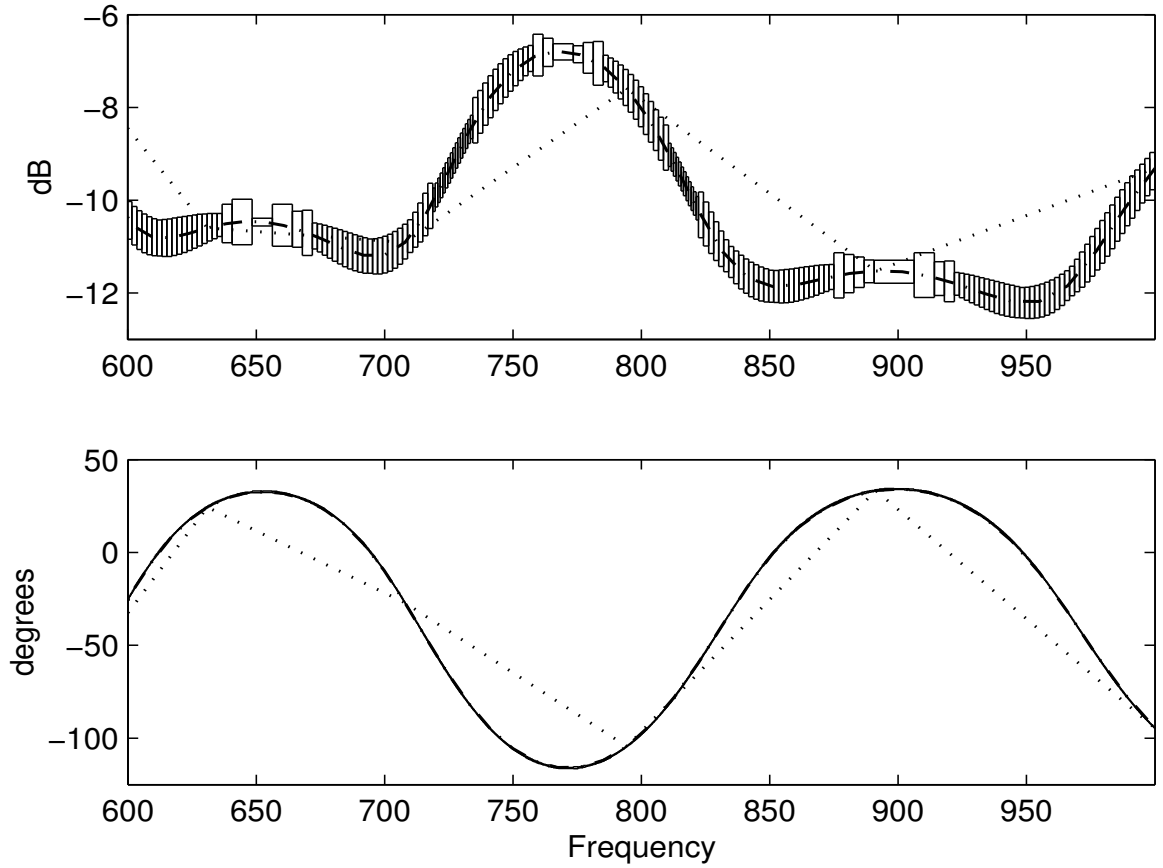

FIGURE 2.6. Bode plots for the steam chest rod heating system in Example 2.4, zoomed around the frequency where the large errors occur. (dotted: grid size 100, dash-dot: grid size 1000, solid boxes: proposed algorithm).

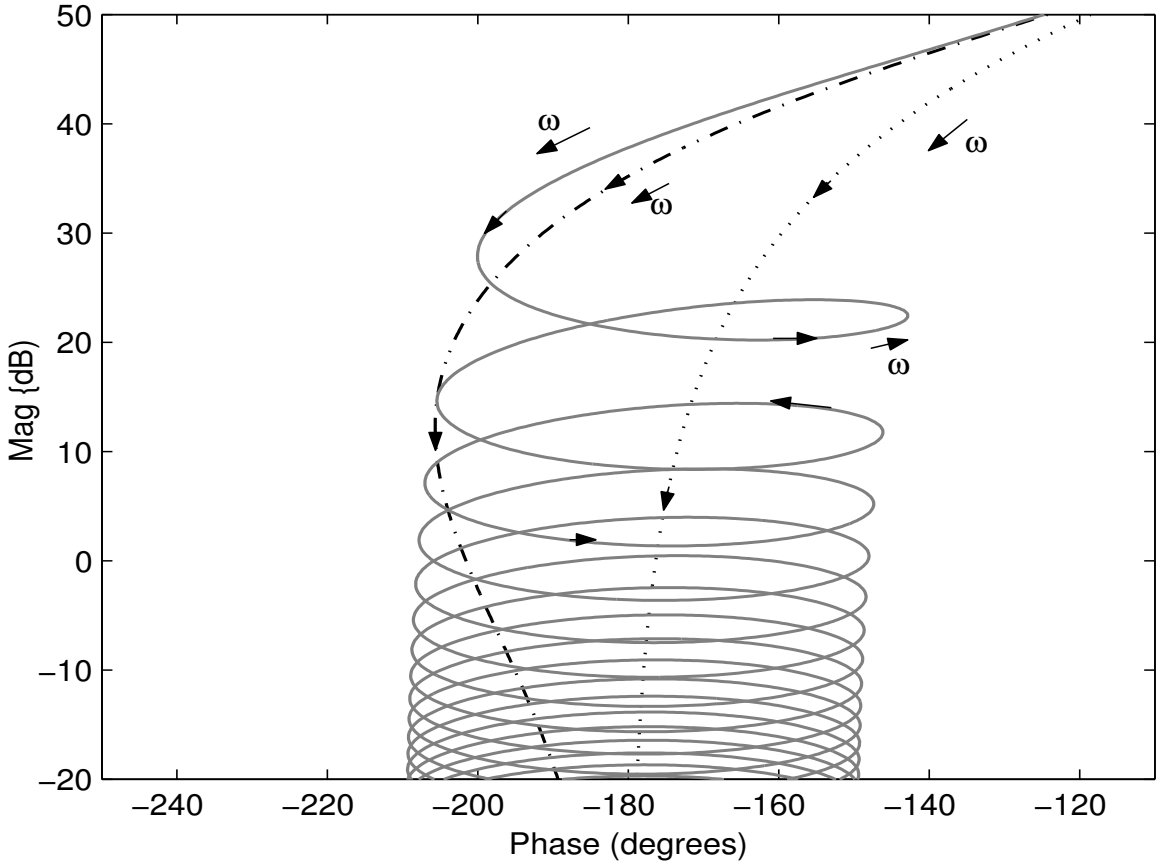

FIGURE 2.7. Comparison of Nichols plots obtained using different methods for the heat exchanger system in Example 2.1. (dotted: Pade 1st order, dash-dot: Pade 2nd order, thick solid: proposed algorithm). Note: Only midpoints of the boxes obtained by the proposed algorithm are plotted.

TABLE 2.1. Maximum errors in Bode magnitude plot using various methods. .

Example	Max. error in mag, dB				
	Grid points		MATLAB's *Bode* [27]		VA
	100	1000	1^{st} order Pade	2^{nd} order Pade	alg.
2.1 Heat exchanger	-10	-10	$+10$	$+9$	$\pm 0.99..$
2.2 Integ. with delay	-37	-15	$+30$	$+30$	$\pm 0.99..$
2.3 Long pipe	-20	-10	–	–	$\pm 0.99..$
2.4 Heating of a rod	-2.5	-0.5	–	–	$\pm 0.99..$

TABLE 2.2. Maximum errors in Bode phase plot using various methods.

Example	Max. error in phase, deg				
	Grid points		MATLAB's *Bode* [27]		VA
	100	1000	1^{st} order Pade	2^{nd}order Pade	alg
2.1 Heat exchanger	-130	-35	$+45$	$+60$	$\pm 0.99..$
2.2 Integ. with delay	$+100$	$+90$	$+5474$	$+5388$	$\pm 0.99..$
2.3 Long pipe	-70	-60	–	–	$\pm 0.99..$
2.4 Heating of a rod	-60	-1	–	–	$\pm 0.99..$

TABLE 2.3. Performance metrics of the proposed VA algorithm to compute Bode plots.

Example	Time, sec	No. of iterations	No. of boxes
2.1 Heat exchanger	15.75	36	44,200
2.2 Integ. with delay	4.59	34	9,435
2.3 Long pipe	4.61	31	10,014
2.4 Heating of a rod	4.43	35	4,816

TABLE 2.4. Performance metrics of the proposed VA algorithm to compute Nyquist plots.

Example	Time, sec	No. of iterations	No. of boxes
2.1 Heat Exchanger	24.85	20	78,152
2.2 Integ. with delay	6.35	19	16,784
2.3 Long pipe	5.93	17	16,062
2.4 Heating of a rod	5.48	19	8,274

3

Gain and phase margins

3.1 Introduction

In section 1.1.2, we described the limitations of existing algorithms in computing the gain and phase margins for nonrational transfer functions. In this chapter, we address these limitations. We present an algorithm that can reliably and accurately compute the robust gain and phase margins and crossover frequencies for systems represented by nonrational transfer functions with nonlinear parametric dependencies. We develop the proposed algorithm using tools of interval analysis [48]. The main features of the proposed algorithm are:

1. The proposed algorithm is applicable to a very general class of nonrational transfer functions and nonlinear parametric dependencies. The magnitude and phase functions need to be only bounded and continuous in the frequency and parameters, and the crossover frequencies be bounded. Subject to these assumptions, there is no restriction on the structure or form of the transfer function. Thus, the magnitude and phase functions can be described by *any* sequence of arithmetic expressions involving the frequency and parameter variables, using the built -in functions of a programming language. On the other hand, the gain and phase margin routine in MATLAB's control systems toolbox, called *allmargin*, is restricted to rational transfer functions.

2. The proposed algorithm can *readily* handle nonrational transfer functions and needs no rational approximation of any nonrational term in the transfer function. Thus, multiple time delays and transcendental terms can be handled with equal ease, without the need for any rational approximation.

3. The proposed algorithm guarantees that *all* gain and phase margins and crossover frequencies are found, see Theorem 3.11 below. This guarantee - that no margins and crossover frequencies are ever missed - meets an important requirement in stability analysis and synthesis of control systems. Existing methods do not offer any such guarantee.

4. The proposed algorithm guarantees that the gain and phase margins and crossover frequencies are computed to a *prescribed* accuracy, see Theorem 3.9 below.

5. The proposed algorithm guarantees that the computed gain and phase margins and crossover frequency values are *reliable* in face of all kinds of computational errors, see Theorem 3.12 below. In contrast, existing methods do not account for such computational errors.

6. The proposed algorithm computes all the gain and phase margins along with the crossover frequencies in a *finite* number of iterations, for a prescribed accuracy. Moreover, an upper bound on the number of algorithmic iterations required is given, see Theorem 3.10 below.

The rest of this chapter is organized as follows. In section 3.2, we present the proposed algorithm for computing robust gain margins. In section 3.3, we describe the proposed algorithm for analyzing systems with fixed parameters. In section 3.4, we study the theoretical and computational properties of the proposed algorithm. In section 3.5, we consider the problem of computing robust phase margins. In section 3.6, we demonstrate the proposed algorithm on some practical nonrational examples. Lastly, in section 3.7, we give the conclusions of this chapter.

3.2 Proposed algorithm for robust gain margins

Consider a linear system represented by the transfer function $g(s, \mathbf{q})$, where s is the Laplace variable and $\mathbf{q} \in \Re^n$ is a vector of the system parameters. Let ω denote the frequency. Define the magnitude, phase, and phase crossover functions respectively as

$$g_{mag}(\omega, \mathbf{q}) \;:=\; 20 \log_{10} |g(s = j\omega, \mathbf{q})| \;\;;\;\; g_{phase}(\omega, \mathbf{q}) := \angle g(s = j\omega, \mathbf{q}) \,\text{deg} \tag{3.1}$$

$$f(\omega, \mathbf{q}) \;:=\; 180^\circ + g_{phase}(\omega, \mathbf{q}) \tag{3.2}$$

Note that the magnitude is expressed in decibels (dB) and phase in degrees.

Suppose there is parametric uncertainty in the system such that the parameter vector $\mathbf{q}$ varies over a bounding box $\mathbf{Q}^0 \in I(\Re^n)$ given by

$$\mathbf{Q}^0 := \{\mathbf{q} \in \Re^n : \underline{q}_i \leq \mathbf{q}_i \leq \overline{q}_i, \text{ where } \underline{q}_i, \overline{q}_i \in \Re, \; i = 1, 2, ..., n\}$$

The parametric uncertainty gives rise to an uncertain linear system. Now define the set $\mathcal{P}_{cf}$ of phase crossover frequencies and the set $\mathcal{G}_m$ of gain margins as

$$\mathcal{P}_{cf} := \{\omega : f(\omega, \mathbf{q}) = \mathbf{0}, \mathbf{q} \in \mathbf{Q}^0\}; \quad \mathcal{G}_m := \{-g_{mag}(\omega, \mathbf{q}) : \mathbf{q} \in \mathbf{Q}^0, \, \omega \in \mathcal{P}_{cf}\}$$

We address the problem of reliably computing $\mathcal{P}_{cf}$ and $\mathcal{G}_m$ to a prescribed accuracy. We assume the following throughout this work.

Assumption 3.1 *The magnitude and phase functions g_{mag} and g_{phase} are bounded and continuous in ω and $\mathbf{q}$ over the domain of interest. Moreover, the crossover frequencies are bounded.*

Subject to the above assumption, the magnitude and phase functions can be described by any sequence of arithmetic expressions involving ω and $\mathbf{q}$ using +, -, *, /, $\sqrt{}$, exp, log, power, trigonometric functions, inverse trigonometric functions, etc.

In the sequel, we prefer to use the more generic term *box* even while referring to an *interval*.

3.2.1 Initial search box

In the proposed algorithm, we need to construct an initial search box $\mathbf{\Omega}^0$ that contains all phase crossover frequencies. We can construct this box as follows. The frequency ω is nonnegative, so $\mathbf{\Omega}^0 \in I(\Re^+)$. On a computer, we can set $\mathbf{\Omega}^0 \leftarrow [0, \text{real}_{\max}]^2$ where $\text{real}_{\max}$ is the largest machine representable number on the computer. We sometimes know the range in which the phase crossover frequencies occur in a particular problem. If so, we can bound $\mathbf{\Omega}^0$ to enclose this range. Finally, by Assumption 3.1, $\mathcal{P}_{cf}$ is bounded, and so we can construct an initial search box that is guaranteed to contain all crossover frequencies, by adopting a procedure of Moore [48, chapter 6].

3.2.2 Algorithm

The proposed algorithm is a binary search algorithm based on subdivision and an interval version of the well known zero exclusion test [8]. It consists of three parts: an initialization part, an iterative part, and a termination part.

In the initialization part, first, natural inclusion functions G_{mag}, G_{phase}, and F for the magnitude, phase, and phase crossover functions are constructed. Let $\mathbf{x} := (\omega, \mathbf{q})$. Then, following section 3.2.1, an initial box $\mathbf{\Omega}^0$ that encloses the set $\mathcal{P}_{cf}$ is constructed, along with the work box $\mathbf{X}^0 = \left(\mathbf{\Omega}^0, \mathbf{Q}^0\right)$. Next, the two lists that are needed in the algorithm are initialized: a working list $\mathcal{L}$ that contains boxes for processing is initialized with the work box $\mathbf{X}^0$, and the list $\mathcal{L}^{sol}$ that contains solution boxes is initialized to the empty list. Then, the algorithm branches to the iterative part.

In the iterative part, all boxes $\mathbf{X}_{(i)}$ present in the working list $\mathcal{L}$ are chosen, and a branch and bound strategy is applied to discard irrelevant boxes using the interval zero exclusion test and subdivisions.

A box is accepted as a solution box when both the domain (frequency) and range (magnitude) accuracy tolerances are satisfied for that box, see Remark 3.2. For all such boxes, the pairs $\left(\mathbf{X}_{(i)}, -G_{mag}\left(\mathbf{X}_{(i)}\right)\right)$ are deposited in the solution list $\mathcal{L}^{sol}$. The remaining boxes are subdivided along the longest direction of the box $\mathbf{X}_{(i)}$, and the resulting subboxes are put in $\mathcal{L}$. The entire iterative part is repeated till no more boxes are left in $\mathcal{L}$ for processing. Following this, the algorithm branches to the termination part.

In the termination part, the set of all phase crossover frequencies is constructed as the union of all boxes $\mathbf{\Omega}_{(i)}$ present in $\mathcal{L}^{sol}$. Likewise, the set of all gain margins is constructed as the union of all boxes $\left(-G_{mag}\left(\mathbf{X}_{(i)}\right)\right)$ present in the same list. The algorithm now exits.

The entire process is greatly speeded up by concurrently (rather than sequentially) processing all the subboxes present in a given iteration. Concurrent processing is possible through the use of vectorized interval operations for function evaluations, subdivisions, width checks, etc., see [51] for details.

We next present the proposed algorithm.

Algorithm (Computation of all robust gain margins and phase crossover frequencies)

Input: Expressions for the magnitude, phase and phase cross over functions, the uncertain parameter vector $\mathbf{Q}^0$, an accuracy tolerance ε_ω for the phase crossover frequency, and an accuracy tolerance ε_{dB} for the gain margin.

Output: The set $\mathcal{G}_m^{a\,\mathrm{lg}}$ of all gain margins along with the set $\mathcal{P}_{cf}^{a\,\mathrm{lg}}$ of all phase crossover frequencies, computed to their respective accuracy tolerances.

Note: The algorithm is to be executed in the order given below, except when otherwise indicated.

BEGIN Algorithm

1. (**Initialization part**) From the expressions for the magnitude, phase, and phase crossover functions, construct natural inclusion functions $G_{mag}, G_{phase},$ and F.

2. Construct initial boxes and lists:

   (a) Following section 3.2.1, construct an initial search box $\mathbf{\Omega}^0$ that encloses the phase crossover frequency set $\mathcal{P}_{cf}$. Next, construct the work box $\mathbf{X}^0 = \left(\mathbf{\Omega}^0, \mathbf{Q}^0\right)$.

   (b) Set $k \leftarrow 0$ and initialize lists $\mathcal{L}^{sol} \leftarrow \{\}$, $\mathcal{L} \leftarrow \{\mathbf{X}^0\}$.

3. (**Iterative part**) Start a new iteration:

   (a) Set $k \leftarrow k+1$ and $l_r \leftarrow$ length of $\mathcal{L}$.

   (b) Pick all boxes $\mathbf{X}_{(i)}$, $i = 1, 2, ..., l_r$ from $\mathcal{L}$ and delete their entries from $\mathcal{L}$.

4. Test phase using an interval zero exclusion test:

   (a) Evaluate $F\left(\mathbf{X}_{(i)}\right), i = 1, 2, ..., l_r$.

   (b) (Interval zero exclusion test): IF $0 \notin F\left(\mathbf{X}_{(i)}\right)$ THEN discard $\mathbf{X}_{(i)}$, $i = 1, 2, ..., l_r$.

   (c) If no more $\mathbf{X}_{(i)}$ remain, go to step 8.

5. Solution set:

   (a) For each remaining $\mathbf{X}_{(i)}$, evaluate $w\left(\mathbf{X}_{(i)}\right)$ and $w\left(G_{mag}\left(\mathbf{X}_{(i)}\right)\right)$. IF $w\left(\mathbf{X}_{(i)}\right) < \varepsilon_\omega$ and $w\left(G_{mag}\left(\mathbf{X}_{(i)}\right)\right) < \varepsilon_{dB}$ THEN enter the pair $\left(\mathbf{X}_{(i)}, -G_{mag}\left(\mathbf{X}_{(i)}\right)\right)$ in $\mathcal{L}^{sol}$ and discard $\mathbf{X}_{(i)}$ from further processing.

   (b) If no more $\mathbf{X}_{(i)}$ remain, go to step 8.

6. Subdivision phase:

   For each remaining box $\mathbf{X}_{(i)}$, find a coordinate direction k_i parallel to which $\mathbf{X}_{(i)}$ has an edge of greater length. Subdivide $\mathbf{X}_{(i)}$ in direction k_i getting subboxes $\mathbf{X}^1_{(i)}$ and $\mathbf{X}^2_{(i)}$ such that $\mathbf{X}_{(i)} = \mathbf{X}^1_{(i)} \cup \mathbf{X}^2_{(i)}$. Enter the subboxes $\mathbf{X}^1_{(i)} \cup \mathbf{X}^2_{(i)}$ in $\mathcal{L}$.

7. End current iteration: Return to step 3.

8. (**Termination part**) Construct the sets

$$\mathcal{P}^{a\,\mathrm{lg}}_{cf} \leftarrow \bigcup_{\mathbf{\Omega}_{(i)} \in \mathcal{L}^{sol}} \mathbf{\Omega}_{(i)}, \quad \mathcal{G}^{a\,\mathrm{lg}}_{m} \leftarrow \bigcup_{\mathbf{X}_{(i)} \in \mathcal{L}^{sol}} -G_{mag}\left(\mathbf{X}_{(i)}\right)$$

   output $\mathcal{P}^{a\,\mathrm{lg}}_{cf}$ and $\mathcal{G}^{a\,\mathrm{lg}}_{m}$, and EXIT.

END Algorithm.

Remark 3.1 *We require the continuity of the magnitude and phase functions in ω in order that the natural inclusion functions G_{mag} and G_{phase} be continuous. The latter property ensures that the widths of the intervals evaluations $G_{mag}(\mathbf{X})$ and $F(\mathbf{X})$ tend to zero as the width of $\mathbf{X}$ tends to zero, so that in turn, convergence and arbitrary accuracy can be achieved.*

Remark 3.2 *Suppose that a phase crossover frequency exists in a box $\mathbf{\Omega}$. If this box is used as an enclosure of the phase crossover frequency, then clearly, the error can be no greater than the width of the box itself. Therefore, we use the* domain *accuracy tolerance condition*

$$w(\mathbf{X}) < \varepsilon_\omega \tag{3.3}$$

where, ε_ω is a prescribed domain tolerance. Moreover, the magnitude function g_{mag} may be flat, with unknown flatness, near a phase crossover frequency. To deal with such cases, we use the range *accuracy tolerance condition on the magnitude (and hence, on the gain margin):*

$$w\left(G_{mag}(\mathbf{X})\right) < \varepsilon_{dB} \tag{3.4}$$

where, ε_{dB} is a prescribed range tolerance. A box is accepted as a solution box only when both the domain *and* range *accuracy tolerances are satisfied for that box.*

3.3 Fixed parameters

The problem of computation of gain margins and crossover frequencies for *fixed* parameter systems is a problem of interest in its own right, for instance, in classical like approaches to control system design. In this case, the parameter box $\mathbf{Q}^0$ reduces to a fixed real parameter vector $\mathbf{q}^0$, as there is no uncertainty. The algorithm for computing the gain margins for *fixed* parameter systems can then obtained from the above algorithm for uncertain parameter systems, by making the following simplifications.

- In step 2a: Replace the uncertain parameter vector $\mathbf{Q}^0$ in $\mathbf{X}^0 = \left(\mathbf{\Omega}^0, \mathbf{Q}^0\right)$ with the fixed parameter vector $\mathbf{q}^0$.
- In step 6: For fixed parameter case, the only interval quantity in the definition of box $\mathbf{X}_{(i)}$ is $\mathbf{\Omega}_{(i)}$, so subdivision needs to be done only for $\mathbf{\Omega}_{(i)}$. It is also not
- In step 2a: Replace the uncertain parameter vector $\mathbf{Q}^0$ in $\mathbf{X}^0 = \left(\mathbf{\Omega}^0, \mathbf{Q}^0\right)$ with the fixed parameter vector $\mathbf{q}^0$.
- In step 6: For fixed parameter case, the only interval quantity in the definition of box $\mathbf{X}_{(i)}$ is $\mathbf{\Omega}_{(i)}$, so subdivision needs to be done only for $\mathbf{\Omega}_{(i)}$. It is also not required to find the subdivision direction, since $\mathbf{\Omega}_{(i)}$ is unidimensional. Thus, the current statement in this step can be replaced with the following one:

  For each remaining box $\mathbf{X}_{(i)}$, subdivide $\mathbf{\Omega}_{(i)}$ to get subboxes $\mathbf{X}^1_{(i)}$ and $\mathbf{X}^2_{(i)}$ such that $\mathbf{X}_{(i)} = \mathbf{X}^1_{(i)} \cup \mathbf{X}^2_{(i)}$. Enter the subboxes $\mathbf{X}^1_{(i)} \cup \mathbf{X}^2_{(i)}$ in $\mathcal{L}$.

3.4 Properties

We next investigate the various properties of the proposed algorithm for robust gain margin. The properties of the proposed algorithm for the system with *fixed* parameters readily follow as a special case, and hence are not given here.

First, we give a result that justifies the interval zero exclusion test in step 4b in the algorithm.

Lemma 3.1 *Let $\mathbf{X} \in I\left(\mathbf{X}^0\right)$ be a subbox, where $\mathbf{X}^0$ is as in step 2a of the algorithm. If $0 \notin F(\mathbf{X})$ then $\mathbf{X}$ can be discarded in the algorithm.*

Proof. By inclusion property in Theorem I.1

$$\{f(\boldsymbol{\omega}, \mathbf{q}) : \boldsymbol{\omega} \in \mathbf{\Omega}, \mathbf{q} \in \mathbf{Q}\} =: \bar{f}(\mathbf{\Omega}, \mathbf{Q}) \subseteq F(\mathbf{\Omega}, \mathbf{Q})$$

Since, by hypothesis $0 \notin F(\mathbf{X})$, we have $0 \notin F(\mathbf{X}) \Rightarrow 0 \notin \bar{f}(\mathbf{X})$. Therefore, $\mathbf{X}$ is irrelevant in the search for phase crossover frequency points, and can be discarded. ■

The following result justifies step 8 to determine the non-existence of phase crossover frequency, in which case, the gain margin is infinite.

Lemma 3.2 *If $\mathcal{L}^{sol} = \emptyset$ in step 8 of the algorithm, then no phase crossover frequency exists.*

Proof. At any iteration, a part or the whole of a box $\mathbf{X}$ is discarded only in the interval zero exclusion test Step 4b. By Lemma 3.1, no crossover frequency points are lost in the discarding process of this step. Therefore, at any iteration, all (if any) phase crossover frequency points are either in list $\mathcal{L}$ or $\mathcal{L}^{sol}$, but are never lost. When the algorithm reaches Step 8, the list $\mathcal{L}$ is empty, so all (if any) crossover frequencies must now be in $\mathcal{L}^{sol}$. However, if $\mathcal{L}^{sol}$ is also empty at this step, then, clearly, no crossover frequencies exist. Hence, the gain margin can be set to infinity. ■

3.4.1 Convergence

To study the convergence properties of the proposed algorithm, we assume that the tolerance criterion can never be satisfied, i.e., $\varepsilon_\omega, \varepsilon_{dB} = 0$, and that list sizes are not a limitation. Further, to avoid trivial cases, we assume in this subsection that at least one crossover frequency exists in $\mathbf{\Omega}^0$.

Definition 3.1 *Let $\mathcal{L}_k$ denote the list $\mathcal{L}$ present at the start of kth iteration of the algorithm, and denote the ith box of this list as $\mathbf{X}_{ki}$. Define the unions*

$$\mathcal{U}_k = \bigcup_{\mathbf{\Omega}_{ki} \in \mathcal{L}_k} \mathbf{\Omega}_{ki}, \quad \mathcal{V}_k = \bigcup_{\mathbf{Q}_{ki} \in \mathcal{L}_k} \mathbf{Q}_{ki}, \quad \mathcal{W}_k = \bigcup_{\mathbf{X}_{ki} \in \mathcal{L}_k} \mathbf{X}_{ki} \tag{3.5}$$

Note that $\mathcal{U}_1 = \mathbf{\Omega}^0, \mathcal{V}_1 = \mathbf{Q}^0, \mathcal{W}_1 = \mathbf{X}^0$.

Lemma 3.3 *The unions $\mathcal{U}_k, \mathcal{V}_k, \mathcal{W}_k$ are compact sets at any k.*

Proof. At a given k, $\mathcal{U}_k$ is a collection of all boxes $\mathbf{\Omega}_{ki}$ present in list $\mathcal{L}_k$. Clearly, each box $\mathbf{\Omega}_{ki}$ is closed and bounded, i.e., is compact. As the finite union of compact sets is compact, $\mathcal{U}_k$ is compact. Similarly for $\mathcal{V}_k, \mathcal{W}_k$. ■

Lemma 3.4 $\mathcal{P}_{cf} \subseteq \bigcap_{k=1}^{\infty} \mathcal{U}_k$

Proof. It is sufficient to show that $\mathcal{P}_{cf} \subseteq \mathcal{U}_k$ for any k. The assertion of the lemma then follows. Consider the algorithm with $k = 1$. Since, $\mathcal{U}_1 = \mathbf{\Omega}^0$, we have $\mathcal{P}_{cf} \subseteq \mathcal{U}_1$. By Lemma 3.1, the discarding process using the interval zero exclusion test in step 4b does not delete any point in $\mathbf{\Omega}^0$ that belongs to $\mathcal{P}_{cf}$. Moreover, none of these points can be lost in the subsequent subdivision step 6, because every box is replaced by both its subboxes in the list $\mathcal{L}$. Thus, at

the end of first iteration, all the points in $\mathcal{P}_{cf}$ are retained in $\mathcal{U}_2$, i.e., $\mathcal{P}_{cf} \subseteq \mathcal{U}_2$. By induction on k, we have $\mathcal{P}_{cf} \subseteq \mathcal{U}_k$ for any $k \Rightarrow \mathcal{P}_{cf} \subseteq \bigcap_{k=1}^{\infty} \mathcal{U}_k$. ■

Lemma 3.5 *The sequence* $\{\mathcal{U}_k\}_{k=1}^{\infty}$ *has the property*

$$\mathcal{U}_1 \supseteq \mathcal{U}_2 \supseteq \mathcal{U}_3 ...$$

Proof. In the first iteration ($k = 1$) of the algorithm, the box $\mathbf{X}^0$ is picked from the list $\mathcal{L}_1$, and then is subdivided; the boxes resulting from the subdivision then replace $\mathbf{X}^0$ in $\mathcal{L}_1$, giving $\mathcal{L}_2$ (the state of the list at $k = 2$). Hence, $\mathcal{U}_2 \subseteq \mathcal{U}_1$. The proof is completed by induction on k. ■

Lemma 3.6 *Let* w_k *denote the maximum width of the boxes* $\mathbf{X}_{ik}$ *of the* k*th list* $\mathcal{L}_k$ *computed by the algorithm. Then,*

$$w_k \to 0 \text{ as } k \to \infty$$

Proof. Follows from the lemma in Ratschek [59]. ■

Lemma 3.7

$$w(F(\mathbf{X})) \to 0 \quad \text{as } w(\mathbf{X}) \to 0$$

Proof. By Theorem I.3, all natural inclusion functions have first order convergence. So, there is a constant $\alpha > 0$ independent of the box $\mathbf{X}$ such that

$$w(F(\mathbf{X})) - w(\bar{f}(\mathbf{X})) \leq \alpha \; w(\mathbf{X}) \tag{3.6}$$

or

$$w(F(\mathbf{X})) - w(\bar{f}(\mathbf{X})) \to 0 \text{ as } w(\mathbf{X}) \to 0 \tag{3.7}$$

Since, by Assumption 3.1, f is continuous in $\mathbf{x}$, $w(\bar{f}(\mathbf{X})) \to 0$ as $w(\mathbf{X}) \to 0$. Together with (3.7) this leads to the assertion of the lemma. ■

Lemma 3.8 $\bigcap_{k=1}^{\infty} \mathcal{U}_k \subseteq \mathcal{P}_{cf}$.

Proof. Let $\mathbf{x} = (\omega, \mathbf{q}) \in \mathcal{W}_k$, for all k. Note that this in turn assumes that the corresponding ω belongs to $\mathcal{U}_k$ for all k. We first show that $f(\mathbf{x}) = 0$. Now, since by hypothesis $\mathbf{x} \in \mathcal{W}_k$ for all k, it follows that for any k, an item $\mathbf{X}'_{(k)}$ must occur in the list $\mathcal{L}_k$ such that $\mathbf{x} \in \mathbf{X}'_{(k)}$. That is,

$$\mathbf{x} \in \mathbf{X}'_{(k)} \in \mathcal{L}_k, \text{ for all } k \tag{3.8}$$

By Lemma 3.6,

$$w(\mathbf{X}'_{(k)}) \to 0 \quad \text{as } k \to \infty \tag{3.9}$$

From (3.8), (3.9) and Definition I.8 it follows that the sequence $\left\{\mathbf{X}'_{(k)}\right\}_{k=1}^{\infty}$ tends to $\mathbf{x}$:

$$\mathbf{X}'_{(k)} \to \mathbf{x} \quad \text{as } k \to \infty \tag{3.10}$$

Since f is continuous by Assumption 3.1, this gives

$$\bar{f}\left(\mathbf{X}'_{(k)}\right) \to f\left(\mathbf{x}\right) \quad \text{as } k \to \infty$$

By the inclusion property in Theorem I.1,

$$f\left(\mathbf{x}\right) \in \bar{f}(\mathbf{X}'_{(k)}) \subseteq F(\mathbf{X}'_{(k)}) \quad \text{for any } k \tag{3.11}$$

Moreover, from (3.9) and Lemma 3.7

$$w\left(F(\mathbf{X}'_{(k)})\right) \to 0 \quad \text{as } k \to \infty \tag{3.12}$$

From (3.11), (3.12), and Definition I.8

$$F\left(\mathbf{X}'_{(k)}\right) \to f\left(\mathbf{x}\right) \quad \text{as } k \to \infty \tag{3.13}$$

Further, since $\mathbf{X}'_{(k)}$ is not yet discarded

$$0 \in F(\mathbf{X}'_{(k)}) \quad \text{for any } k \tag{3.14}$$

From (3.12), (3.14)

$$F\left(\mathbf{X}'_{(k)}\right) \to 0 \quad \text{as } k \to \infty \tag{3.15}$$

Comparing (3.13) and (3.15) gives

$$f(\mathbf{x}) = 0$$

or $\omega \in \mathcal{P}_{cf}$. That is, we have shown that if ω belongs to $\mathcal{U}_k$ for all k then ω belongs to the phase crossover frequency enclosure set $\mathcal{P}_{cf}$. This completes the proof. ■

The following theorem summarizes the convergence property of the proposed algorithm.

Theorem 3.9 *The following hold:*

1. *The collection of solution boxes* $\mathbf{\Omega}$ *generated in the list* $\mathcal{L}$ *of the algorithm converges to the set* $\mathcal{P}_{cf}$ *of all phase crossover frequencies.*

2. *The above convergence is such that the collection always encloses* $\mathcal{P}_{cf}$ *at any iteration.*

3. *The collection of magnitude boxes* $(-G_{mag}\left(\mathbf{X}\right))$ *generated in the list* $\mathcal{L}$ *of the algorithm converges to set* $\mathcal{G}_m$ *of all gain margins. Moreover, this convergence is such that the collection always encloses* $\mathcal{G}_m$ *at any iteration.*

Proof. of part 1: By Lemma 3.5, the unions $\mathcal{U}_k$ form a chain

$$\mathcal{U}_1 \supseteq \mathcal{U}_2 \supseteq \mathcal{U}_3 ... \tag{3.16}$$

By Lemmas 3.4 and 3.8

$$\mathcal{P}_{cf} = \bigcap_{k=1}^{\infty} \mathcal{U}_k \tag{3.17}$$

Now, (3.16) and (3.17) together imply that the unions $\mathcal{U}_k$ converge to $\mathcal{P}_{cf}$.

of part 2: follows from Lemma 3.4 which implies that $\mathcal{P}_{cf} \subseteq \mathcal{U}_k$ for any k.

of part 3: Follows from the results of parts 1, 2, and the assumed continuity of g_{mag} in ω and $\mathbf{q}$ as given by Assumption 3.1. ■

3.4.2 Termination

We show that, for a prescribed accuracy, the proposed algorithm terminates in a *finite* number of iterations. We also give an upper bound on the number of iterations required by the proposed algorithm.

Theorem 3.10 *The algorithm terminates in at most* $(n+1)\,\gamma$ *iterations, where* γ *is given by*

$$\gamma := \max\left\{\log_2\left(\frac{w(\mathbf{X}^0)}{\varepsilon_\omega}\right), \log_2\left(\frac{\alpha w(\mathbf{X}^0)}{\varepsilon_{dB}}\right)\right\} + 1$$

where, α *is a (Lipschitz) constant for the function* g_{mag} *in equation (I.1) in the Appendix. Further, the maximum total number of subdivisions is given by* $2(2^\gamma - 1)$.

Proof. First, note that as $\mathbf{X}$ is a $(n+1)-$dim box, after $(n+v)$ successive subdivisions, where v is some positive integer, we obtain $w(\mathbf{X}) \leq w(\mathbf{X}^0)/2^v$. Thus, in at most $(n+1)\,\gamma'$ successive subdivisions, where $\gamma' := \log_2\left(w(\mathbf{X}^0)/\varepsilon_\omega\right) + 1$, we obtain $w(\mathbf{X}) < \varepsilon_\omega$ and the domain (crossover frequency) accuracy tolerance is satisfied for that subbox.

Next, by Theorem I.3, all natural inclusion functions have first order convergence. So, there is a constant $\alpha > 0$ independent of the box $\mathbf{X}$ such that

$$w\left(G_{mag}(\mathbf{X})\right) - w(\bar{g}_{mag}\left(\mathbf{X}\right)) \leq \alpha w(\mathbf{X})$$

As widths of intervals are never negative, $w(\bar{g}_{mag}\left(\mathbf{X}\right)) \geq 0$. Therefore,

$$w(\mathbf{X}) < \varepsilon_{dB}/\alpha \Rightarrow w\left(G_{mag}(\mathbf{X})\right) < \varepsilon_{dB}$$

and the range (gain margin) accuracy condition is satisfied for that subbox. Proceeding as above, we find that in at most $(n+1)\,\gamma''$ successive subdivisions, where $\gamma'' := \log_2\left(\alpha \cdot w(\mathbf{X}^0)/\varepsilon_{dB}\right) + 1$, the range accuracy condition is satisfied for that subbox.

The processing of a subbox is completed when both the accuracy conditions are satisfied, and from above arguments, this is achieved in at most $(n+1)\,\gamma$ successive subdivisions, where $\gamma = \max(\gamma', \gamma'')$. The first part of the theorem is therefore proved.

To prove the second part of the theorem, note that the algorithm produces a binary tree whose nodes are the regions obtained through successive subdivisions. The root of this tree is the node corresponding to the initial region $\mathbf{X}^0$. The maximum total number of iterations is equal to the depth γ of this tree, where γ is as given above. Further, the maximum total number of subdivisions occurs for a balanced tree, and from [15] equals $2 \cdot (2^\gamma - 1)$. This completes the proof. ■

Theorem 3.11 *The following hold:*

1. $\mathcal{P}_{cf} \subseteq \mathcal{P}_{cf}^{a\,\mathrm{lg}}$
2. $\mathcal{G}_m \subseteq \mathcal{G}_m^{a\,\mathrm{lg}}$

Proof. of part 1: Let $\mathcal{L}_{k-1}^{sol}$ denote the list $\mathcal{L}^{sol}$ at the end of the $(k-1)$th iteration of the algorithm. Proceeding as in the proof of Lemma 3.4, we can show that

$$\mathcal{P}_{cf} \quad \subseteq \left(\bigcup_{\mathbf{\Omega}_i \in \mathcal{L}_{k-1}^{sol}} \mathbf{\Omega}_i \right) \bigcup \mathcal{U}_k, \qquad \text{at any iteration } k$$

By Theorem 3.10, for given $\varepsilon_\omega, \varepsilon_{dB} > 0$, the algorithm terminates at a finite iteration number, denoted here as β. Now, when the algorithm terminates at iteration $k = \beta$, the list $\mathcal{L}$ is empty. Hence,

$$\mathcal{P}_{cf} \quad \subseteq \quad \bigcup_{\mathbf{\Omega}_i \in \mathcal{L}_{\beta}^{sol}} \mathbf{\Omega}_i \quad =: \mathcal{P}_{cf}^{a\,\mathrm{lg}}$$

of part 2: Follows from the inclusion property in Theorem I.1 and the result of part 1. ■

3.4.3 Reliability

We say that a computed crossover frequency set resp. gain margin set is *reliable*, if it has been computed taking into account all kinds of computational errors, such as roundoff, truncation, and approximation errors. To account for all kinds of computational errors, machine interval arithmetic (MIA) [38] can be used. MIA can provide mathematically rigorous results from floating point operations on computers, see [48].

The below theorem shows the reliability of the results computed by a MIA implementation of the proposed algorithm.

Theorem 3.12 (Reliability) *Let $\mathcal{P}_{cf,mia}^{a\,\mathrm{lg}}$ and $\mathcal{G}_{m,mia}^{a\,\mathrm{lg}}$ denote the set of all crossover frequencies and the set of all gain margins as computed by a MIA implementation of the algorithm. Then,*

$$\mathcal{P}_{cf} \subseteq \mathcal{P}_{cf}^{a\,\mathrm{lg}} \subseteq \mathcal{P}_{cf,mia}^{a\,\mathrm{lg}}; \quad \mathcal{G}_m \subseteq \mathcal{G}_m^{a\,\mathrm{lg}} \subseteq \mathcal{G}_{m,mia}^{a\,\mathrm{lg}}$$

Proof. Follows from Theorem 3.11 and the inclusion property of MIA given in [38]. ■

3.5 Computing phase margins

The set $\mathcal{G}_{cf}$ of gain crossover frequencies and set $\mathcal{P}_m$ of phase margins over $\mathbf{Q}$ are defined as

$$\mathcal{G}_{cf} := \{\omega : g_{mag}(\omega, \mathbf{q}) = 0 \text{ dB}, \mathbf{q} \in \mathbf{Q}\}; \quad \mathcal{P}_m := \{f(\omega, \mathbf{q}) : \omega \in \mathcal{G}_{cf}, , \mathbf{q} \in \mathbf{Q}\}$$

To reliably compute $\mathcal{G}_{cf}$ and $\mathcal{P}_m$ to a prescribed accuracy, we can use the above proposed algorithm with obvious changes. Then, all the results of the previous section carry over to this case. The details are omitted.

3.6 Illustrative examples

First, in section 3.6.1 we demonstrate the proposed algorithm on two nonrational examples with fixed parameters. Then, in section 3.6.2 we demonstrate the proposed algorithm to compute the robust margins on a nonrational example with nonlinear parametric dependencies. Note that existing techniques are not readily applicable to compute the margins and crossover frequencies in these nonrational examples, unless some rational approximation is used.

3.6.1 Fixed parameters

Example 3.1 *Consider a multimodal plant typically found in nuclear reactor control systems [14]:*

$$g(s,\mathbf{q}) = \frac{q_1}{\left(1+q_2 s - \frac{e^{-q_5 s}}{1+q_3 s}\right)(1+q_4 s)^2}$$

where, $q_1 = 16.11, q_2 = 0.24, q_3 = 1.2,\ q_4 = 5, q_5 = 9.3$. *Suppose a proportional controller with a gain of* 6 *dB is used in a negative unity feedback structure to control this plant. We wish to compute (all) the gain margins, phase margins, and crossover frequencies to a prescribed accuracy of* 0.01.

We use a PC Pentium III 650 MHz 256 MB RAM machine for all computations.

Conventional gridding approach: We first compute the required quantities using the widely used control systems toolbox of MATLAB [46], [27]. A routine *allmargin* is available in this toolbox for computing all margins, crossover frequencies, etc. However, the *allmargin* routine can deal only with rational transfer functions. For nonrational transfer functions, a good rational approximation has to be found and supplied to this routine.

First order Pade approximation: We approximate the time delay term with the well known first order Pade approximation [16], and apply the *allmargin* routine to the resulting rational transfer function. The obtained results are given in column 3 of Table 3.1. The *allmargin* routine finds one gain and one phase crossover frequency, and the gain margin is found to be $\sim$ 13 dB while the phase margin is $\sim$ 26 deg . The conclusion based on these results is that the feedback system is *stable* for the given proportional controller gain of 6 dB, with adequate stability margins.

Second order Pade approximation: We next use a second order Pade approximation for the time delay term, and apply the *allmargin*routine. The obtained results are given in column 4 of Table 3.1. The *allmargin* routine finds again one gain and one phase crossover frequency, and

the gain margin is found to be ~ -6.8 dB while the phase margin is ~ -23 deg. The conclusion based on these results is that the feedback system is *unstable* for the given proportional controller gain of 6 dB.

Proposed algorithm: We next implement a machine interval arithmetic version of the proposed algorithm using the interval arithmetic toolbox INTLAB [61] of MATLAB. Executing the algorithm, we obtain the initial search box for crossover frequencies as $\Omega^0 = [0.01, 10]$ rads/sec, and all margins and crossover frequencies are obtained in 36 iterations and 1 second. The results of the proposed algorithm are shown in the last column of Table 3.1.

The proposed algorithm finds 5 phase crossover frequencies and 3 gain crossover frequencies, along with the corresponding gain and phase margins. Some of these gain margins are negative, with the worst case one being about -22 dB. Likewise, some phase margins are also negative, with the worst case one being about -29 deg. A gain decrease of ~ 0.58 dB or a gain increase of ~ 5.39 dB leads to system instability. The conclusion based on these results is that the feedback system is *conditionally stable.*

The controller gain needs to be adjusted to get absolute stability with satisfactory margins. For instance, a decrease of the controller gain by about 29.4 dB gives absolute stability, with satisfactory gain and phase margins of 5 dB and 60 deg.

Summary: The findings of the proposed algorithm are verified through intense gridding and careful analysis of the magnitude and phase plots around the obtained crossover frequencies, and found to be correct. That is, the closed loop system is actually conditionally stable, in contrast to the findings of the *allmargin* routine based on first and second order Pade approximations to the time delay term.

Example 3.2 *The heat exchanger system [56]:*

$$g(s, \mathbf{q}) = \frac{q_1(1 - q_2 e^{-q_3 s})}{(q_4 s + 1)(q_5 s + 1)}$$

where, $q_1 = 1$, $q_2 = 0.5, q_3 = 10, q_4 = 40$ *and* $q_5 = 15$. *Suppose a proportional controller with a gain of* 74 *dB is used in a negative unity feedback structure to control this plant. We wish to compute (all) the gain margins, phase margins, and crossover frequencies to a prescribed accuracy of* 0.01.

Conventional gridding approach: We use first and second order Pade approximations to the time delay term, and apply the *allmargin* routine to the resulting transfer functions.

First order Pade approximation: With a first order Pade approximation, the *allmargin* routine finds one gain and no phase crossover frequency, and the gain margin is found to be ∞ dB while the phase margin is about 4 deg. The conclusion is that the feedback system is *stable* for the given proportional controller gain of 74 dB, but with a poor phase margin.

Second order Pade approximation: With a second order Pade approximation, the *allmargin* routine finds one gain crossover and one phase crossover frequency. The gain margin is about

-35 dB while the phase margin is about -21 deg. The conclusion is that the feedback system is *unstable* for the given proportional controller gain.

Proposed algorithm: We next execute the proposed algorithm. The proposed algorithm takes 32 iterations and 0.5 seconds to compute the required quantities to the prescribed accuracy. It finds 5 gain crossover frequencies in the range $[2.4, 3.5]$ rads/sec. Some of the phase margins are negative, with the worst case one being about -26 deg. It also finds 31 phase crossover frequencies in the range $[0.4, 9.74]$ rads/sec. Some gain margins are also negative, with the worst case one being about -37 dB. Clearly, the feedback system is *conditionally stable.*

Summary: As before, we verify the findings of the proposed algorithm through intense gridding and careful analysis of the magnitude and phase plots around the obtained crossover frequencies, and found them to be correct. That is, the closed loop system is actually *conditionally stable*, in contrast to the findings of the *allmargin* routine based on first and second order Pade approximations to the delay term.

3.6.2 Uncertain parameters

Example 3.3 *Consider the same multimodal plant given in Example 3.1*

$$g(s, \mathbf{q}) = \frac{q_1}{\left(1 + q_2 s - \frac{e^{-q_5 s}}{1+q_3 s}\right)(1 + q_4 s)^2}$$

but this time with parametric uncertainty

$$q_1 = 16.11,\ q_2 \in [0.22, 0.26],\ q_3 \in [1.18, 1.22], q_4 = 5,\ q_5 \in [9.2, 9.4]$$

Suppose a proportional controller with a gain of 6 *dB is used in a negative unity feedback structure to control this plant. We wish to compute (all) gain margins, phase margins, gain crossover frequencies, and phase crossover frequencies over the given uncertainty range of the plant parameters.*

Note that existing techniques are not readily applicable to compute the robust margins and crossover frequencies for this example, due to its *nonrational* nature.

Conventional gridding approach: We first generate the family of fixed transfer functions by gridding the parameter box $\mathbf{Q}$ using a grid of $(8 \times 8 \times 8)$ points. We then apply the *allmargin* routine in [27] to each transfer function in this family. However, the *allmargin* routine is applicable only to rational transfer functions. To handle nonrational transfer functions, some rational approximation is to be found and supplied to the routine.

First order Pade approximation: We approximate the time delay term with the well known first order Pade approximation [16], and apply the *allmargin* routine to the resulting rational transfer function. The obtained results are given in column 3 of Table 3.2. The *allmargin*routine finds one gain and one phase crossover frequency for each member transfer

function, and the robust gain margin is found to be $\sim [12, 14]$ dB while the robust phase margin is $\sim [26, 27]$ deg . The conclusion based on these results is that the feedback system is *robustly stable* for the given proportional controller gain of 6 dB, with adequate stability margins over the given uncertainty range. See also the Nichols plot in Fig. 3.1.

Second order Pade approximation: We next use a second order Pade approximation for the time delay term, and apply the *allmargin* routine. The obtained results are given in column 4 of Table 3.2. The *allmargin*routine finds again one gain and one phase crossover frequency for each member transfer function, and the robust gain margin is found to be $\sim [-7, -6]$ dB, while the robust phase margin is $\sim [-25, -21]$ deg. The conclusion based on these results is that the uncertain feedback system is *unstable* for the given proportional controller gain of 6 dB over the given uncertainty range. See also the Nichols plot in Fig. 3.2.

Proposed algorithm: We next implement a machine interval arithmetic version of the proposed algorithm using the interval arithmetic toolbox INTLAB [61] of MATLAB. The prescribed accuracy tolerances are chosen as 0.001. Executing the algorithm, we obtain the initial search box for crossover frequencies as $\mathbf{\Omega}^0 = [0.1, 10]$ rads/sec, and all robust margins and crossover frequencies are computed in 48 iterations and 3 minutes. The results of the proposed algorithm are shown in the last column of Table 3.2.

The proposed algorithm finds 5 intervals of phase crossover frequencies and 3 intervals of gain crossover frequencies, along with the corresponding intervals of gain and phase margins. Some of these gain margins are negative, with the worst case one being about -27 dB. Likewise, some phase margins are also negative, with the worst case one being about -31 deg . We conclude that the uncertain feedback system is *conditionally* stable.

Summary: The results of the proposed algorithm are verified through intense gridding and careful analysis of the Nichols plots around the obtained crossover frequencies shown in Fig. 3.3 and found to be correct. That is, the closed loop system actually shows multiple stability margin bands. In contrast, with the conventional gridding approach based on the Pade approximation and the *allmargin* routine, the computed results are considerably erroneous, and lead to *incorrect* conclusions about the system stability.

3.7 Conclusions

We have presented an algorithm to compute the robust margins and crossover frequencies for a very general class of transfer functions and parametric dependencies, where the magnitude and phase functions need to be only bounded and continuous in frequency and parameters, and the crossover frequencies be bounded. We emphasize the main features of the proposed algorithm: its provision of *guarantees* that the computed robust margins and crossover frequencies are reliable and accurate, that *all* margins and crossover frequencies are found, and that, for a prescribed accuracy, all the margins and crossover frequencies are computed in a

finite number of iterations. The merits of the proposed algorithm over existing methods have been shown through several practical examples.

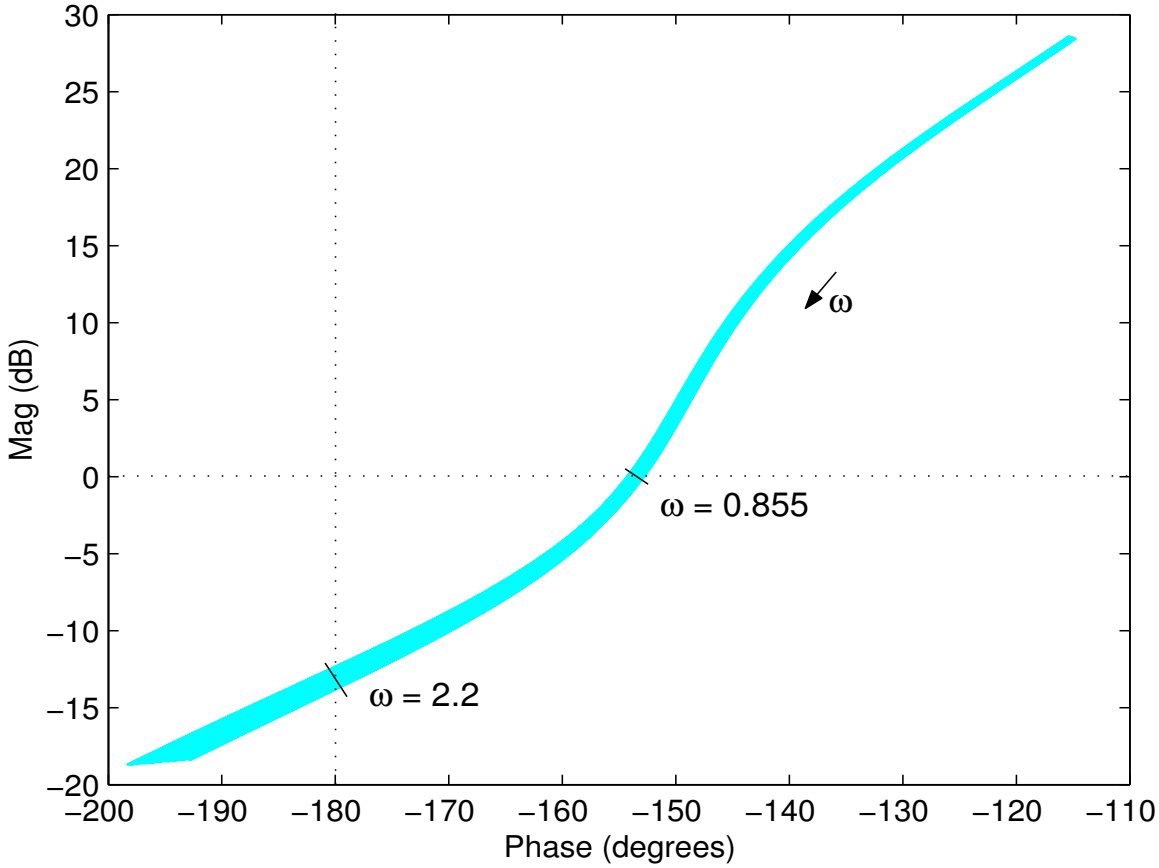

FIGURE 3.1. Nichols plot of the frequency response of the uncertain nuclear reactor system in Example 3.3. The frequency responses of the uncertain system are obtained using Pade approximation of first order. A grid of $(8 \times 8 \times 8)$ is set up for the parameter box $\mathbf{Q}$.

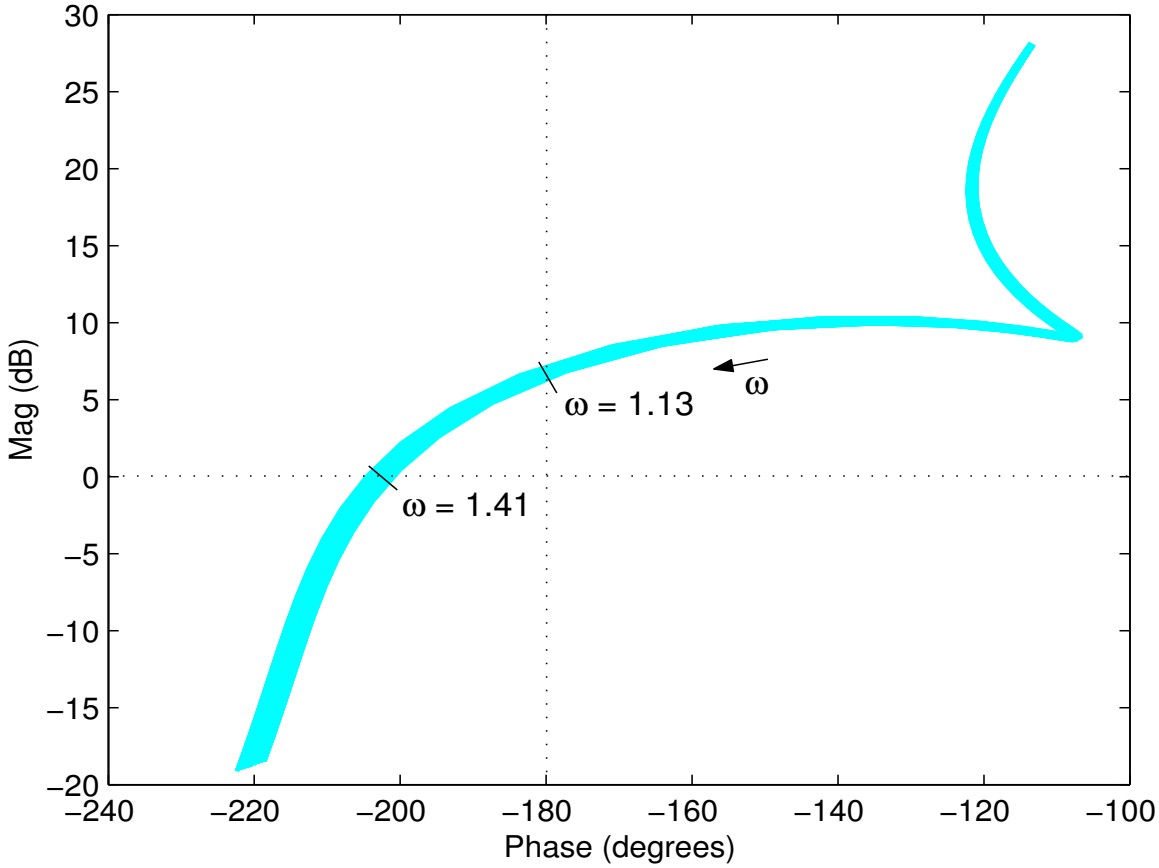

FIGURE 3.2. Nichols plot of the frequency response of the uncertain nuclear reactor system in Example 3.3. The frequency responses of the uncertain system are obtained using Pade approximation of second order. A grid of $(8 \times 8 \times 8)$ is set up for the parameter box $\mathbf{Q}$.

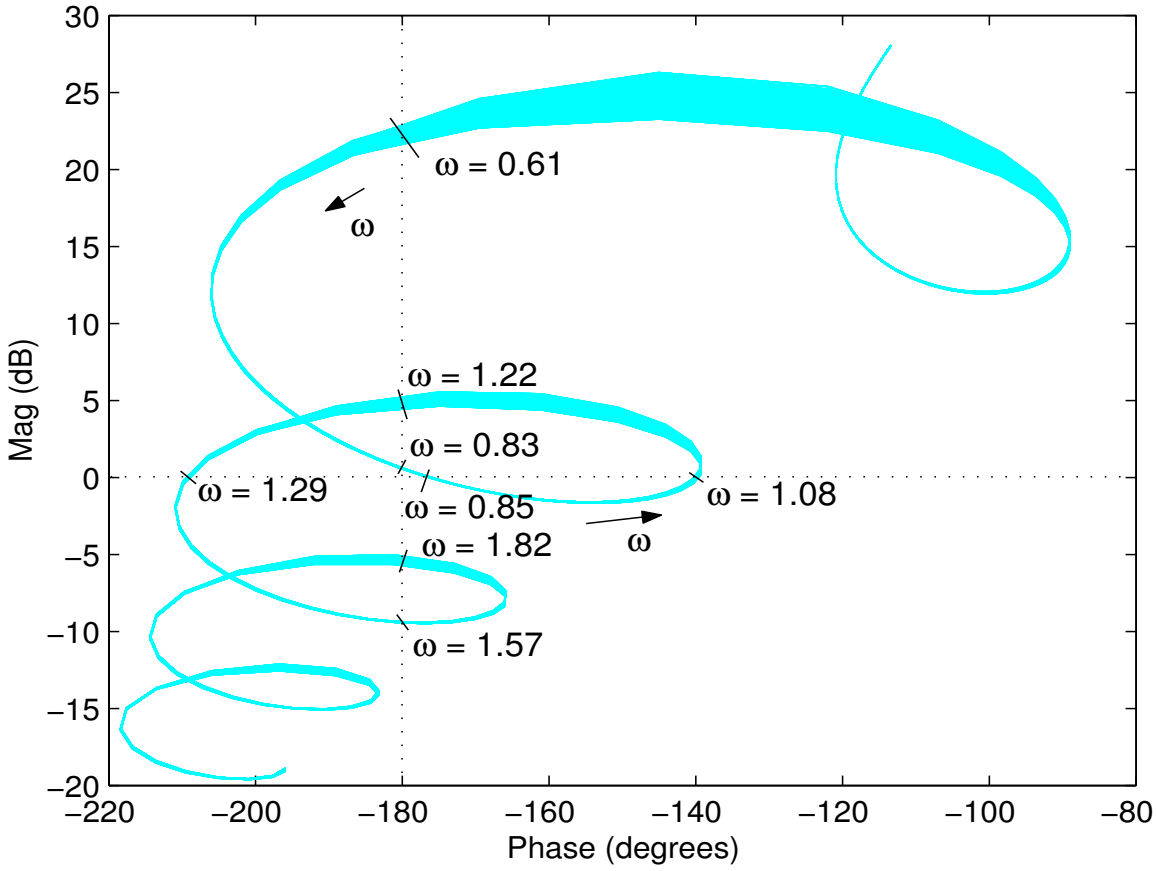

FIGURE 3.3. Nichols plots for the uncertain nuclear reactor system in Example 3.3, obtained using intense gridding of the parameter space.

TABLE 3.1. Performance comparison of the proposed algorithm with that of *allmargin* routine of MATLAB in Example 3.1.

No.	Item	*allmargin* routine of MATLAB		Proposed Algorithm
		Using First Order Pade Approx.	Using Second order Pade Approx.	
1.	$\mathcal{P}_{cf}$	–	–	$[0.6074, 0.6074]$
		–	–	$[0.8354, 0.8355]$
		–	1.1317	$[1.2198, 1.2198]$
		–	–	$[1.5719, 1.5720]$
		2.1781	–	$[1.8234, 1.8234]$
2.	$\mathcal{G}_m$	–	–	$[-22.4225, -22.4184]$
		–	–	$[-0.5829, -0.5813]$
		–	-6.79	$[-4.9237, -4.9208]$
		–	–	$[9.4110, 9.4118]$
		13.05	–	$[5.3924, 5.3943]$
3.	$\mathcal{G}_{cf}$	0.8569	–	$[0.8546, 0.8549]$
		–	–	$[1.0834, 1.0836]$
		–	1.4092	$[1.2931, 1.2932]$
4.	$\mathcal{P}_m$	26.4611	–	$[3.7046, 3.7866]$
		–	–	$[40.1203, 40.1680]$
		–	-22.8089	$[-29.3510, -29.3286]$

TABLE 3.2. Performance comparison of the proposed algorithm with that of *allmargin* routine of MATLAB in Example 3.3.

No.	Item	MATLAB's *allmargin* routine		Proposed Algorithm
		First order Pade approx.	Second order Pade approx.	
1.	$\mathcal{P}_{cf}$	–	–	$[0.6003, 0.6143]$
		–	–	$[0.8250, 0.8463]$
		–	$[1.1046, 1.1608]$	$[1.2075, 1.2324]$
		–	–	$[1.5526, 1.5919]$
		$[2.0767, 2.2931]$	-	$[1.8051, 1.8421]$
2.	$\mathcal{G}_m$	–	–	$[-26.7275, -20.1984]$
		–	–	$[-0.8953, -0.2647]$
		–	$[-7.2973, -6.2670]$	$[-6.1979, -3.8678]$
		–	–	$[9.1494, 9.6747]$
		$[12.3338, 13.8343]$	-	$[4.6213, 6.0769]$
3.	$\mathcal{G}_{cf}$	$[0.8597, 0.8641]$	–	$[0.8498, 0.8596]$
		–	–	$[1.0607, 1.1067]$
		–	$[1.3915, 1.4275]$	$[1.2794, 1.3073]$
4.	$\mathcal{P}_m$	$[25.6709, 27.2518]$	–	$[1.7230, 5.8824]$
		–	–	$[38.7525, 41.6816]$
		–	$[-24.8456, -20.6741]$	$[-30.9688, -27.2496]$

4

Spectral sets - polytopic case

4.1 Introduction

Consider a polytopic family of real nth order polynomials $\left\{p(z,\mathbf{q}), \mathbf{q} \in \mathbf{Q}^0\right\}$ where, $z \in \mathcal{C}$ is the complex variable, $\mathbf{q} \in \Re^n$ is a vector of system parameters occurring affine linearly in the coefficients of polynomial p, and $\mathbf{Q}^0 \subseteq \Re^n$ is a bounding box for $\mathbf{q}$ given as

$$\mathbf{Q}^0 := \{\mathbf{q} \in \Re^n : \underline{q}_i \leq \mathbf{q}_i \leq \overline{q}_i, \text{ where } \underline{q}_i, \overline{q}_i \in \Re, \ i = 1, 2, ..., n\}$$

For this polytope of polynomials, we address the problem of computing the spectral set defined as

$$\mathcal{S} := \{z \in \mathcal{C} : p(z,\mathbf{q}) = 0, \text{for some } \mathbf{q} \in \mathbf{Q}^0\}$$

The spectral set is of considerable interest, for example, in robustness analysis of control systems having real parametric uncertainty, see [8]. As mentioned in section 1.1.3, Barmish and Tempo [4] and Cerone [12] have recently proposed techniques to compute $\mathcal{S}$ for a polytope of polynomials. A feature of these techniques is that they involve only a $2-$dim gridding of a bounded subset of $\mathcal{C}$ rather than a $n-$dim gridding of the parameter box $\mathbf{Q}^0$. The feature is attractive as it circumvents a potential combinatoric explosion in computations with an increase in the number of parameters.

In this chapter, we present a novel algorithm to compute the spectral set $\mathcal{S}$ of a polytope of polynomials. We develop the proposed algorithm using tools of interval analysis [48]. The proposed algorithm retains the attractive feature of requiring subdivision of only a $2-$dim bounded subset of $\mathcal{C}$ instead of the $n-$dim box $\mathbf{Q}^0$. In addition, it has several important advantages over existing techniques:

1. An important requirement from a robust stability analysis and synthesis viewpoint is that no actual points should be absent in the computed spectral set. The existing

techniques are based on gridding, and a well known drawback of all gridding based techniques is that they compute underbounds of the actual sets, because of the very nature of the grid process. Therefore, with existing methods potentially critical points could be absent in the computed spectral sets.

On the other hand, the proposed algorithm guarantees that the computed spectral set always *encloses* the actual set, and moreover, this is the case irrespective of the accuracy prescribed, see Theorem 4.15 below.

2. Existing techniques lack the ability to compute spectral sets to a prescribed accuracy. In particular,

   (a) It is *a priori* unknown how fine the user selected grid must be, in order to achieve a prescribed accuracy, and

   (b) It is *a posteriori* unknown how accurate is the computed set, for a selected fineness of the grid.

   In contrast, the proposed algorithm can compute the spectral set to arbitrary accuracy, see Theorem 4.12 below.

3. The computational complexity of the existing techniques is $O\left(n^2\right)$, whereas with the proposed algorithm it is of $O\left(n\right)$, see Theorem 4.16 below.

4. Existing techniques do not provide any guarantee on the reliability (i.e., trustworthiness) of the computed results in the face of various computational errors. The proposed algorithm, when implemented on an interval arithmetic compiler, computes spectral sets that are *reliable* despite all kinds of computational errors, see Theorem 4.17 below.

5. For a given accuracy, the proposed algorithm computes the spectral set in a *finite* number of iterations. Moreover, an upper bound on the number of iterations required by the proposed algorithm can be given, see Theorem 4.14 below.

The rest of this chapter is organized as follows. In section 4.2, we present the proposed algorithm for computing the spectral set of a polytope of polynomials. In section 4.3, we study the theoretical and computational properties of the proposed algorithm. In section 4.4, we present a few illustrative examples taken from the literature. Lastly in section 4.5, we give the conclusions of the chapter.

4.2 Proposed algorithm

4.2.1 Initial search box

Several results are available in the literature to find an initial region that contains all roots of a given polynomial. The result of Henrici in [29] states that all roots of a polynomial $p(z, \mathbf{q})$ are contained in the disk with center at the origin and radius

$$r_h^{'} = 2 \max_{1 \leq k \leq n} \left| \frac{\lambda_{n-k}}{\lambda_n} \right|^{\frac{1}{k}}$$

where, $\boldsymbol{\lambda}$ is the real vector of polynomial coefficients. It readily follows from Theorem I.1 that all the roots of the family are contained in the disk with center at the origin and radius

$$r_h = 2 \max_{1 \leq k \leq n} \left| \frac{\boldsymbol{\Lambda}_{n-k}}{\boldsymbol{\Lambda}_n} \right|^{\frac{1}{k}}$$

where, $\boldsymbol{\Lambda}$ is the corresponding vector of interval coefficients. The spectral set $\mathcal{S}$ is therefore enclosed by the above described disk, which we call as the *interval Henrici* disk. For the purpose of interval computations, however, it is convenient to choose a box like initial region. An initial box $\mathbf{Z}^0$ which encloses the interval Henrici disk is clearly

$$\mathbf{Z}^0 := ([-r_h, r_h], [-r_h, r_h])$$

Further, since the coefficients of p are assumed to be real, any complex roots of p must occur as conjugate pairs. We may therefore restrict the initial search box to enclose the upper half of interval Henrici disk, so that

$$\mathbf{Z}^0 = ([-r_h, r_h], [0, r_h])$$

We shall assume in the sequel that the proposed algorithm is applied to compute the spectral set with this restricted $\mathbf{Z}^0$, and that the rest of the spectral set containing any roots lying in the lower half of the Henrici disk is subsequently obtained using complex conjugacy.

4.2.2 Algorithm

In the proposed algorithm, the given polynomial p is represented in the Horner's form, see, for instance, [15]. The evaluation of the corresponding natural inclusion function P is done using complex interval arithmetic. For a thorough treatment of complex interval arithmetic, see [1].

The proposed algorithm is essentially a binary search algorithm based on a geometrical subdivision process and a zero exclusion test. Starting with an initial box $\mathbf{Z}^0$ that is guaranteed to enclose the actual spectral set $\mathcal{S}$, the proposed algorithm produces subboxes through successive subdivisions of $\mathbf{Z}^0$, such that the subboxes are disjoint, each point of $\mathcal{S}$ is in one of the subboxes, and no point of $\mathcal{S}$ is in more than one subbox.

The search process in the proposed algorithm is speeded up by *concurrently* processing all the subboxes present in the given iteration. Concurrent processing is possible through the use of vectorized interval operations for function evaluations, subdivisions, width checks, etc., see [51] for details.

We next present the algorithm.

Algorithm (Spectral Set Computation Algorithm)

Input: The Horner's form expression for the polynomial $p(z, \mathbf{q})$, the parameter box $\mathbf{Q}^0$, and accuracy tolerances $\varepsilon_z, \varepsilon_p$ for the domain and range (see below), respectively.

Output: The computed spectral set $\mathcal{S}^{a\,\mathrm{lg}}$.

Note: The algorithm is to be executed in the order given below, except when otherwise indicated.

BEGIN Algorithm

1. From the Horner's form expression for $p(z, \mathbf{q})$, find the natural inclusion function $P(\mathbf{Z}, \mathbf{Q}^0)$.

2. (Initialization phase)

   (a) Following section 4.2.1 construct an initial search box $\mathbf{Z}^0$ that encloses the spectral set $\mathcal{S}$.

   (b) Set $k \leftarrow 0$ and initialize lists $\mathcal{L}^{sol} \leftarrow \{\}$, $\mathcal{L} \leftarrow \{\mathbf{Z}^0\}$.

3. (Start a new iteration)

   (a) Set $k \leftarrow k + 1$ and $l_r \leftarrow$ length of $\mathcal{L}$.

   (b) Pick all boxes $\mathbf{Z}_{(i)}$, $i = 1, 2, ..., l_r$ from $\mathcal{L}$ and delete their entries from $\mathcal{L}$.

4. (Test phase)

   (a) Using complex interval arithmetic, evaluate $P(\mathbf{Z}_{(i)}, \mathbf{Q}^0), i = 1, 2, ..., l_r$.

   (b) IF $0 \notin P(\mathbf{Z}_{(i)}, \mathbf{Q}^0)$ THEN discard $\mathbf{Z}_{(i)}$, $i = 1, 2, ..., l_r$.

   (c) If no more $\mathbf{Z}_{(i)}$ remain, go to step 8.

5. (Solution set)

   (a) For each remaining box $\mathbf{Z}_{(i)}$, evaluate

$$e_{z,i} := w(\mathbf{Z}_{(i)}) \tag{4.1}$$

$$e_{p,i} := w(P(\mathbf{Z}_{(i)}, \mathbf{Q}^0)) - w\left(P\left(\hat{z}, \mathbf{Q}^0\right)\right), \quad \hat{z} \in \mathbf{Z}_{(i)} \tag{4.2}$$

   IF $(e_{p,i} < \varepsilon_p)$ and $(e_{z,i} < \varepsilon_z)$ THEN enter $\mathbf{Z}_{(i)}$ in $\mathcal{L}^{sol}$ and discard $\mathbf{Z}_{(i)}$ from further processing.

(b) If no more $\mathbf{Z}_{(i)}$ remain, go to step 8.

6. (Subdivision phase)

   For each remaining box $\mathbf{Z}_{(i)}$, find a coordinate direction k_i parallel to which $\mathbf{Z}_{(i)}$ has an edge of greater length. Subdivide $\mathbf{Z}_{(i)}$ in direction k_i getting subboxes $\mathbf{Z}^1_{(i)}$ and $\mathbf{Z}^2_{(i)}$ such that $\mathbf{Z}_{(i)} = \mathbf{Z}^1_{(i)} \cup \mathbf{Z}^2_{(i)}$. Enter the subboxes $\mathbf{Z}^1_{(i)} \cup \mathbf{Z}^2_{(i)}$ in $\mathcal{L}$.

7. (End current iteration) Return to step 3.

8. Construct $\mathcal{S}^{a\,\mathrm{lg}} \leftarrow \bigcup_{\mathbf{Z}_{(i)} \in \mathcal{L}^{sol}} \mathbf{Z}_{(i)}$, output $\mathcal{S}^{a\,\mathrm{lg}}$ and EXIT.

END Algorithm.

We next explain the termination conditions used in the proposed algorithm.

4.2.3 Termination conditions

The processing of the subbox is terminated in step 5 when both the *domain* and *range* accuracy tolerances are satisfied for that subbox. To compute a root of $p(z, \mathbf{q}) = 0$, for some $\mathbf{q} \in \mathbf{Q}^0$, we use a domain accuracy tolerance of the form

$$w(\mathbf{Z}) < \varepsilon_z \tag{4.3}$$

where, ε_z is a prescribed domain tolerance. Moreover, $p(z, \mathbf{q})$ may be flat (with unknown flatness) near a root, for some $\mathbf{q} \in \mathbf{Q}^0$. For such cases, an accuracy tolerance $\|p(z, \mathbf{q})\| \leq \varepsilon_p$ on the range is appropriate, where ε_p is a prescribed range tolerance. In the interval case, the corresponding range accuracy tolerance condition is

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) < \varepsilon_p \tag{4.4}$$

However, as shown in Lemma 4.13 below, a more effective range tolerance condition than (4.4) is

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(P(\hat{z}, \mathbf{Q}^0)) < \varepsilon_p, \quad \hat{z} \in \mathbf{Z} \tag{4.5}$$

Condition (4.5) is more effective as it avoids some unnecessary subdivisions resulting from (4.4).

4.3 Properties

We next investigate the various properties of the proposed algorithm. First, we give a result that justifies step 4b in the algorithm.

Lemma 4.1 *Let $\mathbf{Z} \in I\left(\mathbf{Z}^0\right)$ be a subbox, where $\mathbf{Z}^0$ is as in step 2a of the algorithm. If $0 \notin P(\mathbf{Z}, \mathbf{Q}^0)$ then $\mathbf{Z}$ can be discarded in the algorithm.*

Proof. By hypothesis $0 \notin P(\mathbf{Z}, \mathbf{Q}^0)$. By inclusion property in Theorem I.1

$$\{p(z, \mathbf{q}) : z \in \mathbf{Z}, \mathbf{q} \in \mathbf{Q}^0\} =: \bar{p}(\mathbf{Z}, \mathbf{Q}^0) \subseteq P(\mathbf{Z}, \mathbf{Q}^0)$$

Therefore, $0 \notin P(\mathbf{Z}, \mathbf{Q}^0) \Rightarrow 0 \notin \bar{p}(\mathbf{Z}, \mathbf{Q}^0) \Rightarrow \mathbf{Z} \cap \mathcal{S} = \emptyset$. Since $\mathbf{Z}$ does not contain any points of $\mathcal{S}$, it can be discarded. ■

Lemma 4.2 *Let $z \in \mathbf{Z}$. Then, $P(z, \mathbf{Q}^0) = \bar{p}(z, \mathbf{Q}^0)$.*

Proof. In the expression for $p(z, \mathbf{q})$, the variable $\mathbf{q}$ occurs affine linearly. That is, each variable $\mathbf{q}_i$ occurs only once in the polynomial expression. Therefore, by a result of Moore [48], the interval evaluation of $P(z, \mathbf{Q}^0)$ at any point z gives the range of $p(z, \mathbf{q})$ at point z as $\mathbf{q}$ varies over $\mathbf{Q}^0$. That is, $P(z, \mathbf{Q}^0) = \bar{p}(z, \mathbf{Q}^0)$. ■

Lemma 4.3 *Let $z \in \mathbf{Z}$. If $0 \in P(z, \mathbf{Q}^0)$ then $z \in \mathcal{S}$.*

Proof. By Lemma 4.2, $P(z, \mathbf{Q}^0) = \bar{p}(z, \mathbf{Q}^0)$. Therefore, $0 \in P(z, \mathbf{Q}^0) \Rightarrow 0 \in \bar{p}(z, \mathbf{Q}^0) \Rightarrow z \in \mathcal{S}$. ■

4.3.1 Convergence

To study the convergence properties of the proposed algorithm, we assume that the tolerance criterion can never be satisfied (i.e., $\varepsilon_p, \varepsilon_z = 0$). We also assume that list sizes are not a limitation.

Definition 4.1 *Let $\mathcal{L}_k$ denote the list $\mathcal{L}$ present at the start of kth iteration of the algorithm, and denote the ith box of this list as $\mathbf{Z}_{ki}$. Define the union*

$$\mathcal{U}_k = \bigcup_{\mathbf{Z}_{ki} \in \mathcal{L}_k} \mathbf{Z}_{ki} \tag{4.6}$$

Note that $\mathcal{U}_1 = \mathbf{Z}^0$.

Lemma 4.4 *Let $\mathcal{U}_k$ be as in (4.6). Then,*

$$\mathcal{S} \subseteq \bigcap_{k=1}^{\infty} \mathcal{U}_k$$

Proof. Similar to the proof of Lemma 3.4. ■

Lemma 4.5 *$\mathcal{S}$ is a compact set in $\mathcal{C}$.*

Proof. The box $\mathbf{Q}^0$ is closed and bounded, i.e., it is compact. The roots of p are continuous on $\mathbf{Q}^0$. As the image of a continuous mapping on a compact set is compact, so the root set is compact, i.e. $\mathcal{S}$ is compact. ■

Lemma 4.6 *The union $\mathcal{U}_k$ is a non-empty compact set at any k. Further, $\bigcap_{k=1}^{\infty} \mathcal{U}_k$ is a non-empty set.*

Proof. The first part of the Lemma follows from lemma 3.3. We prove the second part. From the proof of Lemma 4.4, $\mathcal{U}_k$ is non-empty. Further, since $\mathcal{S}$ is non-empty, by Lemma 4.4, $\bigcap_{k=1}^{\infty} \mathcal{U}_k$ is non-empty. ■

Lemma 4.7 *The sequence $\{\mathcal{U}_k\}_{k=1}^{\infty}$ has the property*

$$\mathcal{U}_1 \supseteq \mathcal{U}_2 \supseteq \mathcal{U}_3 ...$$

Proof. In the first iteration ($k = 1$) of the algorithm, the box $\mathbf{Z}^0$ (which is the same as $\mathcal{U}_1$) is picked from the list $\mathcal{L}_1$, checked for the zero enclosure and widths, and is subdivided into two subboxes. The boxes resulting from the subdivision then replace $\mathbf{Z}^0$ in $\mathcal{L}_1$, giving $\mathcal{L}_2$ (the state of the list at $k = 2$). Hence, $\mathcal{U}_2 = \mathcal{U}_1$. In the kth iteration of the algorithm, the boxes $\mathbf{Z}_{ki}$ (whose union is $\mathcal{U}_k$) are picked from $\mathcal{L}_k$, (some) irrelevant boxes discarded in Step 4b, and the remaining are subdivided further in Step 6. The subboxes resulting from the subdivision process then form the union $\mathcal{U}_{k+1}$. Hence, $\mathcal{U}_{k+1} \subseteq \mathcal{U}_k$. The proof is now completed by induction on k. ■

Lemma 4.8 *Let w_k denote the maximum width of the boxes $\mathbf{Z}_{ik}$ of the kth list $\mathcal{L}_k$ generated by the algorithm. Then,*

$$w_k \to 0 \text{ as } k \to \infty$$

Proof. As in Lemma 3.6, the proof follows from the Lemma of Ratschek [59]. ■

Lemma 4.9 *Let $z \in \mathbf{Z}$. Then,*

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(\bar{p}(\mathbf{Z}, \mathbf{Q}^0)) \to 0, \qquad \text{as } w(\mathbf{Z}) \to 0$$

Proof. By hypothesis, $z \in \mathbf{Z}$ and $w(\mathbf{Z}) \to 0$. By Definition I.8 this implies that $\mathbf{Z} \to z$ as $w(\mathbf{Z}) \to 0$. As p is continuous in z, this in turn implies

$$\bar{p}(\mathbf{Z}, \mathbf{Q}^0) \to \bar{p}(z, \mathbf{Q}^0) \text{ as } w(\mathbf{Z}) \to 0 \tag{4.7}$$

By Theorem I.3, all natural inclusion functions have first order convergence. So, there is a constant $\alpha > 0$ independent of the box $\mathbf{Z}$ such that

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(\bar{p}(\mathbf{Z}, \mathbf{Q}^0)) \leq \alpha \; w(\mathbf{Z}) \tag{4.8}$$

or

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(\bar{p}(\mathbf{Z}, \mathbf{Q}^0)) \to 0 \text{ as } w(\mathbf{Z}) \to 0$$

From (4.7),

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(\bar{p}(z, \mathbf{Q}^0)) \to 0 \text{ as } w(\mathbf{Z}) \to 0$$

which is the assertion of this lemma. ■

Lemma 4.10 $\bigcap_{k=1}^{\infty} \mathcal{U}_k \subseteq \mathcal{S}$

Proof. Let $z \in \mathcal{U}_k$, for all k. We are to show that $z \in \mathcal{S}$. Since $z \in \mathcal{U}_k$ for all k, it follows first that for any k, an item $\mathbf{Z}'_k$ must occur in the list $\mathcal{L}_k$ such that $z \in \mathbf{Z}'_k$. That is,

$$z \in \mathbf{Z}'_k \ \in \mathcal{L}_k, \ \text{for all } k \tag{4.9}$$

By Lemma 4.8

$$w(\mathbf{Z}'_k) \to 0 \text{ as } k \to \infty \tag{4.10}$$

From (4.9), (4.10) and Lemma 4.9

$$w(P(\mathbf{Z}'_k, \mathbf{Q}^0)) - w(\bar{p}(z, \mathbf{Q}^0)) \to 0 \text{ as } k \to \infty \tag{4.11}$$

By the inclusion property in Theorem I.1

$$\bar{p}(z, \mathbf{Q}^0) \subseteq P(\mathbf{Z}'_k, \mathbf{Q}^0) \text{ for any } k \tag{4.12}$$

From (4.11), (4.12) and Definition I.8

$$P(\mathbf{Z}'_k, \mathbf{Q}^0) \to \bar{p}(z, \mathbf{Q}^0) \text{ as } k \to \infty \tag{4.13}$$

Further, as $\mathbf{Z}'_k$ is not yet discarded in the algorithm

$$0 \in P(\mathbf{Z}'_k, \mathbf{Q}^0) \text{ for any } k \tag{4.14}$$

From (4.9), (4.13) and (4.14)

$$0 \in \bar{p}(z, \mathbf{Q}^0) \tag{4.15}$$

As p is continuous in $\mathbf{q}$

$$0 \in \bar{p}(z, \mathbf{Q}^0) \Rightarrow \exists \text{ some } \mathbf{q} \in \mathbf{Q}^0 \text{ s.t. } p(z, \mathbf{q}) = 0$$

That is, $z \in \mathcal{S}$. This completes the proof. ■

Lemma 4.11 *$d(\mathcal{U}_k, \mathcal{S}) \to 0$ as $k \to \infty$.*

Proof. By Lemma 4.7, the unions $\mathcal{U}_k$ form a chain

$$\mathcal{U}_1 \supseteq \mathcal{U}_2 \supseteq \mathcal{U}_3 ... \tag{4.16}$$

From Lemmas 4.4 and 4.10

$$\mathcal{S} = \bigcap_{k=1}^{\infty} \mathcal{U}_k \tag{4.17}$$

Now, (4.16) and (4.17) together imply that the unions $\mathcal{U}_k$ converge to $\mathcal{S}$. By Definition I.8 this means that $d(\mathcal{U}_k, \mathcal{S}) \to 0$ as $k \to \infty$. This proves the lemma. ■

The following theorem summarizes the convergence property of the proposed algorithm.

Theorem 4.12 *The collection of solution boxes* $\mathbf{Z}$ *generated in the list* $\mathcal{L}$ *of the algorithm converges to the spectral set* $\mathcal{S}$*. Moreover, this convergence is such that the collection always encloses* $\mathcal{S}$ *at any iteration.*

Proof. Let $\mathcal{A}, \mathcal{B}$ be any two non-empty compact sets. From Definition I.7, $d(\mathcal{A}, \mathcal{B}) = 0$ only if $\mathcal{A} = \mathcal{B}$. From Lemma 4.11, $\mathcal{U}_k$ converges to $\mathcal{S}$ via the Hausdorff metric d. The second part of the theorem follows from Lemma 4.4 which implies that $\mathcal{S} \subseteq \mathcal{U}_k$ for any k. ■

4.3.2 Termination

We first examine the effectiveness of the range tolerance condition.

Lemma 4.13 *Let* $\mathbf{Z} \in I\left(\mathbf{Z}^0\right)$ *and* $z \in \mathbf{Z}$*. Let* e_p *denote the maximum error in estimating the range of* p *on* $\mathbf{Z} \times \mathbf{Q}^0$ *using the natural inclusion function* $P(\mathbf{Z}, \mathbf{Q}^0)$*. Then,*

$$e_p \leq w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(\bar{p}(\mathbf{Z}, \mathbf{Q}^0)) \leq w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(P(z, \mathbf{Q}^0)) \leq w(P(\mathbf{Z}, \mathbf{Q}^0)) \tag{4.18}$$

Proof. The proof of the first inequality in (4.18) follows readily from a result of Moore [48]. We therefore prove only the latter inequalities in (4.18). As $z \in \mathbf{Z}$, $\bar{p}(z, \mathbf{Q}^0) \subseteq \bar{p}(\mathbf{Z}, \mathbf{Q}^0)$. Hence, $w(\bar{p}(z, \mathbf{Q}^0)) \leq w(\bar{p}(\mathbf{Z}, \mathbf{Q}^0))$. Subtract both sides from $w(P(\mathbf{Z}, \mathbf{Q}^0))$, and use the result $w(P(z, \mathbf{Q}^0)) = w(\bar{p}(\mathbf{Z}, \mathbf{Q}^0))$ of Lemma 4.2 to get

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(\bar{p}(\mathbf{Z}, \mathbf{Q}^0)) \leq w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(\bar{p}(z, \mathbf{Q}^0)) = w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(P(z, \mathbf{Q}^0)) \tag{4.19}$$

This proves the second inequality in (4.18). To show the last inequality in (4.18), note that as the width of any interval is always non-negative, $w(P(z, \mathbf{Q}^0)) \geq 0$. Hence,

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(P(z, \mathbf{Q}^0)) \leq w(P(\mathbf{Z}, \mathbf{Q}^0)) \tag{4.20}$$

This concludes the proof. ■

Remark 4.1 *We see from Lemma 4.13 that the maximum error* e_p *in the range can be kept within a prescribed range tolerance* ε_p *by ensuring that*

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(P(z, \mathbf{Q}^0)) < \varepsilon_p$$

Further, the fact that range tolerance condition (4.5) is more effective than the one in (4.4) follows from right most inequality in (4.18).

We next show that the proposed algorithm terminates in a *finite* number of iterations. We also give an upper bound on the number of iterations required.

Theorem 4.14 *The algorithm terminates in at most* 2γ *iterations, where*

$$\gamma := \max\left\{\log_2\left(\frac{w(\mathbf{Z}^0)}{\varepsilon_z}\right), \log_2\left(\frac{\alpha \cdot w(\mathbf{Z}^0)}{\varepsilon_p}\right)\right\} + 1$$

and where, α is a (Lipschitz) constant given in (4.8). Further, the maximum total number of subdivisions is given by $2(2^{\gamma}-1)$.

Proof. First, note that $\mathbf{Z} \in \mathcal{C}$ is a $2-$dim box, so that after $2 \cdot v$ successive subdivisions, where v is some positive integer, we will obtain $w(\mathbf{Z}) \leq w(\mathbf{Z}^0)/2^v$. Thus, in at most $2\gamma_z$ successive subdivisions, where, $\gamma_z := \log_2\left(w(\mathbf{Z}^0)/\varepsilon_z\right) + 1$, we will have $w(\mathbf{Z}) < \varepsilon_z$ and the termination condition on e_z in (4.1) will be satisfied.

Next, from Lemma 4.2 and (4.8),

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(P(\hat{z}, \mathbf{Q}^0)) \leq \alpha \cdot w(\mathbf{Z}) \tag{4.21}$$

whereas the termination condition on e_p in (4.2) is

$$w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(P(\hat{z}, \mathbf{Q}^0)) < \varepsilon_p \tag{4.22}$$

From (4.21) and (4.22), we see that if $w(\mathbf{Z}) < \varepsilon_p/\alpha$ then the termination condition on e_p is satisfied. Proceeding as above, we find that in at most $2\gamma_p$ successive subdivisions, where $\gamma_p := \log_2\left(\alpha \cdot w(\mathbf{Z}^0)/\varepsilon_p\right) + 1$, the termination condition on ε_p will be satisfied.

The processing of a subbox is completed when both the termination conditions are satisfied. From the above arguments, this is achieved in at most $\gamma := \max\left(\gamma_z, \gamma_p\right)$ successive subdivisions.

The algorithm produces a binary tree whose nodes are the regions obtained through successive subdivisions. The root of this tree is the initial search box $\mathbf{Z}^0$. The maximum total number of subdivisions occurs for a balanced tree, and from [15] equals $2 \cdot (2^{\gamma}-1)$. The maximum total number of iterations is equal to the depth γ of this tree, where γ is as given above. This completes the proof. ■

Theorem 4.15 $\mathcal{S} \subseteq \mathcal{S}^{a\,\lg}$

Proof. Similar to the proof of Theorem 3.11 ■

4.3.3 Computational complexity

Theorem 4.16 *The number of elementary operations in the algorithm when considering a single box* $\mathbf{Z} \in \mathcal{C}$ *is* $O(n)$, *where* n *is the degree of the polynomial* p.

Proof. Consider the number of operations required for processing of a single box $\mathbf{Z} \in \mathcal{C}$ in the various steps of the algorithm:

- In Step 4a, evaluation of Horner's form of $P(\mathbf{Z}, \mathbf{Q}^0)$ requires n complex interval arithmetic additions and n complex interval arithmetic multiplications, see [15, pg. 778].
- In Step 4b, checking the condition $0 \notin P(\mathbf{Z}, \mathbf{Q}^0)$ requires at most 2 comparisons.

- In Step 5, determination of the terms $w(\mathbf{Z}), w\left(P\left(\mathbf{Z}, \mathbf{Q}^0\right)\right)$ requires 4 real subtractions and 2 comparisons. Evaluation of the term $w\left(P\left(\hat{z}, \mathbf{Q}^0\right)\right)$ requires, firstly, $2n$ real interval arithmetic operations to evaluate $P\left(\hat{z}, \mathbf{Q}^0\right)$ (using Horner's form) and then 2 real subtractions plus 1 comparison to find the width. The term e_z equals $w(\mathbf{Z})$ so needs no extra operations, whereas the term $e_p = w(P(\mathbf{Z}, \mathbf{Q}^0)) - w(P(\hat{z}, \mathbf{Q}^0))$ needs 1 real subtraction. Finally, checking the conditions $(e_p < \varepsilon_p)$ and $(e_z < \varepsilon_z)$ requires 2 comparisons.

- In Step 6, finding a coordinate direction k parallel to which $\mathbf{Z}$ has an edge of greater length requires 2 real subtractions and 1 comparison. Subdividing $\mathbf{Z}$ in direction k getting subboxes $\mathbf{Z}^1$ and $\mathbf{Z}^2$ requires 1 real addition and 1 real division.

Thus, Step 4 needs a maximum of $2n$ complex interval arithmetic operations and 2 comparisons, Step 5 needs a total of $2n$ real interval arithmetic operations, 7 real subtractions, and 5 comparisons, and Step 6 needs a total of 4 real arithmetic operations and 1 comparison.

Therefore, totally, $2n$ complex interval arithmetic operations, $2n$ real interval arithmetic operations, 11 real arithmetic operations, and 8 comparisons are needed. Clearly, the number of elementary operations in the algorithm is linear in n. ■

4.3.4 Reliability

The below theorem shows the reliability of the spectral results computed by a MIA implementation of the proposed algorithm.

Theorem 4.17 (Reliability of spectral set) *Let $\mathcal{S}_{mia}^{a\,\mathrm{lg}}$ denote the spectral set computed by a MIA implementation of the algorithm. Then,*

$$\mathcal{S} \subseteq \mathcal{S}^{a\,\mathrm{lg}} \subseteq \mathcal{S}_{mia}^{a\,\mathrm{lg}}$$

Proof. Similar to the proof of Theorem 3.12. ■

4.4 Illustrative examples

We demonstrate the proposed algorithm on two examples considered by Cerone [12].

Example 4.1 *The polytopic family*

$$p\,(s, \mathbf{q}) = \lambda_0(\mathbf{q}) + \lambda_1(\mathbf{q})s + \lambda_2(\mathbf{q})s^2 + \lambda_3(\mathbf{q})s^3 + \lambda_4(\mathbf{q})\,s^4$$

where, $\mathbf{q} = [q_1, \ldots, q_4]$, $\lambda_i(\mathbf{q}) = (1+q_i)$, *for* $i = 0, \ldots, 3$ *and* $\lambda_4(\mathbf{q}) = (5+q_4)$. *Each parameter* q_i *varies over the interval* $[-0.85, 0.85]$.

Example 4.2 *The polytopic family*

$$p(s, \mathbf{q}) = \lambda_0(\mathbf{q}) + \lambda_1(\mathbf{q})s + \lambda_2(\mathbf{q})s^2 + \ldots + \lambda_9(\mathbf{q})\, s^9$$

where, $\mathbf{q} = [q_1, \ldots, q_9]$, $\lambda_i(\mathbf{q}) = (1+q_i)$, *for* $i = 0, \ldots, 8$ *and* $\lambda_9(\mathbf{q}) = (5+q_9)$. *Each parameter* q_i *varies over the interval* $[-0.4, 0.4]$.

In Cerone's algorithm, there is no accuracy parameter which can be prescribed, whereas in the proposed algorithm, we can specify the domain and range accuracy tolerances. We set the domain and range accuracy tolerances as $\varepsilon_z, \varepsilon_p = 0.01$.

We implement a machine interval arithmetic version of the proposed algorithm in INTLAB [61] on a PC Pentium III 650 MHz 256 MB RAM machine.

In Example 4.1, we find the initial box enclosing the interval Henrici disk to be $\mathbf{Z}^0 = [-1.6343, 1.6343]^2$, while in Example 4.2 it is $\mathbf{Z}^0 = [-1.7524, 1.7524]^2$. To compute $\mathcal{S}^{a\,\mathrm{lg}}(\mathcal{P})$, the proposed algorithm takes 23 and 27 iterations, respectively, and the computed spectral sets are shown in Figs. 4.1 and 4.2.

We observe in both examples that $\mathcal{S}^{a\,\mathrm{lg}}(\mathcal{P})$ indeed encloses the spectral sets reported by Cerone [12]. Moreover, we find it reassuring to see that in both examples $\mathcal{S}^{a\,\mathrm{lg}}(\mathcal{P})$ also encloses all the roots (computed using *roots* routine of MATLAB) of approximately 10^5 fixed polynomials selected randomly from the respective polytopic family.

We stress that the proposed algorithm *guarantees* that the computed spectral set is reliable, and that no actual points would be missing. Further, the proposed algorithm *guarantees* also that the spectral set is computed to the prescribed domain and range accuracy tolerances. Such guarantees are not available with the algorithms of Cerone [12] and Barmish and Tempo [4]. Therefore, the spectral sets computed by these method are not guaranteed to include all the points in the actual spectral set, and, moreover, the accuracy of their results remains unknown.

4.5 Conclusions

We have presented an algorithm to compute the spectral set for the special class of polytopic polynomials. The only requirement is that polynomial coefficients be continuous in the uncertain parameters. We recall the main features of the proposed algorithm are its provision of several *guarantees*: that the computed spectral set is reliable and accurate, that all actual spectral points are included in the computed set, and that the spectral set can be computed in a finite number of iterations for a prescribed accuracy. Further, the computational complexity of the proposed algorithm is $O(n)$, in contrast to $O(n^2)$ for existing algorithms, where n is the degree of the polynomial.

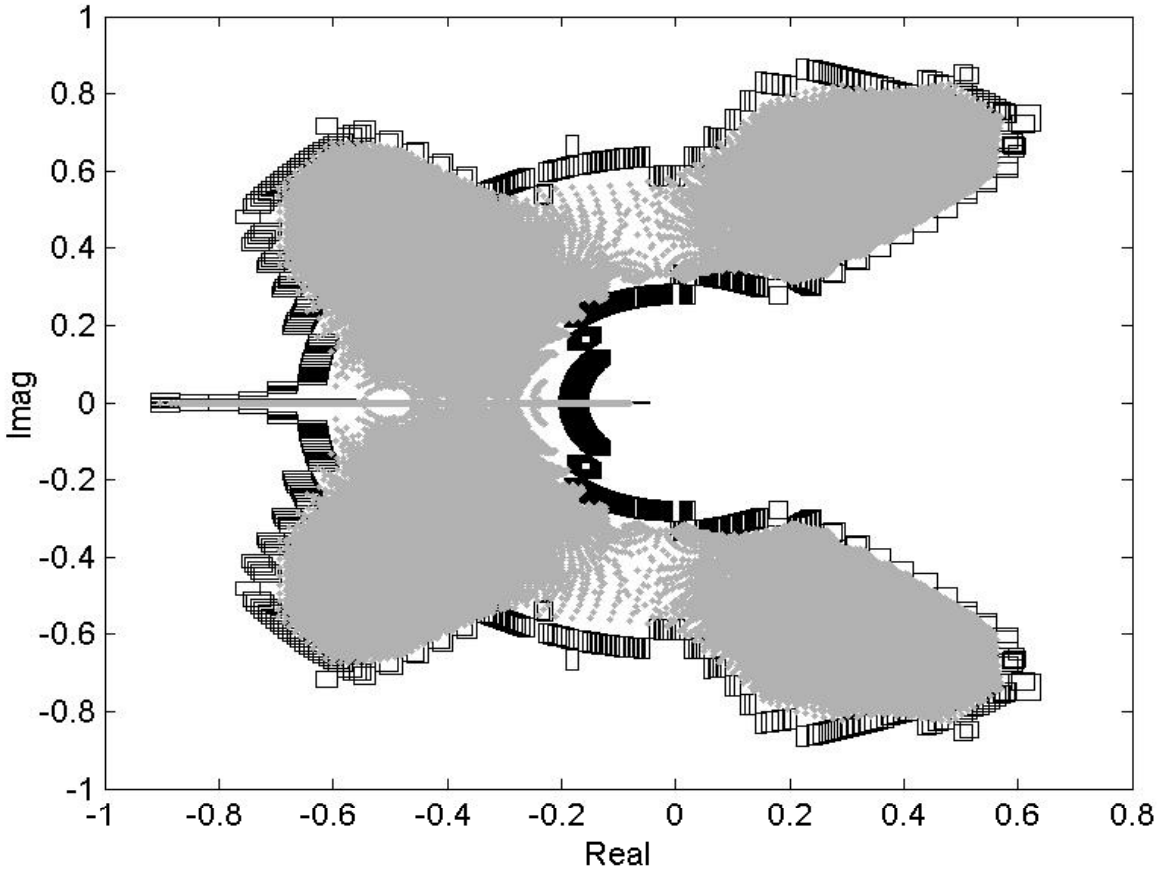

FIGURE 4.1. Spectral set of Example 4.1 computed using the proposed algorithm. The domain and range accuracy tolerances are $\varepsilon_z, \varepsilon_p = 0.01$. Only the outer boundary boxes of the computed set are shown. For comparison, the spectral set of 10^5 fixed polynomials picked randomly from the polynomial family are computed using *roots* routine of MATLAB. These are the inner points shown as light shaded area in the plot.

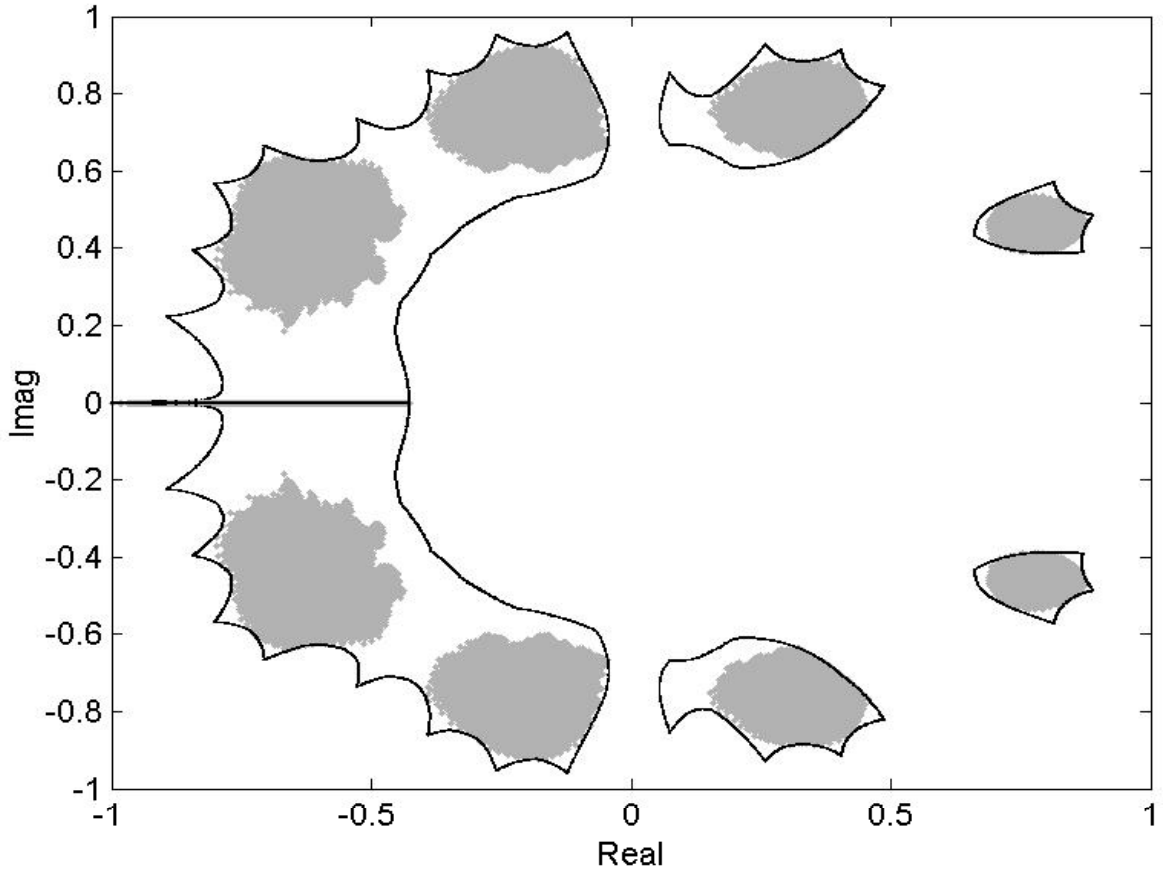

FIGURE 4.2. Spectral set of Example 4.2 computed using the proposed algorithm. The domain and range accuracy tolerances are $\varepsilon_z, \varepsilon_p = 0.01$. Only the outer boundary boxes of the computed set are shown. For comparison, the spectral set of approximately 10^5 fixed polynomials picked randomly from the polynomial family are computed using *roots* routine of the MATLAB. These are the inner points shown as light shaded area in the plot.

5

Spectral sets - general case

5.1 Introduction

Consider a family of real nth order polynomials $\{p(z, \mathbf{q}), \mathbf{q} \in \mathbf{Q}^0\}$ where, $z \in \mathcal{C}$ is the complex variable, $\mathbf{q} \in \Re^n$ is a vector of system parameters occurring *nonlinearly* in the coefficients of p, and $\mathbf{Q}^0 \subseteq \Re^n$ is a bounding box for $\mathbf{q}$ given as

$$\mathbf{Q}^0 := \{\mathbf{q} \in \Re^n : \underline{q}_i \leq \mathbf{q}_i \leq \overline{q}_i, \text{ where } \underline{q}_i, \overline{q}_i \in \Re, \ i = 1, 2, ..., n\}$$

For this polynomial family, we address the problem of computing the spectral set defined as

$$\mathcal{S} := \{z \in \mathcal{C} : p(z, \mathbf{q}) = 0, \text{ for some } \mathbf{q} \in \mathbf{Q}^0\} \tag{5.1}$$

As seen in section 1.1.3 and in the previous chapter, the spectral set is of considerable interest in robustness analysis of linear systems having real parametric uncertainty. However, there is a lack of techniques in the literature to compute the spectral set for polynomials whose coefficients have *nonlinear* dependencies on the parameters $\mathbf{q}$.

In this work, we aim to address this issue by presenting an algorithm to compute the spectral set $\mathcal{S}$ of a family of polynomials with *nonlinear* parametric dependencies. We develop the proposed algorithm using tools of interval analysis [48]. The proposed algorithm has several useful features:

1. The proposed algorithm is applicable to nonlinear parametric dependencies of a very general class where the polynomial coefficients need to be only continuous in the parameters.

2. The proposed algorithm guarantees that the computed spectral set contains *all* points of the actual spectral set, see Theorem 5.13 below. This guarantee meets an important requirement in robust stability analysis and synthesis of control systems.

3. It guarantees that the spectral set is computed to a prescribed accuracy, see Theorem 5.11 below.

4. It computes spectral sets that are *reliable* despite all kinds of computational errors, see Theorem 5.14 below.

5. It computes the spectral set in a *finite* number of iterations, for a prescribed accuracy. Moreover, an upper bound on the number of algorithmic iterations is given, see Theorem 5.12 below.

The rest of this chapter is organized as follows. In section 5.2, we present the proposed algorithm for computing the spectral set. In section 5.3, we study the theoretical and computational properties of the proposed algorithm. In section 5.4, we demonstrate the proposed algorithm on an illustrative example. Lastly, in section 5.5, we give the conclusions of the chapter.

5.2 Proposed algorithm

Let $z := re^{j\theta}, \quad \mathbf{y} := (r, \theta)$. Define the functions

$$\begin{aligned} f_{\mathrm{Re}}(\mathbf{y}, \mathbf{q}) &: = \mathrm{Re}\left\{p(z = re^{j\theta}, \mathbf{q})\right\}; \quad f_{\mathrm{Im}}(\mathbf{y}, \mathbf{q}) := \mathrm{Im}\left\{p(z = re^{j\theta}, \mathbf{q})\right\} \\ f(\mathbf{y}, \mathbf{q}) &: = (f_{\mathrm{Re}}(\mathbf{y}, \mathbf{q}), f_{\mathrm{Im}}(\mathbf{y}, \mathbf{q})) \end{aligned} \tag{5.2}$$

Then, the spectral set $\mathcal{S}$ in (5.1) can be expressed as

$$\mathcal{S} = \{\mathbf{y} : f(\mathbf{y}, \mathbf{q}) = \mathbf{0}, \text{for some } \mathbf{q} \in \mathbf{Q}^0\}$$

We will assume the following throughout this chapter.

Assumption 5.1 *The polynomial p (and thereby the function f) are continuous in the parameters $\mathbf{q}$ on the domain $\mathbf{Q}^0$.*

Remark 5.1 *As p is a polynomial in z, it follows that f is continuously differentiable everywhere in $\mathbf{y}$.*

5.2.1 Initial search box

The result of Henrici in [29] states that all roots of a polynomial $p(z, \mathbf{q})$ are contained in the disk with center at the origin and radius

$$r_h' = 2 \max_{1 \le k \le n} \left| \frac{\lambda_{n-k}}{\lambda_n} \right|^{\frac{1}{k}}$$

where, $\boldsymbol{\lambda}$ is the real vector of polynomial coefficients. For the family of polynomials under consideration, it readily follows from Theorem I.1 that all the roots of the family are contained in the disk with center at the origin and radius

$$r_h = 2 \max_{1 \le k \le n} \left| \frac{\boldsymbol{\Lambda}_{n-k}}{\boldsymbol{\Lambda}_n} \right|^{\frac{1}{k}}$$

where, $\boldsymbol{\Lambda}$ is the corresponding vector of interval coefficients. The spectral set $\mathcal{S}$ is therefore enclosed by the above described disk, which we call as the *interval Henrici* disk. Recalling that $\mathbf{y} := (r, \theta)$, an initial search box $\mathbf{Y}^0$ equivalent to the interval Henrici disk is clearly

$$\mathbf{Y}^0 = ([0, r_h], [0, 2\pi])$$

Further, since the coefficients of p are assumed to be real, any complex roots of p must occur as conjugate pairs. We may therefore restrict the initial search box to enclose the upper half of interval Henrici disk, so that

$$\mathbf{Y}^0 = ([0, r_h], [0, \pi])$$

We shall assume in the sequel that the proposed algorithm is applied to compute the spectral set in this restricted $\mathbf{Y}^0$, and that the rest of the spectral set containing any roots lying in the lower half of the Henrici disk is subsequently obtained using complex conjugacy.

For brevity, in the sequel we refer to a point belonging to the spectral set as 'a spectral point'.

5.2.2 Algorithm

The proposed algorithm can be briefly described as a binary search algorithm based on a geometrical subdivision process, root exclusion test, and the Generalized Krawczyk operator. It consists of three parts: an initialization part, an iterative part, and a termination part.

In the initialization part, a natural inclusion function F is constructed for f, and, following section 5.2.1, an initial box $\mathbf{Y}^0$ that encloses the actual spectral set is also constructed. Next, the two lists that are needed in the algorithm are initialized here: a working list $\mathcal{L}$ that contains boxes for processing is initialized with the box $\mathbf{X}^0 = \left(\mathbf{Y}^0, \mathbf{Q}^0\right)$, and the list $\mathcal{L}^{sol}$ that contains solution boxes having spectral points is initialized to the empty list. Then, the algorithm branches to the iterative part.

In the iterative part, all boxes $\mathbf{X}_{(i)}$ present in the working list $\mathcal{L}$ are chosen, and a branch and bound strategy is applied to discard irrelevant parts (or the whole) of these boxes using the zero exclusion test, the Generalized Krawczyk test, and subdivisions.

A box is accepted as a solution box if its width does not exceed a prescribed domain accuracy tolerance ε_x, see Remark 5.3. For all such boxes, $\mathbf{Y}_{(i)}$ is deposited in the solution list $\mathcal{L}^{sol}$. The remaining boxes are subdivided along their longest directions, and the resulting

subboxes are put in $\mathcal{L}$. The entire iterative part is repeated till no more boxes are left in $\mathcal{L}$ for processing. Then, the algorithm branches to the termination part.

In the termination part, the spectral set $\mathcal{S}^{a\,\mathrm{lg}}$ is constructed as the union of all boxes $\mathbf{Y}_{(i)}$ present in $\mathcal{L}^{sol}$. The algorithm now exits.

The entire process is greatly speeded up by concurrently (rather than sequentially) processing all the subboxes present in a given iteration. Concurrent processing is possible through the use of vectorized interval operations for function evaluations, subdivisions, width checks, etc., see [51] for details.

We next present the proposed algorithm.

Algorithm (Spectral set computation algorithm)

Input: An expression for the function f in (5.2), the parameter box $\mathbf{Q}^0$, and the domain accuracy tolerance ε_x.

Output: The computed spectral set $\mathcal{S}^{a\,\mathrm{lg}}$.

Note: The algorithm is to be executed in the order given below, except when otherwise indicated.

BEGIN Algorithm

1. (**Initialization part**) From the expression for f, find a natural inclusion function F.
2. Construct initial boxes and lists:
   (a) Following section 5.2.1, construct an initial search box $\mathbf{Y}^0$ that encloses the spectral set $\mathcal{S}$, and then construct $\mathbf{X}^0 := \left(\mathbf{Y}^0, \mathbf{Q}^0\right)$.
   (b) Set $k \leftarrow 0$ and initialize lists $\mathcal{L}^{sol} \leftarrow \{\}$, $\mathcal{L} \leftarrow \{\mathbf{X}^0\}$.
3. (**Iterative part**) Start a new iteration:
   (a) Set $k \leftarrow k+1$ and $l_r \leftarrow$ length of $\mathcal{L}$.
   (b) Pick all boxes $\mathbf{X}_{(i)} = \left(\mathbf{Y}_{(i)}, \mathbf{Q}_{(i)}\right)$, $i = 1, 2, ..., l_r$ from $\mathcal{L}$ and delete their entries from $\mathcal{L}$.
4. Test phase using an interval zero exclusion test and Generalized Krawczyk test:
   (a) Evaluate $F(\mathbf{X}_{(i)}), i = 1, 2, ..., l_r$.
   (b) (Interval zero exclusion test): IF $0 \notin F(\mathbf{X}_{(i)})$ THEN discard $\mathbf{X}_{(i)}$, $i = 1, 2, ..., l_r$.
   (c) (Generalized Krawczyk test): For each remaining $\mathbf{X}_{(i)} = \left(\mathbf{Y}_{(i)}, \mathbf{Q}_{(i)}\right)$, do the following:
      i. Find a natural inclusion function F'_y for the derivative f'_y, and set $C \leftarrow \left\{m\left(F'_y\right)\right\}^{-1}$. Evaluate the Generalized Krawczyk operator $\mathcal{K}\left(\mathbf{X}_{(i)}\right)$ defined in (I.2) as

$$\mathcal{K}\left(\mathbf{X}_{(i)}\right) = \hat{\mathbf{y}}_{(i)} - CF\left(\mathbf{X}_{(i)}\right) + \left\{I - CF'_y\left(\mathbf{X}_{(i)}\right)\right\}\left(\mathbf{Y}_{(i)} - \hat{\mathbf{y}}_{(i)}\right), \quad \hat{\mathbf{y}}_{(i)} = m\left(\mathbf{Y}_{(i)}\right)$$

ii. Find $\mathbf{Y}'_{(i)} = \mathcal{K}\left(\mathbf{X}_{(i)}\right) \bigcap \mathbf{Y}_{(i)}$.

iii. IF $\mathbf{Y}'_{(i)} = \varnothing$, THEN discard $\mathbf{X}_{(i)}$ ELSE set $\mathbf{X}_{(i)} \leftarrow \left(\mathbf{Y}'_{(i)}, \mathbf{Q}_{(i)}\right)$.

(d) If no more $\mathbf{X}_{(i)}$ remain, go to step 8.

5. Solution set:

(a) For each remaining $\mathbf{X}_{(i)}$, evaluate $w(\mathbf{X}_{(i)})$. IF $w(\mathbf{X}_{(i)}) < \varepsilon_x$ THEN enter $\mathbf{Y}_{(i)}$ in $\mathcal{L}^{sol}$ and discard $\mathbf{X}_{(i)}$ from further processing.

(b) If no more $\mathbf{X}_{(i)}$ remain, go to step 8.

6. Subdivision phase:

For each remaining box $\mathbf{X}_{(i)}$, find a coordinate direction k_i parallel to which $\mathbf{X}_{(i)}$ has an edge of greater length. Subdivide $\mathbf{X}_{(i)}$ in direction k_i getting subboxes $\mathbf{X}^1_{(i)}$ and $\mathbf{X}^2_{(i)}$ such that $\mathbf{X}_{(i)} = \mathbf{X}^1_{(i)} \cup \mathbf{X}^2_{(i)}$. Enter the subboxes $\mathbf{X}^1_{(i)} \cup \mathbf{X}^2_{(i)}$ in $\mathcal{L}$.

7. End current iteration: Return to step 3.

8. (**Termination part**) Construct spectral set as $\mathcal{S}^{a\,lg} \leftarrow \bigcup_{\mathbf{Y}_{(i)} \in \mathcal{L}^{sol}} \mathbf{Y}_{(i)}$, output $\mathcal{S}^{a\,lg}$, and EXIT.

END Algorithm.

Remark 5.2 *(See Remark 3.1) We require the continuity properties of f in $\mathbf{q}$ in order that the corresponding natural inclusion function be continuous. The latter property ensures that the width of the interval evaluation $F(\mathbf{X})$ tends to zero as the width of $\mathbf{X}$ tends to zero, so that in turn, convergence and arbitrary accuracy can be achieved.*

Remark 5.3 *(See Remark 3.2) Suppose a spectral point exists in a box. If this box is used as an enclosure of the spectral point, then clearly, the error can be no greater than the width of the box itself. Therefore, the processing of a subbox is terminated in the algorithm when the* domain *accuracy tolerance of the form*

$$w(\mathbf{X}) < \varepsilon_x \tag{5.3}$$

is satisfied for that subbox.

5.3 Properties

We next investigate the various properties of the proposed algorithm.

First, we give a result that justifies the interval zero exclusion test in step 4b in the algorithm.

Lemma 5.1 *Let* $\mathbf{X} \in I\left(\mathbf{X}^0\right)$ *be a subbox, where* $\mathbf{X}^0$ *is as in step 2a of the algorithm. If* $0 \notin F(\mathbf{X})$ *then* $\mathbf{X}$ *can be discarded in the algorithm.*

Proof. Similar to the proof of Lemma 3.1. ■

The next result justifies the Generalized Krawczyk test in step 4c in the algorithm.

Theorem 5.2 *Let* $\mathbf{X} \in I\left(\mathbf{X}^0\right)$. *Then,*

1. *(**Non-existence test for spectral points**):*

$$\text{If } \mathcal{K}(\mathbf{X}) \bigcap \mathbf{Y} = \emptyset, \text{ then there are no spectral points in } \mathbf{Y} \quad (\text{for any } \mathbf{q} \in \mathbf{Q})$$

2. *(**Existence test for spectral points**):*

$$\text{If } \mathcal{K}(\mathbf{X}) \subseteq \mathbf{Y}, \text{ then at least one spectral point exists in } \mathbf{Y} \text{ for every } \mathbf{q} \in \mathbf{Q}$$

Proof. Consider a fixed but arbitrary $\mathbf{q} \in \mathbf{Q}$. Define the function g as

$$g(\mathbf{y}, \mathbf{q}) := \mathbf{y} - Cf(\mathbf{y}, \mathbf{q}), \quad \text{for } \mathbf{y} \in \mathbf{Y} \tag{5.4}$$

where, $C \in \Re^{l \times l}$ is an arbitrary nonsingular real matrix. Since f is continuous by Assumption 5.1, g is also continuous. Further, the nonempty search box $\mathbf{Y}$ is clearly convex and compact. Then, by the well-known Brouwer's fixed point theorem

$$\bar{g}(\mathbf{Y}, \mathbf{q}) \subseteq \mathbf{Y} \text{ implies the existence of some } \mathbf{y}^* \in \mathbf{Y} : g(\mathbf{y}^*, \mathbf{q}) = \mathbf{y}^*$$

Moreover, as C is nonsingular by hypothesis, this further implies $f(\mathbf{y}^*, \mathbf{q}) = 0$. That means, for a nonsingular C,

$$\bar{g}(\mathbf{Y}, \mathbf{q}) \subseteq \mathbf{Y} \text{ implies the existence of a spectral point in } \mathbf{Y} \tag{5.5}$$

By the mean value theorem [48, chapter 6],

$$f(\mathbf{y}, \mathbf{q}) \in f(\hat{\mathbf{y}}, \mathbf{q}) + F'_y(\mathbf{Y}, \mathbf{q})(\mathbf{Y} - \hat{\mathbf{y}}), \quad \text{for all } \mathbf{y} \in \mathbf{Y}$$

where $\hat{\mathbf{y}} = m(\mathbf{Y})$. Substituting the above in (5.4) gives

$$\begin{aligned} g(\mathbf{y}, \mathbf{q}) &= \mathbf{y} - Cf(\mathbf{y}, \mathbf{q}) \\ &\in \mathbf{y} - C\left\{f(\hat{\mathbf{y}}, \mathbf{q}) + F'_y(\mathbf{Y}, \mathbf{q})(\mathbf{Y} - \hat{\mathbf{y}})\right\} \\ &\in \mathbf{y} - Cf(\hat{\mathbf{y}}, \mathbf{q}) - CF'_y(\mathbf{Y}, \mathbf{q})(\mathbf{Y} - \hat{\mathbf{y}}), \quad \text{for all } \mathbf{y} \in \mathbf{Y} \end{aligned}$$

Therefore, by Theorem I.1

$$\begin{aligned} \bar{g}(\mathbf{Y}, \mathbf{q}) &\subseteq \mathbf{Y} - Cf(\hat{\mathbf{y}}, \mathbf{q}) - CF'_y(\mathbf{Y}, \mathbf{q})(\mathbf{Y} - \hat{\mathbf{y}}) \\ &\subseteq \hat{\mathbf{y}} - Cf(\hat{\mathbf{y}}, \mathbf{q}) + \left\{I - CF'_y(\mathbf{Y}, \mathbf{q})\right\}(\mathbf{Y} - \hat{\mathbf{y}}) \\ &\subseteq \mathcal{K}(\mathbf{Y}, \mathbf{q}) \end{aligned}$$

From (5.5) it follows that $\mathcal{K}(\mathbf{Y},\mathbf{q}) \subseteq \mathbf{Y}$ implies the existence of at least one spectral point in $\mathbf{Y}$.

Further, if there is a point $\mathbf{y}^* \in \mathbf{Y}$ for which $f(\mathbf{y}^*,\mathbf{q}) = 0$, then from (5.4), $g(\mathbf{y}^*,\mathbf{q}) = \mathbf{y}^*$. Hence, $\mathbf{y}^* \in \bar{g}(\mathbf{Y},\mathbf{q}) \subseteq \mathcal{K}(\mathbf{Y},\mathbf{q})$. In other words, any spectral point in $\mathbf{Y}$ is also in $\mathcal{K}(\mathbf{Y},\mathbf{q})$. Therefore, if $\mathcal{K}(\mathbf{Y},\mathbf{q}) \bigcap \mathbf{Y} = \emptyset$ then there are no spectral points in $\mathbf{Y}$.

The arguments above are given for a fixed but arbitrary $\mathbf{q} \in \mathbf{Q}$. The assertions of the theorem for the (entire) parameter box $\mathbf{Q}$ follow readily by repeating the same arguments for every $\mathbf{q} \in \mathbf{Q}$. ■

Remark 5.4 *If condition (1) in Theorem 5.2 is satisfied, then we can discard* $\mathbf{X}$ *in the search for spectral points. We note that this exclusion test based on Generalized Krawczyk operator is in addition to the interval zero exclusion test in step 4b.*

Remark 5.5 *As proven above, any spectral point in* $\mathbf{Y}$ *is also in* $\mathcal{K}(\mathbf{X})$. *So if* $\mathcal{K}(\mathbf{X}) \bigcap \mathbf{Y} \neq \emptyset$, *we can replace the box* $\mathbf{X}$ *with the smaller box* $\mathbf{X}' := (\mathcal{K}(\mathbf{X}) \bigcap \mathbf{Y}, \mathbf{Q})$ *and continue with the algorithm,* without *losing any spectral points that may be present in* $\mathbf{X}$.

5.3.1 Convergence

To study the convergence properties of the proposed algorithm, we assume that the tolerance criterion can never be satisfied (i.e., $\varepsilon_x = 0$). We also assume that list sizes are not a limitation.

Definition 5.1 *Let* $\mathcal{L}_k$ *denote the list* $\mathcal{L}$ *present at the start of* k*th iteration of the algorithm, and denote the* i*th box of this list as* $\mathbf{X}_{ki}$. *Define the unions*

$$\mathcal{U}_k = \bigcup_{\mathbf{Y}_{ki} \in \mathcal{L}_k} \mathbf{Y}_{ki}, \quad \mathcal{V}_k = \bigcup_{\mathbf{Q}_{ki} \in \mathcal{L}_k} \mathbf{Q}_{ki}, \quad \mathcal{W}_k = \bigcup_{\mathbf{X}_{ki} \in \mathcal{L}_k} \mathbf{X}_{ki} \tag{5.6}$$

Note that $\mathcal{U}_1 = \mathbf{Y}^0, \mathcal{V}_1 = \mathbf{Q}^0, \mathcal{W}_1 = \mathbf{X}^0$.

Lemma 5.3 $\mathcal{S}$ *is a compact set in* $\Re^2$.

Proof. The parameter box $\mathbf{Q}^0$ is closed and bounded, i.e., compact. Now, as is well-known, the roots of a polynomial are continuous in its coefficients. Further, by Assumption 5.1, the coefficients of polynomial p are continuous on $\mathbf{Q}^0$. Therefore, the roots of p are continuous on $\mathbf{Q}^0$. As the image of a continuous mapping on a compact set is compact, so the root set is compact, i.e., $\mathcal{S}$ is compact. ■

Lemma 5.4 *The unions* $\mathcal{U}_k, \mathcal{V}_k, \mathcal{W}_k$ *are compact sets at any* k.

Proof. Similar to the proof of Lemma 3.3. ■

Lemma 5.5 $\mathcal{S} \subseteq \bigcap_{k=1}^{\infty} \mathcal{U}_k$

Proof. It is sufficient to show that $\mathcal{S} \subseteq \mathcal{U}_k$ for any k. The assertion of the lemma then follows. Consider the algorithm with $k = 1$. Firstly, note that by construction, all spectral points in $\mathcal{S}$ are contained in $\mathbf{Y}^0$. Since, $\mathcal{U}_1 = \mathbf{Y}^0$, we have $\mathcal{S} \subseteq \mathcal{U}_1$. By Lemma 5.1, the discarding process using the interval zero exclusion test in step 4b does not delete any point in $\mathbf{Y}^0$ that belongs to $\mathcal{S}$. Further, by Theorem 5.2 (see also Remarks 5.4 and 5.5), the Generalized Krawczyk test in Step 4c also does not delete any point in $\mathbf{Y}^0$ that belongs to $\mathcal{S}$. Moreover, none of these points can be lost in the subsequent subdivision step 6, because every box is replaced by both its subboxes in the list $\mathcal{L}$. Thus, at the end of first iteration, all the points in $\mathcal{S}$ are retained in $\mathcal{U}_2$, i.e., $\mathcal{S} \subseteq \mathcal{U}_2$. By induction on k, we have $\mathcal{S} \subseteq \mathcal{U}_k$ for any $k \Rightarrow \mathcal{S} \subseteq \bigcap_{k=1}^{\infty} \mathcal{U}_k$. ■

Lemma 5.6 *The sequence* $\{\mathcal{U}_k\}_{k=1}^{\infty}$ *has the property*

$$\mathcal{U}_1 \supseteq \mathcal{U}_2 \supseteq \mathcal{U}_3 ...$$

Proof. Similar to the proof of Lemma 4.7, with additional Step 4c where the Generalized Krawczyk test is used to delete some irrelevant portion of each $\mathbf{X}_{(i)}$. ■

Lemma 5.7 *Let* w_k *denote the maximum width of the boxes* $\mathbf{X}_{ik}$ *of the* k*th list* $\mathcal{L}_k$ *generated by the algorithm. Then,*

$$w_k \to 0 \text{ as } k \to \infty$$

Proof. As in Lemma 3.6, the proof follows from the lemma in Ratschek [59]. ■

Lemma 5.8

$$w(F(\mathbf{X})) \to 0 \quad \text{as } w(\mathbf{X}) \to 0$$

Proof. Similar to the proof of Lemma 3.7. ■

Lemma 5.9 $\bigcap_{k=1}^{\infty} \mathcal{U}_k \subseteq \mathcal{S}$

Proof. Similar to the proof of Lemma 3.8. ■

Lemma 5.10 $d(\mathcal{U}_k, \mathcal{S}) \to 0$ *as* $k \to \infty$.

Proof. Similar to the proof of Lemma 4.11. ■

The following theorem summarizes the convergence property of the proposed algorithm.

Theorem 5.11 *The collection of solution boxes* $\mathbf{Y}$ *generated in the list* $\mathcal{L}$ *of the algorithm converges to the spectral set* $\mathcal{S}$*. Moreover, this convergence is such that the collection always encloses* $\mathcal{S}$ *at any iteration.*

Proof. Similar to the proof of Theorem 4.12. ■

5.3.2 Termination

We show that the proposed algorithm terminates in a *finite* number of iterations. We also give an upper bound on the number of iterations required by the proposed algorithm.

Theorem 5.12 *The algorithm terminates in at most* $(n+2)\,\gamma$ *iterations, where* γ *is given by*

$$\gamma := \left\{ \log_2 \left(\frac{w(\mathbf{X}^0)}{\varepsilon_x} \right) + 1 \right\}$$

Further, the maximum total number of subdivisions is given by $2(2^\gamma - 1)$.

Proof. First, note that $\mathbf{X}$ is a $(n+2)-$ dim box, so that after $(n+2)\cdot v$ successive subdivisions, where v is some positive integer, we will obtain $w(\mathbf{X}) \leq w(\mathbf{X}^0)/2^v$. Thus, the processing of a subbox is completed in at most $(n+2)\,\gamma$ successive subdivisions, where $:= \log_2 \left(w(\mathbf{X}^0)/\varepsilon_x\right) + 1$, because then $w(\mathbf{X}) < \varepsilon_x$. The algorithm produces a binary tree whose nodes are the regions obtained through successive subdivisions. The root of this tree is the node corresponding to the initial region $\mathbf{X}^0$. The maximum total number of iterations is equal to the depth γ of this tree, where γ is as given above. Further, the maximum total number of subdivisions occurs for a balanced tree, and from [15] equals $2 \cdot (2^\gamma - 1)$. This completes the proof. ■

Theorem 5.13 $\mathcal{S} \subseteq \mathcal{S}^{a\,\mathrm{lg}}$.

Proof. Similar to the proof of Theorem 3.11. ■

5.3.3 Reliability

The below theorem shows the reliability of the spectral results computed by a MIA implementation of the proposed algorithm.

Theorem 5.14 (Reliability of spectral set) *Let* $\mathcal{S}_{mia}^{a\,\mathrm{lg}}$ *denote the spectral set computed by a MIA implementation of the algorithm. Then,*

$$\mathcal{S} \subseteq \mathcal{S}^{a\,\mathrm{lg}} \subseteq \mathcal{S}_{mia}^{a\,\mathrm{lg}}$$

Proof. Similar to the proof of Theorem 3.12. ■

5.4 Illustrative example

We demonstrate the proposed algorithm on the following examples with nonlinear parametric dependency. We implement a MIA version of the proposed algorithm in INTLAB [61] on a PC Pentium III 850 MHz 512 MB RAM machine. Note that existing techniques are not readily applicable to compute the spectral set in these examples.

Example 5.1 *Consider the characteristic polynomial of the standard second order mechanical system of mass, spring and damper, with nonlinear parametric dependency:*

$$\begin{aligned} p(s,\mathbf{q}) &= \lambda_0(\mathbf{q}) + \lambda_1(\mathbf{q})s + s^2, \quad \mathbf{q} = [q_1, q_2] \\ \lambda_0(\mathbf{q}) &= q_2^2, \quad \lambda_1(\mathbf{q}) = 2q_1q_2 \end{aligned}$$

where, $q_1 \in [0.2, 0.8], q_2 \in [1, 5]$. *We set the domain accuracy tolerance as* $\varepsilon_x = 0.01$.

The initial box enclosing the interval Henrici disk is obtained as

$$\mathbf{Y}^0 = ([0, 1.6343], [0, 2\pi])$$

The proposed algorithm takes 25 iterations to compute the spectral set plotted in Fig. 5.1.

Next, we validate the obtained results as follows. We pick 2500 fixed polynomials randomly from the polynomial family, and compute analytically the spectral set using the formula for finding the roots of a quadratic equation. The analytically found spectral set is also plotted in the same figure. We observe from this figure that the spectral set computed with the proposed algorithm fairly tightly encloses the analytically found spectral set.

Example 5.2 *Consider the characteristic polynomial of the third order system with nonlinear parametric dependency:*

$$\begin{aligned} p(s,\mathbf{q}) &= \lambda_0(\mathbf{q}) + \lambda_1(\mathbf{q})s + \lambda_2(\mathbf{q})s^2 + s^3, \quad \mathbf{q} = [q_1, q_2, q_3] \\ \lambda_0(\mathbf{q}) &= \sqrt{q_2}, \quad \lambda_1(\mathbf{q}) = \cos(q_3 + q_1q_2), \text{ and } \lambda_2(\mathbf{q}) = e^{q_1}\ln(q_3) \end{aligned}$$

where, $q_1 \in [0.9, 1.1]$, $q_2 \in [18, 22]$, *and* $q_2 \in [9, 11]$. *We set the domain accuracy tolerance as* $\varepsilon_x = [0.01, 0.005]$.

The initial box enclosing the interval Henrici disk is obtained as

$$\mathbf{Y}^0 = ([0, 14.4074], [0, 2\pi])$$

The proposed algorithm takes 28 iterations to compute the spectral set plotted in Fig. 5.2.

Next, we validate the obtained results as follows. We pick 1331 fixed polynomials randomly from the polynomial family, and compute the spectral set using the widely used *roots* routine of MATLAB [46]. The spectral set computed as above is also plotted in the same figure. We observe from this figure that the spectral set computed with the proposed algorithm fairly tightly encloses the analytically found spectral set.

5.5 Conclusions

We have presented an algorithm to compute the spectral set for a very general class of uncertain polynomials where the polynomial coefficients need to be only continuous in the

parameters. We recall the main features of the proposed algorithm are its provision of several *guarantees*: that the computed spectral set is reliable and accurate, that all actual spectral points are included, and that the spectral set can be computed in a finite number of iterations, for a prescribed accuracy.

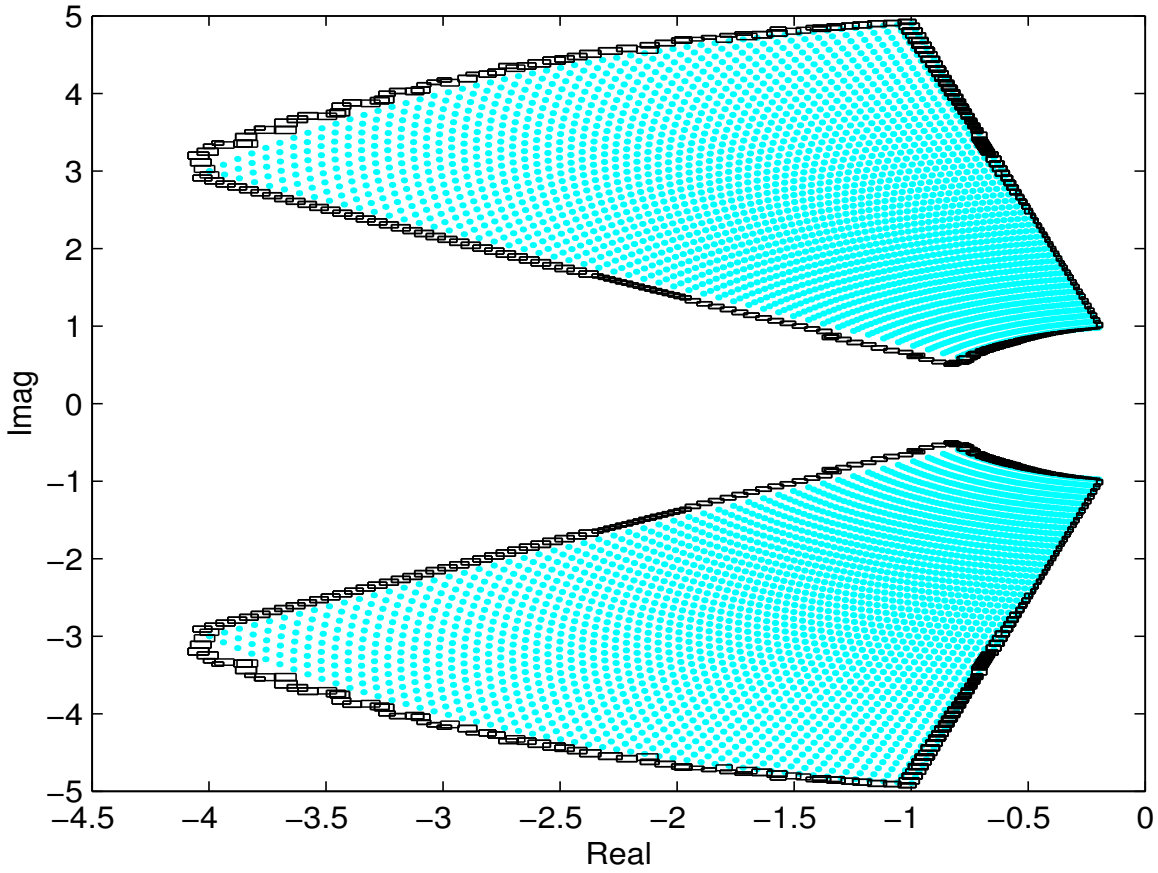

FIGURE 5.1. Spectral set of Example 5.1 computed using the proposed algorithm. Only the outer boundary boxes of the computed set are shown. For comparison, the analytically found spectral set of 2500 fixed polynomials picked randomly from the polynomial family are also plotted. These are the inner points shown as light shaded area in the plot.

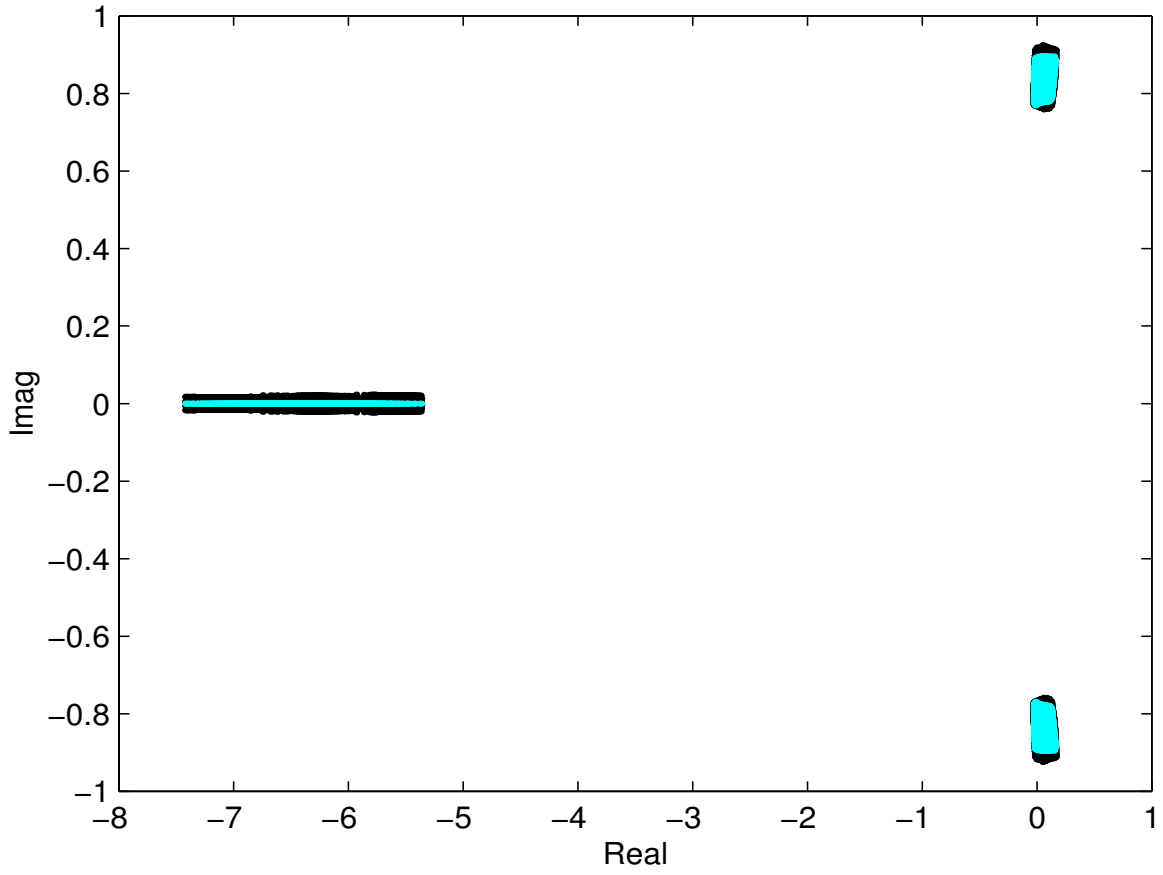

FIGURE 5.2. Spectral set of Example 5.2 computed using the proposed algorithm. The outer boundary of the plots show the midpoints of the solution boxes generated with the proposed algorithm. For comparison, the spectral set computed using *roots* routine of MATLAB over 1331 randomly chosen fixed polynomials is also shown. These are the inner points shown as light shaded area in the plot.

6

Limit cycles

6.1 Introduction

For uncertain nonlinear systems, the popular describing function approach [3], [26] can be used to compute the so-called limit cycle *locus*. The limit cycle locus for a given uncertain parameter is the locus of the limit cycle points as the parameter varies over its given range. The uncertain parameter may belong to the linear or nonlinear element in the system. Indeed, the concept of limit cycle locus is inspired by the well known tool of root locus [21] in linear control system analysis and design.

The limit cycle locus can be useful in a variety of situations in nonlinear system analysis and design, for instance, in

1. obtaining graphical insight: The limit cycle locus graphically illustrates the variation of a system's limit cycle behavior as a function of each uncertain parameter. This information is useful in identifying those uncertain parameters of the system which have strong influence on the limit cycle behavior. Variations in such parameters can then be restricted according to the desired limit cycle behavior.

2. tuning a given controller: If we assume a fixed structure controller, such as an amplifier, lag, lead, or PID controller, to be present in the linear part of the system, then the effects of the various controller parameters on the limit cycle behavior can be observed by computing the limit cycle locus for each of them. The obtained locus can be subsequently used to tune the controller for obtaining a prescribed limit cycle behavior or to overcome an existing one.

Though the concept of limit cycle locus is evidently useful, to our knowledge it does not exist yet in the literature. Perhaps, it has been hitherto found difficult to construct and use

it. The conventional graphical technique [26] for finding the limit cycle locus becomes quite tedious when uncertainties are present in the system parameters, because the method requires plotting a number of polar plots and describing function curves, depending on the parameters involved and their ranges.

Next, consider the more general problem of computing limit cycle *sets* for uncertain nonlinear systems. In section 1.2.2, we saw that there is a lack of methods in the literature to readily compute limit cycle sets, for the large and important class of uncertain nonlinear systems comprising of

- a linear part represented by a *nonrational* transfer function with coefficients having *nonlinear* parametric dependencies, and
- a nonlinear part represented by a *nonrational* describing function with *nonlinear* parametric uncertainty structures.

Such nonlinear systems are commonly found, for instance, in chemical process control - in heat exchanger systems [11], and distillation columns [58], and in nuclear reactor control systems [32].

In this chapter, we propose two tools of limit cycle analysis applicable to the above described class of uncertain nonlinear control systems:

1. The tool of limit cycle *locus*, and
2. An algorithm to compute the limit cycle locus as well as the limit cycle set. The algorithm for computing the limit cycle locus is actually a special case of the one for computing the limit cycle set, and is obtained from the latter simply by restricting the number of uncertain parameters to those of interest, see Remark 6.1.

The proposed algorithm is developed using tools of interval analysis [48], and has several useful features:

1. The proposed algorithm is readily applicable to a very general class of uncertain nonlinear systems, with the only requirement being that the transfer function of the linear element and the describing function of the nonlinear element should be continuous in the parameters and continuously differentiable in the amplitude and frequency of the periodic input signal to the nonlinear element[1]. No rational approximation of any nonrational term is required in the algorithm.

[1] Subject to this requirement, the uncertain linear element can be a rational or nonrational transfer function having parametric dependencies such as, interval, affine linear, multilinear or nonlinear. The nonlinear element can also have a describing function that is nonrational with nonlinear parametric uncertainty structure, and be frequency dependent or independent, with or without memory.

2. The proposed algorithm guarantees that *all* limit cycles are found, see Theorem 5.13. This guarantee meets an important requirement in stability analysis and synthesis of nonlinear control systems. Existing techniques, on the other hand, are unable to provide this important guarantee.

3. The proposed algorithm guarantees that the limit cycles are computed to a *prescribed accuracy*, see Theorem 5.11. Existing techniques lack the ability to compute the limit cycles to a prescribed accuracy.

4. If no limit cycle point exists, then the proposed algorithm determines this fact with a mathematical and computational guarantee, see Lemma 6.1 below.

5. The proposed algorithm computes limit cycles values that are *reliable*[2], despite all kinds of computational errors, see Theorem 5.14. Existing techniques do not provide any guarantee on the reliability (i.e., trustworthiness) of the computed results in the face of various computational errors.

6. For a prescribed accuracy, the proposed algorithm computes the enclosures of the limit cycle points in a *finite* number of iterations. Moreover, an upper bound on the number of algorithmic iterations required is also computable, see Theorem 5.12.

The rest of this chapter is organized as follows. In section 6.2, we present an algorithm for computing the limit cycles for uncertain nonlinear systems. In section 6.3, we examine the theoretical and computational properties of the proposed algorithm. In section 6.4, we demonstrate the proposed algorithm on several challenging examples. In section 6.5, we give the conclusions of the chapter.

6.2 Proposed algorithm

Consider the closed loop system of Fig. 6.1, where $g(s, \mathbf{q}_g)$ denotes the transfer function of the linear element with parameter vector $\mathbf{q}_g$, $h(a, \omega, \mathbf{q}_h)$ denotes the describing function of the nonlinear element with parameter vector $\mathbf{q}_h$, and a, ω denote the amplitude and frequency of the periodic input signal to nonlinear element. Let $\mathbf{y} := (a, \omega) \in \Re^2$, $\mathbf{q} := (\mathbf{q}_g, \mathbf{q}_h) \in \Re^n$. Then, under certain assumptions [3], the nonlinear system exhibits a limit cycle if there exists a solution $\mathbf{y}^*$ to the characteristic equation $f(\mathbf{y}, \mathbf{q}) = \mathbf{0}$, where $f(\mathbf{y}, \mathbf{q})$ is defined as

$$\begin{aligned} f(\mathbf{y}, \mathbf{q}) &: = (f_{\mathrm{Re}}(\mathbf{y}, \mathbf{q}), f_{\mathrm{Im}}(\mathbf{y}, \mathbf{q})) \\ f_{\mathrm{Re}}(\mathbf{y}, \mathbf{q}) &: = \mathrm{Re}\{1 + h(a, \omega, \mathbf{q}_h) g(j\omega, \mathbf{q}_g)\}; f_{\mathrm{Im}}(\mathbf{y}, \mathbf{q}) := \mathrm{Im}\{1 + h(a, \omega, \mathbf{q}_h) g(j\omega, \mathbf{q}_g)\} \end{aligned} \tag{6.1}$$

[2]However, the proposed method will retain any errors in the limit cycle computations due to the approximate nature of the describing function method itself.

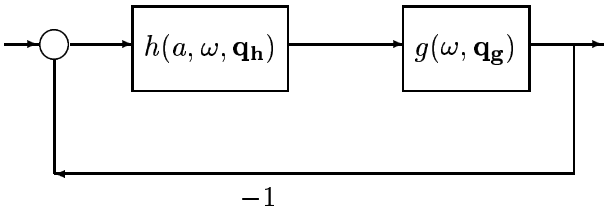

FIGURE 6.1. Uncertain nonlinear system

The solution $\mathbf{y}^*$ is called a limit cycle point, and the corresponding a^* and ω^* are called the limit cycle amplitude and frequency, respectively.

Now, suppose there is parametric uncertainty in the system such that the parameter vector $\mathbf{q}$ varies over a bounding box $\mathbf{Q}^0 \in I(\Re^n)$ given by

$$\mathbf{Q}^0 := \{\mathbf{q} \in \Re^n : \underline{q}_i \leq \mathbf{q}_i \leq \overline{q}_i, \text{ where } \underline{q}_i, \overline{q}_i \in \Re, \ i = 1, 2, ..., n\}$$

The parametric uncertainty gives rise to an uncertain nonlinear system. We address the problem of computing the set of limit cycle points for this system given by

$$\mathcal{L}_C\left(f, \mathbf{Q}^0\right) := \{\mathbf{y} \in \Re^2 : f(\mathbf{y}, \mathbf{q}) = 0, \text{for some } \mathbf{q} \in \mathbf{Q}^0\}$$

The following assumption is made throughout the work.

Assumption 6.1 *The transfer function g and the describing function h are continuous in $\mathbf{q}$ and continuously differentiable in $\mathbf{y}$. Subject to this assumption, the functions g and h can be described by any sequence of arithmetic expressions involving $\mathbf{y}$ and $\mathbf{q}$ using $+$, $-$, $*$, $/$, $\sqrt{}$, exp, log, power, trigonometric functions, inverse trigonometric functions, etc.*

6.2.1 Initial search box

In the proposed algorithm, we need to construct an initial search box $\mathbf{Y}^0$ that contains all (if any) limit cycle points. We can construct this box as follows. The amplitude a and frequency ω are nonnegative, so $\mathbf{Y}^0 \in I\left(\Re^{2+}\right)$. On a computer, we can set $\mathbf{Y}^0 \leftarrow [0, \text{real}_{\max}]^2$ where $\text{real}_{\max}$ is the largest machine representable number on the computer. Further, we sometimes know the ranges in which limit cycle amplitudes and frequencies occur in a particular problem. If so, we can bound $\mathbf{Y}^0$ to enclose these ranges. Finally, we can construct an initial search

box in which all limit cycle points are guaranteed to lie by adopting a procedure of Moore [48, chapter 6].

6.2.2 Algorithm

The proposed algorithm for the limit cycle computation is identical to the algorithm for spectral set computation given in the chapter 5, except for some changes in the termination part.

In the termination part, the obtained solution list $\mathcal{L}^{sol}$ is first examined. If $\mathcal{L}^{sol}$ is found empty, then the algorithm exits with the message – 'no limit cycle point exists'. Else, the limit cycle set $\mathcal{L}_C^{a\,\mathrm{lg}}$ is constructed as the union of all boxes $\mathbf{Y}_{(i)}$ present in $\mathcal{L}^{sol}$, and following [3, 26], the limit cycle stability for each solution box is assessed, and the obtained information is displayed. The algorithm now exits.

We next present the proposed algorithm.

Algorithm (Limit cycle computation algorithm)

Input: An expression for the function f in (6.1), the parameter box $\mathbf{Q}^0$, and the prescribed domain accuracy tolerance ε_x.

Output: The message "no limit cycle point exists", or the computed limit cycle set $\mathcal{L}_C^{a\,\mathrm{lg}}$.

BEGIN Algorithm

The algorithm arises from the one given in chapter 5, by modifying Step 8 and adding a new Step 9 as follows:

8. (**Termination part**) Determine non-existence of limit cycles or construct limit cycle set:

   (a) IF $\mathcal{L}^{sol} = \emptyset$ THEN print "no limit cycle point exists " and EXIT algorithm.

   (b) IF $\mathcal{L}^{sol} \neq \emptyset$ THEN Construct $\mathcal{L}_C^{a\,\mathrm{lg}} = \bigcup_{\mathcal{L}^{sol}} \mathbf{Y}_{(i)}$, and output $\mathcal{L}_C^{a\,\mathrm{lg}}$.

9. Determine stability of limit cycle boxes: for each $\mathbf{Y}_{(i)} \in \mathcal{L}^{sol}$, do the following: determine $\mathbf{Y}_{(i)}$ as a stable, unstable, or indeterminate box as per known methods for fixed parameter systems in [3, 26], display the stability information, and EXIT algorithm.

END Algorithm.

Remark 6.1 *When the limit cycle locus w.r.t. an uncertain parameter* $\mathbf{q}_i$ *is sought, then the parameter box* $\mathbf{Q}^0$ *reduces to the interval over which the parameter varies. The other parameters are kept at some fixed values, such as their nominal ones. However, in a more general situation we may want to compute the limit cycle locus when more than one uncertain parameter varies simultaneously - see Example 6.3 below that concerns controller tuning. To allow the proposed algorithm to compute the limit cycle locus in such situations, we consider a parameter* box *instead of a single parameter* interval *in the sequel.*

6.3 Properties

The convergence, termination, and reliability properties of the algorithm have already been proven in Chapter 5.

The following result justifies step 8a that determines the non-existence of a limit cycle point.

Lemma 6.1 *If $\mathcal{L}^{sol} = \emptyset$ in step 8a, then no limit cycle point exists.*

Proof. At any iteration, a part or the whole of a box $\mathbf{X}$ is discarded only in the interval zero exclusion test Step 4b or in the Generalized Krawczyk test Step 4c. By Lemma 5.1 and Theorem 5.2 (see also Remark 5.4), no limit cycle points are lost in the discarding process of these steps. Therefore, at any iteration, all (if any) limit cycle points are either in list $\mathcal{L}$ or $\mathcal{L}^{sol}$, but are never lost. When the algorithm reaches Step 8a, list $\mathcal{L}$ is empty, so all (if any) limit cycle points must now be in $\mathcal{L}^{sol}$. However, if $\mathcal{L}^{sol}$ is also empty at this step, then, clearly, no limit cycle point exists. ■

6.4 Illustrative examples

We implement the proposed algorithm using the interval arithmetic toolbox INTLAB [61] on a PC Pentium-III 850 MHz machine with 256 MB RAM. We first demonstrate the proposed algorithm for computing the limit cycle locus in section 6.4.1, and then for computing the more general limit cycle set in section 6.4.2.

6.4.1 Limit cycle locus

Example 6.1 *This example demonstrates the applicability of the proposed algorithm to nonrational transfer functions having general nonlinear parametric dependency, and nonlinearities of memory type. We emphasize that this example cannot be readily solved using existing techniques, due to the nonrational nature of the transfer function and nonlinear form of the parametric dependencies. The proposed algorithm readily solves this problem, and no rational approximation of any nonrational term in the transfer function is required.*

The linear element is a low pass nonrational transfer function having nonlinear parametric dependencies

$$g(s,\mathbf{q}) = \frac{(1+\sqrt{q_1 q_3})s)e^{-q_2 q_3 s}}{\ln(q_3)s^2 + 6\cos(\frac{q_3}{2.5} + q_1 q_2)s + 1}$$

where, $q_1 \in [0.1, 0.2]$, $q_2 \in [0.2, 0.3]$, $q_3 \in [13, 17]$.

The nonlinear element is of memory type in the form of a relay with hysteresis. The describing function of the nonlinear element is also nonrational and is given by

$$h(a,\omega,\mathbf{q}) = \frac{4q_4}{\pi a}\left(\sqrt{1-\left(\frac{q_5}{a}\right)^2} - j\left(\frac{q_5}{a}\right)\right) \tag{6.2}$$

where, the relay output $q_4 = \pm 1$ *and the hysterisis has uncertainty varying over interval* $q_5 \in [0.3, 0.4]$.

We choose the nominal parameter values as $q_1 = 0.15,\ q_2 = 0.25, q_3 = 15,\ q_4 = 1,\ q_5 = 0.35$, the initial search box for limit cycle points as $\mathbf{Y}^0 = ((0.3, 10], [0.1, 10])$, and set the prescribed domain accuracy tolerance to $\varepsilon_x = 0.001$.

The proposed algorithm computes a set of $562, 3140, 11673,$ and 628 boxes enclosing the limit cycle loci. The algorithm takes $33, 36, 38,$ and 33 iterations and approximately $155, 124, 374,$ and 57 seconds for the parameters q_1, q_2, q_3 and q_5, respectively. The results are shown in Fig. 6.2, along with the nominal limit cycle points shown as crossmarked circles.

For each parameter, a single branch of the limit cycle locus corresponding to the stable limit cycle points is observed. The frequency of the limit cycle increases with the parameter q_1, but decreases with the parameters q_2 and q_5. Whereas, the limit cycle amplitude increases with each of the parameters q_1, q_2, and q_5. However, the limit cycle frequency and amplitude both first decrease and then increase with the parameter q_3.

Example 6.2 *This example demonstrates the ready applicability of the proposed algorithm to frequency dependent nonlinearities.*

The linear element is a low pass transfer function given by

$$g(s,\mathbf{q}) = \frac{q_1}{q_2 s^2 + q_3 s + q_4}, \quad q_1 \in [10, 20], q_2 \in [1, 2], q_3 \in [5, 10], q_4 \in [1, 2]$$

The nominal parameter values are taken as $q_1 = 15, q_2 = 1.5, q_3 = 7.5,$ *and* $q_4 = 1.5$. *The nonlinear element is a Clegg integrator having a frequency dependent describing function*

$$h\,(a,\omega,\mathbf{q}) = \frac{4}{\pi\omega}\left(1 - j\frac{\pi}{4}\right)$$

The Clegg integrator is a nonlinear integrator that can be used as a more efficient compensator than a linear integrator, see [26, pp 79 - 81]. Note that the describing function of a Clegg integrator is dependent on the frequency but not on the amplitude of the input signal.

We choose the initial search box for limit cycle points as $\mathbf{Y}^0 = ([1, 1], [0.1, 10])$, *and set the prescribed domain accuracy tolerance to* $\varepsilon_x = 0.001$.

Applying the proposed algorithm, we obtain the result that $\mathcal{L}^{sol} = \emptyset$. By Lemma 6.1, we have a *mathematical* guarantee that no limit cycle point exists in the considered search box. Since a MIA implementation of the proposed algorithm is used, by Theorem 5.14 we also obtain a *computational* guarantee of the same.

Example 6.3 *This example is selected from [24] and demonstrates how the proposed algorithm can be used to tune a controller for achieving a prescribed limit cycle behavior.*

The linear element comprises of the plant $g(s,\mathbf{q})$ *and the controller* $c(s,\mathbf{q})$ *given by*

$$g\,(s,\mathbf{q}) = \frac{q_1}{s^2+q_2 s+q_3}, \quad c(s,\mathbf{q}) = \frac{q_4}{q_5 s^2+q_6 s+1}$$

while the nonlinear element is saturation with output $\pm q_7$ *and unity relay gain* q_8*. The nonrational describing function of the nonlinear element is*

$$h(a,\omega,\mathbf{q}) = \frac{2q_8}{\pi}\left(\sin^{-1}\left(\frac{q_7}{a}\right)+\left(\frac{q_7}{a}\right)\sqrt{1-\left(\frac{q_7}{a}\right)^2}\right)$$

The parameter intervals are

$$\begin{aligned} q_1 &\in [0.1,10],\ q_2 \in [0.1,10], q_3 \in [0.1,10], q_4 \in [0,50], \\ q_5 &\in [0.005,0.015],\ q_6 \in [0.01,0.2], q_7 \in [0,2], q_8 = 1 \end{aligned}$$

We choose the nominal parameter values as

$$q_1 = 1, q_2 = 1.4, q_3 = 1, q_4 = 20, q_5 = 0.01, q_6 = 0.1, q_7 = 1\,,$$

the initial search box for limit cycle points as $\mathbf{Y}^0 = ((0,100],[0.01,100])$*, and set the prescribed domain accuracy tolerance to* $\varepsilon_x = 0.01$*.*

The proposed algorithm computes for each parameter, a set of approximately 1040 boxes enclosing the limit cycle locus, in about 34 iterations and 7 seconds. all limit cycles are found to be stable. The obtained results are plotted in Fig. 6.3.

We observe from this figure that the limit cycle disappears for

$$q_1 < 0.407, q_2 < 2.9327, q_4 < 9.375, q_6 < 0.034492, q_7 < 0.6172$$

when these parameter values are changed individually, with all other parameters fixed at their nominal values.

We next consider tuning of the controller in order to achieve a prescribed limit cycle behavior. We designate the following cases, see [24]:

- Case-1: the prescribed limit cycle amplitude and frequency are $(2.46, 3.16)$
- Case-2: the prescribed limit cycle amplitude and frequency are $(1.2, 6.2)$

We can readily apply the proposed algorithm and find the controller parameter values to achieve the prescribed limit cycle behavior for both cases. For this, in the algorithm we restrict the search box $\mathbf{Y}^0$ for limit cycle points to the prescribed limit cycle amplitude and frequency - that is, the search box $\mathbf{Y}^0$ now becomes a degenerate box, or a point. Further,

we also specify the parameter vector $\mathbf{q}$ as the controller parameter vector (q_4, q_5, q_6), and the parameter box $\mathbf{Q}^0$ as the initial search region in the controller parameter space. Accordingly, we take $\mathbf{Q}^0 = ([0, 50], [0.005, 0.015], [0.01, 0.2])$, and set $\mathbf{Y}^0$ equal to the prescribed limit cycle amplitude and frequency in each case. The proposed algorithm then gives the results shown in Fig. 6.4.

Fig. 6.4 shows that instead of single set of controller parameter values, in each case there is a *band* of controller parameter values that can achieve the prescribed limit cycle behavior. The designer can therefore choose the most appropriate combination of controller parameter values from this set, based on some specified criterion, such as minimum control effort. For comparison purposes, we also show in Fig. 6.4 the controller parameter values found by Ferreres and Fromion [24] using the μ - analysis approach. We see that the latter values are indeed enclosed within the bands generated by the proposed algorithm.

6.4.2 Limit cycle set

We next demonstrate the proposed algorithm to find the limit cycle set for Example 6.1. We stress that existing techniques are unable to readily compute the limit cycle set for this example, due to the nonrational nature of the transfer and describing functions and nonlinear form of parametric dependencies. The proposed algorithm readily solves this problem, without requiring any rational approximation of the nonrational terms.

Example 6.4 *: Consider the same uncertain nonlinear system given in Example 6.1. We now choose the initial search box for limit cycle points as* $\mathbf{Y}^0 = ((0.4, 100], [0.01, 100])$*, and set the prescribed domain accuracy tolerance to* $\varepsilon_x = 0.1$*.*

To compute the limit cycle set, all uncertain parameters are varied over their respective ranges in the proposed algorithm. The proposed algorithm then takes 26 iterations to compute the limit cycle set comprising of $14,997$ boxes The results are plotted in Fig. 6.5.

We next proceed to compare these results versus those obtained with the graphical method and nonlinear simulations. We pick randomly a few combinations of parameter values from the given parameter ranges, and designate them as follows.

Case 1: $q_1 = 0.1,\ q_2 = 0.2,\ q_3 = 13, q_4 = 1,\ q_5 = 0.3.$

Case 2: $q_1 = 0.15,\ q_2 = 0.25, q_3 = 15,\ q_4 = 1,\ q_5 = 0.35.$

Case 3: $q_1 = 0.2, q_2 = 0.3, q_3 = 17,\ q_4 = 1,\ q_5 = 0.4.$

For these fixed parameter cases, the proposed algorithm computes the limit cycle amplitude and frequencies shown in Table 6.1, column 3. The stability property of each limit cycle is also found by the algorithm, and is given in column 6 of the same Table.

Graphical method: As stated earlier in section 6.1, the graphical method [26] for finding the limit cycle points becomes rather tedious and unsuitable when the values of system

parameters vary over given ranges. However, for the few selected cases of fixed parameters given above, we can apply the graphical method without much difficulty. Accordingly, the limit cycle points are found graphically as the intersections of the polar plots of $g(j\omega, \mathbf{q}_g)$ and $-1/h(a, \omega, \mathbf{q}_h)$shown in Fig. 6.6 for the various cases, and reported in Table 6.1, column 4. In all cases, we find the results of the proposed algorithm to be nearly identical to those of the graphical method.

Nonlinear simulations: Closed loop nonlinear simulations for the various cases are performed using the SIMULINK toolbox of MATLAB [46], and the simulation results are plotted in Fig. 6.7.

From the figure, we record the limit cycle amplitudes and frequencies, and report them in Table 6.1, column 5. The Table shows some minor differences between the results of nonlinear simulations and the proposed algorithm. However, such minor differences are perhaps to be expected, due to the approximate nature of the describing function method itself.

6.5 Conclusions

We introduced the tool of limit cycle locus for uncertain nonlinear systems. Through several examples, we demonstrated its usefulness in gaining insight into the effect of each uncertain parameter on the limit cycle behavior of the system. We also showed through an example how the locus can be used to tune the parameters of a controller, to suppress an existing limit cycle or achieve a new prescribed one. Based on the experience gained in our studies, we believe that it may be worthwhile preparing charts of limit cycle locus for various sets of controller tuning, and supply them to plant operators.

We also proposed an interval analysis based algorithm to compute the set of limit cycle points for a very wide class of linear and nonlinear elements, including those represented by nonrational functions with interval, affine linear, multilinear, or nonlinear parametric dependencies, and nonlinearities that are memoryless, with memory, frequency independent, or frequency dependent. The proposed algorithm requires the characteristic function to be only continuous in the parameters and continuously differentiable in the amplitude and frequency of the periodic input signal to the nonlinear element.

The main features of the proposed algorithm are its provision of *guarantees* that the computed limit cycles are reliable and accurate, that all actual limit cycles are found, and that, for a prescribed accuracy, all limit cycles are found in a *finite* number of iterations. We successfully demonstrated the proposed algorithm to compute all limit cycles on a challenging nonrational example with nonlinear parametric dependencies. This example cannot be solved with any of the existing techniques.

Finally, it must be noted that any errors in the limit cycle locus results, due to the approximate nature of the describing function method itself, will remain.

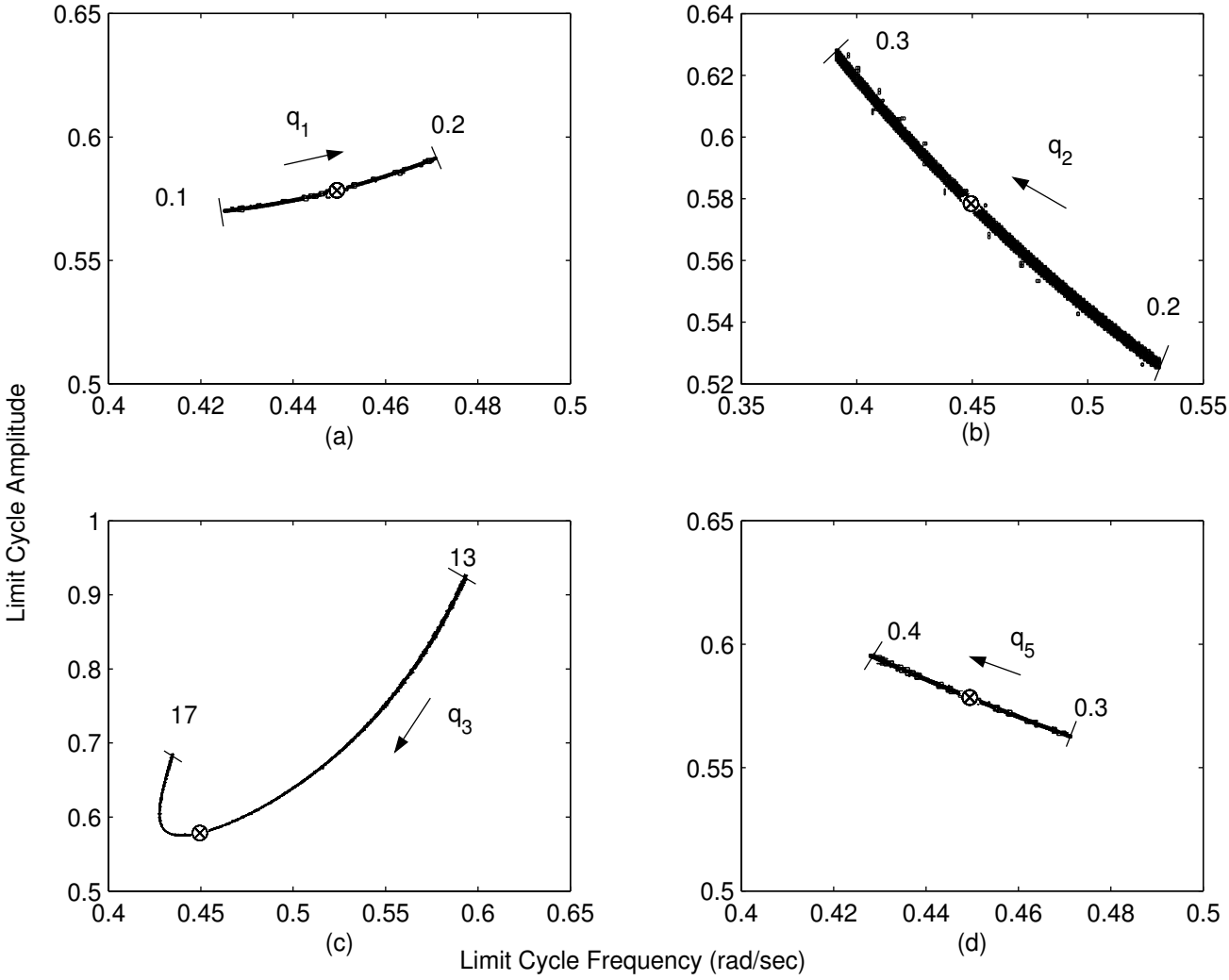

FIGURE 6.2. Limit cycle locus for (a) parameter q_1, (b) parameter q_2, (c) parameter q_3, and (d) parameter q_5 in Example 6.1. The nominal limit cycle is shown as cross marked circles.

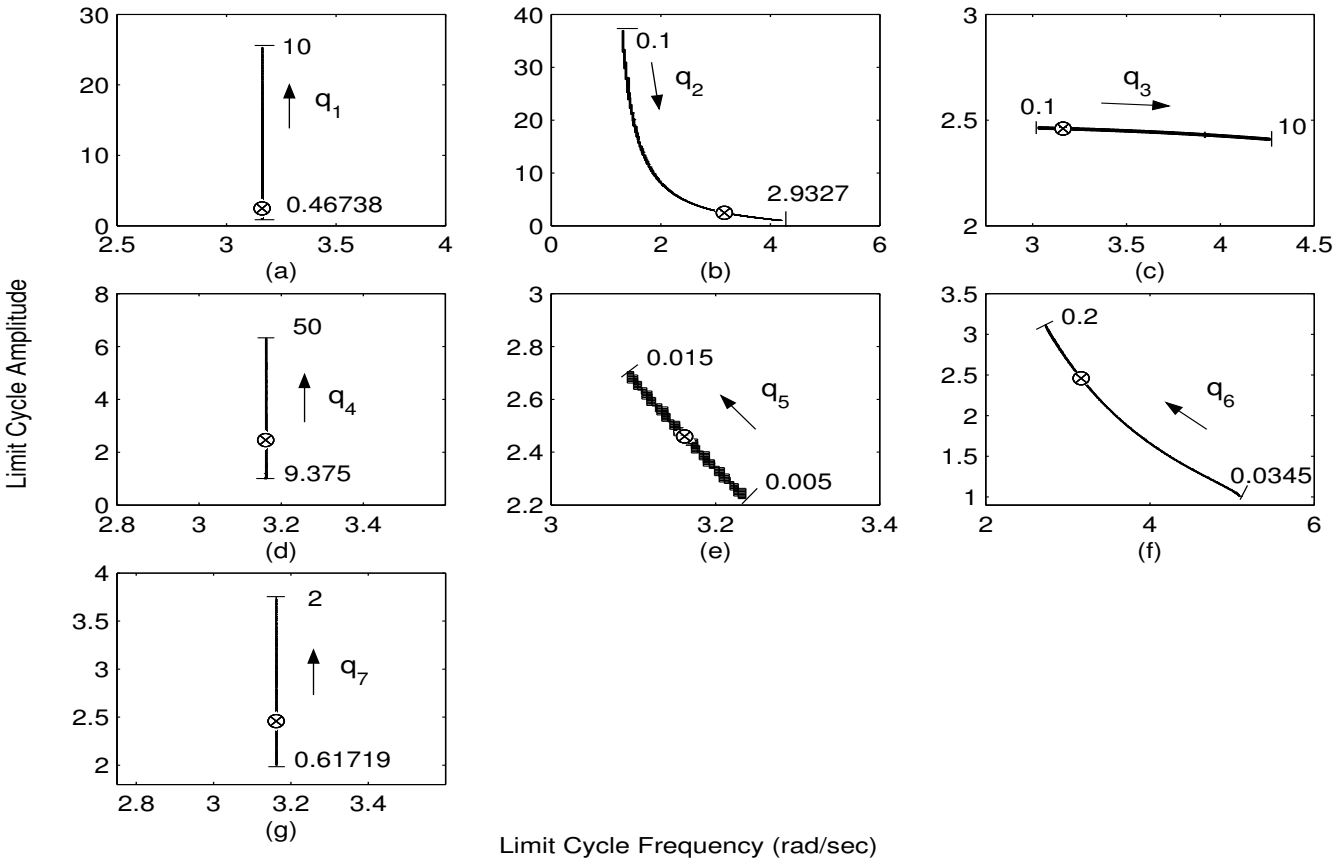

FIGURE 6.3. Limit cycle locus for (a) plant gain q_1, (b) plant parameter q_2, (c) plant parameter q_3, (d) controller gain q_4, (e) controller parameter q_5, (f) controller parameter q_6, and (g) saturation output q_7 in Example 6.3. The nominal limit cycle for Case-1 is shown as cross marked circle. All limit cycle points are found to be stable.

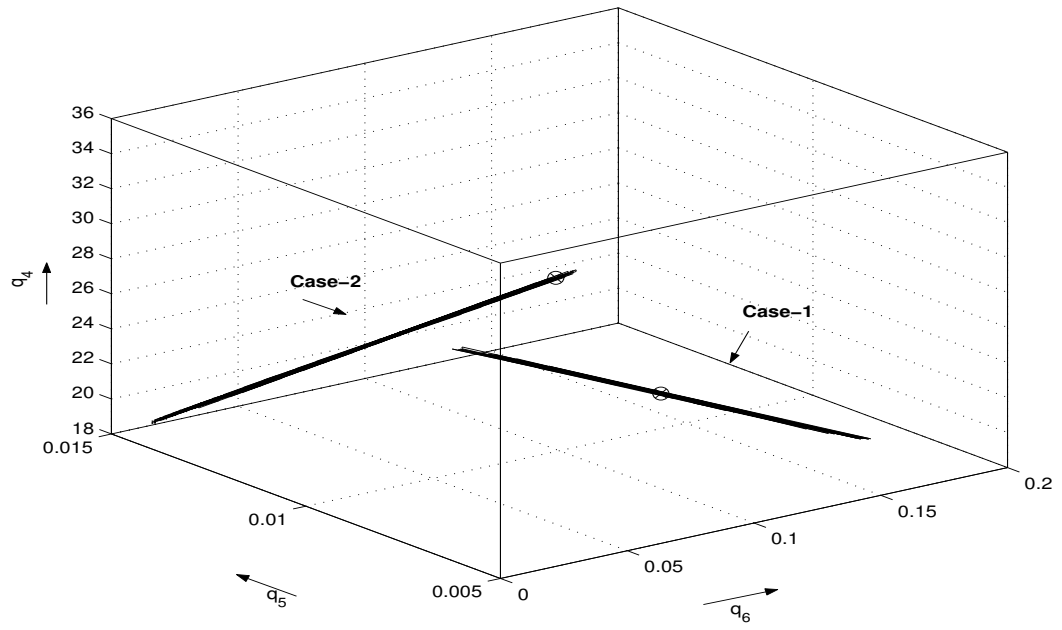

FIGURE 6.4. The band of controller parameter values generated by the proposed algorithm to achieve a prescribed limit cycle behavior in Example 6.3. Case-1 is for limit cycle amplitude and frequency of $(2.46, 3.16)$, while case-2 is for limit cycle amplitude and frequency of $(1.2, 6.2)$. The points marked with cross-circle are the controller parameter values given in [24] based on the μ- analysis method.

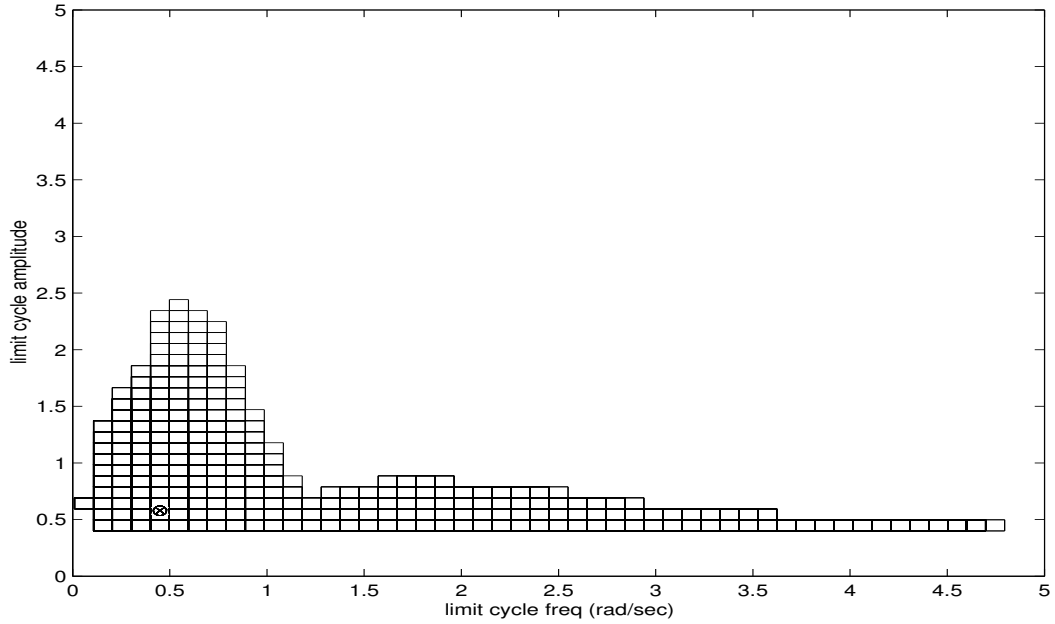

FIGURE 6.5. The limit cycle set computed using proposed algorithm in Example 6.4. The nominal limit cycle is shown as cross marked circle.

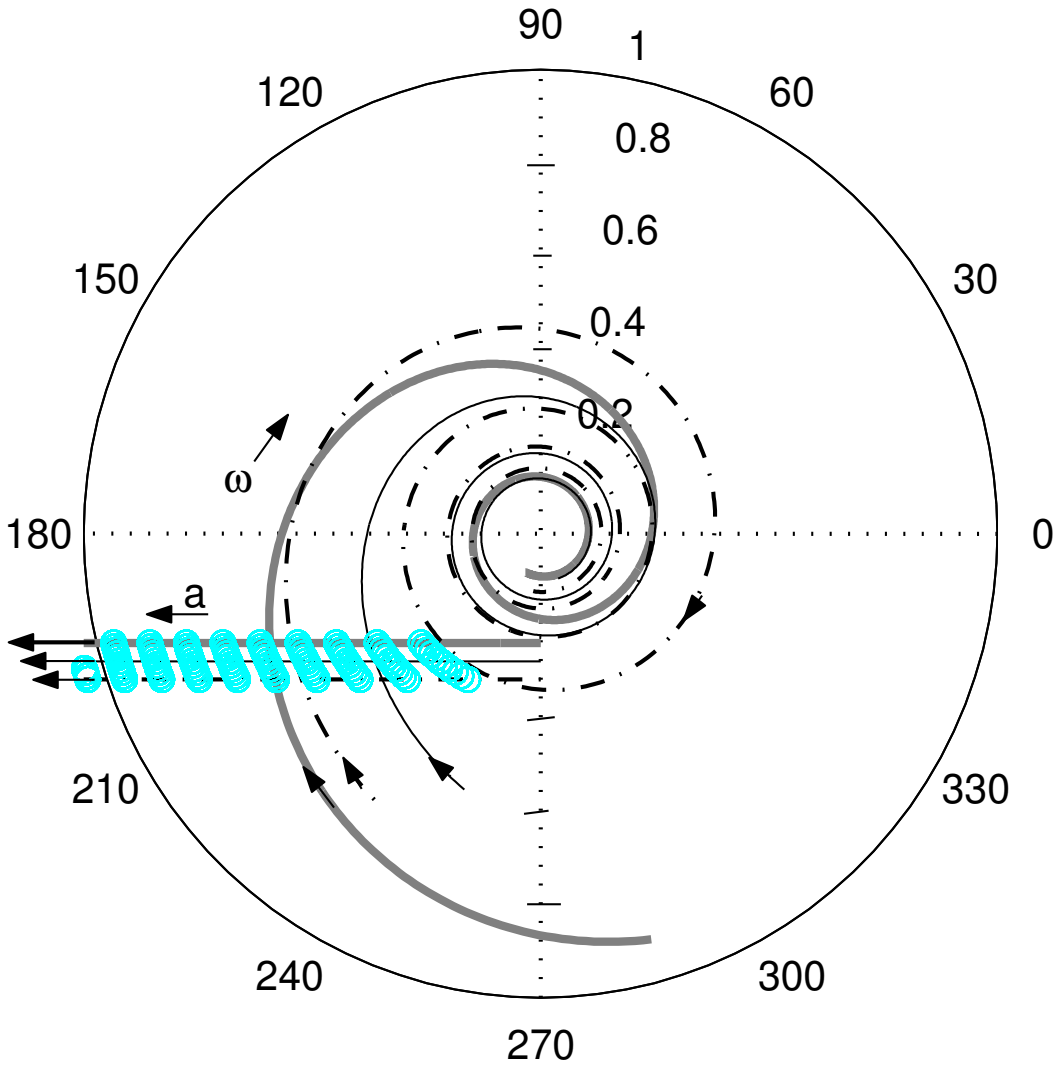

FIGURE 6.6. Polar plots of $g(j\omega, \mathbf{q}_g)$ shown as curves, and the describing function plots $-1/h\,(a, \omega, \mathbf{q}_h)$ shown as horizontal lines, for the three cases in Example 6.4. These plots are used to graphically obtain the limit cycle points for the three cases, serving as a cross-check for the results obtained with the proposed algorithm. The results of the proposed algorithm are shown by a region of circles plotted at the midpoints of the limit cycle boxes. (Case-1: thick solid line, Case-2: thin solid line, Case-3: dash-dot line).

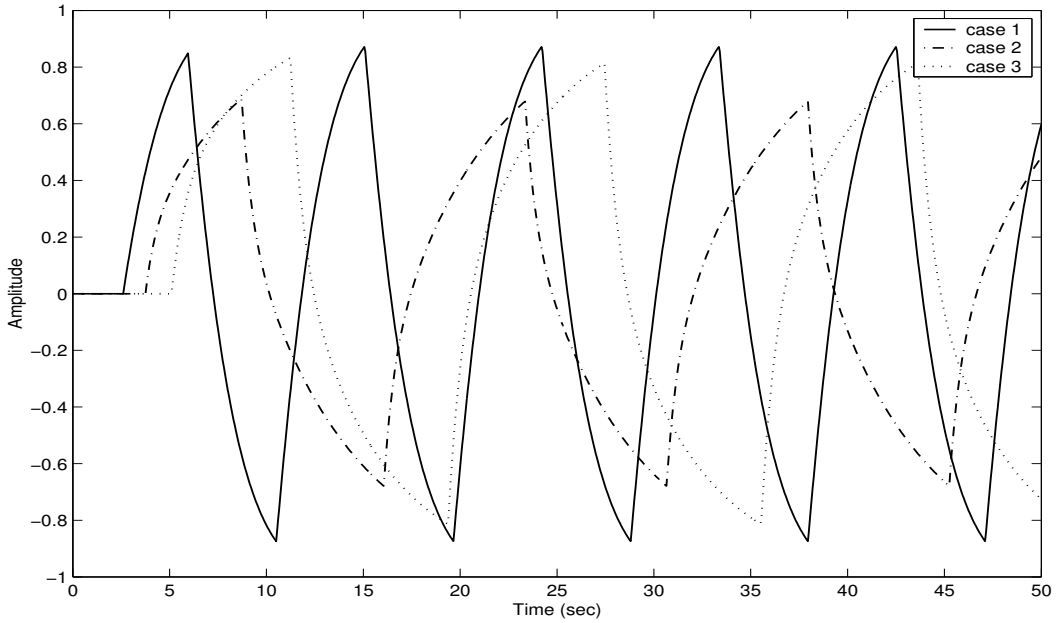

FIGURE 6.7. The limit cycle behaviour for three cases of the nonlinear system in Example 6.4 obtained by the closed loop simulation using SIMULINK.

TABLE 6.1. Comparison of limit cycle points obtained using three methods for three cases in Example 6.4.

Case	**Variable**	**Prop. Alg.**	**Graphical**	**Simulation**	**Property**
Case-1	Amplitude-1	$[0.8111, 0.8126]$	0.8113	0.8715	Stable
	Frequency-1	$[0.6766, 0.6777]$	0.6771	0.6867	
Case-2	Amplitude-1	[0.5781.0.5787]	0.5784	0.6793	Stable
	Frequency-1	$[0.4494, 0.4497]$	0.4494	0.4304	
Case-3	Amplitude-1	$[0.7760, 0.7774]$	0.7768	0.8135	Stable
	Frequency-1	$[0.3802, 0.3808]$	0.3805	0.3891	
	Amplitude-2	[0.4182.0.4184]	0.4183	–	Unstable
	Frequency-2	$[1.4124, 1.4128]$	1.4126	–	

7

Conclusions and future scope

We proposed reliable and accurate algorithms based on interval analysis of Moore [48] for the analysis of the linear and nonlinear, fixed and uncertain parameter systems. Specifically, we proposed algorithms for

1. computation of the Bode, Nyquist and Nichols frequency response plots for the class of nonrational transfer functions,
2. computation of gain and phase margins for the class of uncertain nonrational transfer functions with nonlinear parametric dependencies,
3. computation of the spectral set of uncertain polynomials with affine, multilinear, or nonlinear parametric dependencies, and
4. computation of limit cycles for uncertain nonlinear systems covering a large class of nonrational linear and nonlinear elements with nonlinear parametric dependencies. We also proposed a novel tool called *limit cycle locus* for the analysis of uncertain nonlinear systems.

The main properties of all the proposed algorithms are in terms of provision of various guarantees:

- guarantee that the computed results are reliable and accurate,
- guarantee that the computed solution set always encloses the actual solution set, without missing any of the solution points,
- guarantee that for a prescribed accuracy, the algorithms converges to the solution set in a finite number of iterations.

We proved theoretically all these properties for each proposed algorithm. We have also demonstrated the proposed algorithms on several real life examples. In most cases, these examples cannot be satisfactorily solved by currently available methods.

Some directions in which the present work can be extended are as follows.

1. Throughout this work, we used the natural inclusion function form to compute enclosures of the function ranges. However, various higher order convergence forms, such as mean value forms [48], Taylor forms [6], and Taylor-Bernstein forms [42] can instead be employed as inclusion function forms to obtain further improvements in the computational efficiency of the algorithms.

2. Throughout this work, we considered the parametric type of uncertainties. The work can be extended to address uncertain systems with the nonparametric type of uncertainties.

3. The work on frequency response plots can be extended to address linear systems with parametric uncertainties. One then obtains alternate algorithms for computation of the frequency response *envelopes* to the existing techniques, see [41].

4. The work on robust gain and phase margins can be extended to address the case of parametric stability margins, see [8].

5. The work on limit cycle analysis can be readily extended to address nonlinear systems with multiple nonlinear elements, see [7].

6. The performance of each proposed algorithm can be considerably improved by implementing it on parallel processing machines, or on application specific hardware systems [5], [9], [19].

Appendix I

Interval Analysis

A real interval $\mathbf{X}$ is a closed and bounded set $\mathbf{X} := [\underline{x}, \bar{x}] = \{x \in \Re : \underline{x} \leq x \leq \bar{x}\}$, where $\underline{x}$ and $\bar{x}$ are called the lower and upper endpoints of the interval $\mathbf{X}$. Two intervals are equal if their corresponding endpoints are equal. The intersection of two intervals $\mathbf{X}$ and $\mathbf{Y}$ is empty, if either $\underline{x} > \bar{y}$ or $\bar{x} < \underline{y}$. Else, the intersection of $\mathbf{X}$ and $\mathbf{Y}$ is again an interval $\mathbf{X} \cap \mathbf{Y} = [\max(\underline{x}, \underline{y}), \min(\bar{x}, \bar{y})]$. A set inclusion $\mathbf{X} \subseteq \mathbf{Y}$ is true only when $\underline{y} \leq \underline{x}$ and $\bar{y} \geq \bar{x}$.

An interval vector is a vector whose elements are intervals. Let n be the number of elements of the real vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Re^n$. Then, $\mathbf{X}$ denotes the $n-$ dim interval vector $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$. We use the term *box* as a synonym for an interval vector, and *rectangle* for a $2-$ dim interval vector.

Let $I(\mathbf{X})$ be the set of all boxes contained in $\mathbf{X}$. The width of an interval $\mathbf{X}$ is defined as $w(\mathbf{X}) = \bar{x} - \underline{x}$ if $\mathbf{X} \in I(\Re)$, and as $w(\mathbf{X}) = \max\{w(\mathbf{X}_1), \ldots, w(\mathbf{X}_n)\}$, if $\mathbf{X} \in I(\Re^n)$. The midpoint of an interval $\mathbf{X}$ is defined as $m(\mathbf{X}) = (\bar{x} + \underline{x})/2$ if $\mathbf{X} \in I(\Re)$, and as $m(\mathbf{X}) = \{m(\mathbf{X}_1), \ldots, m(\mathbf{X}_n)\}$, if $\mathbf{X} \in I(\Re^n)$. The relations $\in, =, \subset, \cap, \cup$ are all defined componentwise.

I.1 Natural inclusion functions and properties

Definition I.1 *Let $f : \Re^n \to \Re^m$ be a function defined over a box $\mathbf{X} \in I(\Re^n)$. Then, the range of f over $\mathbf{X}$ is denoted as $\bar{f}(\mathbf{X})$, i.e., $\bar{f}(\mathbf{X}) := \{f(\mathbf{x}) : \mathbf{x} \in \mathbf{X}\}$.*

Definition I.2 (Inclusion function) *We call a function $F : I(\mathbf{X}) \to I(\Re^m)$ an inclusion function for f, if $\bar{f}(\mathbf{Y}) \subseteq F(\mathbf{Y})$ for all $\mathbf{Y} \in I(\mathbf{X})$.*

Definition I.3 (Inclusion monotonicity) *An inclusion function F is said to be inclusion monotonic if $\mathbf{Z} \subseteq \mathbf{Y} \Rightarrow F(\mathbf{Z}) \subseteq F(\mathbf{Y})$ for all $\mathbf{Z}, \mathbf{Y} \in I(\mathbf{X})$.*

Definition I.4 (Natural inclusion function) *If f is a function computable as an expression, algorithm, or computer program involving four elementary arithmetic operations interspersed with evaluations of unary functions, then a natural inclusion function of f, whose value over a box $\mathbf{X}$ is denoted by $F(\mathbf{X})$, is obtained by the replacing each occurrence of each component x_i of $\mathbf{x}$ by the corresponding interval component $\mathbf{X}_i$ of $\mathbf{X}$, by executing all operations according to standard interval arithmetic rules, and by computing the exact ranges of all the unary functions.*

Remark I.1 *For instance, if $f(\mathbf{x}) = 1-5x_1+1/3x_2^2+5$, then $F(\mathbf{X}) = 1-5\mathbf{X}_1+1/3\mathbf{X}_2^2+5$ is a natural inclusion function of f on $\mathbf{X}$. As another example, if $f(\mathbf{x}) = x_1 \sin x_2 - x_3 \log x_2$, then $F(\mathbf{X}) = \mathbf{X}_1 * ISIN(\mathbf{X}_2) - \mathbf{X}_3 * ILOG(\mathbf{X}_2)$ is a natural inclusion function of f on $\mathbf{X}$, where $ISIN$ and $ILOG$ are the pre-declared interval* sin *and* log *functions in some programming language. A natural inclusion function is usually the simplest to construct inclusion function.*

Definition I.5 (Convergence order) *An inclusion function F for f is said to have a convergence order m if*

$$w(F(\mathbf{Y})) - w(\bar{f}(\mathbf{Y})) = \alpha w(\mathbf{Y})^m \tag{I.1}$$

for all $\mathbf{Y} \in I(\mathbf{X})$, where α and m are positive constants.

Some key properties of a natural inclusion function are given by Moore [47] :

Theorem I.1 (Inclusion property) $\bar{f}(\mathbf{X}) \subseteq F(\mathbf{X})$

Theorem I.2 (Inclusion monotonicity) *Natural inclusion functions are inclusion monotonic.*

Theorem I.3 *Natural inclusion functions have first order convergence.*

I.2 Generalized Krawczyk operator

Let $f : \Re^{l+n} \to \Re^m$ be a function defined over the box $\mathbf{X} = (\mathbf{Y}, \mathbf{Q})$, where $\mathbf{Y} \in I\left(\Re^l\right)$ is the box of unknowns and $\mathbf{Q} \in I(\Re^n)$ is the box of parameters. Suppose f is (Gateaux) differentiable with respect to $\mathbf{y}$, and let f'_y denote the derivative of f with respect to $\mathbf{y}$. Let F and F'_y denote respectively the natural inclusion functions of f and f'_y on $\mathbf{X}$. Let $\hat{\mathbf{y}} \in \mathbf{Y}$ and let $C \in \Re^{l \times l}$ be any nonsingular real matrix. Then, the Krawczyk operator introduced in [39] can be generalized to cover the parameter dependent case, as follows.

Definition I.6 *The Generalized Krawczyk operator $\mathcal{K}(\mathbf{X})$ is defined as*

$$\mathcal{K}(\mathbf{X}) := \hat{\mathbf{y}} - CF(\mathbf{X}) + \left\{I - CF'_y(\mathbf{X})\right\}(\mathbf{Y} - \hat{\mathbf{y}}) \tag{I.2}$$

Remark I.2 *The nonsingular matrix $C \in \Re^{l\times l}$ in above equation is used as a preconditioning matrix, so as to achieve better numerical properties using the Krawczyk operator. A typical choice is to set C as the inverse of the midpoint matrix of F'_y, i.e., $C = \left\{m\left(F'_y\right)\right\}^{-1}$. For details of preconditioning matrices, see [36].*

I.3 Interval topology

Definition I.7 (Hausdorff distance) *Let $\mathcal{A}, \mathcal{B}$ be compact, non-empty subsets of $\Re^n$ and $\mathbf{x} \in \Re^n$. Define $d_0(\mathbf{x}, \mathcal{B}) := \min_{b\in\mathcal{B}} \|\mathbf{x} - b\|_2$; $d_0(\mathcal{A}, \mathcal{B}) := \max_{a\in\mathcal{A}} d_0(a, \mathcal{B})$; $d(\mathcal{A}, \mathcal{B}) := \max\{d_0(\mathcal{A}, \mathcal{B}), d_0(\mathcal{B}, \mathcal{A})\}$.*

The Hausdorff-distance d is a metric for the sets of compact non-empty subsets of $\Re^n$. With this metric, a topology is defined, and convergence, etc., can be defined in the usual manner. In particular, the following definition can be made:

Definition I.8 *A sequence $\left\{\mathbf{X}_{(k)}\right\}_{k=1}^{\infty}$ of interval vectors $\mathbf{X}_{(k)} \in I\left(\Re^n\right)$ is said to converge to a point $\mathbf{x} \in \Re^n$, if $d(\mathbf{X}_{(k)}, \mathbf{x}) \to 0$ as $k \to \infty$.*

I.4 Complex interval arithmetic

The above concepts are readily extended to the field of complex numbers and complex intervals. For an introduction to complex interval arithmetic, the reader is referred to [38, 57].

References

[1] G. Alefeld and J. Herzberger. *Introduction to interval computations.* Academic Press, New York, 1983.

[2] T. C. Anthony, B. Wie, and S. Carroll. Pulse modulated control synthesis of a flexible spacecraft. *AIAA Journal of Guidance, Control, and Dynamics*, 13(6):1014–1022, 1990.

[3] D. P. Atherton. *Nonlinear control engineering.* Van Nostrand, 1975.

[4] B. R. Barmish and R. Tempo. On the spectral set for a family of polynomials. *IEEE Trans. on Automat. Control*, 36:111–115, 1991.

[5] S. Berner. Parallel methods for verified global optimization: practice and theroy. *Journal of Global Optimization*, 9:1–22, 1996.

[6] M. Berz and G. Hoffstatter. Computation and application of Taylor polynomials with interval remainder bounds. *Reliable Computing*, 4:83–97, 1998.

[7] V. Bhargava and D. N. Rao. Application of describing function technique to memoryless nonlinearities in tandem. *International Journal of Control*, 12:1–8, 1970.

[8] S. P. Bhattacharyya, H. Chapellat, and L. H. Keel. *Robust control - the parametric approach.* Prentice Hall, New York, 1995.

[9] A. Bik, M. Girkar, P. Grey, and X. Tian. Efficient exploitation of parallelism on Pentium III and Pentium IV processor based systems. *Intel Technology Journal*, pages 1–12, Q1-2001.

[10] H. W. Bode. *Network analysis and feedback design.* Van Nostrand, New York, 1945.

[11] P. S. Buckley. *Techniques of process control.* John Wiley and Sons, New York, 1964.

[12] V. Cerone. A fast technique for the generation of the spectral set of a polytope of polynomials. *Automatica*, 33(2):277–280, 1997.

[13] C. H. Chang and M. K. Chang. Analysis of gain margins and phase margins of a nonlinear reactor control system. *IEEE Transactions on Nuclear Science*, 41(4):1686–1691, 1994.

[14] Y. L. Chen and K. W. Han. Stability analysis of a nonlinear reactor control system. *IEEE Transactions on Nuclear Science*, NS-18:18–25, 1971.

[15] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* Prentice Hall of India, New Delhi, 2001.

[16] J. J. D'Azzo and C. H. Houpis. *Linear control system analysis and design.* McGraw-Hill, New York, 4th edition, 1995.

[17] R. R. E. DeGaston and M. G. Safanov. Exact calculation of the multiloop stability margin. *IEEE Trans. on Automat. Control*, 33:156–171, 1988.

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

[19] L. C. W. Dixon and M. Jha. Parallel algorithm for global optimization. *Journal of Optimization Theory and Applications*, 79:385–395, 1993.

[20] E. O. Doebelin. *Control system principles and design.* John Wiley and Sons, New York, 1985.

[21] W. R. Evans. *Control system dynamics.* McGraw Hill, New York, 1954.

[22] M. S. Fadali and N. Chachavalvoong. Describing function analysis of uncertain nonlinear systems using the Kharitonov approach. In *Proc. of American Control Conference*, pages 2908–2912, Seattle, 1995.

[23] M. S. Fadali, L. E. LaForge, and A. Sonbol. Linear time computation of robust stability margins. In *Proc. of American Control Conference*, Arlington VA, 2001.

[24] G. Ferreres and V. Fromion. Nonlinear analysis in the presence of parametric uncertainties. *International Journal of Control*, 69(5):695–716, 1998.

[25] M. Fu, S. Dasgupta, and V. Blondel. Robust stability under a class of nonlinear parametric perturbations. *IEEE Trans. on Automat. Control*, 40(2):213–223, 1995.

[26] A. Gelb and W. E. V. Velde. *Multiple-input describing functions and nonlinear system design.* McGraw-Hill, New York, 1968.

[27] A. Grace, A. J. Laub, J. N. Little, and C. M. Thompson. *Control system toolbox for use with MATLAB: user guide.* The Mathworks Inc., MA, USA, 2002.

[28] E. Hansen. *Global optimization using interval analysis.* Marcel Dekker, 1992.

[29] P. Henrici. *Applied and computational complex analysis vol-I.* Wiley International Edition, John Wiley and Sons, New York, 1974.

[30] I. M. Horowitz. *Synthesis of feedback systems.* Academic Press, New York, 1963.

[31] I. M. Horowitz. *Quantitative feedback design theory (QFT).* QFT Publications, Boulder, Colorado, 1993.

[32] C. L. Hu and K. W. Han. Stability analysis of a nuclear reactor control system with multiple transport lags. *IEEE Trans. on Nucl. Sci.*, NS-20(2):83–93, 1973.

[33] S. Impram and N. Munro. Describing function analysis of nonlinear systems with parametric uncertainties. In *Proc. of UKACC Int. Conf. on CONTROL 98*, pages 112–116, Swansea, UK, 1998.

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

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

[36] R. B. Kearfott. *Rigorous global search: continuous problems.* Kluwer Academic Publishers, Dordrecht, 1996.

[37] L. H. Keel and S. P. Bhattacharyya. Robust parametric classical control design. *IEEE Trans. on Automat. Control*, 39(7):1524–1530, 1994.

[38] R. Klatte, U. Kulisch, M. Neaga, D. Ratz, and C. Ullrich. *PASCAL-XSC language reference with examples.* Springer-Verlag, Berlin, 1993.

[39] R. Krawczyk. Newton algorithmen zur bestimmung von nullstellen mit fehler-schranken. *Computing*, 4:187–201, 1969.

[40] S. Kwon and J. T. Cain. Interval analysis application for stability margin computation of linear uncertain systems. In *Proc. of the Twentyeighth Southeastern Symposium on System Theory*, 1996.

[41] A. Levkovich, E. Zeheb, and N. Cohen. Frequency response envelopes of a family of uncertain continuous-time systems. *IEEE Trans. on Circuits and Systems - I, Fundamental Theory and Applications*, 42(3):155–165, 1995.

[42] Q. Lin and J. G. Rokne. Interval approximation of higher order to the ranges of functions. *Computers Math. Applic.*, 31(7):101–109, 1996.

[43] H. M. Mahon, O. D. Crisalle, H. Latchman, and K. H. Yen. A new method for computing robustness margins for real parametric uncertainties. In *Proc. of American Control Conference*, pages 2301–2303, 1998.

[44] S. Malan, M. Milanese, and M.Taragna. Robust analysis and design of control systems using interval arithmetic. *Automatica*, pages 1363–1372, 1997.

[45] M. Malek-Zavarei and M. Jamshidi. *Time delay systems: analysis, optimization and application.* North-Holland, Amsterdam, 1987.

[46] The MathWorks Inc. *MATLAB user guide, version 5.3.* MA, USA, 2000.

[47] R. E. Moore. *Interval analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

[48] R. E. Moore. *Methods and applications of interval analysis.* SIAM, Philadelphia, 1979.

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

[50] I. J. Nagarath and M. Gopal. *Control system engineering.* Wiley Eastern, New Delhi, 1995.

[51] P. S. V. Nataraj and S. Sheela. A new subdivision strategy for range computations. *Reliable Computing*, 8:1–10, 2002.

[52] A. Neumaier. *Interval methods for systems of equations.* Cambridge University Press, Cambridge, England, 1990.

[53] B. Newman. Dynamics and control of limit cycling motions in boosting rockets. *Journal of Guidance Control, and Dynamics*, 8(2):280–286, 1995.

[54] H. Nyquist. Regeneration theory. *Bell Syst. Tech. J.*, 11:126–147, 1932.

[55] K. Ogata. *Modern control engineering.* Prentice Hall of India, New Delhi, 1997.

[56] B. A. Ogunnike and W. H. Ray. *Process dynamics, modelling and control.* Oxford University Press, New York, 1994.

[57] M. S. Petkovic and L. D. Petkovic. *Complex interval arithmetic and its applications.* Wiley-VCH, Verlag, Berlin, 1998.

[58] O. Rademaker, J. E. Rijnsdorp, and A. Maarleveld. *Dynamics and control of continuous distillation units.* Elsevier, Amsterdam, 1975.

[59] H. Ratschek. Inclusion functions and global optimization. *Mathematical Programming*, 33:300–317, 1985.

[60] H. Ratschek and J. Rokne. *Computer methods for the range of functions.* Chichester: Ellis Horwood Limited, 1984.

[61] S. M. Rump. INTLAB - interval laboratory. In T. Csendes, editor, *Developments in reliable computing.* Kluwer Academic Publishers, Dordrecht, 1999.

[62] A. Sideris and R. S. S. Sanchez-Pena. Fast computation of the multivariable stability margin for real interrelated uncertain parameters. *IEEE Trans. on Automat. Control*, 34:1272–1276, 1989.

[63] D. D. Siljak. *Nonlinear systems: parametric analysis and design.* John Wiley and Sons, New York, 1969.

[64] J. E. Tierno. Describing function analysis in the presence of uncertainty. *AIAA Journal of Guidance, Control and Dynamics*, 20(5):956–961, 1997.

[65] A. Vicino, A. Tesi, and M. Milanese. Computation of nonconservative stability perturbation bounds for systems with nonlinearly correlated uncertainties. *IEEE Trans. on Automat. Control*, 35:835–841, 1990.

[66] B. H. Wilson, B. Eriylmaz, and B. Shafai. Improving control design for nonlinear parametric uncertainty. *International Journal of Control*, 66(6):863–883, 1997.

[67] S. K. Yang and C. L. Chen. Determining the root locations of systems with real parameter perturbations. *AIAA Journal of Guidance, Control and Dynamics*, 17(1):219–221, 1994.

Publications

1. P. S. V. Nataraj and J. J. Barve, *Generation of Nyquist plots to a prescribed accuracy*, Presented at IETE International Conference on Quality, Reliability and Control, Mumbai, India, December 2001.

2. P. S. V. Nataraj and J. J. Barve, *Reliable computation of gain and phase margins for nonrational transfer functions using interval arithmetic*, International Journal of Multimedia and Applications, To appear.

3. P. S. V. Nataraj and J. J. Barve, *Reliable computation of frequency response plots for nonrational transfer functions to prescribed accuracy*, Presented at SIAM Workshop on Validated Comuting, May 2002, Toronto, Canada.

4. P. S. V. Nataraj and J. J. Barve, *Reliable computation of frequency response plots for nonrational transfer functions to prescribed accuracy*, Reliable Computing, Under revision.

5. P. S. V. Nataraj and J. J. Barve, *Reliable and accurate algorithm to compute limit cycle locus for uncertain nonlinear systems*, Proc. IEE Pt. D: Control Theory and Applications, Submitted.

Acknowledgments

First of all, I thank The God Almighty – Lord Ganesha – for giving me an opportunity and ability to carry out this research work, for blessings me to successfully completing this work, and enabling me to contribute a drop in the sea of the scientific knowledge in this world. I truly believe that without His blessings nothing is possible in this world.

I am very thankful to my parents for their blessings, love and continuous inspiration. In fact, they sowed and nurtured the seeds of academic and research interest in me right from my childhood. They always made sacrifices and silently faced all the hardships in their life to give us the better facilities. I, very happily and emotionally bow to touch their feet and DEDICATE this thesis to THEM.

I am very much thankful to my supervisor Prof. P. S. V. Nataraj, who introduced me to the new area of robust control and interval analysis. He inspired me to carry out research by applying interval analysis in control system engineering during my M. Tech. studies, and thereby motivating me to pursue this research work after completion of my M.Tech. He has always been very helpful, friendly and patient. I really thank for the technical, and equally important moral and psychological support that he extended, whenever it required the most, particularly during some unavoidable short discouraging times, that typically comes during the research work while not getting the results as per expectations. Besides, I am also thankful to the family members of Prof. P. S. V. Nataraj for their affection, which always make me feel like their family member, rather than a student.

I am thankful also to respected Prof. K. P. Madhavan, Prof. R. D. Gudi, Prof. B. Bandyopadhyay, Prof. Ajit Verma, and Prof. Girish Kumar for thought provoking questions leading to useful research tips and directions during the seminars and discussions. My sincere thanks are due to Prof. D. P. Atherton for his valuable comments and criticism on the sixth chapter of my thesis, and also to unknown referees for their valuable comments and suggestions for

the publications based on the work in second chapter. Nontheless I am also grateful to Prof. S. Rump, for providing a fantastic software tool INTLAB, which was extensively used allowing me to concentrate more on the research work rather than solving the 'trivial' programming and syntax errors. Nontheless, I do remember and thank all my teachers who are responsible for expanding my knowledge and understanding during my school and college days.

I am also thankful to Dr. H. M. Desai, Vice Chancellor of D. D. Institute of Technology, who sponsored me for carrying out this research work externally, and also to Prof. H. B. Dave for readily and happily accepting to guide me, in case of requirements, as a co-supervisor. I am also thankful to Prof. R. K. Patel, the Head of Instrumentation and Control Engineering department at D. D. Institute of Technology for supporting me in sponsorship and during research work. I also acknowledge the support of my colleagues at D. D. Institute of Technology for their cooperation during my sponsorship and the research work. I am also thankful to my friends, particularly to Prakash, Sachin, Kubal, and Umapathy for providing a pleasant company and direct or indirect help during my frequent short stays at Indian Institute of Technology.

I am also thankful to Dr. Gautam Sardar, Pune for permitting me to continue my thesis writing work alongwith my duties after joining TCS in December 2002. Incidently, Dr. Gautam Sardar was also partly responsible for inspiring me to pursue research in the field of applications of interval analysis, being my college senior during my M. Tech. studies, and also a researcher in the same area. I also thank my colleagues in the Manufacturing Practice, TCS for their goodwill and cooperation.

I am very thankful to my wife Rashmi for her love, tolerance, and patience during the tough schedules of the research work. I also thank my lovely sons Bhavik and Nachiketa. Though they are too young to speak how they missed me during my busy schedules of the research work. I take this opportunity to express my gratitudes for their patience, tolerance and understanding.

I also thank my brother, sister and their family members for their love, affection and cooperation during my research work. I am also thankful to all the family members of my "greater" family, including my in-laws for their blessings, love, and tolerance during my research work.

Lastly, I sadly miss my Late Father-in-Law thanking him for his blessings. He was also very much interested in my higher studies, and I wished him to be alive on the earth, instead of being in the heaven, to see me complete this research work.

- Jayesh Barve