DISSERTATION

Algorithms for the Constrained Design of Digital Filters with Arbitrary Magnitude and Phase Responses

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

O.Univ.Prof. Dipl.-Ing. Dr.techn. W. F. G. Mecklenbräuker
Institut für Nachrichtentechnik und Hochfrequenztechnik

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik

von

Dipl.-Ing. Mathias Lang

Wien, im Juni 1999

Kurzfassung

In dieser Arbeit werden neue Algorithmen zum Entwurf digitaler Filter im Frequenzbereich vorgestellt. Es wird sowohl der Entwurf nichtrekursiver FIR (finite impulse response) Filter, als auch der Entwurf rekursiver IIR (infinite impulse response) Filter behandelt.

Im Gegensatz zu vielen Standardverfahren ermöglichen die hier entwickelten Methoden den Entwurf von Systemen, die beliebig spezifizierte Betrags- und Phasengänge approximieren. Derartige Filter können zur optimalen Entzerrung von Betrags- und Phasenverzerrungen eingesetzt werden. Probleme dieser Art treten bei Übertragungssystemen oder bei überabtastenden Analog-Digital-Wandlern auf. Eine weitere Anwendung der vorgestellten Verfahren ist der Entwurf von Filtern mit geringer Signalverzögerung. Im Fall von FIR Filtern kann eine starke Reduktion der Gruppenlaufzeit durch Verzicht auf exakte Linearphasigkeit zugunsten eines nur annähernd linearen Phasengangs in den Durchlaßbereichen erzielt werden. Auch IIR Filter können derart entworfen werden, daß ihr Phasengang in den Durchlaßbereichen annähernd linear ist. Die Gruppenlaufzeit dieser IIR Filter ist häufig wesentlich geringer als die Gruppenlaufzeit linearphasiger FIR Filter mit entsprechendem Dämpfungsverhalten.

Die hier präsentierten Algorithmen ermöglichen die Berücksichtigung von Entwurfsbedingungen, welche häufig bei praktischen Entwurfsproblemen auftreten. Das Erreichen einer vorgeschriebenen Mindestdämpfung oder das Einhalten einer maximalen Abweichung vom gewünschten Betrags- und Phasengang sind typische Bedingungen, die mit den vorgeschlagenen Methoden vorgeschrieben werden können. Beim Entwurf von IIR Filtern ist die Beschränkung der Polradien eine wichtige Nebenbedingung. Dadurch kann nicht nur die Stabilität der Filter garantiert werden, sondern es können auch unerwünschte Effekte bei der Implementierung der entworfenen Filter mit Festkommaarithmetik vermieden werden. Ein Entwurfsverfahren, das die Beschränkung der Polradien von IIR Filtern ermöglicht, wird in dieser Arbeit vorgestellt.

Während der Entwicklung der hier vorgestellten Entwurfstechniken wurde großer Wert auf Recheneffizienz gelegt. Dies und die Tatsache, daß diese Arbeit zu allen vorgeschlagenen Verfahren Matlabprogramme enthält, ermöglicht eine unmittelbare Anwendung der hier präsentierten Ergebnisse auf viele praktische Entwurfsprobleme.

Abstract

This thesis presents several new algorithms for designing digital filters subject to specifications in the frequency domain. Finite impulse response (FIR) as well as infinite impulse response (IIR) filter design problems are considered.

Unlike many standard filter design algorithms, all methods proposed here solve the problem of simultaneously approximating specified magnitude and phase responses. Filters of this type can be used to optimally equalize magnitude and phase distortions. Such problems occur e.g. in data transmission systems or in oversampled analog-to-digital conversion. Another application of the proposed algorithms is the design of filters with low group delay in the passbands. In the FIR case, the exactly linear phase property is given up in order to reduce the delay while maintaining a good approximation to phase linearity in the passbands. IIR filters can also be designed to have an approximately linear passband phase response. Their passband group delay is usually considerably smaller than the delay of linear phase FIR filters with equivalent magnitude responses.

An important feature of the algorithms presented in this thesis is that they allow for design constraints which often arise in practical filter design problems. Meeting a required minimum stopband attenuation or a maximum deviation from the desired magnitude and phase responses in the passbands are common design constraints that can be handled by the methods proposed here. For the design of IIR filters, an important constraint is the prescription of a maximum pole radius. This not only allows to guarantee stability but also helps to avoid undesirable effects when implementing the designed filter with fixed-point arithmetic. An algorithm solving this constrained IIR design problem is presented.

During the development of the proposed design techniques special emphasis has been placed on their computational efficiency. This and the fact that this thesis includes Matlab programs implementing all proposed algorithms makes all results of this dissertation directly applicable to many practical design problems.

Acknowledgments

It is my pleasure to thank my advisor Professor W. F. G. Mecklenbräuker for his interest and support throughout the development of this thesis. I also like to thank my second advisor Professor H. W. Schüßler for many inspiring discussions and for constructive comments improving the presentation of the material. I feel honored by his interest in my work.

I especially like to thank Dr. G. Doblinger for employing me during my Ph.D. studies and for his advice and support.

Many thanks to Ivan Selesnick for his continuing interest in my work, for many useful remarks, and for his friendship.

Thanks to all my colleagues at the Institut für Nachrichtentechnik und Hochfrequenztechnik at Vienna University of Technology for their support and for many entertaining hours.

I gratefully acknowledge the financial support by the Austrian Science Fund (Fonds zur Förderung der wissenschaftlichen Forschung).

Contents

Chapter 1

Introduction

Digital filters are integral parts of many digital signal processing systems, including control systems, systems for audio and video processing, communication systems, and systems for medical applications. Due to the increasing number of applications involving digital signal processing and digital filtering, the variety of requirements that have to be met by digital filters has increased as well. Consequently, there is a need for flexible techniques that can design digital filters satisfying sophisticated specifications.

This thesis presents methods for the frequency domain design of digital filters. The proposed methods allow for specifications on the magnitude and phase of the filter's frequency response. The design problem is a complex approximation problem where a complex desired frequency response prescribing the desired magnitude and phase responses is approximated by the actual frequency response of the digital filter. We consider the design of finite impulse (FIR) and infinite impulse response (IIR) digital filters. In the case of FIR filters, we formulate the design problem using the complex least squares and complex Chebyshev criteria. By introducing peak constraints on magnitude and phase approximation errors, we arrive at problem formulations offering the possibility to trade mean squared approximation errors versus maximum approximation errors. Because of this flexibility, the problem formulation can be adapted to a broad range of applications. For IIR filters, we also present a design method that can take into account simultaneous specifications on magnitude and phase responses. The proposed method uses a complex least squares criterion and imposes a constraint on the pole radii of the filter's transfer function.

The main contribution of this thesis are the numerical techniques for solving the resulting optimization problems. The basic motivation for the development of these techniques was the interest in fast algorithms that allow the user to quickly compare different specifications and different design criteria in order to arrive at the most satisfying solution adapted to the application within a reasonable amount of time. We consider this to be an important issue since filter design specifications arising from practical problems are never uniquely determined. Therefore, the most appropriate type of design specification is not known until the usefulness of several design results obtained from different specifications and different error criteria has been evaluated. We understand filter design as an interactive process and believe that the algorithms presented in this thesis are an appropriate tool for finding optimal and practical solutions to a broad class of design problems.

The text contains several Matlab[1] programs that implement the algorithms developed in this thesis. In order to keep the programs relatively simple and short, the validity of input variables is not checked in most cases, and there is no special user interface. An effort has been made to keep the programs computationally efficient. In addition, we have tried to make them easy to use and yet flexible enough to solve a variety of practical design problems.

1.1 Notation and Definitions

This section introduces the notation and some definitions that will be used throughout this thesis. The design specifications are formulated in the frequency domain by choosing a complex desired frequency response $D\left(e^{j\omega}\right)$ which prescribes the desired magnitude and phase response. The complex function $D\left(e^{j\omega}\right)$ is defined on the domain of approximation Ω which is a subset of the interval $[0, 2\pi)$. In most cases, the domain Ω is the union of several disjoint frequency bands which are separated by transition bands where no desired response is specified. We denote the union of all passbands by Ω^p:

$$\Omega^p = \left\{\omega \in \Omega \middle| \ \left|D\left(e^{j\omega}\right)\right| > 0\right\}.$$

The union of all stopbands is denoted by Ω^s:

$$\Omega^s = \left\{\omega \in \Omega \middle| \ D\left(e^{j\omega}\right) = 0\right\}.$$

If the designed filter is to have real-valued coefficients, only the domain $\Omega \cap [0, \pi]$ is considered. In this case, the symmetry $D\left(e^{j\omega}\right) = D^*\left(e^{-j\omega}\right)$ is assumed implicitly. In the following chapters we will focus on the design of filters with real-valued coefficients, unless stated otherwise. It is, however, straightforward to extend the proposed methods to the design of filters with complex coefficients.

For formulating the design problems it is useful to define a complex error function by

$$E_c(\omega) = H\left(e^{j\omega}\right) - D\left(e^{j\omega}\right), \tag{1.1}$$

[1]Matlab is a trademark of *The MathWorks Inc.*

where $H\left(e^{j\omega}\right)$ is the actual frequency response of the filter. Often, a real-valued positive weighting function $W(\omega)$ is used, e.g. by considering a weighted error function $W(\omega)E_c(\omega)$, in order to give different weights on the approximation error in different frequency regions.

In Chapters 2 to 4 we will consider the design of FIR filters. In this case, the frequency response is given by

$$H\left(e^{j\omega}\right) = \sum_{n=0}^{N-1} h(n)e^{-jn\omega} \tag{1.2}$$

where N denotes the length of the impulse response $h(n)$. The degree of the FIR filter is $N-1$. For further discussions it will be advantageous to use vector/matrix notation. Define a column vector of FIR filter coefficients by

$$\boldsymbol{h} = [h(0), h(1), \ldots, h(N-1)]^T,$$

where the superscript ^{T} denotes the transpose operation. Furthermore, define a column vector of complex exponentials

$$\boldsymbol{e}(\omega) = [1, e^{j\omega}, e^{2j\omega}, \ldots, e^{(N-1)j\omega}]^T. \tag{1.3}$$

With these vectors the frequency response $H\left(e^{j\omega}\right)$ of an FIR filter given by (1.2) can be written as

$$H\left(e^{j\omega}\right) = \boldsymbol{e}^H(\omega)\boldsymbol{h}, \tag{1.4}$$

where the superscript ^{H} denotes conjugate transposition. We will use the notation $H\left(e^{j\omega}, \boldsymbol{h}\right)$ to explicitly express the dependence of $H\left(e^{j\omega}\right)$ on the coefficients $\boldsymbol{h}$ whenever necessary. Likewise, we will write the complex error function as $E_c(\omega, \boldsymbol{h})$ whenever its dependence on the filter coefficients is to be emphasized. Note the linear dependence of $H\left(e^{j\omega}, \boldsymbol{h}\right)$ and of $E_c(\omega, \boldsymbol{h})$ on the coefficients $\boldsymbol{h}$ in the FIR case. For this reason, FIR filter design problems are much simpler to solve than comparable IIR filter design problems. We will consider the design of IIR filters in Chapter 5.

1.2 Outline

Chapter 2 discusses the optimality criteria that we consider to be important for digital filter design. All criteria are formulated for the complex approximation case. The complex least squares and Chebyshev criteria are discussed in some detail because they are the basis for the more advanced criteria that have become popular recently. A survey of algorithms for the design of FIR filters completes the discussion of standard criteria. Matlab programs implementing some important algorithms are provided. The subsequent discussion of constrained complex design criteria shows that these criteria are more flexible than the standard criteria. The corresponding solutions of filter design problems are more practical than pure least squares or Chebyshev approximations. We define several frequency domain error functions that can be used to formulate the design problems, and we discuss the mathematical properties of the resulting optimization problems for the FIR case. Chapter 2 provides the background for the following detailed discussion of algorithms solving constrained complex filter design problems.

Chapter 3 presents new approaches to the constrained design of FIR filters with arbitrary magnitude and phase responses. Motivated by the discussion in Chapter 2, we consider constraints on the complex error function, and independent constraints on magnitude and phase errors. Section 3.1 shows how these design problems can be approximated by linear and quadratic programming problems, and analyzes the errors resulting from these approximations. In Section 3.2, we present new exchange algorithms solving the complex constrained least squares problems defined in Chapter 2. These algorithms are much faster and more memory efficient than previously published exchange methods. In Section 3.3, we propose a new iterative reweighted least squares (IRLS) method that extends Lawson's algorithm to constrained approximation problems. We apply this algorithm to least squares and Chebyshev problems with constraints on the complex error function. The advantage of this approach is its low memory requirement because no matrices are stored. We provide Matlab programs implementing all proposed design algorithms.

Chapter 4 gives several FIR filter design examples, some of which have been used in the literature before. We compare the results obtained by the methods proposed in this thesis to previously published results regarding computational efficiency, accuracy, and usefulness of the chosen design criterion. All filters given in this chapter have been computed by the Matlab programs published in this thesis.

Chapter 5 addresses the problem of designing causal stable IIR filters satisfying specifications on magnitudes and phases of their frequency responses. We use a complex weighted least squares design criterion. The proposed problem formulation includes a constraint on the pole radii of the designed transfer function. An arbitrary maximum pole radius can be specified. Incorporating this constraint solves the stability problem, avoids undesired behavior of the frequency response in transition bands, and makes it easier to implement the designed filters using fixed-point arithmetic. The proposed design algorithm modifies the Gauss-Newton method for nonlinear least squares problems to incorporate a constraint on the pole radii. The formulation of this constraint is based on Rouché's theorem. The subproblems of the modified Gauss-Newton method are solved by the multiple exchange algorithm presented in Chapter 3. We give several design examples comparing the results obtained by the proposed algorithm to those of other approaches. A Matlab program implementing the algorithm developed in this chapter is provided.

Chapter 6 concludes this thesis and discusses some related problems that remain open.

Appendix A reviews some basic results of optimization theory focusing on convex problems. The discussion covers Lagrange multipliers, Kuhn-Tucker conditions, convexity, and the concept of duality in mathematical programming. We conclude with a brief introduction to algorithms solving linearly constrained optimization problems. A Matlab program solving the subproblems occurring in the multiple exchange method presented in Chapter 3 is provided.

Appendix B discusses the use of Levinson's algorithm for solving Toeplitz systems of linear equations and for explicitly computing the inverse of a Hermitian Toeplitz matrix. These problems occur in the algorithms presented in Chapter 3. Matlab programs implementing versions of Levinson's algorithm for the problems of interest are provided.

Appendix C provides a proof of Rouché's theorem which is used in Chapter 5. Furthermore, we give a proof of a sufficient condition on the coefficients of a polynomial such that the moduli of its roots do not exceed a specified value.

Chapter 2

Design Criteria

2.1 Complex Least-Squares Approximation

Least-Squares (LS) optimality is the criterion of choice for many practical problems in engineering and applied mathematics. One reason for this is that minimizing error energy or error power is desired in many applications. Moreover, computing the solution to the LS problem is straightforward if the error depends linearly on the unknown parameters.

In the case of frequency domain FIR filter design, the error function given by (1.1) is a linear function with respect to the unknown filter coefficients $\boldsymbol{h}$. Hence, the LS frequency domain FIR filter design problem is a linear LS problem which can be solved by computing the solution to a system of linear equations.

2.1.1 Continuous Approximation

The error measure to be minimized is given by

$$\epsilon(\boldsymbol{h}) = \int_{\Omega} W(\omega) \left| E_c(\omega, \boldsymbol{h}) \right|^2 d\omega, \tag{2.1}$$

where $W(\omega)$ is a positive weighting function and $E_c(\omega)$ is the complex error function defined by (1.1). Using (1.1) and (1.4) we get

$$\begin{aligned} \epsilon(\boldsymbol{h}) &= \int_{\Omega} W(\omega) \left| H\left(e^{j\omega}, \boldsymbol{h}\right) - D\left(e^{j\omega}\right) \right|^2 d\omega \\ &= \int_{\Omega} W(\omega) \left| \boldsymbol{e}^H(\omega)\boldsymbol{h} - D\left(e^{j\omega}\right) \right|^2 d\omega \\ &= \int_{\Omega} W(\omega) \left\{ \boldsymbol{h}^H \boldsymbol{e}(\omega)\boldsymbol{e}^H(\omega)\boldsymbol{h} - 2\mathrm{Re}\left(D^*(e^{j\omega})\boldsymbol{e}^H(\omega)\boldsymbol{h} \right) + \left| D\left(e^{j\omega}\right) \right|^2 \right\} d\omega \\ &= \boldsymbol{h}^H \boldsymbol{R}\boldsymbol{h} - \boldsymbol{s}^H \boldsymbol{h} - \boldsymbol{h}^H \boldsymbol{s} + t, \end{aligned} \tag{2.2}$$

where we used the abbreviations

$$\begin{aligned} \boldsymbol{R} &= \int_{\Omega} W(\omega)\boldsymbol{e}(\omega)\boldsymbol{e}^{H}(\omega)d\omega, \\ \boldsymbol{s} &= \int_{\Omega} W(\omega)D\left(e^{j\omega}\right)\boldsymbol{e}(\omega)d\omega, \\ t &= \int_{\Omega} W(\omega)\left|D\left(e^{j\omega}\right)\right|^{2}d\omega. \end{aligned} \tag{2.3}$$

Since

$$\boldsymbol{h}^{H}\boldsymbol{R}\boldsymbol{h} = \int_{\Omega} W(\omega)\left|H\left(e^{j\omega},\boldsymbol{h}\right)\right|^{2}d\omega > 0 \quad \text{for } \boldsymbol{h} \neq \boldsymbol{0} \text{ and } W(\omega) \not\equiv 0, \quad \omega \in \Omega,$$

the matrix $\boldsymbol{R}$ is positive definite and the error measure $\epsilon(\boldsymbol{h})$ is a convex function with respect to the filter coefficients $\boldsymbol{h}$ and has one unique minimum. From (2.2) the error measure $\epsilon(\boldsymbol{h})$ can be written as

$$\epsilon(\boldsymbol{h}) = \left(\boldsymbol{h} - \boldsymbol{R}^{-1}\boldsymbol{s}\right)^{H}\boldsymbol{R}\left(\boldsymbol{h} - \boldsymbol{R}^{-1}\boldsymbol{s}\right) - \boldsymbol{s}^{H}\boldsymbol{R}^{-1}\boldsymbol{s} + t. \tag{2.4}$$

Obviously, the error measure is minimized by choosing the coefficients

$$\boldsymbol{h} = \boldsymbol{R}^{-1}\boldsymbol{s}. \tag{2.5}$$

Hence, the solution to the complex LS approximation problem can be computed by solving a system of linear equations. Moreover, this can be done efficiently exploiting the Hermitian Toeplitz structure of $\boldsymbol{R}$ (see Appendix B). If $D\left(e^{j\omega}\right) = D^{*}(e^{-j\omega})$ and $W(\omega) = W(-\omega)$ is satisfied on $[-\pi,\pi)$ then $\boldsymbol{R}$ and $\boldsymbol{s}$ are real-valued and consequently the optimum solution $\boldsymbol{h}$ is also real-valued.

Figure 2.1 shows the magnitudes of the frequency responses of two real-valued low pass filters of length 61 that optimally approximate

$$D\left(e^{j\omega}\right) = \begin{cases} e^{-j\tau\omega}, & 0 \leq \omega \leq 0.2\pi \\ 0, & 0.3\pi \leq \omega \leq \pi \end{cases} \tag{2.6}$$

in a least squares sense. Note that we defined a transition band $[0.2\pi, 0.3\pi]$ where no desired response is specified in order to avoid Gibbs' phenomenon. The weighting function $W(\omega)$ was chosen as 1 in the passband and 100 in the stopband. The filters have been computed according to (2.5). The desired frequency response has a linear phase in the passband. The group delay value τ has been chosen differently for both designs. For the design shown by the dashed curve in Figure 2.1 a value $\tau = (N-1)/2 = 30$ samples has been chosen. This leads to an exactly linear phase filter with symmetric impulse response. The other design shown as the solid curve in Figure 2.1 optimally approximates $D\left(e^{j\omega}\right)$

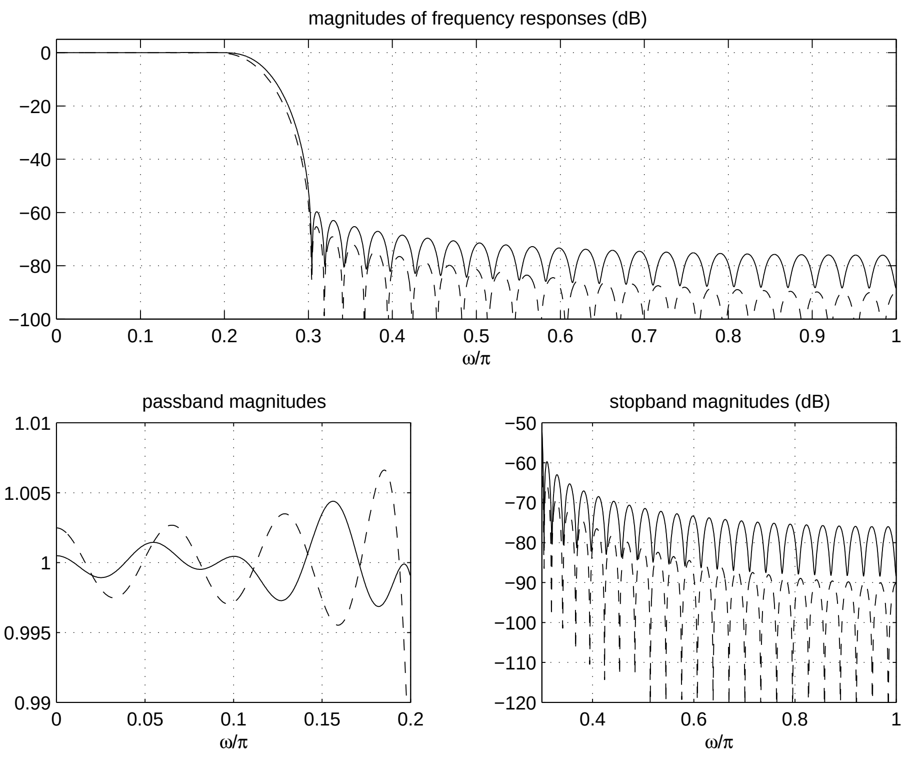


Figure 2.1: Complex least squares low pass filter designs (length $N = 61$). Solid curves: low delay filter (delay: 20 samples). Dashed curves: linear phase filter (delay: 30 samples).

in (2.6) with a reduced delay $\tau = 20$ samples. The phase error and the group delay of this filter are shown in Figure 2.2. Note that the approximation to the desired linear phase response is very good. The group delay error is smaller than 0.5 samples over more than 95% of the passband, although the desired group delay is not approximated explicitly in the complex approximation problem. Hence, a considerable reduction of delay can be achieved at the cost of a larger magnitude error while still retaining a good approximation to a linear phase response in the passband.

2.1.2 Discrete Approximation

There are several useful design specifications for which the integrals in (2.3) can be solved analytically. An important example is the case of multiband filters with piecewise constant $|D(e^{j\omega})|$, a piecewise constant weighting function $W(\omega)$, and piecewise linear desired phase responses. However, often these integrals cannot be solved analytically. In this case one either has to resort to numerical integration or formulate the problem a priori as a discrete approximation problem. The latter is also necessary if the desired response is only specified at certain frequency points. This will occur if the specifications have been derived from measurements.

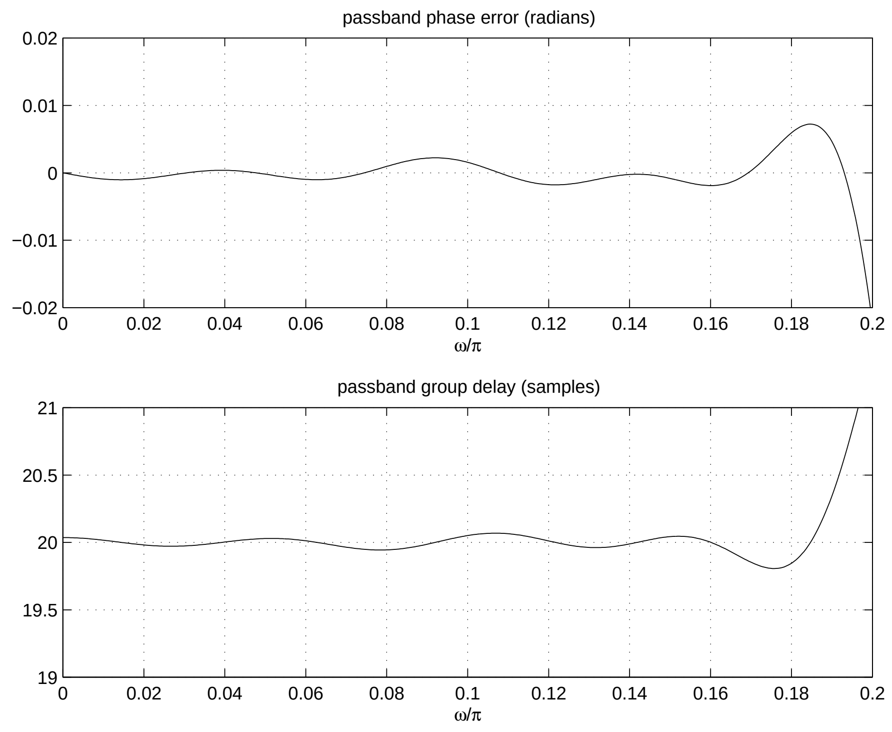


Figure 2.2: Complex least squares low pass filter design (length $N = 61$, desired delay: 20 samples). Upper plot: phase error. Lower plot: group delay.

The error measure for the discrete complex LS approximation problem is defined by

$$\epsilon(\boldsymbol{h}) = \sum_{i=0}^{L-1} W(\omega_i)\left|E_c(\omega_i, \boldsymbol{h})\right|^2. \tag{2.7}$$

If the discrete approximation problem is used as an approximation to a continuous problem, the frequency points ω_i should be distributed evenly over the domain of approximation and their number L should satisfy $L \geq 10N$, where N is the length of the filter to be designed. Minimizing $\epsilon(\boldsymbol{h})$ in (2.7) is equivalent to solving the overdetermined linear system

$$W^{1/2}(\omega_i)H\left(e^{j\omega_i}, \boldsymbol{h}\right) \approx W^{1/2}(\omega_i)D\left(e^{j\omega_i}\right), \quad i = 0, 1, \ldots, L-1,$$

in a least-squares sense. In matrix notation, this overdetermined system reads

$$\boldsymbol{W}^{1/2}\boldsymbol{C}^H\boldsymbol{h} \approx \boldsymbol{W}^{1/2}\boldsymbol{d}, \tag{2.8}$$

where $\boldsymbol{W}$ is an $L \times L$ diagonal matrix with elements $W(\omega_i)$ on its diagonal, $\boldsymbol{C}$ is an $N \times L$ complex matrix with columns $\boldsymbol{e}(\omega_i)$, $i = 0, 1, \ldots, L-1$, and $\boldsymbol{d}$ is a complex vector containing the elements $D\left(e^{j\omega_i}\right)$. The normal equations associated with the overdetermined linear system (2.8) are given by

$$\boldsymbol{C}\boldsymbol{W}\boldsymbol{C}^H\boldsymbol{h} = \boldsymbol{C}\boldsymbol{W}\boldsymbol{d}. \tag{2.9}$$

Hence, the solution to the discrete complex LS approximation problem is

$$\boldsymbol{h} = \left(\boldsymbol{C}\boldsymbol{W}\boldsymbol{C}^H\right)^{-1}\boldsymbol{C}\boldsymbol{W}\boldsymbol{d}. \tag{2.10}$$

If for every specified frequency point ω_m except for $\omega_m = 0$ there is another specified point $\omega_n = -\omega_m$ and if $D(e^{j\omega_m}) = D^*(e^{j\omega_n})$ and $W(\omega_m) = W(\omega_n)$ is satisfied, then the system matrix and the right hand side vector in (2.9) are real-valued and the optimum solution $\boldsymbol{h}$ is also real-valued.

Just like in the continuous approximation case, the normal equations (2.9) can be solved efficiently due to the Hermitian Toeplitz structure of the system matrix $\boldsymbol{C}\boldsymbol{W}\boldsymbol{C}^H$. However, from a numerical point of view it is better to solve the overdetermined system (2.8) using the QR-decomposition of the matrix $\boldsymbol{W}^{1/2}\boldsymbol{C}^H$ in (2.8) instead of solving the normal equations (2.9) [68]. This is due to the fact that the matrix condition number is squared by going from the original problem (2.8) to the normal equations (2.9). Hence, the accuracy of the solution computed via QR-decomposition in single precision is the same as the accuracy of the solution computed by solving the normal equations using double precision. On the other hand, from an efficiency point of view, solving the normal equations is preferable because the computational effort is smaller. This is even more pronounced if the Hermitian Toeplitz structure is exploited. Moreover, the memory requirement is much lower because the Hermitian Toeplitz system matrix of the normal equations is completely described by its first row.

A different approach to solving linear LS problems is to construct a set of polynomials that are orthonormal with respect to the desired weighting function $W(\omega)$ (see e. g. [103]). This approach has been proposed for the design of linear phase FIR filters in [86]. Using a set of orthonormal polynomials completely uncouples the system of linear equations, and hence solving this system is trivial. However, the computational burden is now shifted to the construction of the orthonormal set of polynomials. The advantage of this approach is that it is less prone to numerical difficulties. These difficulties may arise when designing filters of very high order ($N > 1000$). Numerical problems may also occur due to the specification of a transition band that is too wide for the specified filter order. In this case the matrices $\boldsymbol{R}$ in (2.5) and $\boldsymbol{C}\boldsymbol{W}\boldsymbol{C}^H$ in (2.10) become ill-conditioned. However, this mismatch between the desired response and the specified filter order is undesirable anyway, and the filter has to be redesigned to obtain a practical solution.

We present another design example in order to show that high-order filters can be designed by solving the normal equations if the corresponding specification is chosen carefully. This time we use the discrete least squares formulation and compute the

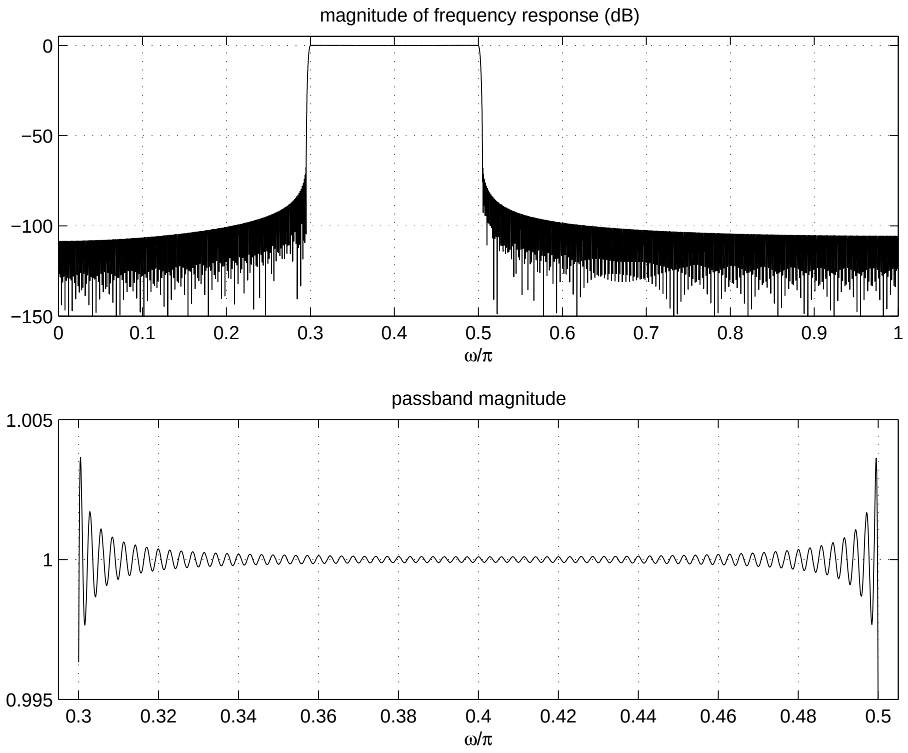


Figure 2.3: Least squares design of linear phase bandpass filter (length $N = 1401$).

optimum filter according to (2.10). The desired response is given by

$$D\left(e^{j\omega}\right) = \begin{cases} 0, & 0 \leq \omega \leq 0.295\pi \\ e^{-j700\omega}, & 0.3\pi \leq \omega \leq 0.5\pi \\ 0, & 0.505\pi \leq \omega \leq \pi \end{cases} \tag{2.11}$$

and the filter length is chosen to be $N = 1401$. The weighting function $W(\omega)$ is chosen to be 1 in the passband and 100 in the stopbands. The desired response $D\left(e^{j\omega}\right)$ is evaluated at 5000 points evenly distributed over the passband and the stopbands. Due to the desired group delay of 700 samples in (2.11) the optimum filter must have a linear phase response as is shown at the end of Section 2.1.3. The optimum filter coefficients are computed using the program `lslevin` given in Section 2.1.4. Figure 2.3 shows the magnitude of the frequency response of the designed filter. If considerable numerical errors had occurred during the design process, there would be artifacts in the frequency response that usually occur in the transition bands [86]. Note that least squares optimal linear phase FIR filters with piecewise linear desired magnitude responses can also be designed using the Matlab program `firls` contained in the Matlab Signal Processing Toolbox. However, this program does not take the special matrix structure into account. It stores the full matrix and solves the normal equations by Gaussian elimination. Hence, the computational effort and the memory requirements are very high if high-order filters are to be designed. Moreover, unlike `lslevin`, the program `firls` is restricted to the design of filters with exactly linear phase responses.

2.1.3 Independent Weighting of Magnitude and Phase Errors

The example given in Section 2.1.1 (Figures 2.1 and 2.2) shows that an almost linear phase response in the passband is possible even if the desired group delay is considerably smaller than the $(N-1)/2$ samples delay of an exactly linear phase filter. However, we have to accept a larger magnitude error that increases with decreasing desired group delay. Figure 2.2 shows that the phase approximation of the filter with reduced delay is very good. Hence, we might think of a way to trade magnitude error for phase error. This is not possible by directly approximating the complex desired frequency response.

Define the magnitude error $E_m(\omega, \boldsymbol{h})$ and the phase error $E_\phi(\omega, \boldsymbol{h})$ by

$$E_m(\omega, \boldsymbol{h}) = \left|H\left(e^{j\omega}, \boldsymbol{h}\right)\right| - \left|D\left(e^{j\omega}\right)\right| \tag{2.12}$$

and by

$$E_\phi(\omega, \boldsymbol{h}) = \arg\left\{H\left(e^{j\omega}, \boldsymbol{h}\right)\right\} - \arg\left\{D\left(e^{j\omega}\right)\right\}, \tag{2.13}$$

respectively. Let $\Omega^p = \{\omega \in \Omega |\ |D(e^{j\omega})| > 0\}$ and $\Omega^s = \{\omega \in \Omega |\ |D(e^{j\omega})| = 0\}$ denote the passbands and stopbands, respectively. The phase error $E_\phi(\omega)$ is only defined in the passbands. Trading magnitude versus phase errors could be achieved by using an error measure

$$\epsilon(\boldsymbol{h}) = \int_\Omega W_m(\omega)|E_m(\omega, \boldsymbol{h})|^2 d\omega + \int_{\Omega^p} W_\phi(\omega)|E_\phi(\omega, \boldsymbol{h})|^2 d\omega. \tag{2.14}$$

In frequency regions where the weighting functions satisfy $W_m(\omega) > W_\phi(\omega)$ the magnitude error is decreased at the cost of an increased phase error and vice versa. However, the magnitude as well as the phase error are nonlinear functions with respect to the filter coefficients $\boldsymbol{h}$, and hence minimizing $\epsilon(\boldsymbol{h})$ given by (2.14) results in a nonlinear LS problem. Solving this problem is by far more complicated than solving a linear LS problem and global optimality of the solution obtained by some iterative optimization procedure cannot be guaranteed. However, there is a good approximation to this nonlinear problem that requires only a slight modification of the original error measure defined in (2.1) and that retains the linearity of the problem. From (2.1) we have

$$\begin{aligned}
\epsilon(\boldsymbol{h}) &= \int_{\Omega^p} W(\omega)\left|E_c(\omega, \boldsymbol{h})\right|^2 d\omega + \int_{\Omega^s} W(\omega)\left|E_c(\omega, \boldsymbol{h})\right|^2 d\omega \qquad (2.15)\\
&= \int_{\Omega^p} W(\omega)\left|E_c(\omega, \boldsymbol{h})\right|^2 d\omega + \int_{\Omega^s} W(\omega)\left|H\left(e^{j\omega}, \boldsymbol{h}\right)\right|^2 d\omega\\
&= \epsilon_p(\boldsymbol{h}) + \epsilon_s(\boldsymbol{h}),
\end{aligned}$$

where $\epsilon_p(\boldsymbol{h})$ and $\epsilon_s(\boldsymbol{h})$ denote the passband and stopband contributions to the error $\epsilon(\boldsymbol{h})$, respectively. The passband error contribution $\epsilon_p(\boldsymbol{h})$ can be written as

$$\begin{aligned}\epsilon_p(\boldsymbol{h}) &= \int_{\Omega^p} W(\omega)\left|E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right|^2 d\omega \\ &= \int_{\Omega^p} W(\omega)\left\{\mathrm{Re}\left[E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right]\right\}^2 d\omega \\ &+ \int_{\Omega^p} W(\omega)\left\{\mathrm{Im}\left[E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right]\right\}^2 d\omega, \qquad (2.16)\end{aligned}$$

where $\phi_d(\omega) = \arg\left\{D\left(e^{j\omega}\right)\right\}$ is the desired phase response. Since $E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}$ can be written as

$$E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)} = H\left(e^{j\omega},\boldsymbol{h}\right)e^{-j\phi_d(\omega)} - \left|D\left(e^{j\omega}\right)\right|,$$

its real and imaginary parts are given by

$$\begin{aligned}\mathrm{Re}\left[E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right] &= \mathrm{Re}\left[H\left(e^{j\omega},\boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right] - \left|D\left(e^{j\omega}\right)\right|, \qquad (2.17)\\ \mathrm{Im}\left[E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right] &= \mathrm{Im}\left[H\left(e^{j\omega},\boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right]. \qquad (2.18)\end{aligned}$$

The magnitude and phase errors defined by (2.12) and (2.13) can be expressed as

$$\begin{aligned}E_m(\omega,\boldsymbol{h}) &= \mathrm{Re}\left[H\left(e^{j\omega},\boldsymbol{h}\right)e^{-j\phi_H(\omega)}\right] - \left|D\left(e^{j\omega}\right)\right|, \qquad (2.19)\\ E_\phi(\omega,\boldsymbol{h}) &= \arcsin\left\{\frac{\mathrm{Im}\left[H\left(e^{j\omega},\boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right]}{\left|H\left(e^{j\omega},\boldsymbol{h}\right)\right|}\right\}, \qquad (2.20)\end{aligned}$$

where $\phi_H(\omega) = \arg\left\{H\left(e^{j\omega}\right)\right\}$ is the phase of the actual frequency response. Comparing (2.17) and (2.18) with (2.19) and (2.20) and assuming small magnitude and phase errors leads to the approximations

$$\begin{aligned}E_m(\omega,\boldsymbol{h}) &\approx \mathrm{Re}\left[E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right] \qquad (2.21)\\ E_\phi(\omega,\boldsymbol{h}) &\approx \frac{1}{\left|D\left(e^{j\omega}\right)\right|}\mathrm{Im}\left[E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right]. \qquad (2.22)\end{aligned}$$

Note that these approximations are only defined in the passband because they make use of the desired phase response $\phi_d(\omega)$ which does not exist in the stopband. The approximations (2.21) and (2.22) are very close to the exact expressions if the magnitude and phase errors are small. From the passband error measure $\epsilon_p(\boldsymbol{h})$ given by (2.16) we see now immediately how to define a new passband error measure $\tilde{\epsilon_p}(\boldsymbol{h})$ that allows for independent weighting of magnitude and phase errors in the passband without introducing any nonlinear functions:

$$\begin{aligned}\tilde{\epsilon_p}(\boldsymbol{h}) &= \int_{\Omega^p} W_m(\omega)\left\{\mathrm{Re}\left[E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right]\right\}^2 d\omega \qquad (2.23)\\ &+ \int_{\Omega^p} \frac{W_\phi(\omega)}{\left|D\left(e^{j\omega}\right)\right|^2}\left\{\mathrm{Im}\left[E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right]\right\}^2 d\omega.\end{aligned}$$

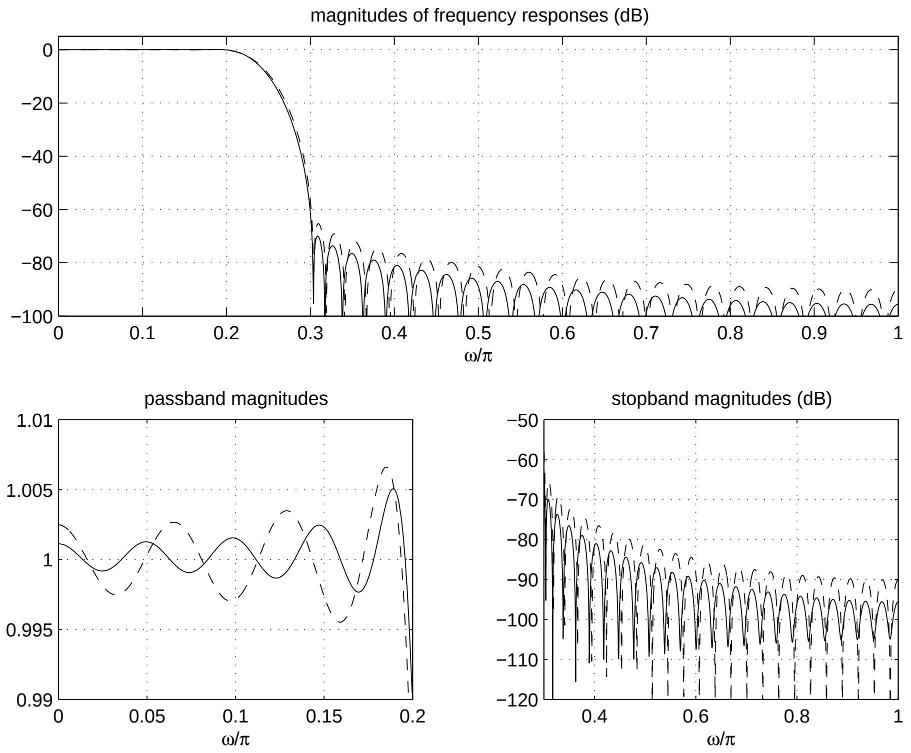


Figure 2.4: Complex least squares low pass filter designs with independent weighting of magnitude and phase errors (length $N = 61$). Solid curves: low delay filter (delay: 20 samples). Dashed curves: linear phase filter (delay: 30 samples).

The error measure to be minimized is then

$$\tilde{\epsilon}(\boldsymbol{h}) = \tilde{\epsilon}_p(\boldsymbol{h}) + \epsilon_s(\boldsymbol{h}). \tag{2.24}$$

Minimizing $\tilde{\epsilon}(\boldsymbol{h})$ is a linear LS problem as desired. The difference to the original complex LS problem is that the system matrix is no longer Toeplitz but exhibits a Toeplitz plus Hankel structure if magnitude and phase weightings are chosen differently. However, the Toeplitz plus Hankel structure can be exploited by specialized algorithms as well [84, 142]. The least squares problem with independent weighting of magnitude and phase errors can be formulated as a discrete approximation problem in a completely analogous manner.

We design another filter according to the specifications given by (2.6) with $\tau = 20$ samples. We improve the magnitude behavior by trading magnitude versus phase error using the error measure (2.24). The weighting functions are chosen to be piecewise constant. The passband magnitude weighting is 1, the stopband weighting is 100, and the phase weighting is 10^{-3}. The magnitudes of the frequency responses of the designed filter as well as of the exactly linear phase filter are shown in Figure 2.4. Unlike in the standard complex LS design case shown in Figure 2.1, the magnitude response of the low

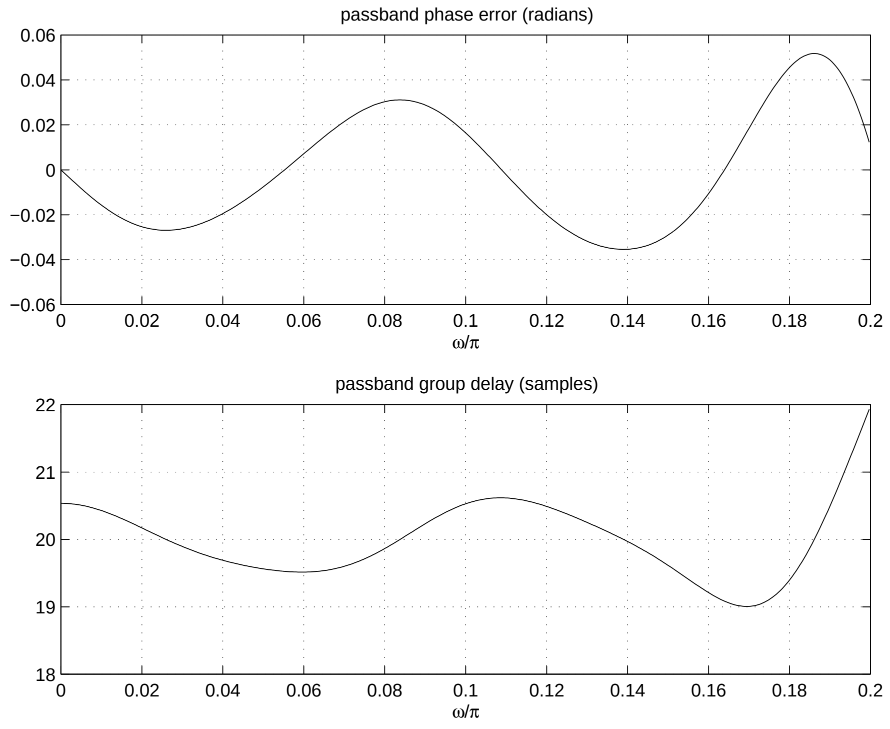


Figure 2.5: Complex least squares low pass filter design with independent weighting of magnitude and phase errors (length $N = 61$, desired delay: 20 samples). Upper plot: phase error. Lower plot: group delay.

delay filter is now superior to the one of the linear phase filter. Figure 2.5 shows the phase error and the group delay of the low delay filter in the passband. Comparing Figure 2.5 to Figure 2.2 shows the increase of phase and group delay error due to the small phase error weighting. However, the phase approximation is still sufficient for many practical filter applications. Note that the group delay error is smaller than one sample over more than 95% of the passband. This design example shows that reducing the delay and at the same time improving the magnitude behavior compared to a linear phase filter is possible if a relatively small phase error can be tolerated.

It should be noted that trading magnitude versus phase errors is not possible if the desired phase is linear and satisfies

$$\phi_d(\omega) = -\frac{N-1}{2}\omega. \tag{2.25}$$

In this case the optimum filter has an exactly linear phase response and the phase error vanishes. This result holds for the standard complex LS approximation problem as well as for the linear LS problem with independent magnitude and phase weighting. To see this, note that the optimum solution to the complex LS problem is characterized by

$$\boldsymbol{Rh} = \boldsymbol{s} \tag{2.26}$$

with $\boldsymbol{R}$ and $\boldsymbol{s}$ given by (2.3). Define

$$\tilde{\boldsymbol{e}}(\omega) = e^{j\phi_d(\omega)}\boldsymbol{e}(\omega). \tag{2.27}$$

Using (2.27), $\boldsymbol{R}$ and $\boldsymbol{s}$ can be written as

$$\begin{aligned} \boldsymbol{R} &= \int_\Omega W(\omega)\tilde{\boldsymbol{e}}(\omega)\tilde{\boldsymbol{e}}^H(\omega)d\omega \\ \boldsymbol{s} &= \int_\Omega W(\omega)\left|D\left(e^{j\omega}\right)\right|\tilde{\boldsymbol{e}}(\omega)d\omega \end{aligned} \tag{2.28}$$

If the desired phase satisfies (2.25), then

$$\tilde{\boldsymbol{e}}(\omega) = \boldsymbol{J}\tilde{\boldsymbol{e}}^*(\omega) \tag{2.29}$$

holds with

$$\boldsymbol{J} = \begin{pmatrix} \mathbf{0} & & & 1 \\ & & 1 & \\ & \cdots & & \\ 1 & & & \mathbf{0} \end{pmatrix}.$$

From (2.28) and (2.29) $\boldsymbol{JR} = \boldsymbol{R}^*\boldsymbol{J}$ and $\boldsymbol{Js} = \boldsymbol{s}^*$ follow. Hence, from (2.26) we get

$$\boldsymbol{JRh} = \boldsymbol{Js} = \boldsymbol{s}^*. \tag{2.30}$$

On the other hand,

$$\boldsymbol{JRh} = \boldsymbol{R}^*\boldsymbol{Jh}. \tag{2.31}$$

Equating the right hand sides of (2.30) and (2.31) and taking the complex conjugate leads to

$$\boldsymbol{RJh}^* = \boldsymbol{s}. \tag{2.32}$$

Comparing (2.32) to (2.26) shows that the optimum solution must satisfy

$$\boldsymbol{h} = \boldsymbol{Jh}^* \tag{2.33}$$

due to the non-singularity of $\boldsymbol{R}$. The symmetry defined by (2.33) is sufficient for the linear phase property of the filter with impulse response coefficients $\boldsymbol{h}$. There exists a similar proof for the linear phase property of the solution that minimizes the modified error measure $\tilde{\epsilon}(\boldsymbol{h})$ defined by (2.24). Hence, using independent magnitude and phase weighting functions is useless if the desired phase response satisfies (2.25). However, for all other choices of desired phase responses improving magnitude behavior at the cost of a larger phase error is possible.

2.1.4 Matlab Programs

The Matlab program `mpls` implements the least squares filter design with independent weighting of magnitude and phase errors described in Section 2.1.3. It computes approximations on a continuous domain by using closed-form solutions for all integrals. It solves the special design problems with piecewise constant magnitudes of the desired complex frequency responses, piecewise constant desired weighting functions, and piecewise linear desired passband phase responses. The program implicitly assumes the symmetry $D\left(e^{j\omega}\right) = D^*(e^{-j\omega})$ to be satisfied. The continuous complex least squares design problem discussed in Section 2.1.1 is solved if magnitude and phase weighting functions satisfy $W_m(\omega) = W_\phi(\omega)/\left|D\left(e^{j\omega}\right)\right|^2$ (cf. Equation (2.23)). The filters shown in Figures 2.1 and 2.2 have been designed using the commands

```
om=pi*[0 .2 .3 1]; m=[1 0]; tau1=[30 0]; tau2=[20 0]; w=[1 100];
h1 = mpls(61,om,m,tau1,w,w);
h2 = mpls(61,om,m,tau2,w,w);
```

The filter with independent magnitude and phase weighting shown in Figures 2.4 and 2.5 has been designed by

```
om=pi*[0 .2 .3 1]; m=[1 0]; tau=[20 0]; wm=[1 100]; wp=[1e-3 0];
h = mpls(61,om,m,tau,wm,wp);
```

The program `mpls` uses six input arguments. `N` is the filter length, `om` is a vector of bandedges between 0 and π, the vector `m` contains the desired magnitude values in the different frequency bands, `tau` holds the corresponding desired group delay values, and `wm` and `wp` are the values of the magnitude and phase weights in each band. The vector `om` containing the band edges must have twice as many elements as the vectors `m`, `tau`, `wm`, and `wp`.

```
function h = mpls(N,om,m,tau,wm,wp)
% h = mpls(N,om,m,tau,wm,wp)
% Complex Least Squares Design of FIR Filters with independent
% weighting of magnitude and phase errors
%
% h:   filter impulse response
% N:   filter length
% om:  band edges (0 <= om <= pi)
% m:   desired magnitude per band (constant)
% tau: desired group delay per band (constant)
%      (stopband values irrelevant)
% wm:  magnitude weighting function per band (constant)
% wp:  phase weighting function per band (constant)
%      (stopband values irrelevant)
%
% example:
```

```
% om=pi*[0 .4 .5 1]; m=[1 0]; tau=[20 0]; wm=[1 100]; wp=[.1 0];
% h=mpls(61,om,m,tau,wm,wp);
%
% Author: Mathias C. Lang, Vienna University of Technology, March '98

om=om(:); m=m(:); tau=tau(:); wm=wm(:); wp=wp(:);

% Compute objective function matrices Q (NxN) and q (Nx1)
% (objective = h'Qh - 2q'h)
nb = length(m);     % number of bands
Qtvec = zeros(N,1); Qhcvec = zeros(N,1); Qhrvec = zeros(N,1);
q = zeros(N,1);
n = (0:N-1)';
for i=1:nb,
    fu = om(2*i)/pi;
    fl = om(2*i-1)/pi;
    if m(i),
      Qtvec = Qtvec + ...
         (wm(i) + wp(i)/m(i)^2)*...
         (fu*sinc(n*fu)-fl*sinc(n*fl));
      Qhcvec = Qhcvec + ...
         (wm(i) - wp(i)/m(i)^2)*...
         (fu*sinc((n-2*tau(i))*fu)-fl*sinc((n-2*tau(i))*fl));
      Qhrvec = Qhrvec + ...
         (wm(i) - wp(i)/m(i)^2)*...
         (fu*sinc((n+N-1-2*tau(i))*fu)-fl*sinc((n+N-1-2*tau(i))*fl));
      q = q + ...
         wm(i)*m(i)*2*(fu*sinc((n-tau(i))*fu) - fl*sinc((n-tau(i))*fl));
    else
      Qtvec = Qtvec + wm(i)*2*(fu*sinc(n*fu)-fl*sinc(n*fl));
    end
end
Q = toeplitz(Qtvec) + hankel(Qhcvec,Qhrvec);

% Compute WLS solution
h = Q\q;
```

Note that the program `mpls` does not take the Toeplitz plus Hankel structure of the system matrix into account. This structure, however, should be exploited when high-order filters are to be designed.

The program `lslevin` computes a discrete weighted least squares approximation to a given complex desired frequency response according to (2.10). Independent weighting of magnitude and phase errors is not implemented in `lslevin`. The program uses Levinson's algorithm to solve the normal equations (2.9). Levinson's algorithm and the corresponding Matlab program `levin` are described in Appendix B. The program `lslevin` computes and stores only the first row of the Hermitian Toeplitz system matrix of the normal equations. Hence, the memory requirements are extremely low, even for very high filter orders.

```
function h = lslevin(N,om,D,W)
% h = lslevin(N,om,D,W)
% Complex Least Squares FIR filter design using Levinson's algorithm
%
% h       filter impulse response
% N       filter length
% om      frequency grid (0 <= om <= pi)
% D       complex desired frequency response on the grid om
% W       positive weighting function on the grid om
%
% example: length 61 bandpass, band edges [.23,.3,.5,.57]*pi,
% weighting 1 in passband and 10 in stopbands, desired passband
% group delay 20 samples
%
% om=pi*[linspace(0,.23,230),linspace(.3,.5,200),linspace(.57,1,430)];
% D=[zeros(1,230),exp(-j*om(231:430)*20),zeros(1,430)];
% W=[10*ones(1,230),ones(1,200),10*ones(1,430)];
% h = lslevin(61,om,D,W);
%
% Author: Mathias C. Lang, Vienna University of Technology, July 1998

om = om(:); D = D(:); W = W(:); L = length(om);
DR = real(D); DI = imag(D); a = zeros(N,1); b = a;

% Set up vectors for quadratic objective function
% (avoid building matrices)
dvec = D; evec = ones(L,1); e1 = exp(j*om);
for i=1:N,
   a(i) = W.'*real(evec);
   b(i) = W.'*real(dvec);
   evec = evec.*e1; dvec = dvec.*e1;
end

a=a/L; b=b/L;

% Compute weighted l2 solution
h = levin(a,b);
```

The filter shown in Figure 2.3 has been designed using the commands

```
om=pi*[linspace(0,.295,1500),linspace(.3,.5,1000),linspace(.505,1,2500)];
D=[zeros(1,1500),exp(-j*om(1501:2500)*700),zeros(1,2500)];
W=[100*ones(1,1500),ones(1,1000),100*ones(1,2500)];
h=lslevin(1401,om,D,W);
```

2.2 Complex Chebyshev Approximation

2.2.1 Introduction

Frequency domain Chebyshev design of FIR filters has been widely used since McClellan and Parks published their algorithm which is applicable to the design of linear phase FIR filters [88, 82]. Although there has been some earlier work on Chebyshev design of linear phase FIR filters [39, 43], it was the Parks-McClellan algorithm that became famous and has been used almost exclusively for Chebyshev FIR filter design since then. The reason for this is its efficiency compared to other methods and its availability as a reliable computer program [83]. The Chebyshev criterion was preferred over other design criteria because it perfectly matches the problem of fitting a frequency response to a specified tolerance scheme using the minimum possible filter order. Much later is has been argued that this problem formulation might actually not be well suited for many applications [2]. This topic will be treated in Section 2.3.2 in more detail.

If the phase response of the filter to be designed is restricted to be linear, the resulting approximation problem is real-valued. This follows from the fact that the approximating function is the real-valued amplitude function given by

$$A(\omega,\boldsymbol{h}) = H\left(e^{j\omega},\boldsymbol{h}\right)e^{j\left(\frac{N-1}{2}\omega-k\frac{\pi}{2}\right)}, \tag{2.34}$$

where N is the filter length and the constant k satisfies $k=0$ for even symmetry and $k=1$ for odd symmetry of the impulse response [87]. In this case the alternation theorem (see e. g. [20]) represents a very strong characterization of the optimal Chebyshev solution to the approximation problem. This characterization theorem is utilized by the Remez algorithm[1] [20] which is the core of the Parks-McClellan program for optimal linear phase FIR filter design in the Chebyshev sense. If the filter to be designed is not restricted to exhibit an exactly linear phase response, then there is no unique formulation of the corresponding Chebyshev approximation problem because several error functions might be taken into account. One generalization of the real Chebyshev approximation problem is the complex Chebyshev approximation problem where the complex desired frequency response $D\left(e^{j\omega}\right)$ is approximated by the actual frequency response $H\left(e^{j\omega}\right)$ by solving for

$$\boldsymbol{h} = \arg\min\left\{\max_{\omega\in\Omega} W(\omega)\left|E_c(\omega,\boldsymbol{h})\right|\right\}, \tag{2.35}$$

with the complex error function $E_c(\omega,\boldsymbol{h})$ defined by (1.1). However, unlike in the real approximation case, the solution to problem (2.35) does not exhibit a strong characterization that would directly lead to an efficient algorithm for its computation. The main

[1]If we refer to the Remez algorithm we actually mean the *Second Algorithm of Remez* which is a multiple exchange algorithm as opposed to the single exchange algorithm referred to as *First Algorithm of Remez*.

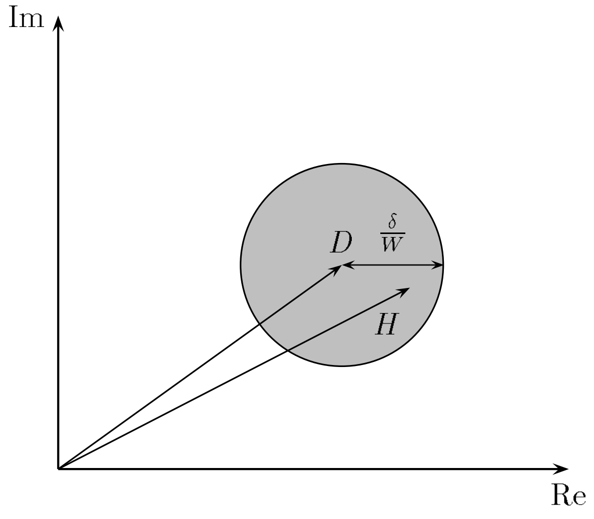


Figure 2.6: Complex Chebyshev approximation illustrated in the complex plane for one frequency point $\omega \in \Omega$: $H\left(e^{j\omega}\right)$ approximates the desired response $D\left(e^{j\omega}\right)$ with a maximum error $\delta/W(\omega)$.

problem is that the exact number of extremal frequencies where the magnitude of the complex error function attains its maximum is not known in advance. Other problem formulations take magnitude and phase errors or magnitude and group delay errors into account. However, these problems are highly nonlinear and extremely hard to solve. This is especially true if the desired filter order is high. Moreover, the solutions obtained by some nonlinear optimization method might only be inferior local solutions. Hence, methods dealing with these problems are usually slow and prone to local minima even for moderate filter lengths. However, there are good approximations to the problem of simultaneously approximating specified magnitude and phase responses in a Chebyshev sense [124, 92].

Figure 2.6 shows a graphical explanation of the complex Chebyshev design problem in the complex plane. For every frequency point $\omega \in \Omega$ the optimum approximating frequency response $H\left(e^{j\omega}\right)$ is confined to a circular region with the desired response $D\left(e^{j\omega}\right)$ in its center. The radius of the circle is given by $\delta/W(\omega)$, where δ is the Chebyshev error defined by

$$\delta = \min_{\boldsymbol{h}} \left\{ \max_{\omega\in\Omega} W(\omega)\left|E_c(\omega,\boldsymbol{h})\right| \right\}. \tag{2.36}$$

Consequently, the complex error function $E_c(\omega)$ corresponding to the optimum solution to the complex Chebyshev approximation problem remains inside a circle with radius $\delta/W(\omega)$ centered at the origin of the complex plane.

2.2.2 Overview of Algorithms

In recent years many methods for solving the complex Chebyshev approximation problem (2.35) have been proposed. However, almost all of them belong to one of three groups of algorithms that will be outlined in the following.

Generalized Remez Algorithms. These algorithms aim at generalizing the Remez algorithm for real Chebyshev approximation to the complex approximation case. This idea is appealing because of the computational efficiency of the Remez algorithm. Other algorithms based on optimization methods are much slower and need more memory. The first approach to generalize the Remez algorithm to the complex case has been published by Preuss [94, 95]. This algorithm solves a sequence of complex interpolation problems prescribing values for the complex error function on a heuristic basis. Due to its simplicity this method is fast if it converges. However, many cases have been observed where the algorithm does not converge [49, 90]. If it converges the solution need not necessarily be the optimal solution to the problem. However, it has been observed that the resulting maximum error is usually close to the one of the optimal solution [109]. This method has been improved by Schulist [108]. The improvements proposed in [108] mainly accelerate the design algorithm but do not significantly improve its convergence behavior. Similar generalized Remez algorithms have been published in [9, 10] but they do not have any significant advantages compared to the original method by Preuss.

Another multiple exchange method which may be regarded as a generalization of the Remez algorithm has been proposed by Tseng [132, 134]. This method deviates more from the original Remez multiple exchange algorithm for real-valued Chebyshev approximations than previous work in the sense that it only retains the two basic steps in each iteration of the Remez algorithm:

1. Solve a subproblem on a finite set of frequencies.

2. Find a new set of frequencies for the next iteration step such that the Chebyshev error of the next subproblem will increase.

In the real approximation case both steps are easily solved. Step 1 is just an interpolation problem and the alternation theorem provides a simple rule to implement step 2. However, since in the complex case the number of extremal frequencies is unknown and the alternation theorem cannot be applied, both steps are considerably more complicated. It turns out that in the complex approximation case the subproblem in step 1 is a complex Chebyshev approximation problem on a finite frequency grid. In [132, 134] these subproblems are solved by Lawson's algorithm [67]. The exchange rule needed in step 2 is based on heuristics. It is known that Lawson's algorithm exhibits a slow convergence rate and hence, it is no good candidate for being used in an iteration loop. For this reason, the method proposed in [132, 134] is rather slow and does not yield very accurate results. Tseng improved his algorithm in [133, 135] where he replaced Lawson's algorithm by Newton's method and improved the frequency exchange rule in step 2. However, this improved algorithm lost many of the desirable properties of the original Remez algorithm. Especially the computational effort is dramatically higher. Moreover, the same results can be achieved by using more flexible optimization methods with comparable computational effort.

An interesting new approach to generalize the Remez algorithm to the complex approximation problem has been presented by Karam and McClellan [48, 49]. They do not use any heuristics but extend the alternation theorem to the complex case. They show that a generalized alternation of the complex function

$$\tilde{E}(\omega) = W(\omega)E_c(\omega)e^{j\frac{N-1}{2}\omega} \tag{2.37}$$

is a sufficient condition for optimality of the corresponding solution. Generalized alternation means that the complex function $\tilde{E}(\omega)$ satisfies

$$|\tilde{E}(\omega_i)| = \max_{\omega\in\Omega} |\tilde{E}(\omega_i)|, \quad i = 1, 2, \ldots$$

and

$$\tilde{E}(\omega_i) + \tilde{E}(\omega_{i+1}) = 0, \quad i = 1, 2, \ldots$$

for at least $N+1$ frequency points $\omega_1 < \omega_2 < \ldots < \omega_{N+1}$. The frequencies ω_i are called extremal frequencies. Moreover, they show that this generalized alternation property is also necessary if the number of extremal frequencies is $N+1$. It is known that for polynomial complex Chebyshev approximation with Chebyshev systems the number of extremal frequencies is between $N+1$ and $2N+1$ [73]. However, in all cases where the number of extremal frequencies is greater than $N+1$ the generalized alternation property need not be satisfied for optimality. Based on this generalized alternation theorem Karam and McClellan apply the Remez algorithm to the real-valued function

$$\tilde{E}_R(\omega) = \text{Re}\left\{\tilde{E}(\omega)e^{-j\theta}\right\},$$

with $\theta = \arg\{\tilde{E}(\omega_1)\}$. Figure 2.7 illustrates the generalized alternation property of $\tilde{E}(\omega)$. The plot on the left-hand side shows the trace of the weighted complex error function of an optimum solution to a complex Chebyshev approximation problem. The plot on the right-hand side shows the corresponding rotated error function $\tilde{E}(\omega)e^{-j\theta}$ which is purely real-valued at its extremal points. The algorithm by Karam and McClellan is guaranteed to find the optimal solution if this solution has exactly $N+1$ extremal points.

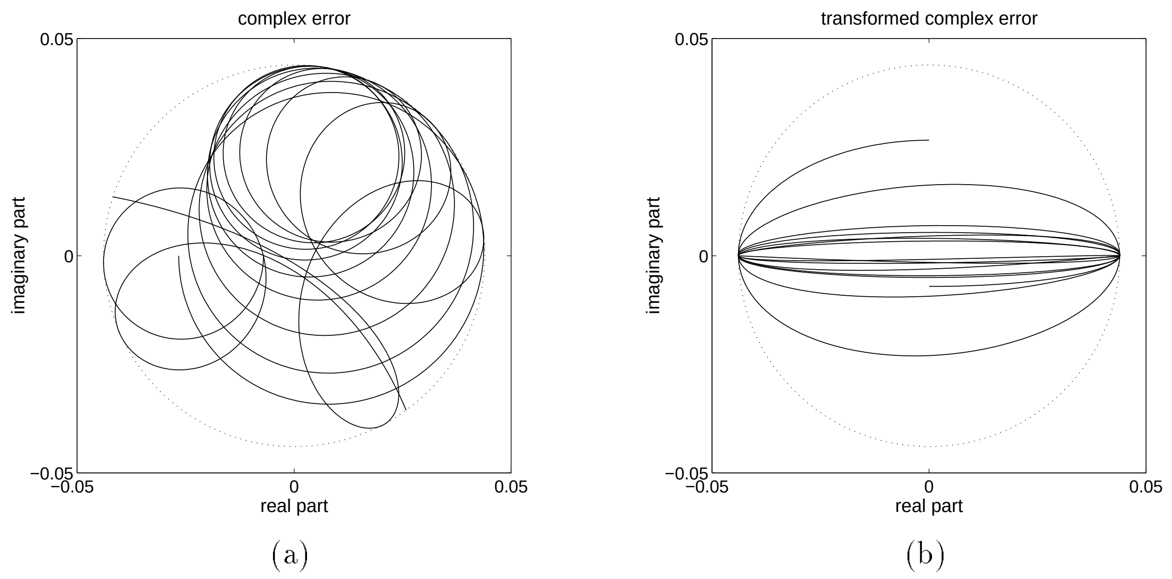


Figure 2.7: Generalized alternation property. Left plot: weighted complex error function $W(\omega)E_c(\omega)$. Right plot: alternating transformed error function $\tilde{E}(\omega)e^{-j\theta}$.

Of course, this property is not known in advance. The authors show that the solutions obtained by their algorithm are in general optimal solutions on some subset $\Omega' \subset \Omega$. However, this subset may be arbitrarily small. Since the algorithm does not converge to the optimal solution in general but may yield solutions that are close to the optimal solution, Karam and McClellan proposed to use it as an initial design procedure followed by a more general optimization method [50]. However, it will be shown in Section 2.2.3 that this post-optimization proposed in [50] is very inefficient and not useful for the design of filters with lengths $N > 50$.

Linear Programming and Semi-infinite Optimization. The complex Chebyshev approximation problem (2.35) is a convex optimization problem [42]. Hence, general algorithms for solving convex optimization problems can be applied. The main advantage of this approach is that additional design constraints can be taken into account. In the case of linear phase filter design the Chebyshev approximation problem reduces to a linear programming problem. The original problem reads

$$\underset{\boldsymbol{h}}{\text{minimize}} \ \max_{\omega\in\Omega} W(\omega)|A(\omega,\boldsymbol{h}) - A_D(\omega)|, \tag{2.38}$$

where $A(\omega,\boldsymbol{h})$ is the real-valued amplitude function defined by (2.34) and $A_D(\omega)$ is the real-valued desired amplitude function. This problem is equivalent to the constrained optimization problem

$$\begin{aligned} &\underset{\boldsymbol{h},\,\delta}{\text{minimize}} && \delta \\ &\text{subject to} && W(\omega)|A(\omega,\boldsymbol{h}) - A_D(\omega)| \le \delta, \quad \omega \in \Omega, \end{aligned} \tag{2.39}$$

where a new unknown parameter δ has been introduced. Problem (2.39) can be reformulated as follows:

$$\begin{array}{ll} \underset{\boldsymbol{h},\delta}{\text{minimize}} & \delta \\ \text{subject to} & \begin{array}{rcl} W(\omega)[A(\omega,\boldsymbol{h})-A_D(\omega)] & \leq & \delta \\ -W(\omega)\left[A(\omega,\boldsymbol{h})-A_D(\omega)\right] & \leq & \delta, \quad \omega\in\Omega. \end{array} \end{array} \tag{2.40}$$

Problem (2.40) is a linear programming problem because the objective function as well as the constraints are linear with respect to the unknown parameters $\boldsymbol{h}$ and δ. However, the number of constraints is infinite because they are to be satisfied for all $\omega\in\Omega$. For this reason, problem (2.40) is referred to as a linear semi-infinite programming problem. The term *semi-infinite* denotes the fact that there is a finite number of unknowns but an infinite number of constraints. By replacing the set Ω by a discrete set of frequencies Ω_d, problem (2.40) becomes a standard linear programming problem and is readily solved by any linear programming algorithm [34]. This approach has been discussed in [97].

Considering the complex approximation problem (2.35) for the design of filters with arbitrary magnitude and phase responses, we arrive at the following optimization problem:

$$\begin{array}{ll} \underset{\boldsymbol{h},\delta}{\text{minimize}} & \delta \\ \text{subject to} & W(\omega)|E_c(\omega,\boldsymbol{h})|\leq\delta, \quad \omega\in\Omega. \end{array} \tag{2.41}$$

This is a convex semi-infinite programming problem since the objective function is linear – and hence convex – and the feasible set defined by the constraints in (2.41) is convex due to the convexity of the function $|E_c(\omega,\boldsymbol{h})|$ with respect to the filter coefficients $\boldsymbol{h}$. Splitting the constraints on the magnitude of the error in two sets of linear constraints as in (2.40) is not possible here because the error is complex. However, generalizing this idea results in the problem formulation

$$\begin{array}{ll} \underset{\boldsymbol{h},\delta}{\text{minimize}} & \delta \\ \text{subject to} & \text{Re}\left[W(\omega)E_c(\omega,\boldsymbol{h})e^{j\alpha}\right]\leq\delta, \quad \omega\in\Omega, \quad \alpha\in[0,2\pi). \end{array} \tag{2.42}$$

Problem (2.42) is equivalent to problem (2.41). It is a linear semi-infinite programming problem because the constraints are linear with respect to the parameters $\boldsymbol{h}$ and δ. However, the problem is infinite with respect to the frequency variable ω and the angle variable α. This formulation of the complex Chebyshev approximation problem was first presented in [37] and was later analyzed in [126]. Discretizing the continuous domain Ω is not sufficient for the problem to become a standard linear programming problem. Linear

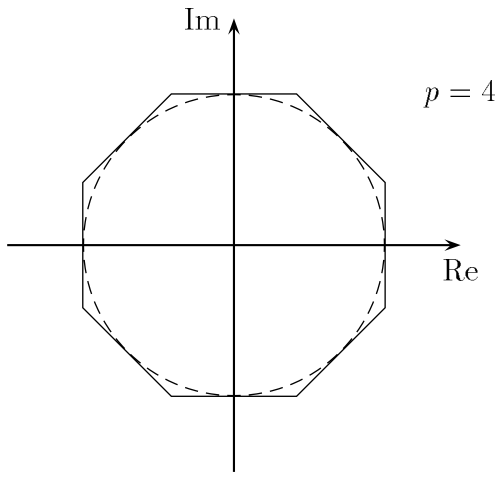


Figure 2.8: Linear programming formulation of the complex Chebyshev approximation problem: the circular error region is replaced by a regular polygon with $2p$ vertices.

programming can only be applied if also the angle variable α is discretized. However, this means that unlike in the linear phase case the approximation problem cannot be solved exactly on a discrete frequency grid by linear programming methods. Solving a discretized version of problem (2.42) by linear programming methods has been proposed in [37]. In [127] the same discretization method has been used for optimizing antenna arrays. Chen and Parks were the first to use this method for the design of FIR filters according to the complex Chebyshev criterion [18]. Usually an even number of equidistant angles

$$\alpha_k = \frac{k\pi}{p}, \quad k = 0, 1, \ldots, 2p-1, \quad p \geq 2. \tag{2.43}$$

is chosen. With this choice, the circular error region in the complex plane is approximated by a regular polygon with $2p$ vertices. This modified error region is illustrated by Figure 2.8 for the case $p = 4$. Let δ_d denote the Chebyshev error of the solution to the complex Chebyshev approximation problem where the continuous domain Ω has been replaced by a finite set of frequencies Ω_d:

$$\delta_d = \min_{\boldsymbol{h}} \max_{\omega \in \Omega_d} W(\omega)|E_c(\omega, \boldsymbol{h})|. \tag{2.44}$$

Let $\delta_d(p)$ denote the Chebyshev error of the solution obtained by solving the linear programming problem (2.42) with $\omega \in \Omega_d$ and $2p$ equidistant angles α_k, $k = 0, 1, \ldots, 2p-1$, according to (2.43). Note that we define $\delta_d := \lim_{p \to \infty} \delta_d(p)$. From geometrical considerations it follows that $\delta_d(p)$ is bounded by

$$\delta_d(p) \leq \delta_d / \cos(\pi/2p). \tag{2.45}$$

For $p \to \infty$, $\delta_d(p)$ approaches δ_d quadratically. Hence, we can solve the discrete complex Chebyshev problem with arbitrary accuracy using linear programming by increasing the number of angles α_k. However, the dimension of the problem increases considerably. If L denotes the number of frequency points in Ω_d, the total number of constraints in the linear programming problem is $2Lp$. Choosing $p = 2$ corresponds to simultaneous Chebyshev approximation of the real and imaginary parts of the desired frequency response and results in a maximum degradation of 3 dB compared to the optimum solution to the original problem. Solving this simpler problem by linear programming has already been proposed in [12].

Linear programming can also be used to approximately solve the problem of simultaneous but independent approximation of prescribed magnitude and phase responses in the Chebyshev sense. This has been proposed for the special case of FIR allpass design by Steiglitz in [124]. The corresponding nonlinear semi-infinite optimization problem reads

$$\begin{array}{lrcccl} \underset{\boldsymbol{h},\,\delta}{\text{minimize}} & \delta & & & & \\ \text{subject to} & -\delta & \leq & W_m(\omega)E_m(\omega,\boldsymbol{h}) & \leq & \delta, \quad \omega \in \Omega \\ & -\delta & \leq & W_\phi(\omega)E_\phi(\omega,\boldsymbol{h}) & \leq & \delta, \quad \omega \in \Omega^p, \end{array} \tag{2.46}$$

where $E_m(\omega,\boldsymbol{h})$ and $E_\phi(\omega,\boldsymbol{h})$ are magnitude and phase errors defined by (2.12) and (2.13), $W_m(\omega)$ and $W_\phi(\omega)$ are magnitude and phase weighting functions, and Ω^p denotes the passband. A linear semi-infinite problem approximating problem (2.46) can be formulated by using linear approximations for magnitude and phase errors as given by (2.21) and (2.22). These approximations are only valid in the passbands. The stopband approximation, however, can be formulated like in the complex Chebyshev approximation problem (2.42). The complete linearized problem reads

$$\begin{array}{lrcccl} \underset{\boldsymbol{h},\,\delta}{\text{minimize}} & \delta & & & & \\ \text{subject to} & -\delta & \leq & W_m(\omega)\text{Re}\left[E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right] & \leq & \delta, \quad \omega \in \Omega^p \\ & -\delta & \leq & \dfrac{W_\phi(\omega)}{\left|D\left(e^{j\omega}\right)\right|}\text{Im}\left[E_c(\omega,\boldsymbol{h})e^{-j\phi_d(\omega)}\right] & \leq & \delta, \quad \omega \in \Omega^p \\ & & & W_m(\omega)\text{Re}\left[H\left(e^{j\omega},\boldsymbol{h}\right)e^{j\alpha}\right] & \leq & \delta, \quad \omega \in \Omega^s, \quad \alpha \in [0,2\pi), \end{array} \tag{2.47}$$

with $\phi_d(\omega) = \arg\{D\left(e^{j\omega}\right)\}$. Figure 2.9 shows a graphical interpretation of the magnitude/phase approximation problem (2.46) and its linearized form (2.47) in the complex plane. Only the passband approximation is illustrated because in the stopband there is no difference to the complex Chebyshev problem where the optimal response $H\left(e^{j\omega}\right)$ is confined to a circle around the origin with radius $\delta/W_m(\omega)$. If the semi-infinite linear

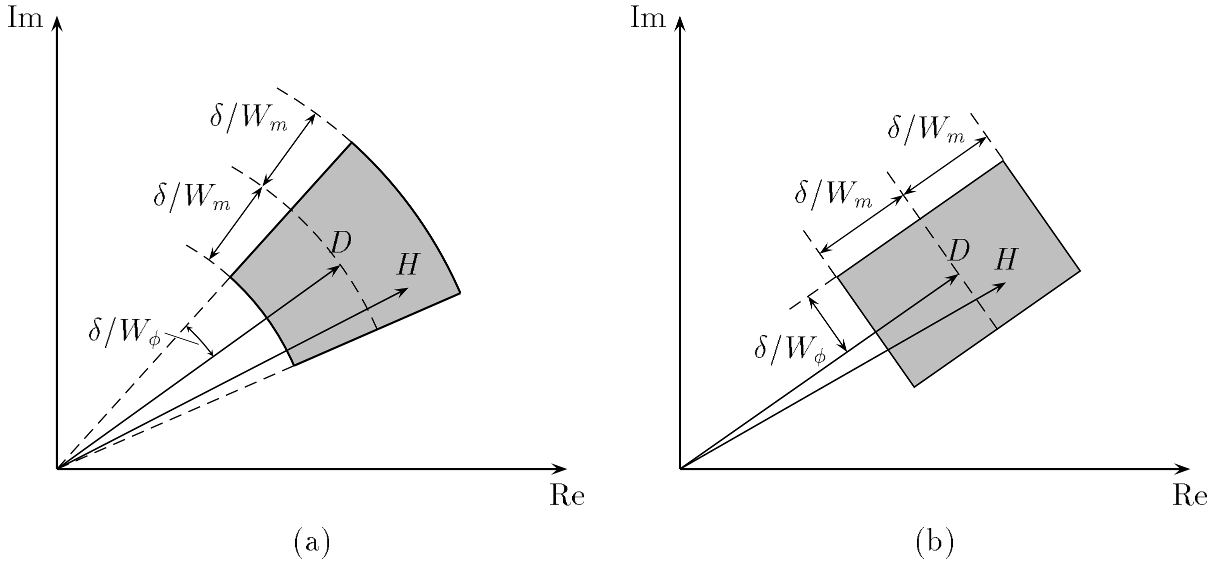


Figure 2.9: Magnitude and phase Chebyshev approximation illustrated in the complex plane for one passband frequency point $\omega \in \Omega^p$. The magnitude error is weighted by $W_m(\omega)$, the phase error is weighted by $W_\phi(\omega)$, and δ is minimized. (a) exact problem formulation. (b) linearized problem formulation.

problem (2.47) is to be solved by a standard linear programming algorithm, discretization of ω and α is necessary. The graphical interpretation shown in Figure 2.9 (b) does not change in this case because it applies to the passbands. Only in the stopbands the circle is replaced by a regular polygon as shown in Figure 2.8.

So far we have shown that there are two formulations of the Chebyshev FIR design problem with arbitrary magnitude and phase responses that lead to finite linear programming problems. The first is a discretized version of the complex Chebyshev approximation problem (2.42), and the second is a discretized version of problem (2.47) which by itself is a linearized formulation of the magnitude/phase Chebyshev approximation problem (2.46). It is possible to find a unified linear programming formulation that contains both formulations described so far as special cases [58]. This can be achieved by transforming the complex error function according to[2]

$$\tilde{E}(\omega) = \begin{cases} W_m(\omega)\mathrm{Re}\left[E_c(\omega)e^{-j\phi_d(\omega)}\right] + j\dfrac{W_\phi(\omega)}{|D(e^{j\omega})|}\mathrm{Im}\left[E_c(\omega)e^{-j\phi_d(\omega)}\right], & \omega \in \Omega^p \\ W_m(\omega)E_c(\omega), & \omega \in \Omega^s. \end{cases} \tag{2.48}$$

A transformation similar to (2.48) has been proposed by Preuss in [95] in the context of a generalized Remez algorithm for complex Chebyshev approximation to allow for a different weighting of magnitude and phase errors. This transformed error function $\tilde{E}(\omega)$ is

[2]Note that (2.48) is a new definition of $\tilde{E}(\omega)$ and there is no connection to the previous definition (2.37).

used instead of $E_c(\omega)$ in problem formulation (2.42). It simply replaces the circular error region by an elliptic region in the passbands. The shape of the ellipse is determined by the magnitude and phase weighting functions $W_m(\omega)$ and $W_\phi(\omega)$. However, the solution obtained by this transformation in combination with some complex Chebyshev design algorithm is far from the solution that is optimal with respect to the magnitude/phase Chebyshev problem (2.46). This situation changes if we use the finite linear programming formulation derived from (2.42) by discretizing ω and α. Nothing is gained in the stopbands where the number of different angles α_k given by (2.43) should be large to make the discretization error small. However, in the passbands we can choose another set of angles α_k. For a large number of angles in the passband the error region in the complex plane approaches an ellipse and we get complex Chebyshev approximation with different weighting of magnitude and phase errors as proposed in [95]. On the other hand, if we use a small number of different angles α_k in the passband, we get closer to problem formulation (2.47) which is a very good approximation to the magnitude/phase Chebyshev approximation problem. Hence, by varying the number of angles α_k in the passband we get from complex Chebyshev approximation with optional weighting of magnitude and phase errors to the approximate solution of the magnitude/phase Chebyshev problem. The two earlier linear programming approaches [18] and [124] are contained as special cases. The linear programming approach to complex Chebyshev approximation by Chen and Parks [18] results for $W_m(\omega) = W_\phi(\omega)/|D\left(e^{j\omega}\right)|$, and the magnitude/phase approximation approach by Steiglitz [124] results for the choice $p = 2$ in (2.43) in the passbands and arbitrary $W_m(\omega)$ and $W_\phi(\omega)$. It should be noted that Chen and Parks proposed an additional weighting of the phase error by adding linear constraints to the problem [18]. This results in a very large number of constraints which is avoided by using the proposed problem formulation based on the transformed error function (2.48).

The drawback of all linear programming approaches that discretize the set Ω and the angle variable α in (2.42) or (2.47) is the large number of constraints in the corresponding problem. Solving this linear programming problem requires a large amount of memory and a high computational effort. However, it is possible to solve the linear semi-infinite problems (2.42) and (2.47) exactly by solving a sequence of relatively small finite problems. At the optimum solution only a finite number of constraints in (2.42) or (2.47) will be satisfied with equality. These constraints are called *active constraints*. If the corresponding pairs of frequencies ω_i and angles α_i were known in advance, it would suffice to consider only these constraints. The solution to this finite problem also solves the original semi-infinite problem. Of course, the relevant frequencies and angles are not known in advance, but they can be found by an iterative procedure. This is achieved by semi-infinite programming algorithms. The basic idea is similar to the Remez exchange algorithm, and in a sense these algorithms could also be called "generalized Remez algorithms". However, since there are several differences to the Remez algorithm for linear phase FIR filter design we prefer to treat these methods as a separate group of algorithms. All these algorithms solve a sequence of finite linear programming problems and change the set of constraints from step to step such that in the last step the active constraints of the optimal solution are a subset of the actual constraint set. The corresponding exchange procedure in the Remez algorithm is a simple multiple exchange where all old frequency points are replaced by the new minima and maxima of the actual real-valued error function. This simple multiple exchange procedure, however, does not work in the general complex approximation case. Linear semi-infinite programming problems can be solved using either single or multiple exchange algorithms. Single exchange methods update the constraint set by only one new constraint in each iteration step whereas multiple exchange methods use several new constraints in each step. For both types of methods the update of the constraint set can be done in such a way that the total number of constraints remains the same (*explicit exchange*) or that it changes from iteration to iteration (*implicit exchange*). Some methods even never delete any old constraints but keep adding new ones in every iteration step. For these methods the term *exchange* algorithm is actually not appropriate.

The first algorithm of Remez is a single exchange algorithm that adds one point corresponding to the largest error in every iteration step. This algorithm is also applicable to semi-infinite programming problems. Another single exchange procedure specialized to the complex Chebyshev approximation problem has been developed by Tang [130]. In this method the number of constraints is fixed. Hence, in every iteration step one old constraint is replaced by a new one. The drawback of single exchange algorithms is their slow convergence. Since the subproblems are usually simpler to solve than the subproblems of multiple exchange algorithms this would not be a severe drawback. However, the error function must be evaluated in each iteration step and hence, it is advantageous to use multiple exchange algorithms where the number of iteration steps necessary to solve the problem with sufficient accuracy is considerably smaller. Single exchange methods have been used for complex Chebyshev FIR filter design in [8, 110, 111, 139, 52, 53]. All these methods apply Tang's algorithm [130]. Schulist extended Tang's algorithm to include additional design constraints such as constraints on the magnitude, its derivative and on the group delay [110, 111]. In [139] the numerical stability of Tang's algorithm has been improved and hence, the design of high-order filters ($N > 100$) has become possible. However, because of the above mentioned reasons these single exchange methods are not very efficient. Tang's algorithm [130] has also been used for the design of filters with complex coefficients [52, 53]. All methods discussed here can easily be generalized to the case of complex filter coefficients. Any difficulties occur only due to the complex nature of the approximation problem itself, and not because of complex filter coefficients. Optimum Chebyshev filters with complex coefficients and linear phase responses can be designed by the standard Remez multiple exchange algorithm because these problems are real approximation problems [81].

Burnside and Parks were the first to use a multiple exchange algorithm to solve the semi-infinite programming problem corresponding to the complex Chebyshev FIR filter design problem [13, 14]. The algorithm is very simple since in every iteration step it adds new constraints corresponding to the maxima of the magnitude of the actual error function without ever deleting any old constraints. This algorithm was originally presented in [140]. The advantage of this method is that its convergence rate is fast compared to single exchange methods as well as to many other multiple exchange methods that delete old constraints. The reason for this is that old constraints that have been deleted due to some exchange rule might be needed in later iteration steps. The related drawback is that the number of constraints keeps increasing with every iteration step. Hence, especially for the design of high-order filters large amounts of memory are required. The authors of [90, 91] used a multiple exchange method for solving the semi-infinite programming problem corresponding to the complex Chebyshev FIR filter design problem. They used an algorithm which drops certain old constraints in every iteration step. In [90] a discretization method was used where the problem is solved on increasingly finer frequency grids. Since the authors used extremely dense grids the number of constraints in each subproblem is relatively large and the computational effort is high. The method they presented in [91] is more practical since for each local error maximum only one constraint is added to the old constraint set. These maxima are computed on continuous frequency intervals. This aspect of continuous approximation as an advantage over other methods which use a dense finite frequency grid in their numerical implementation (e. g. [14, 130]) seems to be overemphasized in [91]. The important point is rather that using an exchange algorithm makes the complexity of the optimization subproblems independent of the number of frequency points used to represent the approximation domain. This is the important difference to standard linear programming formulations where the number of grid frequencies directly contributes to the problem size and dramatically influences the memory requirements and computational effort. Hence, by using multiple exchange algorithms the number of frequency points can be made large enough such that the solution to the problem on a discrete frequency set is a very good approximation to the solution to the continuous problem. The only difference in computational effort resulting from an increased grid density lies in the computation of the error function on this grid. However, compared to the effort for solving the subproblems in multiple exchange methods the error evaluation on a grid is negligible if the FFT is used. The evaluation of the error maxima becomes a bit more complicated if it is to be done on the continuous domain. However, the basic optimization algorithm remains unchanged. If desired it is easily possible to achieve a true continuous approximation by choosing some method to compute local maxima on a continuous domain. This has also been proposed in [119] for the design of linear phase FIR filters by a multiple exchange algorithm. The main advantage of the method [91] over the multiple exchange approach by Burnside and Parks [13, 14] is the use of an exchange rule that drops some of the old constraints in every iteration step and hence does not result in an unlimited growth of the constraint set. The price paid for this is a considerable reduction in convergence speed due to the occasional deletion of constraints that become important at a later iteration step. If enough memory is available the approach chosen by Burnside and Parks is superior to the one given in [91] because it usually converges much faster.

Iterative Reweighted Least Squares. Iterative reweighted least squares (IRLS) algorithms compute a sequence of weighted least squares approximations which, under suitable conditions, converge to the optimum solution to the Chebyshev approximation problem. In every iteration step the weighting function is updated according to the actual error function in order to find the weighting function that results in a best Chebyshev approximation. Since these algorithms are not based on the alternation theorem, they can be applied to more general problems than the Remez multiple exchange algorithm including complex and multivariate approximations. IRLS methods are based on the following fundamental relationship between least squares approximation and Chebyshev approximation:

The N coefficients in the vector $\boldsymbol{h}$ optimally solve problem (2.35) if and only if there exist r points $\omega_1, \ldots, \omega_r$ and r numbers $\mu_1 > 0, \ldots, \mu_r > 0$ $(N+1 \leq r \leq 2N+1)$, such that

$$\sum_{\ell=1}^{r} \mu_\ell W^2(\omega_\ell) E_c(\omega_\ell, \boldsymbol{h}) e^{jn\omega_\ell} = 0, \quad n = 0, 1, \ldots, N-1, \tag{2.49}$$

with

$$W(\omega_\ell)|E_c(\omega_\ell, \boldsymbol{h})| = \max_{\omega \in \Omega} W(\omega)|E_c(\omega, \boldsymbol{h})|, \quad \ell = 1, \ldots, r.$$

This characterization theorem can be derived either from the theory of best Chebyshev approximations (see e. g. [73]) or from optimization theory using the Kuhn-Tucker conditions (see Section 3.3). The N linear equations (2.49) are the normal equations corresponding to the discrete weighted least squares problem

$$\underset{\boldsymbol{h}}{\text{minimize}} \sum_{\ell=1}^{r} \mu_\ell W^2(\omega_\ell) |E_c(\omega_\ell, \boldsymbol{h})|^2. \tag{2.50}$$

We will show this fact in Section (3.3). Consequently, the optimal solution to the complex Chebyshev approximation problem (2.35) can be computed by solving a discrete weighted least squares problem as given by (2.50). However, neither the weights μ_ℓ nor the frequencies ω_ℓ needed in (2.50) are known in advance. IRLS algorithms find the weights μ_ℓ iteratively by updating the weighting function according to the actual magnitude of the error function. It is not necessary to explicitly find the frequencies ω_ℓ because it suffices if the weighting function approaches zero everywhere except for the extremal frequencies ω_ℓ. This is exactly what IRLS algorithms achieve.

Lawson was the first to present an IRLS method for solving linear Chebyshev approximation problems [67]. He exclusively considered approximations on finite point sets but extending the algorithm to the continuous case is possible [24]. However, since the complexity of the weighted least squares subproblems is independent of the number of frequencies in the grid provided that these problems are solved using the normal equations, the grid density can be chosen high enough such that the continuous frequency intervals of interest are sufficiently well represented. Denoting the number of grid points by L and the iteration index by k, Lawson's algorithm iterates as follows:

0. Choose $w_i^{(0)} > 0, \quad i = 0, 1, \ldots, L-1, \quad k := 0.$

1. Solve
$$\boldsymbol{h}^{(k)} = \arg\min \left\{ \sum_{i=0}^{L-1} w_i^{(k)} W^2(\omega_i) |E_c(\omega_i, \boldsymbol{h})|^2 \right\}.$$

2. Set
$$w_i^{(k+1)} = \frac{w_i^{(k)} W(\omega_i) |E_c(\omega_i, \boldsymbol{h}^{(k)})|}{\sum_{i=0}^{L-1} w_i^{(k)} W(\omega_i) |E_c(\omega_i, \boldsymbol{h}^{(k)})|}, \quad k := k+1, \quad \text{go to 1.}$$

After the algorithm has converged, most of the weights w_i, $i = 0, \ldots, L-1$, will be zero or very small while the others approach the optimum values μ_ℓ in (2.49). The extremal frequencies ω_ℓ are found implicitly as those frequencies corresponding to nonzero weights w_i.

Lawson's algorithm has two drawbacks that are often considered obstructive to its practical application. The first drawback occurs due to the multiplicative weight update in step 2. As soon as a weight is zero it cannot be changed anymore. If a weight w_i becomes very small a considerable increase will take many iterations. Hence, if some weight decreases to zero although it corresponds to an extremal frequency of the optimal solution the algorithm has to be restarted. However, this rarely occurs in practice. The second drawback is its slow final convergence rate. Especially in the complex approximation case the convergence rate can be slower than linear [32]. This drawback is the actual limit to the practical use of Lawson's algorithm if very accurate solutions are desired.

However, experiments show that solutions close to the optimum are usually found after a small number of iteration steps.

Several modifications to Lawson's algorithm have been proposed in order to accelerate its rate of convergence (e. g. [32, 102]). These modifications mainly affect the way in which the weight update is performed. The modifications reported in [32, 102] have been observed to lead to divergence in some cases. However, if these modified algorithms converge, they converge to the optimum solution.

IRLS algorithms have been applied to filter design in [7, 11, 15, 17, 22, 72, 132, 134]. In [11, 15] the authors use IRLS methods to compute general L_p approximations, $1 \leq p \leq \infty$, whereas most other authors concentrate on Chebyshev approximation. An interesting approach to filter design using IRLS methods has been published independently in [21] and [72]. There, the weight update is performed using the envelope of the magnitude of the complex error function. In [21] a piecewise constant envelope is used, whereas in [72] the envelope is piecewise linear. Using an error envelope instead of the error itself for the weight update is advantageous because the weights cannot become zero, and hence restarting the algorithm is not necessary. Moreover, convergence speed is increased considerably. However, since all weights are nonzero the subproblem solved in the last iteration step of these algorithms is not equivalent to the problem characterizing the optimum solution (2.50). Consequently, the solutions computed by these algorithms are suboptimal. However, it turns out that for practical purposes these solutions are close enough to the true optimum solutions.

The main advantage of IRLS methods is their computational efficiency. In every iteration step a linear system of equations is solved and even if convergence is slow the computational effort is usually still small compared to linear programming or semi-infinite optimization methods. Another important advantage is that high-order filters can be designed using only a small amount of memory. This is the case if the normal equations (2.9) are used for solving the weighted least squares problems. Then the Toeplitz structure of the system matrix can be exploited and only its first row needs to be stored. The advantage of IRLS methods over generalized Remez algorithms is simply the fact that the latter do not converge in general whereas suitably formulated IRLS methods always converge to a solution which is at least very close to the true optimum.

2.2.3 Comparison of Algorithms

After discussing the various algorithmic approaches to complex Chebyshev approximation it is interesting to compare these methods regarding computational effort and quality of the solution with help of practical design examples. In this section two examples – one low-order and one high-order – are discussed.

Example 1: Reduced Delay Bandpass Filter ($N = 31$)
A bandpass filter of length $N = 31$ is designed according to the following specification:

$$D\left(e^{j\omega}\right) = \begin{cases} 0, & 0 \leq \omega \leq 0.2\pi \\ e^{-j12\omega}, & 0.3\pi \leq \omega \leq 0.56\pi \\ 0, & 0.66\pi \leq \omega \leq \pi \end{cases} \tag{2.51}$$

The passband weighting is 1 and the stopband weightings are 10. This example has been used in [87] and in [18]. We use four different algorithms described in Section 2.2.2 to design this filter. The first is the general linear programming formulation of the complex Chebyshev approximation problem as proposed in [58]. This method is implemented by the Matlab program `mpcheb` given in Section 2.2.4. For a semi-infinite programming algorithm we choose the method by Burnside and Parks [14] due to its simplicity. The program `burnpark` in Section 2.2.4 provides a very simple implementation of this algorithm. Lawson's algorithm in its original form implemented by the program `lawson` provided in Section 2.2.4 is used to obtain another solution to the same problem. Finally, we discuss the properties of Karam's and McClellan's generalized Remez algorithm [49] as implemented by the program `cremez` in the Matlab Signal Processing Toolbox 4.0. The design problem is posed as a complex Chebyshev approximation problem according to (2.35) without magnitude or phase weighting.

The design problem is solved on a grid of frequencies. To make a fair comparison we choose the grid that is used by the program `cremez` in the Matlab Signal Processing Toolbox. For a filter length $N = 31$ and the specification (2.51) `cremez` uses a grid of $L = 411$ points. This grid is rather coarse but it turns out that increasing the number of grid points does not change the solution significantly. The same grid is used for all four algorithms.

For the linear programming design with the program `mpcheb` we use a polygon with eight vertices to approximate the circular error region in the complex plane ($p = 4$ in (2.43)). Since no magnitude or phase weighting is used, the problem formulation is equivalent to the problem proposed by Chen and Parks [18]. The linear programming problem has $N + 1 = 32$ unknowns and $2Lp = 3288$ inequality constraints.

It takes Burnside's and Parks' algorithm 12 iterations to converge to a solution with a relative magnitude deviation of the local error maxima smaller than 10^{-3}. The linear programming problem in the final iteration step has 452 constraints.

Lawson's algorithm as implemented by the program `lawson` given in Section 2.2.4 is stopped after 50 iterations when the l_2-norm of the change of the solution vector $\boldsymbol{h}$ between two adjacent iteration steps is in the order of 10^{-5}. This is enough to obtain a sufficiently accurate solution to the problem.

The generalized Remez algorithm by Karam and McClellan [49] converges to a subop-

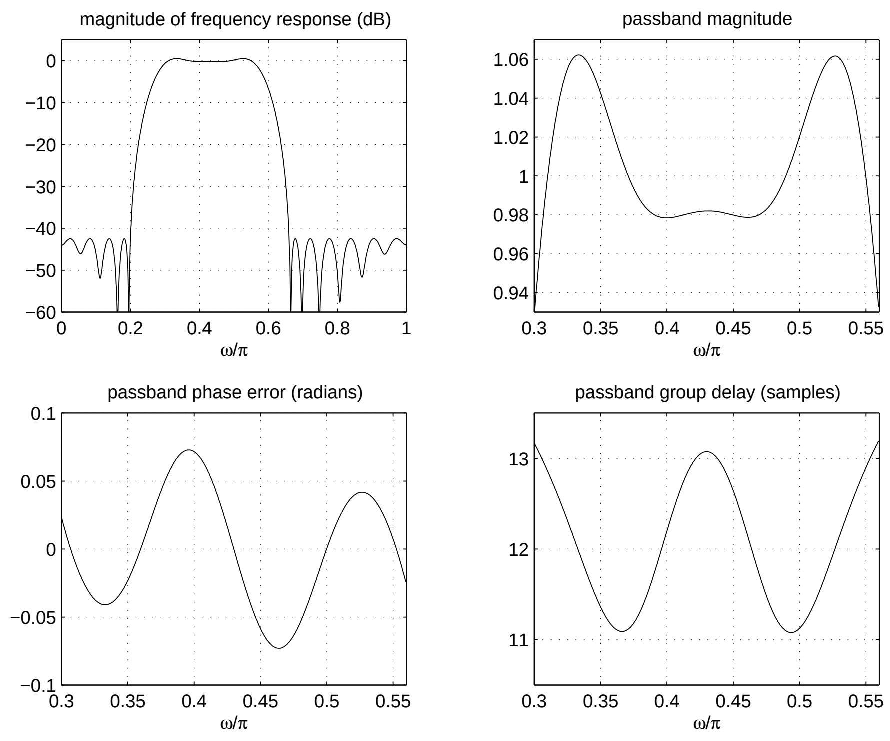


Figure 2.10: Reduced delay band pass filter (length $N = 31$).

timal solution after 7 multiple exchange steps. This solution is optimal on a subset comprising 53% of the original frequency set. Hence, a post-optimization stage is necessary which converges after 4 iteration steps to the final solution. Most of the computational effort is spent in this second optimization phase.

The designed filter is shown in Figure 2.10. Only the solution obtained by Burnside's and Parks' algorithm is shown because the individual solutions could hardly be distinguished on the plot. All designs were performed on a 166 MHz Pentium PC using Matlab 5.0. For each design the number of floating point operations was counted and the execution time was measured. Table 2.1 shows the results in terms of the Chebyshev error δ defined by (2.36), the required number of floating point operations, and the design time. The solution obtained by linear programming exhibits the largest Chebyshev error due to the linearization of the problem. Moreover, the associated computational effort and the memory requirements are larger than for the other methods. The multiple exchange algorithm by Burnside and Parks provides the solution with the smallest error and it is relatively efficient for this low-order design. Lawson's algorithm is most efficient and requires only little memory. However, the Chebyshev error of the solution is slightly larger than the one obtained by Burnside's and Parks' method. The generalized exchange algorithm by Karam and McClellan does not provide a satisfactory solution without using

Method	δ	flops	time
Linear Programming	$7.69 \cdot 10^{-2}$	$5.1 \cdot 10^{8}$	123 seconds
Burnside & Parks	$7.52 \cdot 10^{-2}$	$7.2 \cdot 10^{7}$	27 seconds
Lawson	$7.56 \cdot 10^{-2}$	$1.4 \cdot 10^{7}$	12 seconds
Karam & McClellan	$7.54 \cdot 10^{-2}$	$4.4 \cdot 10^{8}$	72 seconds

Table 2.1: Results for reduced delay bandpass filter design ($N = 31$). Chebyshev errors δ, numbers of floating point operations, and design times for different algorithms. Design times and numbers of floating point operations were measured using Matlab 5.0 on a 166 MHz Pentium PC.

the inefficient post-optimization procedure implemented in the program `cremez`. The solution obtained by this optimization procedure is close to the one obtained by Burnside's and Parks' method, but the computational effort is higher.

The differences between the four solutions are negligible from a practical point of view. Hence, for this low order design example only the design effort might be a criterion to choose a certain method. However, the computational effort is relatively small for all four design algorithms and consequently all four methods are almost equally well suited for low-order filter designs. It appears, however, that there is not much sense in using the generalized Remez algorithm by Karam and McClellan because most designs require the costly post-optimization procedure which can be avoided by directly using some more efficient optimization method based on semi-infinite programming such as Burnside's and Parks' algorithm.

Example 2: Reduced Delay Lowpass Filter ($N = 250$)
We use the following specification of a length $N = 250$ reduced delay low pass filter originally given in [14]:

$$D\left(e^{j\omega}\right) = \begin{cases} e^{-j100\omega}, & 0 \leq \omega \leq 0.46\pi \\ 0, & 0.5\pi \leq \omega \leq \pi \end{cases} \tag{2.52}$$

Three designs are computed using Burnside's and Parks' algorithm, Karam's and McClellan's algorithm, and Lawson's algorithm. The filters are computed using the Matlab programs `burnpark`, `cremez`, and `lawson` given in Section 2.2.4. Linear programming is not used because of the large amount of memory that would be required to obtain an accurate solution for this filter order. Burnside's and Parks' algorithm is stopped as soon as the relative deviation between the total maximum error and the maximum error of the actual subproblem is smaller than 10^{-3}. Lawson's algorithm is stopped after 20 iteration steps because the accuracy of the solution is considered to be sufficient. However, the

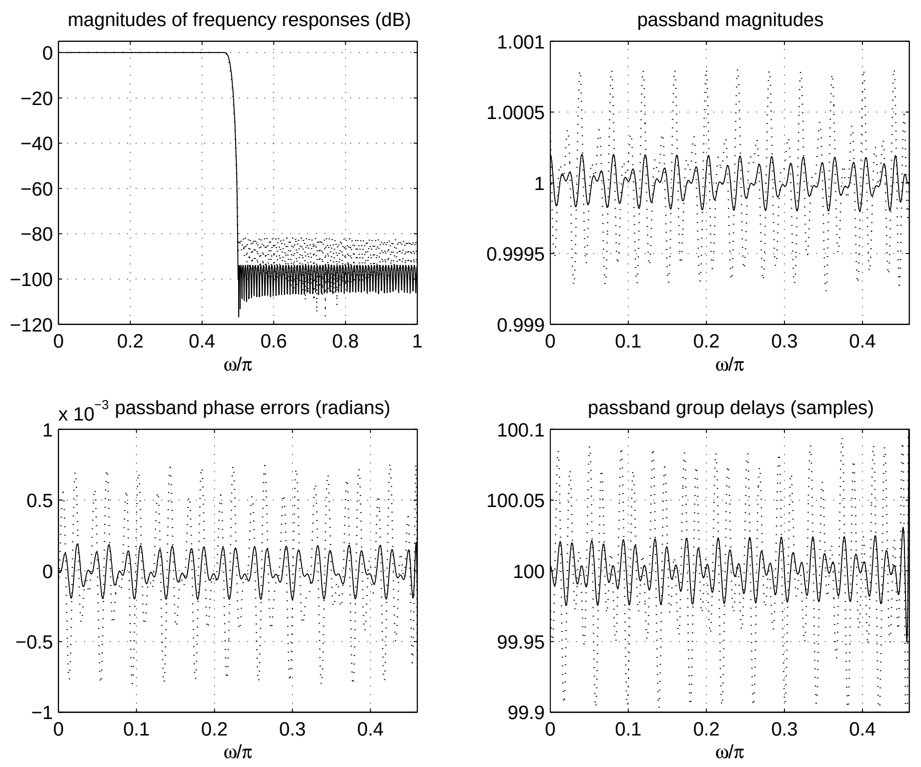


Figure 2.11: Reduced delay low pass filters (length $N = 250$). Solid curves: Burnside's and Parks' algorithm. Dotted curves: Karam's and McClellan's algorithm with post-optimization.

same accuracy as the one obtained by Burnside's and Parks' algorithm can be achieved by adding another 23 iteration steps. Since only little is gained, the improvement does not seem worth the effort. Karam's and McClellan's algorithm yields a suboptimum solution after 14 multiple exchange steps which is optimum on 93% of the original frequency set. However, the maximum error is one order of magnitude larger than the optimum error. Using the post-optimization procedure implemented in the program `cremez` results in an extremely high computational effort and the corresponding solution is approximately 12 dB from the optimum. Figure 2.11 shows the designed filters for Burnside's and Parks' method (solid curves) and for Karam's and McClellan's algorithm with post-optimization (dotted curves). The solution obtained by Lawson's algorithm could not be distinguished from the one obtained by Burnside's and Parks' algorithm on the plot. Table 2.2 summarizes the results for the different algorithms. It turns out that for this specification Lawson's algorithm is by far the most efficient one providing a very accurate solution.

2.2.4 Matlab Programs

This section provides simple Matlab programs implementing some of the algorithms discussed in Section 2.2.3. Their purpose is to illustrate how these algorithms work rather

Method	δ	flops	time
Burnside & Parks	$2.02 \cdot 10^{-4}$	$9.8 \cdot 10^{10}$	2 hours
Lawson	$2.03 \cdot 10^{-4}$	$3.1 \cdot 10^{8}$	1.5 minutes
Karam & McClellan	$8.12 \cdot 10^{-4}$	$7.7 \cdot 10^{11}$	8 hours

Table 2.2: Results for reduced delay lowpass filter ($N = 250$). Chebyshev errors δ, numbers of floating point operations, and design times for different algorithms. Design times and numbers of floating point operations were measured using Matlab 5.0 on a 166 MHz Pentium PC.

than to provide a fast and numerically optimal implementation.

The program `mpcheb` implements the general linear programming approach to complex Chebyshev approximation with arbitrary magnitude and phase weighting as described in Section 2.2.3 and in [58]. The input variables are the filter length `N`, the frequency grid given by the vector `om` on which the desired frequency response (complex vector `D`) and the weighting functions (real vectors `Wm` and `Wp` for magnitude and phase weighting) are defined. The scalars `pp` and `ps` are the linearization parameters p used in (2.43) in the passbands and stopbands, respectively. They define the number of vertices of the regular polygons approximating the circular error region in the complex plane. The best linear approximation to the magnitude/phase approximation problem is achieved by using a rectangular error region in the passbands. This is achieved by choosing `pp` $= 2$.

```
function h = mpcheb(N,om,D,Wm,Wp,pp,ps)
% h = mpcheb(N,om,D,Wm,Wp,pp,ps)
% Complex Chebyshev FIR Filter Design using linear programming
% with independent weighting of magnitude and phase errors
%
% h       filter impulse response
% N       filter length
% om      frequency grid (0 <= om <= pi)
% D       complex desired frequency response on the grid om
% Wm      positive magnitude weighting function on the grid om
% Wp      positive phase weighting function on the grid om
%         (irrelevant in the stopbands)
% pp,ps   integer linearization parameters for passbands and stopbands
%         (pp,ps >= 2);
%         for complex Chebyshev approximation the linearization error is
%         sec(pi/(2*p))-1;
%         for magnitude/phase approximation choose pp=2;
%
% example (Chen & Parks): length 31 lowpass, band edges [.12 .24]*pi,
% weighting 1 in passband and 10 in stopband, desired passband group
% delay 12 samples, magnitude more important than phase
%
% om=pi*[linspace(0,.12,50),linspace(.24,1,250)];
% D=[exp(-j*om(1:50)*12),zeros(1,250)];
% Wm=[ones(1,50),10*ones(1,250)]; Wp=[.1*ones(1,50),zeros(1,250)];
% h = mpcheb(31,om,D,Wm,Wp,2,6);
%
% Author: Mathias C. Lang, Vienna University of Technology, Nov. 97

om = om(:); D = D(:); Wm = Wm(:); Wp = Wp(:); L = length(om);
Dmag = abs(D); Dph = angle(D);
pass = find(Dmag > 0); stop = find(Dmag == 0);
Lp = length(pass); Ls = length(stop);
Wp(pass) = Wp(pass)./Dmag(pass);

A = zeros(pp*Lp + ps*Ls,N); b = zeros(pp*Lp + ps*Ls,1);
alpha_p = pi*(0:pp-1)/pp; alpha_s = pi*(0:ps-1)/ps;

for i=1:pp,       % passbands
   I = (i-1)*Lp+(1:Lp);
   A(I,:) = Wm(pass,ones(N,1)).*...
      cos(om(pass)*(0:N-1) + Dph(pass,ones(N,1)))*cos(alpha_p(i))+...
      Wp(pass,ones(N,1)).*...
      sin(om(pass)*(0:N-1) + Dph(pass,ones(N,1)))*sin(alpha_p(i));
   b(I) = Wm(pass).*Dmag(pass)*cos(alpha_p(i));
end

if Ls > 0,
   for i=1:ps,       % stopbands
      I = pp*Lp + (i-1)*Ls+(1:Ls);
      A(I,:) = Wm(stop,ones(N,1)).*cos(alpha_s(i) - om(stop)*(0:N-1));
   end
end

A = [-ones(2*(pp*Lp+ps*Ls),1),[A;-A]]; b = [b;-b];
x0 = [max(b)+1;zeros(N,1)];   % feasible initial solution
x = lp([1;zeros(N,1)],A,b,[],[],x0);
h = x(2:N+1);
```

The program `burnpark` is an implementation of Burnside's and Parks' algorithm [14]. It should be noted that the authors made their implementation available. Their program is written in Matlab and C which results in a considerably increased execution speed. However, to make a fair comparison we chose to use this very simple Matlab program for the design examples in Section 2.2.3. For solving the linear programming subproblems the function `lp` contained in the Matlab Optimization Toolbox is used. This implementation of a linear programming algorithm is not very well suited for our purpose. Hence, the design algorithm is relatively slow. Our implementation uses a frequency grid because this allows to specify arbitrary desired frequency responses most easily. The grid can be chosen to be very dense without considerably increasing the computational effort. The complexity is determined mainly by the desired filter length. The input variables are similar to those of the program `mpcheb`. Independent weighting of magnitude and phase errors is not implemented.

```
function [h,delta] = burnpark(N,om,D,W);
% [h,delta] = burnpark(N,om,D,W)
% Complex Chebyshev FIR filter design using
% Burnside & Parks method
%
% h      filter impulse response
% delta  maximum approximation error
% N      filter length
% om     frequency grid (0 <= om <= pi)
% D      complex desired frequency response on the grid om
% W      positive weighting function on the grid om
%
% example: length 41 lowpass, band edges [.4 .5]*pi,
% weighting 1 in passband and 10 in stopband,
% desired passband group delay 12 samples
%
% om=pi*[linspace(0,.4,400),linspace(.5,1,500)];
% D=[exp(-j*om(1:400)*12),zeros(1,500)];
% W=[ones(1,400),10*ones(1,500)];
% [h,delta] = burnpark(41,om,D,W);
%
% Author: Mathias C. Lang, Vienna University of Technology, Nov. 97

tol = 1e-3;
om = om(:); D = D(:); W = W(:); L = length(om);

% set up initial problem
L_ini = 2*N;
I_ini = (1:max(fix((L-1)/(L_ini-1)),1):L);
om_ini = om(I_ini); D_ini = D(I_ini); W_ini = W(I_ini);

AR = cos(om_ini*(0:N-1)); AI = sin(om_ini*(0:N-1)); w = 1./W_ini;
br = real(D_ini); bi = imag(D_ini);
A = [-w -AR; -w AR; -w AI; -w -AI];
b = [-br; br; -bi; bi]; c = [1;zeros(N,1)];
x0 = [max(abs([br;bi]./[w;w]));zeros(N,1)];  % feasible starting solution

% start iteration
while 1,

% solve lp-problem
[x,lam] = lp(c,A,b,[],[],x0);
delta = x(1)
h = x(2:N+1);
x0 = x;

% compute complex error function
```

```
E = W.*(D-freqz(h,1,om));
magE = abs(E); argE = angle(E);
maxerr = max(magE);
x0(1) = maxerr;   % make x0 feasible for next iteration

% compute local maxima of error magnitude
Imax = locmax(magE);
ommax = om(Imax); argEmax = argE(Imax);
magDmax = abs(D(Imax)); argDmax = angle(D(Imax));
Wmax = W(Imax);

% make a plot
plot(om/pi,magE,ommax/pi,magE(Imax),'or'); grid
xlabel('\omega/\pi'); title('Magnitude of Complex Error');
drawnow

% check stopping criterion
if (maxerr-delta)/delta < tol, break; end

% build new constraints
Anew = -[1./Wmax cos(ommax*(0:N-1)+argEmax(:,ones(1,N)))];
bnew = -magDmax.*cos(argDmax-argEmax);

% update constraint set
A = [A;Anew]; b = [b;bnew];

end
```

The program `locmax` called by `burnpark` determines the indices of the local maxima of a sequence of real numbers.

```
function I = locmax(x)
% LOCMAX: I = locmax(x)
% finds indices of local maxima of data x
x = x(:); n = length(x);
if n, I = find(x > [x(1)*(1-eps)-1;x(1:n-1)] & ...
               x > [x(2:n);x(n)*(1-eps)-1]);
else, I = []; end
```

An implementation of Lawson's algorithm is provided by the program `lawson`. In order to retain simplicity, no modifications to avoid zero weights or to increase speed of convergence have been made. However, for practical filter design problems this program usually works fine. Since final convergence may be slow, the number of iterations can be prescribed by the input variable `iter`. Usually, initial convergence is relatively fast such that a reasonably good solution is obtained after only a few iterations. The program `lawson` calls the program `lslevin` introduced in Section 2.1.4 to solve the weighted least squares subproblems.

```
function h = lawson(N,om,D,W,iter)
% h = lawson(N,om,D,W,iter)
% Complex Chebyshev FIR filter design using Lawson's algorithm
%
% h      filter impulse response
% N      filter length
% om     frequency grid (0 <= om <= pi)
% D      complex desired frequency response on the grid om
% W      positive weighting function on the grid om
% iter   maximum number of iterations
%
% example: length 61 bandpass, band edges [.23,.3,.5,.57]*pi,
% weighting 1 in passband and 10 in stopbands, desired passband
% group delay 20 samples
%
% om=pi*[linspace(0,.23,230),linspace(.3,.5,200),linspace(.57,1,430)];
% D=[zeros(1,230),exp(-j*om(231:430)*20),zeros(1,430)];
% W=[10*ones(1,230),ones(1,200),10*ones(1,430)];
% h = lawson(61,om,D,W,20);
%
% Author: Mathias C. Lang, Vienna University of Technology, Jan. 1998

om = om(:); D = D(:); W = W(:); L = length(om);

% Initialize, define
tol = 1e-6; h = zeros(N,1); evec = exp(-j*om);

% Initial IRLS weighting w
w = W.^2;

% Main loop
for i = 1:iter,

   % Compute weighted l2 solution
   h0 = h;
   h = lslevin(N,om,D,w);

   % Compute magnitude of complex error
   E = abs(polyval(h(N:-1:1),evec) - D);

   % Stop if nothing changes
   dh = norm(h-h0)
   if dh < tol, break; end

   % Update weight function
   update = W.*E;
   w = w.*update/(w'*update);

end   % main loop
```

2.3 Constrained Complex Approximation

The design methods discussed so far aim at minimizing a specified error measure. The mean squared error and the maximum error are error measures of practical relevance. However, it may be desired not only to minimize these error measures but also to specify error bounds that must not be exceeded. In fact, the basic motivation for using a Chebyshev error criterion for the frequency domain design of filters is to satisfy a tolerance scheme using the smallest filter order. However, if the design problem is formulated as the complex Chebyshev approximation problem

$$\underset{\boldsymbol{h}}{\text{minimize}} \ \max_{\omega\in\Omega} W(\omega)|E_c(\omega,\boldsymbol{h})|,$$

then there is no possibility to explicitly specify this tolerance scheme in the mathematical problem formulation. The maximum approximation error attained by the solution is rather a result of the design process. The approximation error can only be influenced by using a weighting function. Several designs with different weighting functions have to be computed until the approximation error satisfies its desired bounds. A more useful design method should allow for a specification like *"minimize the maximum error in the passband and fix the minimum stopband attenuation at 60 dB"*. This is an example of constrained Chebyshev approximation where the error measure is minimized in some frequency band and it is upper bounded in some other frequency band. We could think of a similar problem formulation using the least squared error criterion. However, as opposed to bounds for maximum errors, bounds on mean squared errors rarely arise in practical problems. Interesting designs result from using a least squared error criterion in combination with bounds on maximum errors. This constrained least squares problem formulation can result in a mix of least squares and Chebyshev error criteria. In frequency bands where the error bounds are loose, the least squared error criterion is dominant, whereas in bands where the maximum error bounds are tight, the error will exhibit a Chebyshev behavior with its maximum given by the specified constraint.

It turns out that there are basically two motivations for generalizing the standard error criteria:

1. prescribe error bounds,

2. mix different error criteria.

The problem formulations including prescriptions of error bounds will be discussed in Sections 2.3.2 and 2.3.3. They directly follow from the desired error criterion and from the error measure that is to be bounded. Due to its practical relevance we will focus on constraining maximum errors.

It is more difficult to find a useful problem formulation that provides the possibility to mix Chebyshev and least squared error criteria. One possibility is the constrained least squares problem formulation already mentioned and discussed in more detail in Section 2.3.2. A completely different approach is the use of general L_p-norms as optimality criteria. For $p = 2$ and $p \to \infty$ the least squares and Chebyshev optimality criteria are obtained, respectively. For $2 < p < \infty$ the solutions are in between and a continuous transition from L_2 to L_∞ can be achieved. This approach has been discussed in [15] for the design of linear phase FIR filters and has been extended in [11] to complex approximation problems with arbitrary desired magnitude and phase responses. Yet another way to formulate a problem for mixed criteria filter design is to use the error measure

$$\epsilon(\boldsymbol{h}) = \alpha\epsilon_C(\boldsymbol{h}) + (1-\alpha)\epsilon_{LS}(\boldsymbol{h}), \quad 0 \le \alpha \le 1, \tag{2.53}$$

with the Chebyshev and least squares error measures $\epsilon_C(\boldsymbol{h})$ and $\epsilon_{LS}(\boldsymbol{h})$ given by

$$\begin{aligned} \epsilon_C(\boldsymbol{h}) &= \max_{\omega\in\Omega} W_C(\omega)\left|E_c(\omega,\boldsymbol{h})\right|, && (2.54)\\ \epsilon_{LS}(\boldsymbol{h}) &= \int_\Omega W_{LS}(\omega)\left|E_c(\omega,\boldsymbol{h})\right|^2 d\omega. && (2.55)\end{aligned}$$

By varying the parameter α from $\alpha = 0$ to $\alpha = 1$ a continuous transition from complex least squares to complex Chebyshev approximation is achieved. In analogy to complex Chebyshev approximation, we can formulate the problem of minimizing the error measure (2.53) as a semi-infinite constrained optimization problem by introducing an additional unknown parameter δ:

$$\begin{aligned} &\underset{\boldsymbol{h},\,\delta}{\text{minimize}} && \alpha\delta + (1-\alpha)\epsilon_{LS}(\boldsymbol{h}) && (2.56)\\ &\text{subject to} && W_C(\omega)|E_c(\omega,\boldsymbol{h})| \le \delta, \quad \omega\in\Omega. \end{aligned}$$

This problem formulation is interesting for investigating the trade-off between complex Chebyshev and complex least squares approximation. As an example, we take the specification of the reduced delay band pass filter of length $N = 31$ from Section 2.2.3. Using problem formulation (2.56), we design filters minimizing the error measure (2.53) with the parameter α varying from 0 to 1. Figure 2.12 shows the results with respect to the Chebyshev and least squares error measures $\epsilon_{LS}(\boldsymbol{h})$ and $\epsilon_C(\boldsymbol{h})$. The end points of this trade-off curve ($\alpha = 0$ and $\alpha = 1$) correspond to the standard least squares and Chebyshev solutions, respectively. We see that the respective other error measure is relatively large for these extreme cases. We move along the trade-off curve by varying α. Good solutions that exhibit relatively low least squared errors as well as low maximum errors are obtained at the knee of this curve. We pick one example ($\alpha = 0.76$) and plot the magnitude of

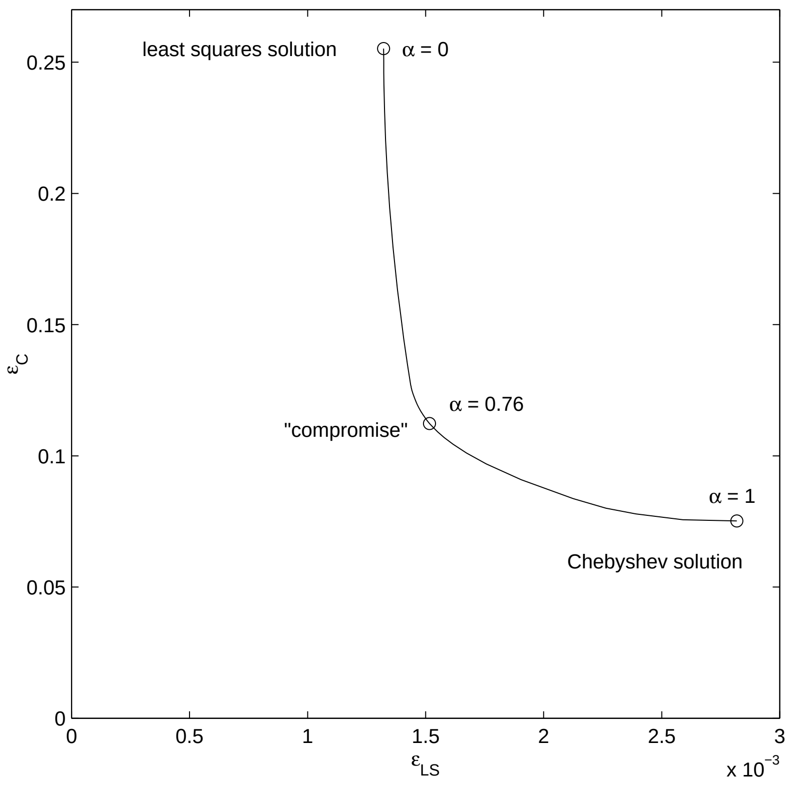


Figure 2.12: Trade-off curve between Chebyshev and least squares approximation for reduced delay bandpass filter ($N = 31$).

the frequency responses in Figure 2.13. The least squares and Chebyshev solution are shown as the dashed and dotted curves, respectively. The solid curves corresponds to the "compromise" filter. This filter exhibits transitions from the passband to the stopbands that are almost as steep as those of the Chebyshev filter. However, the stopband behavior is very similar to the least squares case. The compromise filter has a Chebyshev error 56% less than the one of the optimum least squares filter, and its least squared error is 46% smaller than the one achieved by the Chebyshev filter. The passband details of the three filters regarding magnitude responses and phase errors are shown in Figure 2.14. Trade-off curves as shown by Figure 2.12 have been published before in [2] where the constrained least squares design of linear phase FIR filters is discussed.

Arbitrary trade-offs between Chebyshev and least squares designs can be computed using problem formulation (2.56). However, we do not know how to choose the parameter α in advance to obtain filters satisfying desired specifications in terms of tolerance schemes. Hence, the practical relevance of this problem formulation is relatively small. The constrained least squares problem formulation is much more useful because it allows for the prescription of error bounds. Moreover, all points on the trade-off curve shown in

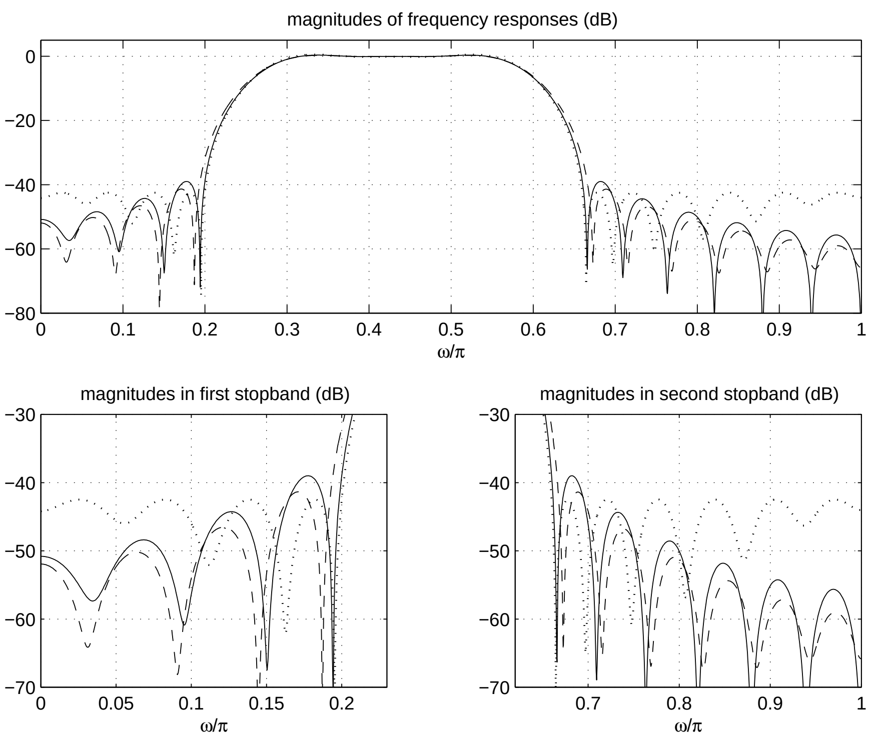


Figure 2.13: Magnitude responses of reduced delay bandpass filters ($N = 31$). Dotted curves: complex Chebyshev approximation. Dashed curves: complex least squares approximation. Solid curves: compromise.

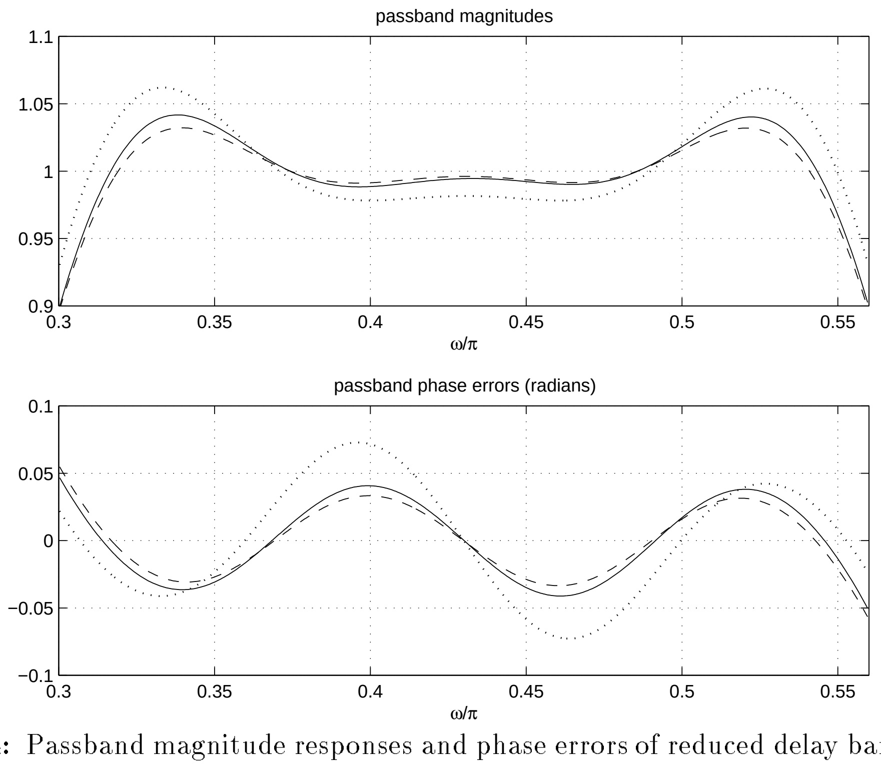


Figure 2.14: Passband magnitude responses and phase errors of reduced delay bandpass filters ($N = 31$). Dotted curves: complex Chebyshev approximation. Dashed curves: complex least squares approximation. Solid curves: compromise.

Figure 2.12 can be obtained by solving constrained least squares problems.

2.3.1 Error Functions

In frequency domain design of filters with arbitrarily specified magnitude and phase responses the questions arises which error functions to incorporate into the design problem. As opposed to the design of linear phase filters, the general frequency domain filter design problem is a complex approximation problem. Hence, error functions that can be considered are the complex error function $E_c(\omega)$, the magnitude error function $E_m(\omega)$, and the phase error function $E_\phi(\omega)$. Instead of the phase error, one could consider the group delay error

$$E_\tau(\omega) = \tau(\omega) - \tau_d(\omega), \tag{2.57}$$

where $\tau(\omega)$ is the actual group delay given by

$$\tau(\omega) = -\frac{\partial \arg\{H\left(e^{j\omega}\right)\}}{\partial\omega} = \operatorname{Re}\left\{\frac{\sum\limits_{k=-\infty}^{\infty} kh(k)e^{-jk\omega}}{\sum\limits_{k=-\infty}^{\infty} h(k)e^{-jk\omega}}\right\}, \tag{2.58}$$

and $\tau_d(\omega)$ is the desired group delay.

If the complex error function is used, it is not necessary to include any other error functions in the design problem, because the complex error incorporates magnitude and phase errors. Assume $|E_c(\omega)| \leq \delta_c(\omega)$ is satisfied, either by minimizing a norm of the weighted complex error function or by imposing constraints. Then simple geometric considerations lead to the following inequalities for magnitude and phase errors (see also Figure (2.15)):

$$|E_m(\omega)| \leq \delta_c(\omega), \tag{2.59}$$

$$|E_\phi(\omega)| \leq \arcsin\left(\frac{\delta_c(\omega)}{|D\left(e^{j\omega}\right)|}\right), \tag{2.60}$$

where the phase error $E_\phi(\omega)$ is only defined in the passbands where $|D\left(e^{j\omega}\right)| > 0$ holds. However, using the complex error function does not provide the possibility to independently optimize or constrain magnitude and phase errors. Hence, another useful possibility for formulating the design problem is to use magnitude and phase error functions. This results in a much more general problem formulation where magnitude and phase errors can be completely decoupled. However, the corresponding problems are more difficult to solve than problems formulated by using the complex error function. This is shown later in this chapter. A third possibility that has been chosen by many authors

(e. g. [16, 17, 26, 121, 128, 129]) is to use magnitude and group delay error functions to formulate the filter design problem.

However, we consider the phase error to be a more relevant quantity than the group delay error. This can be argued as follows. Since in stopbands ($|D(e^{j\omega})| = 0$) phase and group delay errors are not defined and only the magnitude error matters, we consider a design problem where the desired frequency response does not contain any stopbands, i. e. $|D(e^{j\omega})| > 0,\ \omega \in [-\pi, \pi)$. Assume the desired complex frequency response $D(e^{j\omega})$ is specified everywhere on $[-\pi, \pi)$, i. e. there are no "don't care bands". The actual frequency response $H(e^{j\omega})$ approximating $D(e^{j\omega})$ can be written as

$$\begin{aligned} H(e^{j\omega}) &= D(e^{j\omega})\left[1 + \frac{E_m(\omega)}{|D(e^{j\omega})|}\right] e^{jE_\phi(\omega)} \\ &= D(e^{j\omega}) + D(e^{j\omega})\left\{\left[1 + \frac{E_m(\omega)}{|D(e^{j\omega})|}\right] e^{jE_\phi(\omega)} - 1\right\} \\ &= D(e^{j\omega}) + D(e^{j\omega})\, C(e^{j\omega}) \qquad (2.61) \end{aligned}$$

where $E_m(\omega)$ and $E_\phi(\omega)$ are magnitude and phase errors, respectively. Let $h_d(n)$ be the impulse response corresponding to the desired frequency response $D(e^{j\omega})$. From (2.61), the impulse response corresponding to the actual system is

$$h(n) = h_d(n) + \sum_{k=-\infty}^{\infty} c(k) h_d(n-k). \qquad (2.62)$$

The coefficients $c(k)$ in (2.62) are given by

$$c(k) = \frac{1}{2\pi}\int_{-\pi}^{\pi} C(e^{j\omega})\, e^{jk\omega} d\omega. \qquad (2.63)$$

Accordingly, the response $y(n)$ of the actual system to an arbitrary input signal $x(n)$ is

$$y(n) = y_d(n) + \sum_{k=-\infty}^{\infty} c(k) y_d(n-k), \qquad (2.64)$$

where $y_d(n)$ is the response of the ideal system with frequency response $D(e^{j\omega})$ to the input signal $x(n)$. According to (2.64), the actual output signal $y(n)$ consists of the ideal output $y_d(n)$ and undesired echo terms with weights $c(k)$ caused by the magnitude and phase approximation errors. The magnitude of the echo weights $c(k)$ can be upper bounded as follows:

$$|c(k)| = \frac{1}{2\pi}\left|\int_{-\pi}^{\pi}\left\{\left[1 + \frac{E_m(\omega)}{|D(e^{j\omega})|}\right] e^{jE_\phi(\omega)} - 1\right\} e^{jk\omega} d\omega\right| \le$$

$$
\begin{aligned}
&\leq \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\left[1+\frac{E_m(\omega)}{|D(e^{j\omega})|}\right]e^{jE_\phi(\omega)}-1\right|d\omega \;=\\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|e^{jE_\phi(\omega)}-1+\frac{E_m(\omega)}{|D(e^{j\omega})|}e^{jE_\phi(\omega)}\right|d\omega \;\leq\\
&\leq \frac{1}{2\pi}\int_{-\pi}^{\pi}\left[\left|e^{jE_\phi(\omega)}-1\right|+\left|\frac{E_m(\omega)}{D(e^{j\omega})}\right|\right]d\omega \;=\\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi}\left[2\left|\sin\left(\frac{E_\phi(\omega)}{2}\right)\right|+\left|\frac{E_m(\omega)}{D(e^{j\omega})}\right|\right]d\omega \;\leq\\
&\leq \frac{1}{2\pi}\int_{-\pi}^{\pi}\left[|E_\phi(\omega)|+\left|\frac{E_m(\omega)}{D(e^{j\omega})}\right|\right]d\omega \;\leq\\
&\leq \max_{\omega\in[-\pi,\pi)}\left\{|E_\phi(\omega)|+\left|\frac{E_m(\omega)}{D(e^{j\omega})}\right|\right\}. \qquad (2.65)
\end{aligned}
$$

Note that only relative magnitude errors matter. From (2.65) it follows that minimizing or constraining magnitude and phase errors minimizes or constrains the maximum time domain echo amplitudes. This is not the case if magnitude and group delay errors are considered in the design problem. We are not aware of any useful interpretation of group delay error minimization or group delay error constraints. Consequently, in the following we will only consider magnitude and phase errors. This is achieved by using the complex error function or, more generally, by using magnitude and phase error functions for formulating design problems.

After having decided which error functions to use for formulating filter design problems, we now discuss the mathematical properties of these problems. It is of particular interest whether the functions involved are convex. Furthermore, it is relevant if constraints on these error functions result in convex feasible regions with respect to the coefficients $\boldsymbol{h}$. Refer to Appendix A for the definition and implications of convexity.

The magnitude of the complex error function $E_c(\omega,\boldsymbol{h})$ as defined by (1.1) is a convex function with respect to $\boldsymbol{h}$ for all frequencies ω. This is easily shown as follows. Define

$$\boldsymbol{h}(\alpha)=\alpha\boldsymbol{h}_1+(1-\alpha)\boldsymbol{h}_2. \qquad (2.66)$$

From linearity of $E_c(\omega,\boldsymbol{h})$ with respect to $\boldsymbol{h}$ and from the triangular inequality, it follows that

$$
\begin{aligned}
|E_c(\omega,\boldsymbol{h}(\alpha))| &= |\alpha E_c(\omega,\boldsymbol{h}_1)+(1-\alpha)E_c(\omega,\boldsymbol{h}_2)| \qquad (2.67)\\
&\leq \alpha|E_c(\omega,\boldsymbol{h}_1)|+(1-\alpha)|E_c(\omega,\boldsymbol{h}_2)|,\quad \forall\alpha\in[0,1],\quad \forall\boldsymbol{h}_1,\boldsymbol{h}_2\in\mathbb{R}^N,
\end{aligned}
$$

must hold. Inequality (2.67) is the definition of a convex function as given by (A.14). It follows that also the functions $\max_{\omega\in\Omega}W(\omega)|E_c(\omega,\boldsymbol{h})|$ and $\int_\Omega W(\omega)|E_c(\omega,\boldsymbol{h})|^2d\omega$ with arbitrary non-negative $W(\omega)$ are convex. Moreover, the set

$$K = \left\{ \boldsymbol{h} \in \mathbb{R}^N \middle| \; |E_c(\omega, \boldsymbol{h})| \leq \delta_c(\omega) \right\}$$

is convex. Hence, any Chebyshev or least squares design problem which uses $|E_c(\omega, \boldsymbol{h})|$ in its objective function, and which imposes constraints on $|E_c(\omega, \boldsymbol{h})|$ is a convex optimization problem.

The magnitude error function $E_m(\omega, \boldsymbol{h}) = |H(e^{j\omega}, \boldsymbol{h})| - |D(e^{j\omega})|$ is a convex function due to the convexity of $|H(e^{j\omega}, \boldsymbol{h})|$. Convexity of $|H(e^{j\omega}, \boldsymbol{h})|$ with respect to $\boldsymbol{h}$ is shown in a completely analogous manner as convexity of $|E_c(\omega, \boldsymbol{h})|$. However, the magnitude $|E_m(\omega, \boldsymbol{h})|$ is not convex because the magnitude of a convex function cannot be convex unless the function is non-negative. This is generally not the case for the function $E_m(\omega, \boldsymbol{h})$. Hence, all design problems using $|E_m(\omega, \boldsymbol{h})|$ in their objective functions are non-convex problems in general. Moreover, the set defined by constraints on $|E_m(\omega, \boldsymbol{h})|$ is non-convex. Constraints on $|E_m(\omega, \boldsymbol{h})|$ can be written as upper and lower bound constraints on $E_m(\omega, \boldsymbol{h})$. Due to convexity of $E_m(\omega, \boldsymbol{h})$, upper bound constraints correspond to a convex set. However, convexity of $E_m(\omega, \boldsymbol{h})$ implies that lower bound constraints correspond to a non-convex set. Consequently, constraints on $|E_m(\omega, \boldsymbol{h})|$ correspond to a non-convex feasible region, and any design problem imposing constraints on $|E_m(\omega, \boldsymbol{h})|$ is non-convex.

The phase error $E_\phi(\omega, \boldsymbol{h}) \in [-\pi, \pi)$ is defined as the principal value of the phase difference between the actual and the desired complex frequency responses and can be written as follows:

$$E_\phi(\omega, \boldsymbol{h}) = \arg\left\{ H\left(e^{j\omega}, \boldsymbol{h}\right) e^{-j\phi_d(\omega)} \right\}, \quad \omega \in \Omega^p, \tag{2.68}$$

where $\phi_d(\omega) = \arg\{D(e^{j\omega})\}$ is the desired phase response. Note that $E_\phi(\omega, \boldsymbol{h})$ is only defined in the passbands Ω^p. Only its magnitude $|E_\phi(\omega, \boldsymbol{h})|$ is of interest. For $|E_\phi(\omega, \boldsymbol{h})|$ to be convex with respect to $\boldsymbol{h}$ it must satisfy

$$|E_\phi(\omega, \boldsymbol{h}(\alpha))| \leq \alpha |E_\phi(\omega, \boldsymbol{h}_1)| + (1-\alpha)|E_\phi(\omega, \boldsymbol{h}_2)|, \quad \forall \alpha \in [0,1], \quad \forall \boldsymbol{h}_1, \boldsymbol{h}_2 \in \mathbb{R}^N, \tag{2.69}$$

with $\boldsymbol{h}(\alpha)$ given by (2.66). However, it is always possible to construct $\boldsymbol{h}(\alpha)$ for some $\alpha \in (0,1)$ and $\omega \in \Omega^p$ such that $|E_\phi(\omega, \boldsymbol{h}(\alpha))| = \pi$ and $|E_\phi(\omega, \boldsymbol{h}_1)| < \pi$ is satisfied. This contradicts the convexity assumption (2.69). Consequently, $|E_\phi(\omega, \boldsymbol{h})|$ is not convex and results in non-convex objective functions if some norm of the phase error is to be minimized. Let Ω_B denote the union of all frequency regions where error bounds are desired. If the feasible region $K = \left\{ \boldsymbol{h} \in \mathbb{R}^N \middle| \; |E_\phi(\omega, \boldsymbol{h})| \leq \delta_\phi(\omega), \;\; \omega \in \Omega^p \cap \Omega_B \right\}$ defined by constraints on the phase error is convex, the following inequality must hold:

$$|E_\phi(\omega, \boldsymbol{h}(\alpha))| \leq \delta_\phi(\omega), \quad \forall \omega \in \Omega^p \cap \Omega_B, \quad \forall \alpha \in [0,1], \quad \forall \boldsymbol{h}_1, \boldsymbol{h}_2 \in K, \tag{2.70}$$

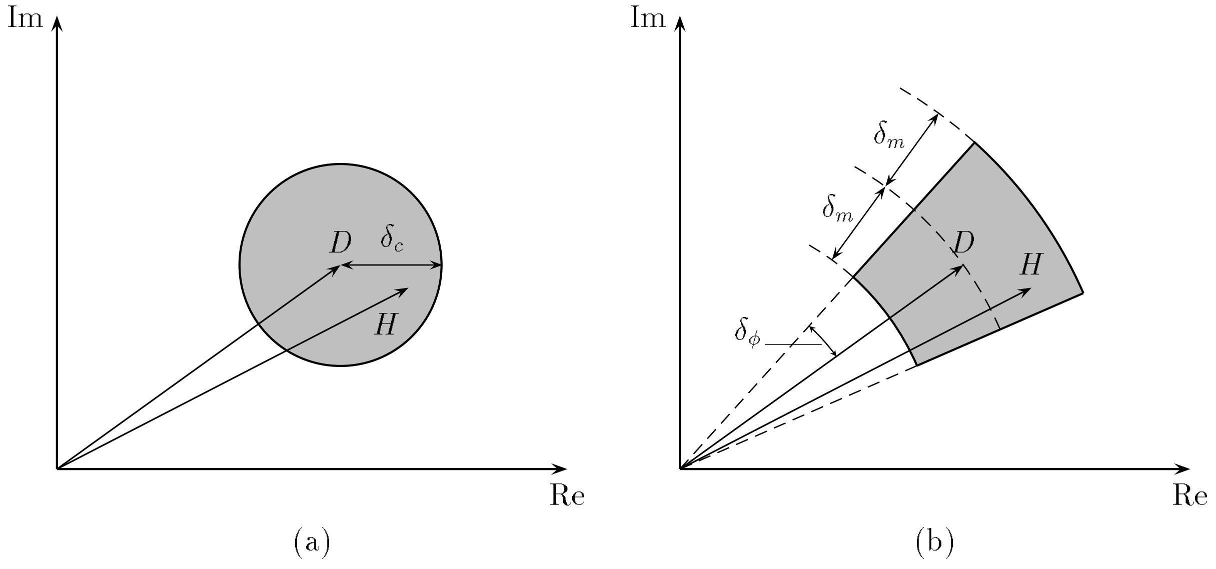


Figure 2.15: Error constraints illustrated in the complex plane for one passband frequency point $\omega \in \Omega^p$. The desired complex frequency response $D\left(e^{j\omega}\right)$ is approximated by $H\left(e^{j\omega}\right)$. (a) The magnitude of the complex error function is upper bounded by the function $\delta_c(\omega)$. (b) Magnitude and phase errors are upper bounded by the functions $\delta_m(\omega)$ and $\delta_\phi(\omega)$, respectively.

with $\boldsymbol{h}(\alpha)$ defined by (2.66). Note that by linearity of $H\left(e^{j\omega}, \boldsymbol{h}\right)$ with respect to $\boldsymbol{h}$,

$$|E_\phi(\omega, \boldsymbol{h}(\alpha))| = \left|\arg\left\{\alpha H(e^{j\omega}, \boldsymbol{h}_1)e^{-j\phi_d(\omega)} + (1-\alpha)H(e^{j\omega}, \boldsymbol{h}_2)e^{-j\phi_d(\omega)}\right\}\right|$$

holds. Hence, inequality (2.70) is satisfied if and only if the following inequality holds for arbitrary complex $z_1, z_2 \neq 0$:

$$|\arg\{\alpha z_1 + (1-\alpha)z_2\}| \leq \max\{|\arg(z_1)|, |\arg(z_2)|\}, \quad \forall \alpha \in [0,1]. \tag{2.71}$$

Note that by $\arg(\cdot)$ we denote the principal value of the argument of a complex number. It is easy to visualize in the complex plane that (2.71) holds only if

$$|\arg(z_1) - \arg(z_2)| \leq \pi \tag{2.72}$$

is satisfied. If z_1 and z_2 satisfy $|\arg(z_1)| \leq \delta_\phi$ and $|\arg(z_2)| \leq \delta_\phi$, then for (2.72) to hold, $\delta_\phi \leq \pi/2$ must be satisfied. Consequently, constraints on the magnitude of the phase error correspond to a convex feasible region if the constraint function satisfies $\delta_\phi(\omega) \leq \pi/2$.

Figure 2.15 illustrates the constraints on the various error functions in the complex plane for one frequency point in the passbands. Figure 2.15 (a) shows the constraint on the magnitude of the complex error function. As opposed to complex Chebyshev approximation, the radius of the circle in the complex plane is now specified by the constraint function $\delta_c(\omega)$. Figure 2.15 (b) shows independent constraints on magnitude and phase errors. In the stopbands, it does not make a difference whether constraints are imposed on the complex error or on the magnitude error since $|E_c(\omega)| = |E_m(\omega)|$ holds. Hence, in both cases the region inside which $H\left(e^{j\omega}\right)$ is forced to remain is a circle around the origin in the complex plane. Note that due to linearity of $H\left(e^{j\omega}, \boldsymbol{h}\right)$ with respect to the coefficients $\boldsymbol{h}$ there is a one to one correspondence between the convexity of the shaded regions in the complex plane shown in Figure 2.15 and the convexity of the feasible region defined in N-dimensional coefficient space. If the feasible region is convex, then the region in the complex plane is convex for any frequency $\omega \in \Omega_B$. On the other hand, if the region in the complex plane is convex for all $\omega \in \Omega_B$ then the feasible region is convex as well. With these considerations in mind it is now obvious from Figure 2.15 (a) that constraints on the magnitude of the complex error imply a convex feasible region. Similarly, from Figure 2.15 (b), upper bound constraints on the magnitude error are obviously convex. However, lower bound constraints on $E_m(\omega)$ cause the shaded region in Figure 2.15 (b) to become non-convex for any frequency. Hence, these constraints result in a non-convex feasible region. Finally, from Figure 2.15 (b), phase constraints are convex if $\delta_\phi(\omega) \leq \pi/2$ is satisfied. All these statements about convexity agree with the previously derived results.

Note that in Figure 2.15 (b), phase constraints are represented by linear inequality constraints with respect to $\text{Re}[H\left(e^{j\omega}, \boldsymbol{h}\right)]$ and $\text{Im}[H\left(e^{j\omega}, \boldsymbol{h}\right)]$ as long as $\delta_\phi(\omega) \leq \pi/2$ is satisfied. Since $\text{Re}[H\left(e^{j\omega}, \boldsymbol{h}\right)]$ and $\text{Im}[H\left(e^{j\omega}, \boldsymbol{h}\right)]$ are linear functions with respect to $\boldsymbol{h}$, the convex phase constraints are also linear with respect to $\boldsymbol{h}$. This can be shown more formally as follows. The constraint $|E_\phi(\omega, \boldsymbol{h})| \leq \delta_\phi(\omega)$ is equivalent to

$$-\delta_\phi(\omega) \leq E_\phi(\omega, \boldsymbol{h}) \leq \delta_\phi(\omega), \tag{2.73}$$

because the phase error $E_\phi(\omega, \boldsymbol{h})$ is a real-valued function. From (2.68), the phase error can be written as

$$E_\phi(\omega, \boldsymbol{h}) = \arctan\left\{\frac{\text{Im}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right]}{\text{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right]}\right\} \tag{2.74}$$

if $|E_\phi(\omega, \boldsymbol{h})| < \pi/2$ holds. Plugging (2.74) into (2.73) and noting that $\tan(x)$ is monotonic and non-negative on $[0, \pi/2)$ yields the following constraints that are equivalent to (2.73) if $\delta_\phi < \pi/2$ holds:

$$-\tan\delta_\phi(\omega) \leq \frac{\text{Im}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right]}{\text{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right]} \leq \tan\delta_\phi(\omega), \quad \delta_\phi(\omega) < \pi/2. \tag{2.75}$$

Multiplying the constraints (2.75) by $\text{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right]$ yields exactly equivalent constraints that are linear with respect to the coefficients $\boldsymbol{h}$. Note that this is only possible and valid for $\delta_\phi(\omega) < \pi/2$. The case $\delta_\phi(\omega) = \pi/2$ can be handled by the linear constraint

$$\text{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right] \geq 0, \quad \delta_\phi(\omega) = \pi/2.$$

The result that phase constraints are exactly represented by linear constraints for $\delta_\phi(\omega) \leq \pi/2$ has been used independently by the author of this thesis in [59] and by the authors of [92].

2.3.2 Constrained Least-Squares Approximation

The constrained least squares filter design problem can be posed as an optimization problem, where approximation error energy is minimized subject to constraints on error functions of interest. From Section 2.3.1, only the complex error function $E_c(\omega, \boldsymbol{h})$ leads to an expression for error energy which is a convex quadratic function with respect to the unknown filter coefficients $\boldsymbol{h}$. If magnitude and phase errors are used to define error energy, the resulting objective function is non-convex which may cause severe difficulties when solving the design problem and prevents the use of simple and efficient algorithms. Moreover, the minimization of error energy is most important in the stopbands of the filter. There, only the complex error and the magnitude error are defined and their magnitudes are equal because $D\left(e^{j\omega}\right) = 0$ holds. In the passbands, magnitude and phase errors can be taken into account by imposing constraints. Hence, the use of the complex error for defining the error energy leads to a simple and meaningful objective function. These considerations lead to two constrained least squares design problems that are of practical interest and that result in tractable optimization problems. One problem imposes constraints on the complex error function, whereas the other constrains magnitude and phase errors independently. Both minimize the same objective function. They are formulated by

$$\begin{aligned} \underset{\boldsymbol{h}}{\text{minimize}} \quad & \int_{\Omega_{min}} W(\omega)|E_c(\omega, \boldsymbol{h})|^2 d\omega \\ \text{subject to} \quad & |E_c(\omega, \boldsymbol{h})| \leq \delta_c(\omega), \quad \omega \in \Omega_B, \end{aligned} \tag{2.76}$$

and by

$$\begin{aligned} \underset{\boldsymbol{h}}{\text{minimize}} \quad & \int_{\Omega_{min}} W(\omega)|E_c(\omega, \boldsymbol{h})|^2 d\omega \\ \text{subject to} \quad & |E_m(\omega, \boldsymbol{h})| \leq \delta_m(\omega), \quad \omega \in \Omega_B, \\ & |E_\phi(\omega, \boldsymbol{h})| \leq \delta_\phi(\omega), \quad \omega \in \Omega^p \cap \Omega_B, \end{aligned} \tag{2.77}$$

where Ω_{min} contains the frequency bands where the mean squared error is to be minimized, Ω_B contains the frequency bands where constraints are to be imposed, and Ω^p is the passband region. The positive functions $\delta_c(\omega)$, $\delta_m(\omega)$, and $\delta_\phi(\omega)$ defining the constraints are part of the design specifications.

Algorithms for solving problems (2.76) and (2.77) will be presented in Chapter 3.

The solutions obtained by solving problem (2.76) are the same as those obtained from problem formulation (2.56) if $\delta_c(\omega) = \delta/W_C(\omega)$, where δ is part of the solution to problem

(2.56) and $W_C(\omega)$ is the weighting function used in problem (2.56). Hence, all points on the trade-off curve shown in Figure 2.12 can be computed by formulating the design problem as in (2.76). Instead of specifying the trade-off parameter α as in (2.56), the function $\delta_c(\omega)$ constraining the complex error function $E_c(\omega)$ is specified. This problem formulation seems more suitable for practical applications. Similar trade-off curves result from problem formulation (2.77).

The constrained least squares filter design problem has been discussed for the first time in [71]. The problem was formulated for linear phase FIR filters where the frequency domain error function is always defined in terms of real-valued amplitude functions. Hence, after frequency discretization, the design problem can be formulated as a standard convex quadratic programming problem with linear constraints. The motivation of the approach in [71] was the minimization of stopband noise power subject to constraints on the maximum passband deviations. A similar motivation lead to the work by Adams [1] – [6] and Sullivan [128, 129]. The work by Adams discusses the design of FIR filters and windows with linear phase. In [2] he presented an exchange algorithm for solving the design problem in the linear phase case. In [6] Adams and Sullivan summarize several different problem formulations for constrained least squares filter design and also give examples for the complex approximation FIR case, and the magnitude design of analog and IIR digital filters. As for the complex approximation FIR case, they refer to Sullivan's work [128, 129] where the magnitude and group delay errors have been constrained. It has been reported in [129] that this algorithm has convergence problems in certain cases. This is also suggested by the fact that the specifications of all complex approximation FIR filter designs presented in [128, 129, 6] are very close to the exactly linear phase solutions. Hence, a very good starting guess is easily available and convergence problems are very unlikely to occur. The algorithm for constrained least squares linear phase FIR filter design presented by Adams in [2] has been generalized to the complex approximation case in [55]. However, this algorithm does not converge for many practical design specifications. This has been verified by using the program provided by the authors of [55] and has also been indicated in [119].

Motivated by the work of Cortelazzo [27] and Weisburn et al. [141], Selesnick et al. presented an interesting approach to the constrained least squares design of linear phase FIR filters in [119]. The authors suggest that the use of specified transition bands is often inappropriate because it assumes that desired and undesired signals are separated by a "guard band". Hence, the method proposed in [119] does not use passband and stopband edges, but defines one cut-off frequency for each transition from passbands to stopbands. In order to overcome Gibbs' phenomenon, peak error constraints are introduced. The transition widths of the filters are a result of the design process. This has the additional advantage that arbitrarily tight error constraints may be formulated and the existence of a solution is always guaranteed. The more restrictive the constraints, the wider become the transition widths. Although the problem formulation suggested in [119] seems interesting for several applications – such as beamforming using antenna arrays – we will not adopt this view in this thesis because we believe that fixed band edges up to which certain error constraints are to be satisfied constitute an important part of sophisticated design specifications. Moreover, band edges for error constraints and band edges for the minimization of some error measure such as the least squared error, need not necessarily be identical in the problem formulations presented in this thesis. If desired, an ideal lowpass function without transition band can be approximated in the least squares sense, while satisfying error constraints in specified frequency bands.

2.3.3 Constrained Chebyshev Approximation

Constrained Chebyshev problems are problems where a maximum error is minimized subject to constraints on possibly different error functions. As in the constrained least squares case we only use $E_c(\omega, \boldsymbol{h})$ for defining the objective function. Again we formulate two different problems. One with constraints on $|E_c(\omega, \boldsymbol{h})|$, and the other with independent constraints on $|E_m(\omega)|$ and $|E_\phi(\omega)|$. These problems read

$$\begin{array}{ll} \underset{\boldsymbol{h}}{\text{minimize}} & \max\limits_{\omega\in\Omega_{min}} W(\omega)|E_c(\omega,\boldsymbol{h})| \\ \text{subject to} & |E_c(\omega,\boldsymbol{h})| \leq \delta_c(\omega), \quad \omega\in\Omega_B, \end{array} \tag{2.78}$$

and

$$\begin{array}{lrcl} \underset{\boldsymbol{h}}{\text{minimize}} & \max\limits_{\omega\in\Omega_{min}} W(\omega)|E_c(\omega,\boldsymbol{h})| && \\ \text{subject to} & |E_m(\omega,\boldsymbol{h})| & \leq & \delta_m(\omega), \quad \omega\in\Omega_B, \\ & |E_\phi(\omega,\boldsymbol{h})| & \leq & \delta_\phi(\omega), \quad \omega\in\Omega^p\cap\Omega_B, \end{array} \tag{2.79}$$

where Ω_{min} are those frequency bands where a minimization of the maximum error is desired, and Ω_B are those bands where constraints are imposed. As opposed to the constrained least squares problem formulation, the bands Ω_{min} and Ω_B should be disjoint to obtain useful problem formulations. In this case the maximum error is fixed for $\omega \in \Omega_B$ while it is minimized for $\omega \in \Omega_{min}$. The functions $\delta_c(\omega)$, $\delta_m(\omega)$, and $\delta_\phi(\omega)$ are arbitrary positive functions specified by the filter designer. Problem (2.78) is a convex optimization problem due to the convexity of $|E_c(\omega, \boldsymbol{h})|$. Problem (2.79) is non-convex due to the non-convex feasible region implied by the lower bounds on the magnitude error $E_m(\omega, \boldsymbol{h})$. The solutions obtained by solving problems (2.78) and (2.79) can also be obtained by unconstrained complex Chebyshev approximation methods discussed in Section 2.2 if appropriate weighting functions are chosen. Hence, the constrained Chebyshev problem is not as interesting as the constrained least squares problem which provides completely new solutions. However, it seems to be better suited to the demands of filter designers because it provides the possibility to explicitly prescribe a tolerance scheme. Since algorithms for complex constrained Chebyshev approximation are not more complicated than good algorithms for the unconstrained complex Chebyshev problem, we suggest to use constrained Chebyshev approximation if a Chebyshev behavior of the solution is desired.

As suggested by Section 2.2 there exists a lot of work on digital FIR filter design using Chebyshev approximation. The constrained Chebyshev problem has been discussed in [125, 80, 93] for the linear phase case. An efficient exchange algorithm extending the early work by Herrmann and Schüßler [39, 40, 41] and by Hofstetter et al. [43] has been presented in [120]. The complex constrained Chebyshev problem has been treated in [90, 91, 92] by semi-infinite programming methods. In [90, 91] the authors use constraints on the complex error function whereas in [92] they constrain magnitude and phase errors using a convex approximation to the original optimization problem. In Section 3.3 we will present an IRLS algorithm that solves problem (2.78) as well as the constrained least squares problem (2.76) efficiently and with very low memory requirements. This is an important improvement over previously published algorithms.

2.4 Summary

In this chapter, we have provided an overview of useful optimality criteria for the design of filters with arbitrary complex desired frequency responses. The concept of complex constrained approximation has been introduced, and we have motivated the use of design algorithms that can incorporate constraints.

In Section 2.1 we have discussed the continuous and discrete complex least squares criteria. Only linear least squares problems have been considered. The problems are linear if the complex error function is used for defining error energy. If we considered magnitude or phase errors, the resulting least squares problems would become nonlinear and would therefore be much harder to solve. However, we have proposed a method for independently weighting magnitude and phase errors in Section 2.1.3. This method retains the linearity of the design problem, and allows trading magnitude versus phase errors. Several design examples have been given to illustrate the methods under consideration. Matlab programs implementing the algorithms discussed in this section have been provided.

Section 2.2 considers the complex Chebyshev criterion. We have defined three classes of algorithms – generalized Remez algorithms, linear programming and semi-infinite optimization algorithms, and iterative reweighted least squares algorithms –, and have assigned the algorithms published so far to one of these classes. We have compared the different classes of algorithms in Section 2.2.3. For this comparison, we have chosen Karam's and McClellan's generalized Remez algorithm [49], a linear programming approach as used by Chen and Parks [18], the multiple exchange (semi-infinite programming) algorithm by Burnside and Parks [14], and Lawson's iterative reweighted least squares algorithm [67]. It turns out that Karam's and McClellan's generalized Remez algorithm does not converge to the optimum solution for most design specifications, unless a very expensive post-optimization process is invoked. The multiple exchange approach by Burnside and Parks works fine, although its computational effort and its memory requirements are relatively high. However, the computational effort is usually still lower than the one needed by Karam's and McClellan's post-optimization procedure. Lawson's algorithm computes practical solutions to design problems for arbitrary filter orders much faster than the other algorithms. Moreover, it needs only very little memory if the Toeplitz structure of the normal equations associated with the least squares subproblems is exploited. However, final convergence of Lawson's algorithm is often very slow, so that it may be difficult to compute solutions with very high accuracy. Nevertheless, a reasonably good accuracy is often achieved after only a few iteration steps. Moreover, there are modifications to Lawson's algorithm that increase its convergence speed [72]. We conclude that Lawson's method or some related IRLS method is usually most suitable for computing complex Chebyshev approximations, unless the desired accuracy is extremely high. In the latter case, we have to resort to less efficient semi-infinite programming methods and exchange algorithms, respectively, such as proposed in [14, 91].

Section 2.2.4 provides Matlab programs implementing the linear programming approach to complex Chebyshev approximation, Burnside's and Parks' multiple exchange method, and Lawson's iterative reweighted least squares algorithm.

In Section 2.3, we have motivated that unconstrained minimization of some error measure is often not the most appropriate mathematical formulation of a filter design problem. It is often desired to impose bounds on certain error functions, or to mix different error criteria. A standard design specification is a tolerance scheme for some frequency domain function. A tolerance scheme is the actual motivation for all algorithms based on Chebyshev approximation. However, unconstrained Chebyshev approximation does not solve this problem, because the maximum approximation error is a result of the design process and cannot be prescribed. An explicit prescription of a tolerance scheme in the mathematical formulation of the problem is only possible if some type of constrained optimization is used. Constraints are not only useful for specifying tolerance schemes, but they also allow the mix of the Chebyshev error criterion with other criteria. We have shown how a continuum of filters trading maximum error versus error energy can be generated. It has turned out that a filter mixing both criteria in an appropriate way can have more desirable properties than filters that are optimum with respect to the least squares or Chebyshev criterion.

In the complex approximation case, there are several error functions that can be considered when formulating the design problem. Furthermore, one has to decide which error functions should be constrained and which should be used for defining an error measure that is minimized. In Section 2.3.1 we have discussed magnitude, phase, group delay and complex error functions. We have motivated that magnitude and phase errors are relevant error functions by using the fact that the amplitudes of echos introduced by linear distortions can be bounded by bounding magnitude and phase errors. The group delay error is not an important quantity and can be ignored if we consider the phase error instead.

In the formulation of the complex constrained least squares problem, we have chosen the complex error function for defining error energy. We have shown that this choice results in a convex quadratic objective function. We have also used the complex error function in the objective function of the complex constrained Chebyshev design problem. We have introduced two different formulations of design constraints. The first constrains the magnitude of the complex error function. This is useful because it also constrains magnitude and phase errors. Moreover, we have shown that constraints on the magnitude of the complex error function correspond to a convex feasible region in the associated optimization problem. This makes the computation of the optimum solution easier. The second formulation of constraints directly imposes bounds on magnitude and phase errors. This formulation is more flexible than the one constraining the complex error function because both error functions can be constrained independently of each other. Moreover, given specified maximum magnitude and phase errors, the corresponding constraints on the complex error function are more restrictive than independent magnitude and phase constraints.

We have shown that upper bounds on the magnitude of the frequency response result in a convex feasible region. The same is true for phase constraints if the phase constraint function does not exceed $\pi/2$. In this case, the phase constraints can even be formulated as linear constraints. Only the lower bounds on the magnitude of the frequency response make the corresponding feasible region non-convex.

We have formulated the constrained least squares and constrained Chebyshev problems with constraints on the complex error function and on magnitude and phase errors, respectively, in Sections 2.3.2 and 2.3.3. Algorithms solving these problems will be considered in Chapter 3.

Chapter 3

Constrained Design of FIR Filters

This chapter presents several new algorithmic approaches to the constrained design of FIR filters with arbitrary magnitude and phase responses. The design problems under consideration are the complex constrained least squares and the complex constrained Chebyshev problems as formulated in Section 2.3. However, we put the emphasis on the constrained least squares problem because it leads to new interesting solutions, whereas the constrained Chebyshev problem formulation merely represents a more practical formulation of the design problem than the standard Chebyshev problem formulation.

Three different groups of methods are considered. The first approach reformulates the filter design problems as standard linear and quadratic programming problems. There are many good algorithms and a lot of professional software exists to compute the solutions to these problems. However, in order to arrive at these problem formulations some approximations have to be made, and hence systematic errors are introduced. It will be shown that for practical purposes these errors are often negligible. The disadvantage of this approach is the large size of the resulting optimization problems if filters of higher order (say $N > 100$) are to be designed. This problem is overcome by the next group of methods called exchange algorithms. They solve a sequence of small subproblems. Their basic idea is similar to the one of the Remez algorithm used in the McClellan-Parks program for Chebyshev design of linear phase FIR filters [83]. An alternative approach presented in this chapter is to generalize Lawson's algorithm described in Section 2.2 making it applicable to constrained design problems. This algorithm belongs to the group of iterative reweighted least squares (IRLS) methods. Finally, the properties of all approaches are summarized and compared.

3.1 Linear and Quadratic Programming

In this section we consider the constrained least squares problems (2.76) and (2.77), and the constrained Chebyshev problems (2.78) and (2.79) with constraints on the complex error and on the magnitude and phase errors, respectively. We showed in Section 2.3.1 that those problems with constraints on the complex error function are convex problems, whereas those with constraints on magnitude and phase errors are non-convex problems due to the non-convex lower magnitude error bounds. In order to formulate the design problems as (finite) linear or quadratic programming problems, two approximations have to be made. First, the constraints formulated on continuous frequency bands are replaced by a finite number of constraints imposed on a discrete set of frequency points. The second approximation is the linearization of all constraints that are nonlinear functions of the filter coefficients $\boldsymbol{h}$. For convex problems such as (2.76) and (2.78), the errors introduced by linearizing the constraints can in principle be made arbitrarily small by increasing the number of linear constraints. This is not the case for the non-convex problems (2.77, 2.79).

3.1.1 Constraints on the Complex Error Function

Consider the feasible region K defined by

$$K = \{\boldsymbol{h} \in \mathbb{R}^N \big| \ |E_c(\omega, \boldsymbol{h})| \leq \delta_c(\omega),\ \omega \in \Omega_B\},$$

with Ω_B being the union of all frequency intervals where constraints on the magnitude of the complex error are specified. We assume the frequency intervals in Ω_B to be represented by a finite number of frequency points. The number of points in the frequency grid should be chosen in proportion to the desired filter length. Since Ω_B consists of a finite number of points, the feasible set K is described by a finite number of nonlinear constraints. Because the set K is convex, it can be represented by an infinite number of linear constraints:

$$K = \{\boldsymbol{h} \in \mathbb{R}^N \big| \ \mathrm{Re}\left[E_c(\omega, \boldsymbol{h})e^{j\alpha}\right] \leq \delta_c(\omega),\ \omega \in \Omega_B,\ \alpha \in [0, 2\pi)\}. \tag{3.1}$$

Replacing the feasible set K by a finite number of linear inequality constraints corresponds to replacing the continuous angle variable α in (3.1) by a finite number of angles as has been done in order to approximately solve the complex Chebyshev approximation problem by linear programming (see Section 2.2). We choose the discrete angles as in (2.43):

$$\alpha_k = \frac{k\pi}{p}, \quad k = 0, 1, \ldots, 2p-1, \quad p \geq 2. \tag{3.2}$$

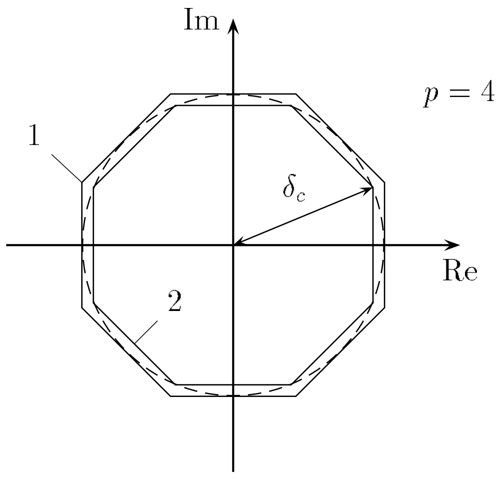


Figure 3.1: Constraints on the complex error function: replacing the circular error region in the complex plane by a regular polygon with $2p$ vertices. 1: new feasible region is larger than the original feasible region. 2: new feasible region is smaller than the original feasible region.

This replaces the circular error region in the complex plane by a regular polygon with $2p$ vertices. As opposed to complex Chebyshev approximation, we are free to choose the size of this polygon since its size is fixed instead of being minimized. Two reasonable choices for the size of the polygon are shown in Figure 3.1. The polygon denoted by '1' in Figure 3.1 corresponds to the feasible region

$$K_1 = \{\boldsymbol{h} \in \mathbb{R}^N | \ \text{Re}\left[E_c(\omega, \boldsymbol{h})e^{j\alpha_k}\right] \leq \delta_c(\omega), \ \omega \in \Omega_B, \ k = 0, 1, \ldots, 2p-1\},$$

and the smaller polygon denoted by '2' in Figure 3.1 is associated with the set

$$K_2 = \{\boldsymbol{h} \in \mathbb{R}^N | \ \text{Re}\left[E_c(\omega, \boldsymbol{h})e^{j\alpha_k}\right] \leq \delta_c(\omega) \cos\left(\frac{\pi}{2p}\right), \ \omega \in \Omega_B, \ k = 0, 1, \ldots, 2p-1\},$$

with angles α_k given by (3.2). Obviously, the following relationship holds:

$$K_2 \subseteq K \subseteq K_1. \tag{3.3}$$

Denote the optimum solutions corresponding to the feasible sets K, K_1, and K_2 by $\boldsymbol{h}_o$, $\boldsymbol{h}_{o1}$, and $\boldsymbol{h}_{o2}$. If $f(\boldsymbol{h})$ denotes the objective function to be minimized, then due to (3.3),

$$f(\boldsymbol{h}_{o1}) \leq f(\boldsymbol{h}_o) \leq f(\boldsymbol{h}_{o2})$$

must hold. Note that $\boldsymbol{h}_{o2}$ is guaranteed to be feasible with respect to the original feasible region K, but the corresponding value of the objective function may be larger than $f(\boldsymbol{h}_o)$.

Choosing K_1 may result in a smaller objective function, but $\boldsymbol{h}_{o1}$ may be infeasible with respect to K. The maximum violation of the original constraints by solution $\boldsymbol{h}_{o1}$ is

$$\Delta(\omega,p)=\delta_c(\omega)\left[\frac{1}{\cos(\pi/2p)}-1\right]=\delta_c(\omega)\left[\frac{1}{2}\left(\frac{\pi}{2p}\right)^2+\frac{5}{24}\left(\frac{\pi}{2p}\right)^4+\dots\right], \tag{3.4}$$

which is easily verified from Figure 3.1. The maximum violation decreases quadratically with increasing p. Note that the parameter p – i. e. the number of vertices of the polygon – may be chosen as a function of frequency.

Having defined feasible sets K_1 and K_2 described by finite numbers of linear inequality constraints that approximate the original feasible set K, we can now formulate the linear and quadratic programming problems that approximate the constrained Chebyshev and least squares problems (2.78) and (2.76) with constraints on the complex error function. The linear programming formulation of the constrained Chebyshev problem (2.78) incorporates additional linear constraints replacing the nonlinear objective function as explained in Section 2.2 (cf. problem formulation (2.42) and the following discussion):

$$\begin{array}{lll} \underset{\boldsymbol{h},\,\delta}{\text{minimize}} & \delta & \\ \text{subject to} & \operatorname{Re}\left[E_c(\omega,\boldsymbol{h})e^{j\alpha_k}\right] \le \delta/W(\omega), & \omega\in\Omega_{min}, \\ & \operatorname{Re}\left[E_c(\omega,\boldsymbol{h})e^{j\alpha_k}\right] \le U(\omega), & \omega\in\Omega_B, \\ & k=0,1,\dots,2p-1,\quad p\ge 2, & \end{array} \tag{3.5}$$

with angles α_k given by (3.2). The positive constraint function $U(\omega)$ in (3.5) is chosen as $U(\omega)=\delta_c(\omega)$ if polygon '1' in Figure 3.1 is to be implemented, or as $U(\omega)=\delta_c(\omega)\cos(\pi/2p)$ if polygon '2' is chosen. The function $W(\omega)$ is a positive weighting function defined on Ω_{min}. Note that the sets Ω_{min} and Ω_B in (3.5) should be disjoint. The linearization parameter p in (3.5) is assumed to be the same for all frequencies. However, the generalization to a frequency dependent $p(\omega)$ is straightforward.

Problem (3.5) is a standard linear programming problem of the form (A.28). If N denotes the filter length, the number of variables in (3.5) is $N+1$. Let L denote the total number of frequency points in Ω_{min} and Ω_B. Assuming a fixed value p for all frequency points, the number of constraints in (3.5) is $2Lp$. The number of frequency points should be chosen according to $L\ge 5N$. Hence, the number of constraints is much larger than the number of variables in (3.5). According to the discussion in Section A.5, an active set method is well suited for solving problem (3.5). When using the simplex algorithm, it is advantageous to solve the dual of (3.5) as given by (A.29).

The formulation of the constrained least squares problem (2.76) as a quadratic programming problem reads

$$\begin{array}{ll} \underset{\boldsymbol{h}}{\text{minimize}} & \displaystyle\int_{\Omega_{min}} W(\omega)|E_c(\omega,\boldsymbol{h})|^2 d\omega \\ \text{subject to} & \text{Re}\left[E_c(\omega,\boldsymbol{h})e^{j\alpha_k}\right] \leq U(\omega), \quad \omega \in \Omega_B, \\ & k = 0,1,\ldots,2p-1, \quad p \geq 2, \end{array} \tag{3.6}$$

with $U(\omega)$ chosen as discussed in connection with the constrained Chebyshev problem (3.5). Note that unlike problem (3.5), problem (3.6) does not require the sets Ω_{min} and Ω_B to be disjoint. Problem (3.6) is a standard quadratic programming problem as formulated by (A.30). The corresponding dual problem (A.32) has no constraints except for non-negativity constraints on its variables. However, the number of variables of the dual is $2mp$ which is usually too large for the dual to become easier to solve than the primal problem, even for moderate filter lengths. Consequently, solving problem (3.6) directly is more advantageous than solving its dual. Note that in problem (3.6) only the set Ω_B is assumed to be a set of discrete frequency points, whereas the set Ω_{min} is the union of continuous frequency intervals. Alternatively, the objective function in (3.6) can be formulated as a weighted sum of squared errors evaluated on a frequency grid.

3.1.2 Constraints on Magnitude and Phase Errors

As shown in Section (2.3.1), constraints on the phase error are exactly represented by linear inequality constraints if the function $\delta_\phi(\omega)$ constraining the phase response satisfies $\delta_\phi(\omega) \leq \pi/2$, $\forall\omega \in \Omega^p$. Hence, only the nonlinear magnitude constraints must be replaced by a finite set of linear constraints. In the stopbands, the error region is a circle around the origin of the complex plane. Consequently, all results from the previous section still apply to the stopbands. The error region in the passbands is shown by Figure 2.15 (b). The modifications of the passband error region due to magnitude constraint linearizations are shown in Figure 3.2. The original magnitude constraints are shown by the dashed arcs. The solid vertical lines correspond to linear constraints replacing the original magnitude constraints. The constraints denoted by '1' in Figure 3.2 correspond to a feasible region which is the smallest set described by two linear inequality constraints per frequency point containing the original feasible set. The constraints denoted by '2' correspond to a feasible region being the largest set described by two linear inequality constraints per frequency point that is contained in the original set. A better approximation to the upper magnitude constraint could be achieved by using more than one linear upper magnitude constraint per frequency point. However, this increases the size of the resulting optimization problem. Note that improving the approximation by using more than one linear constraint per frequency point is not possible for the non-convex lower magnitude

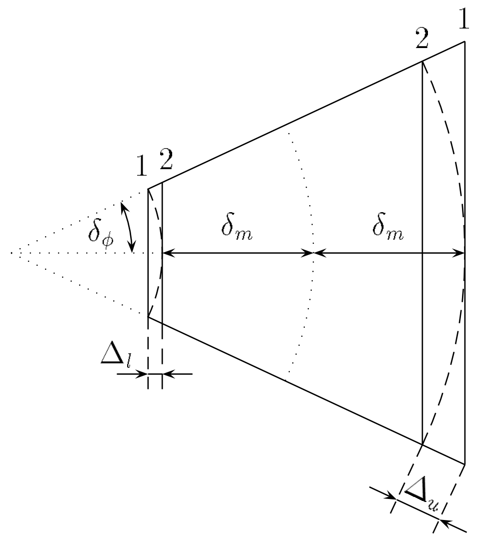


Figure 3.2: Replacing nonlinear magnitude constraints (dashed arcs) by linear constraints (solid vertical lines). 1: new feasible region is larger than the original feasible region. 2: new feasible region is smaller than the original feasible region.

constraint. Choosing the linear constraints denoted by '1' in Figure 3.2 results in the following maximum violations of the original passband magnitude constraints if $\delta_\phi(\omega) < \pi/2$ holds:

$$\begin{aligned}
\Delta_u(\omega) &= \left[\left|D\left(e^{j\omega}\right)\right| + \delta_m(\omega)\right]\left[\frac{1}{\cos\delta_\phi(\omega)} - 1\right] \\
&= \left[\left|D\left(e^{j\omega}\right)\right| + \delta_m(\omega)\right]\left[\tfrac{1}{2}\delta_\phi^2(\omega) + \tfrac{5}{24}\delta_\phi^4(\omega) + \ldots\right], \quad \omega \in \Omega^p, \\
\Delta_l(\omega) &= \left[\left|D\left(e^{j\omega}\right)\right| - \delta_m(\omega)\right]\left[1 - \cos\delta_\phi(\omega)\right] \\
&= \left[\left|D\left(e^{j\omega}\right)\right| - \delta_m(\omega)\right]\left[\tfrac{1}{2}\delta_\phi^2(\omega) - \tfrac{1}{24}\delta_\phi^4(\omega) + \ldots\right], \quad \omega \in \Omega^p,
\end{aligned} \tag{3.7}$$

where $\Delta_u(\omega)$ and $\Delta_l(\omega)$ are the maximum violations of the upper and lower magnitude constraints, respectively. It is straightforward to show that $\Delta_l(\omega) \leq \Delta_u(\omega)$, $\forall\omega \in \Omega^p$, if $0 \leq \delta_\phi(\omega) < \pi/2$, $\forall\omega \in \Omega^p$, holds. From (3.7) it is clear that the violations $\Delta_u(\omega)$ and $\Delta_l(\omega)$ consist only of mixed and higher order terms of the specified functions $\delta_m(\omega)$ and $\delta_\phi(\omega)$. Hence, for small $\delta_m(\omega)$ and $\delta_\phi(\omega)$, the errors introduced by using linearized magnitude constraints are small as well. If the linear constraints denoted by '2' in Figure 3.2 are used, no violations with respect to the original feasible region occur.

Using the linearized passband magnitude constraints shown in Figure 3.2, the linear phase constraints derived in Section 2.3.1 (cf. page 54), and the stopband constraints from the previous section, we can formulate linear and quadratic programming problems approximating the constrained Chebyshev and least squares problems (2.79) and (2.77). If the new feasible region is to completely contain the original feasible region (constraints denoted by '1' in Figures 3.1 and 3.2), define positive functions $U(\omega)$ and $L(\omega)$ according to

$$\begin{aligned} U(\omega) &= \begin{cases} |D(e^{j\omega})| + \delta_m(\omega), & \omega \in \Omega^p \cap \Omega_B, \\ \delta_m(\omega), & \omega \in \Omega^s \cap \Omega_B, \end{cases} \\ L(\omega) &= [|D(e^{j\omega})| - \delta_m(\omega)] \cos\delta_\phi(\omega), \quad \omega \in \Omega^p \cap \Omega_B. \end{aligned} \tag{3.8}$$

If no constraint violations with respect to the original feasible region are tolerated (constraints denoted by '2' in Figures 3.1 and 3.2), then $U(\omega)$ and $L(\omega)$ must be chosen according to

$$\begin{aligned} U(\omega) &= \begin{cases} [|D(e^{j\omega})| + \delta_m(\omega)] \cos\delta_\phi(\omega), & \omega \in \Omega^p \cap \Omega_B, \\ \delta_m(\omega)\cos(\pi/2p), & \omega \in \Omega^s \cap \Omega_B, \end{cases} \\ L(\omega) &= |D(e^{j\omega})| - \delta_m(\omega), \quad \omega \in \Omega^p \cap \Omega_B. \end{aligned} \tag{3.9}$$

The linear programming problem approximating the constrained Chebyshev problem (2.79) reads

$$\begin{aligned} &\underset{\boldsymbol{h}}{\text{minimize}}\ \delta, \quad \text{subject to} \\ &\operatorname{Re}[E_c(\omega,\boldsymbol{h})e^{j\alpha_k}] \le \delta/W(\omega), && \omega \in \Omega_{min}, \\ &\operatorname{Re}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right] \le U(\omega), && \omega \in \Omega^p \cap \Omega_B, \\ &\operatorname{Re}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right] \ge L(\omega), && \omega \in \Omega^p \cap \Omega_B, \\ &\operatorname{Im}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right] \le \tan\delta_\phi(\omega)\operatorname{Re}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right], && \omega \in \Omega^p \cap \Omega_B, \\ &\operatorname{Im}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right] \ge -\tan\delta_\phi(\omega)\operatorname{Re}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right], && \omega \in \Omega^p \cap \Omega_B, \\ &\operatorname{Re}[H(e^{j\omega},\boldsymbol{h})\,e^{j\alpha_k}] \le U(\omega), && \omega \in \Omega^s \cap \Omega_B, \\ &k = 0,1,\ldots,2p-1, \quad p \ge 2. \end{aligned} \tag{3.10}$$

The quadratic programming problem approximating the constrained least squares problem (2.77) reads

$$\begin{aligned} &\underset{\boldsymbol{h}}{\text{minimize}} \quad \int_{\Omega_{min}} W(\omega)|E_c(\omega,\boldsymbol{h})|^2 d\omega, \quad \text{subject to} \\ &\operatorname{Re}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right] \le U(\omega), && \omega \in \Omega^p \cap \Omega_B, \\ &\operatorname{Re}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right] \ge L(\omega), && \omega \in \Omega^p \cap \Omega_B, \\ &\operatorname{Im}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right] \le \tan\delta_\phi(\omega)\operatorname{Re}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right], && \omega \in \Omega^p \cap \Omega_B, \\ &\operatorname{Im}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right] \ge -\tan\delta_\phi(\omega)\operatorname{Re}\left[H(e^{j\omega},\boldsymbol{h})\,e^{-j\phi_d(\omega)}\right], && \omega \in \Omega^p \cap \Omega_B, \\ &\operatorname{Re}[H(e^{j\omega},\boldsymbol{h})\,e^{j\alpha_k}] \le U(\omega), && \omega \in \Omega^s \cap \Omega_B, \\ &k = 0,1,\ldots,2p-1, \quad p \ge 2, \end{aligned} \tag{3.11}$$

where Ω^p and Ω^s are passbands and stopbands, respectively, Ω_B is the union of all bands where constraints are to be imposed, and Ω_{min} is the union of all bands where the objective function (maximum error or error energy) is to be minimized. Note that Ω_B and in (3.10) also Ω_{min} are assumed to be represented by a discrete set of grid points. The first constraint in (3.10) represents the original Chebyshev objective function. The remaining constraints in (3.10) and the constraints in (3.11) are arranged as follows: upper and lower passband magnitude constraints, upper and lower phase constraints, and stopband constraints.

Note that the phase constraints in (3.10) and (3.11) are only valid if $\delta_\phi(\omega) < \pi/2$ holds. The results obtained by using problem formulations (3.10) and (3.11) will only be satisfactory if the phase constraint function $\delta_\phi(\omega)$ (in radians) is not considerably larger than the magnitude constraint function $\delta_m(\omega)$. Otherwise the deviations of the linearized magnitude constraints from the original magnitude constraints will be large.

3.1.3 Design Example

We design bandpass filters with reduced delays according to the specification given in Section 2.2.3. We use the design methods presented in Sections 3.1.1 and 3.1.2. These methods are implemented by the Matlab programs `lqp_ce` and `lqp_mp` given in the next section. The desired response $D\left(e^{j\omega}\right)$ is defined by (2.51). We choose the constrained least squares criterion with a weighting function $W(\omega)$ that is 1 in the passband and 1000 in the stopbands. The linearization parameter p is 6 in all bands, i. e. the circle in the complex plane is approximated by a regular polygon with $2p = 12$ vertices. According to Table 2.1, the maximum error of the optimum Chebyshev solution is $7.52 \cdot 10^{-2}$. Due to the stopband weighting of 10 used in Section 2.2.3, the stopband attenuation of the optimum Chebyshev filter is 42.5 dB. For designing a filter by the method introduced in Section 3.1.1, we choose a constraint function $\delta_c(\omega)$ that constrains the magnitude $|E_c(\omega)|$ of the complex error function to be smaller than $7.2 \cdot 10^{-2}$ in the passband. In the stopbands we require a minimum attenuation of 40 dB. Hence, compared to the optimum Chebyshev solution, we slightly decrease the passband error, but we relax the stopband specification by 2.5 dB. This results in additional degrees of freedom that can be used to minimize the error energy. These specifications are passed to Matlab by the following commands:

```
om=pi*[linspace(0,.2,70),linspace(.3,.56,90),linspace(.66,1,120)];
D=[zeros(1,70),exp(-j*12*om(71:160)),zeros(1,120)];
W=[1000*ones(1,70),ones(1,90),1000*ones(1,120)];
dc=[1e-2*ones(1,70),.072*ones(1,90),1e-2*ones(1,120)];
```

We design two filters using the program `lqp_ce` which implements the method of Section 3.1.1. Section 3.1.4 gives a listing of this program and explains its input parameters. For the first design we choose the polygon denoted by '1' in Figure 3.1. The corresponding filter may violate the specified constraints. The maximum violation determined by the parameter p is given by (3.4). For the second design, we choose the polygon denoted by '2' in Figure 3.1. This filter is guaranteed to satisfy all constraints. These two filters are designed by the commands

```
hce1 = lqp_ce(31,om,D,W,dc,6,1,'ls');
hce2 = lqp_ce(31,om,D,W,dc,6,2,'ls');
```

A third filter is designed by imposing constraints on magnitude and phase errors using the method presented in 3.1.2. We choose the magnitude and phase constraint functions equal to the constraint function $\delta_c(\omega)$ used in the previous designs. We choose polygon '2' in Figure 3.1 for the stopbands, and passband magnitude constraints denoted by '2' in Figure 3.2. Hence, no violations of the original constraints will occur. This filter is designed by the following commands:

```
dm=dc; dp=dc;
hmp = lqp_mp(31,om,D,W,dm,dp,6,2,'ls');
```

A listing of the program `lqp_mp` and the definition of its input parameters can be found in Section 3.1.4. Figure 3.3 shows the results of the three filter designs. The dashed straight lines indicate the constraints specified by the constraint functions $\delta_c(\omega)$, $\delta_m(\omega)$, and $\delta_\phi(\omega)$. The solid curves are the magnitude, phase, and group delay responses of the filter designed with constraints on its magnitude and phase responses. The dashed and dotted curves show the magnitude responses, phase errors, and group delay responses of the filters designed with constraints on the complex error function. The dashed curves correspond to the filter with constraints according to polygon '1' in Figure 3.1. The dotted curves correspond to the filter with constraints according to polygon '2' in Figure 3.1.

In the programs `lqp_ce` and `lqp_mp` the weighted mean square error is computed on a frequency grid:

$$\epsilon(\boldsymbol{h}) = \frac{1}{L}\sum_{i=0}^{L-1} W(\omega_i)\,|E_c(\omega_i,\boldsymbol{h})|^2\,, \tag{3.12}$$

where L denotes the number of frequency points in the grid. The bounds imposed on magnitude and phase errors by bounds on the complex error function are given by (2.59) and (2.60). Since we chose $\delta_c(\omega) = \delta_m(\omega) = \delta_\phi(\omega)$ and since $\arcsin(0.072) \doteq 0.072$, the bounds on magnitude and phase errors are virtually the same for all three designs. However, constraints on the complex error are more restrictive than equivalent but independent constraints on magnitude and phase errors. Hence, the design with constraints on magnitude and phase errors (solid curves in Figure 3.3) achieves the lowest weighted mean square approximation error. Its value as defined by (3.12) is 2.35 which compares

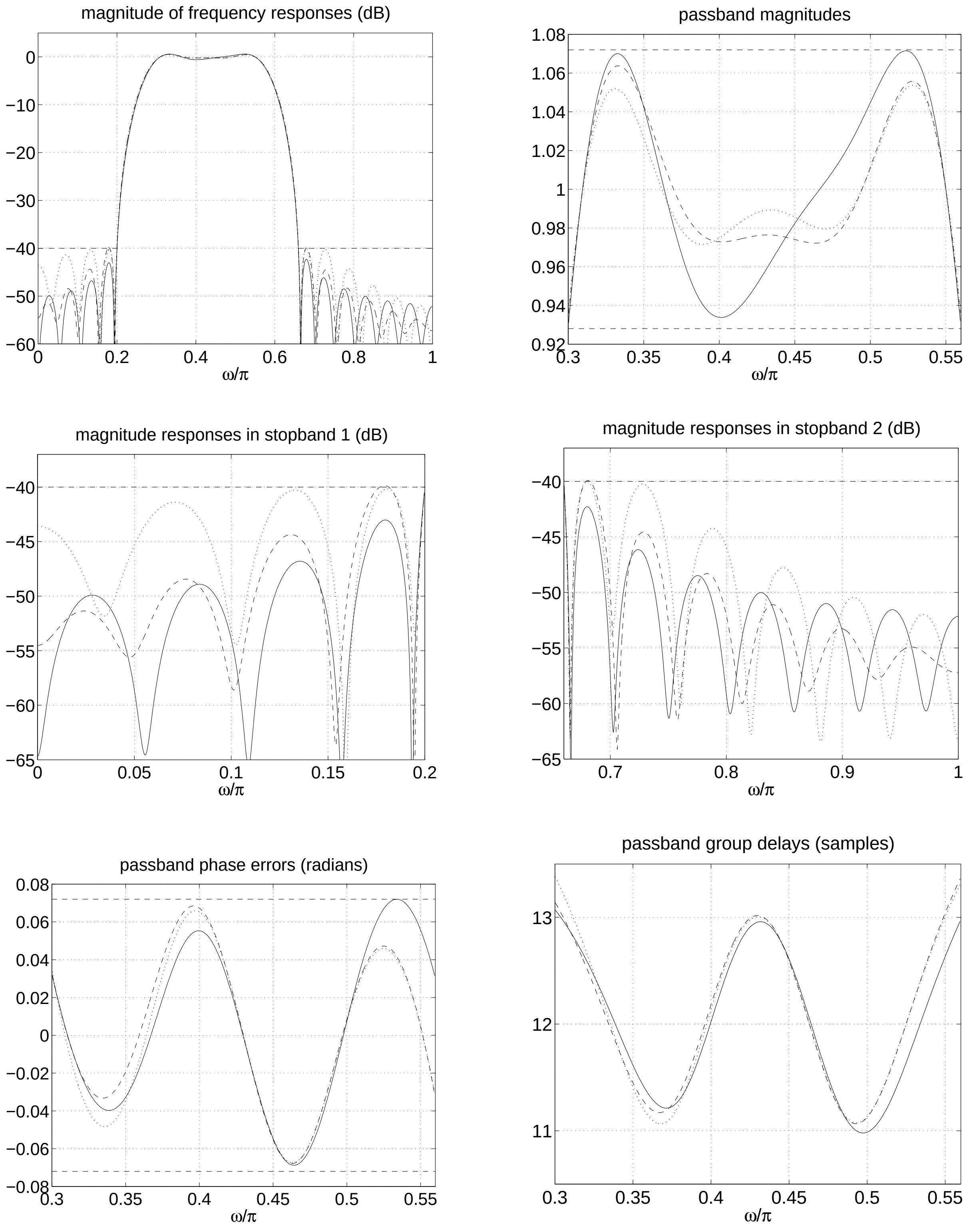


Figure 3.3: Constrained least squares designs of reduced delay bandpass filter (length $N = 31$). Solid curves: design with constraints on magnitude and phase errors. Dashed curves: design with constraints on the complex error (polygon '1' in Fig. 3.1). Dotted curves: design with constraints on the complex error (polygon '2' in Fig. 3.1). Dashed lines: constraints.

to 3.19 and 5.65 for the designs with constraints on the complex error approximated by polygons '1' and '2' in Figure 3.1, respectively. Note that the relatively large values for the weighted mean square approximation errors occur due to the large stopband weighting ($W(\omega) = 1000$ in the stopbands). The differences in error energy are most obvious in the stopbands, as can be seen from Figure 3.3.

The filter designed with constraints on the complex error function using polygon '1' violates the original nonlinear constraints. The maximum passband violation is $2.54 \cdot 10^{-3}$ and occurs at one of the passband edges. This value corresponds to the maximum possible violation given by (3.4). The maximum constraint violation in the stopbands is 0.13 dB which is smaller than the maximum possible violation of 0.3 dB. Violations of the stopband constraints occur at the stopband edges and at the two local error maxima closest to the stopband edges. The constraint violations could be decreased by increasing the value of the parameter p. This would also decrease the difference between the two solutions to the problems with constraints on the complex error (dashed and dotted curves in Figure 3.3).

3.1.4 Matlab Programs

The Matlab program `lqp_ce` solves the Chebyshev and least squares problems with constraints on the complex error function as formulated by (3.5) and (3.6), respectively. The desired complex frequency response $D\left(e^{j\omega}\right)$, the weighting function $W(\omega)$, and the constraint function $\delta_c(\omega)$ must be specified on a frequency grid `om` $\in [0, \pi]$, and are passed to the program as vectors `D`, `W`, and `dc`. The input parameter `p` corresponds to p in (3.2) and determines the number of vertices of the polygon approximating the circle in the complex plane. The value of `rsize` (1 or 2) corresponds to the choices of linear constraints denoted by '1' and '2' in Figure 3.1. The optimization criterion (Chebyshev or least squares) is chosen by the string variable `crit` which must be either `'cheb'` or `'ls'`. The least squares objective function is the weighted sum of squared errors evaluated on the grid `om` as defined by (3.12).

```
function h = lqp_ce(N,om,D,W,dc,p,rsize,crit)
% LQP_CE: complex Chebyshev or least squares FIR filter design
% with constraints on the magnitude of the complex error function
% by linear/quadratic programming
%
% h = LQP_CE(N,om,D,W,dc,p,rsize,crit)
%
% h      filter impulse response
% N      filter length
% om     frequency grid (0 <= om <= pi)
% D      complex desired frequency response on the grid om
% W      positive weighting function on the grid om
% dc     positive constraint function on the grid om;
%        for intervals without constraints specify negative dc
% p      integer linearization parameter (p >= 2)
%        (a circle in the complex plane is approximated by a
%        regular polygon with 2p vertices)
% rsize  for rsize=1, the polygon approximates the circle from
%        the outside, for rsize=2 it approximates the circle from
%        the inside
% crit   string variable defining the design criterion:
%        'cheb' for constrained Chebyshev design
%        'ls' for constrained least squares design
%
% example: Low-delay lowpass filter with passband constraints
% om=pi*[linspace(0,.2,50),linspace(.3,1,200)];
% D=[exp(-j*10*om(1:50)),zeros(1,200)]; W=[ones(1,50),100*ones(1,200)];
% dc=[.02*ones(1,50),-ones(1,200)];
% h = lqp_ce(31,om,D,W,dc,4,1,'ls')
%
% Author: Mathias C. Lang, Vienna University of Technology, July 98

om=om(:); D=D(:); W=W(:); dc=dc(:); L = length(om);
Dmag = abs(D); Dph = angle(D);
if strcmp(crit,'cheb')==1, cheb=1; else, cheb=0; end
IB0 = find(dc==0); LB0 = length(IB0);
IB1 = find(dc>0); LB1 = length(IB1);
AB0 = []; bB0 = []; AB1 = []; bB1 = [];
p2 = 2*p; alpha = pi*(0:p2-1)/p;

if cheb,    % constrained Chebyshev
   Imin = find(dc<0); Lmin = length(Imin);
   if Lmin,
      Amin = zeros(p2*Lmin,N); bmin = zeros(p2*Lmin,1);
      for i=1:p2,
         I = (i-1)*Lmin+(1:Lmin);
         Amin(I,:) = W(Imin,ones(N,1)).*cos(alpha(i) - om(Imin)*(0:N-1));
         bmin(I) = W(Imin).*Dmag(Imin).*cos(alpha(i) + Dph(Imin));
      end
   else, Amin=[]; bmin=[];
   end
else        % constrained least squares
   t=zeros(N,1); c=t;
   for i = 1:N,
      t(i) = W.'*cos((i-1)*om);
      c(i) = W.'*(real(D).*cos((i-1)*om) - imag(D).*sin((i-1)*om));
   end
   t=t/L; c=c/L;
end

% formulate constraints
```

```
if LB0,     % equality constraints
   AB0 = [cos(om(IB0)*(0:N-1));-sin(om(IB0)*(0:N-1))];
   bB0 = [real(D(IB0));imag(D(IB0))];
end
if LB1,     % inequality constraints
   AB1 = zeros(p2*LB1,N); bB1 = zeros(p2*LB1,1);
   U = dc; if rsize==2, U = U*cos(pi/p2); end
   for i=1:p2,
      I = (i-1)*LB1+(1:LB1);
      AB1(I,:) = cos(alpha(i) - om(IB1)*(0:N-1));
      bB1(I) = U(IB1) + Dmag(IB1).*cos(alpha(i) + Dph(IB1));
   end
end
AB = [AB0;AB1]; bB = [bB0;bB1];

% solve optimization problem
if cheb,
   % linear program
   A = [zeros(2*LB0+p2*LB1,1),AB;-ones(p2*Lmin,1),Amin]; b = [bB;bmin];
   h = lp([1;zeros(N,1)],A,b,[],[],[],2*LB0); h=h(2:N+1);
else
   % quadratic program
   h = qp(toeplitz(t),-c,AB,bB,[],[],[],2*LB0);
end
```

The Chebyshev and least squares problems with independent magnitude and phase constraints as formulated by (3.10) and (3.11) are solved by the program `lqp_mp`. It uses two constraint functions specified by the vectors `dm` and `dp` for magnitude and phase errors, respectively. All other input parameters are the same as the ones used by `lqp_ce`. The input variable `rsize` refers to the constraints in Figures (3.1) and (3.2).

```
function h = lqp_mp(N,om,D,W,dm,dp,p,rsize,crit)
% LQP_MP: complex Chebyshev or least squares FIR filter design
% with magnitude and phase constraints
% by linear/quadratic programming
%
% h = LQP_MP(N,om,D,W,dc,p,rsize,crit)
%
% h       filter impulse response
% N       filter length
% om      frequency grid (0 <= om <= pi)
% D       complex desired frequency response on the grid om
% W       positive weighting function on the grid om
% dm      positive magnitude constraint function on the grid om (linear,
%         not dB); for intervals without magnitude constraints specify
%         negative dm
% dp      positive phase constraint function on the grid om (in radians);
%         stopband values irrelevant; for (passband) intervals without
%         phase constraints specify negative dp
% p      integer linearization parameter (p >= 2)
%        (a circle in the complex plane is approximated by a
%        regular polygon with 2p vertices)
% rsize  parameter specifying size of feasible region; for rsize=1,
%        the feasible region is approximated from the outside; for
%        rsize=2 it is approximated from the inside
% crit   string variable defining the design criterion:
%        'cheb' for constrained Chebyshev design
%        'ls' for constrained least squares design
%
% EXAMPLE:
% Chirp All-Pass (Steiglitz, ASSP-29, pp. 171-176, 1981), length 31,
% magnitude and phase errors must not exceed 0.002 (linear/rad):
% N=31; om=pi*(0:500)/500; D=exp(-j*((N-1)/2*om+4/pi*(om-pi/2).^2));
% W=ones(size(D)); dm=.002*W; dp=.002*W;
% h = lqp_mp(N,om,D,W,dm,dp,4,1,'ls')
%
% Author: Mathias C. Lang, Vienna University of Technology, July 98

om=om(:); D=D(:); W=W(:); dm=dm(:); dp=dp(:); L = length(om);
Dmag = abs(D); Dph = angle(D);
dp(find(dp<0 & dm>=0 & Dmag))=pi/4;
if strcmp(crit,'cheb')==1, cheb=1; else, cheb=0; end
Is0 = find(dm==0 & ~Dmag); Ls0 = length(Is0);
Is1 = find(dm>0 & ~Dmag); Ls1 = length(Is1);
Im = find(dm>=0 & Dmag); Lm = length(Im);
Ip = find(dp>=0 & Dmag); Lp = length(Ip);
p2 = 2*p; alpha = pi*(0:p2-1)/p;

if cheb,    % constrained Chebyshev
   Imin = find(dm<0); Lmin = length(Imin);
   if Lmin,
      Amin = zeros(p2*Lmin,N); bmin = zeros(p2*Lmin,1);
      for i=1:p2,
         I = (i-1)*Lmin+(1:Lmin);
         Amin(I,:) = W(Imin,ones(N,1)).*cos(alpha(i) - om(Imin)*(0:N-1));
         bmin(I) = W(Imin).*Dmag(Imin).*cos(alpha(i) + Dph(Imin));
      end
   else, Amin=[]; bmin=[];
   end
else        % constrained least squares
   t=zeros(N,1); c=t;
   for i = 1:N,
      t(i) = W.'*cos((i-1)*om);
      c(i) = W.'*(real(D).*cos((i-1)*om) - imag(D).*sin((i-1)*om));
   end
   t=t/L; c=c/L;
end

% formulate constraints
```

```
if Lm,      % passband magnitude constraints
   U = Dmag(Im) + dm(Im); if rsize==2, U = U.*cos(dp(Im)); end
   L = Dmag(Im) - dm(Im); if rsize==1, L = L.*cos(dp(Im)); end
   Am = cos(om(Im)*(0:N-1) + Dph(Im,ones(N,1)));
   Am = [Am;-Am]; bm = [U;-L];
else, Am=[]; bm=[];
end

if Lp,      % phase constraints
   Ap1 = -sin(om(Ip)*(0:N-1) + Dph(Ip,ones(N,1)));
   Ap2 = tan(dp(Ip,ones(N,1))).*cos(om(Ip)*(0:N-1) + Dph(Ip,ones(N,1)));
   Ap = [Ap1-Ap2;-Ap1-Ap2]; bp = zeros(2*Lp,1);
else, Ap=[]; bp=[];
end

% stopband constraints
if Ls0,
   As0 = [cos(om(Is0)*(0:N-1));-sin(om(Is0)*(0:N-1))];
   bs0 = [real(D(Is0));imag(D(Is0))];
else, As0=[]; bs0=[];
end
if Ls1,
   As1 = zeros(p2*Ls1,N); bs1 = zeros(p2*Ls1,1);
   U = dm(Is1); if rsize==2, U = U*cos(pi/p2); end
   for i=1:p2,
      I = (i-1)*Ls1+(1:Ls1);
      As1(I,:) = cos(alpha(i) - om(Is1)*(0:N-1));
      bs1(I) = U;
   end
else, As1=[]; bs1=[];
end

% solve optimization problem
A = [As0;As1;Am;Ap]; b = [bs0;bs1;bm;bp];
if cheb,
   % linear program
   A = [zeros(2*(Lm+Lp+Ls0)+p2*Ls1,1),A;-ones(p2*Lmin,1),Amin];
   b = [b;bmin];
   h = lp([1;zeros(N,1)],A,b,[],[],[],2*Ls0); h=h(2:N+1);
else
   % quadratic program
   h = qp(toeplitz(t),-c,A,b,[],[],[],2*Ls0);
end
```

Specifying negative values in the constraint vectors `dc`, `dm`, or `dp` will leave the respective frequency regions unconstrained. Both programs use the linear and quadratic programming routines `lp` and `qp` of the Matlab Optimization Toolbox.

3.2 Exchange Algorithms

This section introduces new exchange algorithms for solving the constrained least squares problems (2.76) and (2.77) with constraints on the complex error, and on magnitude and phase errors, respectively. Reformulating these problems as quadratic programming problems with linear constraints as in Section 3.1 has two drawbacks. The first is the large number of constraints resulting in a high computational effort and in high memory requirements. The second drawback is the fact that replacing the nonlinear magnitude constraints by linear constraints introduces errors. These errors can be made arbitrarily small in the stopbands by increasing the number of vertices of the polygon shown in Figure 3.1. However, this also increases the number of linear constraints. In the passbands, the linearization errors depend on second and higher order terms involving the constraint functions $\delta_m(\omega)$ and $\delta_\phi(\omega)$. Hence, these errors are small if $\delta_m(\omega)$ and $\delta_\phi(\omega)$ are small. However, especially in situations where $\delta_\phi(\omega)/\text{rad} \gg \delta_m(\omega)$ holds in parts of the passbands, the linearization errors may become quite large.

Exchange algorithms can solve optimization problems with an arbitrarily large – or even infinite – number of constraints. They solve a sequence of subproblems with relatively small subsets of the total set of constraints. During the iteration process, a set of constraints containing those constraints that will be active at the optimum solution is identified. The solution of the last subproblem subject to these constraints equals the solution of the original problem. The efficiency of these methods strongly depends on the efficiency of the algorithms used for solving the subproblems. Hence, it is desirable to solve subproblems with linear constraints since there exist fast algorithms for linearly constrained problems. The amount of memory required by exchange algorithms is independent of the total number of constraints. Hence, the first drawback of the standard quadratic programming formulation is eliminated. Also the linearization errors can be eliminated. We have to distinguish between problems with convex and non-convex feasible regions. Since any convex feasible region can be represented by an infinite number of linear constraints (see [42]), such problems can directly be handled by exchange algorithms that solve a sequence of linearly constrained subproblems. These algorithms can be viewed as generalizations of cutting plane methods for convex programming problems [42]. Linearization of nonlinear constraints is no longer necessary. The only requirement for algorithms based on cutting plane methods to be applicable is the convexity of the feasible region. This requirement is satisfied for problem (2.76) with constraints on the complex error function. It is also satisfied for upper bounds on the magnitude response and for phase constraints if $\delta_\phi(\omega) \leq \pi/2$ holds. Only lower bounds on the magnitude response as used in the passbands result in a non-convex feasible region (see Section 2.3.1). Hence, for exchange algorithms based on cutting plane methods to be applicable, the lower bounds on the magnitude error have to be replaced by constraints such that the feasible region is convex. Non-convex feasible regions cannot be represented by an infinite number of linear constraints, and exchange algorithms based on cutting plane methods cannot be applied. To the best knowledge of the author of this thesis, exchange algorithms solving a sequence of linearly constrained subproblems for solving problems with an infinite number of constraints and a non-convex feasible region have not been given in the literature yet.

An exchange algorithm that solves convex constrained least squares problems will be discussed in Sections 3.2.1 and 3.2.2.1. In Section 3.2.2.2, the need for approximating the feasible region by a convex set is eliminated, and an exchange algorithm that exactly solves the non-convex constrained least squares problem with magnitude and phase constraints is presented.

3.2.1 Constraints on the Complex Error Function

From Section 2.3.1 we know that the set

$$K = \{\boldsymbol{h} \in \mathbb{R}^N \big| \ \ |E_c(\omega, \boldsymbol{h})| \leq \delta_c(\omega), \ \omega \in \Omega_B\},$$

is convex. Its representation by an infinite number of linear inequality constraints has been given in (3.1):

$$K = \{\boldsymbol{h} \in \mathbb{R}^N \big| \ \ \mathrm{Re}\left[E_c(\omega, \boldsymbol{h}) e^{j\alpha}\right] \leq \delta_c(\omega), \ \omega \in \Omega_B, \ \alpha \in [0, 2\pi)\}. \tag{3.13}$$

The optimum solution $\boldsymbol{h}_o$ to the constrained least squares problem (2.76) with the constraint $\boldsymbol{h} \in K$ will satisfy the inequality $|E_c(\omega, \boldsymbol{h}_o)| \leq \delta_c(\omega)$, $\forall \omega \in \Omega_B$, with equality only at a finite number of frequency points $\omega \in \Omega_{act}$ corresponding to the active set of constraints. Hence, only a finite number of linear constraints in (3.13) is satisfied with equality, where for each frequency point $\omega \in \Omega_{act}$ there is an associated angle $\alpha := \alpha(\omega) = -\phi_E(\omega, \boldsymbol{h}_o)$, $\omega \in \Omega_{act}$, with $\phi_E(\omega, \boldsymbol{h}) = \arg\{E_c(\omega, \boldsymbol{h})\}$. It is sufficient to find the set Ω_{act} and the associated arguments of the complex error function in order to compute the solution to the constrained optimization problem by solving a quadratic programming problem with a relatively small number of linear equality constraints. It is, however, very difficult to directly identify the set Ω_{act} and the corresponding angles α. Hence, the constraints of the subproblems will not be formulated as equality constraints but as inequality constraints. This allows to include more constraints in every iteration step and the set of constraints that will be active at the optimum solution need not be exactly identified. These constraints must only be a subset of the inequality constraints of the last subproblem.

An important question is how to identify frequency points that are likely to correspond to active constraints at the solution. Assume that for a given coefficient vector $\boldsymbol{h}^{(k)}$ the constraint $|E_c(\omega, \boldsymbol{h}^{(k)})| \leq \delta_c(\omega)$, $\forall \omega \in \Omega_B$, is violated. It is reasonable to use the local maximizers of $|E_c(\omega, \boldsymbol{h}^{(k)})| - \delta_c(\omega)$ that violate the constraint as candidate frequencies for active constraints at the solution. Using this strategy, $|E_c(\omega, \boldsymbol{h}^{(k)})|$ is evaluated in every iteration step with $\boldsymbol{h}^{(k)}$ being the solution of the actual subproblem, the local maximizers of $|E_c(\omega, \boldsymbol{h}^{(k)})| - \delta_c(\omega)$ at which the constraint is violated are identified, and new linear constraints are formulated according to

$$\operatorname{Re}\left[E_c(\omega, \boldsymbol{h}) e^{j\alpha(\omega)}\right] \leq \delta_c(\omega), \quad \omega \in \Omega_{viol}, \tag{3.14}$$

where Ω_{viol} is the set of frequencies that correspond to local maximizers of $|E_c(\omega, \boldsymbol{h}^{(k)})|$ violating the constraint. The angles $\alpha(\omega)$ are estimates of the negative argument of the complex error function at the next iteration step. A reasonable choice is $\alpha(\omega) = -\phi_E(\omega, \boldsymbol{h}^{(k)})$. The expression $\operatorname{Re}\left[E_c(\omega, \boldsymbol{h}) e^{j\alpha(\omega)}\right]$ serves as a linearization of the nonlinear function $|E_c(\omega, \boldsymbol{h})|$. A first order Taylor expansion of $|E_c(\omega, \boldsymbol{h})|$ about some $\boldsymbol{h}^{(k)}$ exists for all frequencies satisfying $|E_c(\omega, \boldsymbol{h}^{(k)})| > 0$ and is given by

$$|E_c(\omega, \boldsymbol{h}^{(k)})| + (\boldsymbol{h} - \boldsymbol{h}^{(k)})^T \nabla_{\boldsymbol{h}} |E_c(\omega, \boldsymbol{h}^{(k)})|, \tag{3.15}$$

with the gradient vector

$$\nabla_{\boldsymbol{h}} |E_c(\omega, \boldsymbol{h})| = \frac{\partial |E_c(\omega, \boldsymbol{h})|}{\partial \boldsymbol{h}}.$$

Writing $|E_c(\omega, \boldsymbol{h})|$ as

$$|E_c(\omega, \boldsymbol{h})| = \operatorname{Re}\left[E_c(\omega, \boldsymbol{h}) e^{-j\phi_E(\omega, \boldsymbol{h})}\right] = \operatorname{Re}\left\{\left[\boldsymbol{e}^H \boldsymbol{h} - D\left(e^{j\omega}\right)\right] e^{-j\phi_E(\omega, \boldsymbol{h})}\right\},$$

the gradient vector $\nabla_{\boldsymbol{h}} |E_c(\omega, \boldsymbol{h})|$ can be written as

$$\begin{aligned} \nabla_{\boldsymbol{h}} |E_c(\omega, \boldsymbol{h})| &= \operatorname{Re}\left[\boldsymbol{e}^*(\omega) e^{-j\phi_E(\omega, \boldsymbol{h})} - j E_c(\omega, \boldsymbol{h}) e^{-j\phi_E(\omega, \boldsymbol{h})} \nabla_{\boldsymbol{h}} \phi_E(\omega, \boldsymbol{h})\right] \\ &= \operatorname{Re}\left[\boldsymbol{e}^*(\omega) e^{-j\phi_E(\omega, \boldsymbol{h})}\right] - \operatorname{Re}\left[j |E_c(\omega, \boldsymbol{h})| \nabla_{\boldsymbol{h}} \phi_E(\omega, \boldsymbol{h})\right] \end{aligned} \tag{3.16}$$

for $|E_c(\omega, \boldsymbol{h})| > 0$, where $\nabla_{\boldsymbol{h}} \phi_E(\omega, \boldsymbol{h})$ is the gradient of $\phi_E(\omega, \boldsymbol{h}) = \arg\{E_c(\omega, \boldsymbol{h})\}$ with respect to the coefficient vector $\boldsymbol{h}$. Since the term in brackets on the right-hand side of (3.16) is purely imaginary, the gradient vector can finally be written as

$$\nabla_{\boldsymbol{h}} |E_c(\omega, \boldsymbol{h})| = \operatorname{Re}\left[\boldsymbol{e}^*(\omega) e^{-j\phi_E(\omega, \boldsymbol{h})}\right], \tag{3.17}$$

where $\boldsymbol{e}(\omega)$ is an $N \times 1$ vector with elements $e^{jn\omega}$, $n = 0, 1, \ldots, N-1$, as defined by (1.3). Inserting (3.17) into (3.15) yields the following expression for the first order Taylor series of $|E_c(\omega, \boldsymbol{h})|$ about $\boldsymbol{h}^{(k)}$:

$$|E_c(\omega, \boldsymbol{h}^{(k)})| + (\boldsymbol{h} - \boldsymbol{h}^{(k)})^T \nabla_{\boldsymbol{h}} |E_c(\omega, \boldsymbol{h}^{(k)})| = \operatorname{Re}\left[E_c(\omega, \boldsymbol{h}) e^{-j\phi_E(\omega, \boldsymbol{h}^{(k)})}\right]. \tag{3.18}$$

Hence, the choice $\alpha(\omega) = -\phi_E(\omega, \boldsymbol{h}^{(k)})$ in (3.14) is not only reasonable as stated earlier but also corresponds to a first order Taylor expansion of $|E_c(\omega, \boldsymbol{h})|$ about the current solution $\boldsymbol{h}^{(k)}$.

Another set of frequencies that are candidates for active constraints at the solution are those frequencies corresponding to active constraints at the solution to the respective subproblem. Hence, the current active constraints should be reused in the next iteration step. Note that due to convexity of the feasible region, linear constraints that have been derived from a first order Taylor expansion of the original constraints will never cut away parts of the original feasible region. Consequently, it is not critical to reuse old active constraints. The situation is different if non-convex feasible regions are considered. This case will be discussed in Section 3.2.2.2.

Up to now we have discussed how to find a new set of constraints given the solution to the current subproblem. It remains to choose an appropriate initial solution $\boldsymbol{h}^{(0)}$. For the constrained least squares problem, an obvious choice is to use the solution to the unconstrained least squares problem. Only if the unconstrained solution does not satisfy the constraints, further iteration steps become necessary.

Putting all these considerations together, we arrive at the following algorithm for solving the constrained least squares problem (2.76) with constraints on the complex error function:

0. $k = 0$. Solve the unconstrained quadratic minimization problem for $\boldsymbol{h}^{(0)}$.

1. Determine the local maxima of $|E_c(\omega, \boldsymbol{h}^{(k)})| - \delta_c(\omega),\ \omega \in \Omega_B$. If $|E_c(\omega, \boldsymbol{h}^{(k)})| \leq \delta_c(\omega),\ \omega \in \Omega_B$, is satisfied up to some specified tolerance, stop. Otherwise, go to 2.

2. Determine the set Ω_{viol} of local maximizers of $|E_c(\omega, \boldsymbol{h}^{(k)})| - \delta_c(\omega)$ that satisfy $|E_c(\omega, \boldsymbol{h}^{(k)})| > \delta_c(\omega)$.

3. Formulate a new set of constraints by using the current active constraints and the new constraints

$$\mathrm{Re}\left[E_c(\omega, \boldsymbol{h})e^{-j\phi_E(\omega, \boldsymbol{h}^{(k)})}\right] \leq \delta_c(\omega), \quad \omega \in \Omega_{viol}.$$

4. $k := k + 1$. Compute $\boldsymbol{h}^{(k)}$ by solving the quadratic programming problem subject to the constraints determined in step 3. Go to 1.

Note that the objective function of the quadratic programming subproblems is the same as the one of the original problem (2.76) and remains unchanged throughout all iteration steps. In the first iteration step ($k = 0$), there are of course no current active constraints, and the set of new constraints formulated in step 3 only consists of the constraints determined by the set Ω_{viol}. For solving the quadratic programming subproblems in step 4, it is advantageous to use the dual formulation given by (A.32). Due to the small number of constraints in every subproblem, the numbers of variables of the dual problems is relatively small. Moreover, the dual has only simple non-negativity constraints on its variables instead of general inequality constraints. This fact can be exploited by the algorithm solving the subproblems. In addition, due to these simple constraints, a feasible initial solution to the subproblems is easily found.

The proposed exchange algorithm actually solves a linearly constrained semi-infinite programming problem, i. e. a problem with finitely many variables and an infinite number of constraints. This is due to the continuous angle variable $\alpha \in [0, 2\pi)$ in (3.13), no matter if the problem is solved on a grid of frequencies or on continuous frequency intervals. The algorithm can solve both kinds of problems. The only difference occurs in the implementation of step 1, where the local maxima of $|E_c(\omega, \boldsymbol{h}^{(k)})| - \delta_c(\omega)$ are determined. The determination of local maxima on continuous frequency intervals can be done by first computing the maxima on a dense frequency grid, and refining these maxima using Newton's method (see e. g. [119]). However, the differences between the solutions computed on grids and on continuous intervals are negligible if the number of grid points is large enough. Choosing $10N$ to $20N$ grid points equally distributed over the relevant frequency intervals suffices for the computation of solutions with an accuracy that is sufficient for practical purposes. The authors of [90] and [91] use standard semi-infinite programming techniques for solving constrained Chebyshev FIR filter design problems. They emphasize that one important difference between their method and other methods is the fact that they consider continuous frequency intervals. However, this can be accomplished with any other exchange method by just changing the way the constraint function is evaluated. The way this is done is not an integral part of the optimization algorithm but just a detail of implementation. It should be noted that in [90] the authors also use dense frequency grids instead of continuous intervals. However, they use grids of up to 100.000 points which is unnecessary from a practical point of view and is one reason for the extensive computation times given in [91].

A good overview of general semi-infinite programming methods can be found in [42] and [100]. The algorithm presented in this section can be viewed as a special version of an algorithm by Roleff [105]. This algorithm is an implicit multiple exchange method, which means that the number of constraints may change from one iteration step to the next, and that several constraints are exchanged in every iteration step. In order to prove convergence of Roleff's method, the knowledge of a set enclosing the feasible region is required. This set is usually described by a finite number of linear inequality constraints. In order to satisfy existing convergence proofs, it is necessary to use these inequality constraints in all subproblems of an iterative algorithm. The filter design algorithms presented in [90, 91] are also based on Roleff's method. In [90, 91], the set enclosing the feasible region is constructed by solving an initial optimization problem with a finite number of linear inequality constraints. The active constraints of this initial problem define the set enclosing the feasible region. For guaranteed convergence, these constraints must be part of all subproblems formulated in further iteration steps. However, the authors of [90, 91] state that in their implementation of the algorithm these constraints are dropped as soon as none of them is active any more. The initial problem solved in [90, 91] is an approximation to the constrained Chebyshev problem with constraints on the complex error as formulated by (3.5) in Section 3.1.1. The circle in the complex plane is approximated by a square (i. e. $p = 2$) and the number of frequency points is approximately $N/2$. This initial constrained optimization problem is usually larger than the subproblems of all further iteration steps, and hence requires the largest amount of memory and a considerable part of the total computational effort. The inclusion of this initial problem is one factor that makes algorithms as the ones proposed in [90, 91] relatively inefficient and slow. The algorithm presented in [91] can be used to solve arbitrary convex semi-infinite programming problems. The first author of [91] applied this algorithm also to the constrained least squares design of linear phase FIR filters [93].

The algorithm presented in this section solves an unconstrained least squares problem as an initial problem, and then adds all constraints that are necessary to compute the optimum solution to the semi-infinite programming problem. This greatly reduces memory requirements and computational effort. Note, however, that this is not possible for solving constrained Chebyshev problems, since solving an unconstrained Chebyshev problem is not simpler than solving the constrained problem as is the case for linear least squares approximation. Unconstrained minimization of the objective function of the equivalent semi-infinite programming problem (2.78) is not possible, since in the case of Chebyshev approximation the objective function is linear. However, complex constrained Chebyshev problems as defined by (2.78) can be solved efficiently by the iterative reweighted least squares algorithm presented in Section 3.3.

3.2.2 Constraints on Magnitude and Phase Errors

This section considers the constrained least squares problem (2.77) with constraints on magnitude and phase errors. In Section (3.2.2.1) an approximate solution is computed by replacing the original non-convex feasible region by a convex feasible region and applying the exchange algorithm presented in Section 3.2.1. The errors introduced by this approach are shown to be very small as long as the phase error constraint function $\delta_\phi(\omega)$ is sufficiently small. If this is not the case, the original non-convex problem should be solved

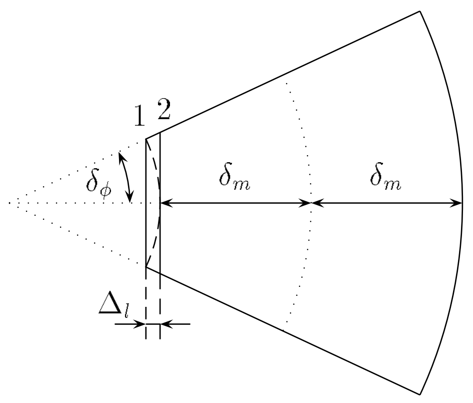


Figure 3.4: Replacing non-convex lower passband magnitude bound (dashed arc) by linear constraints (solid vertical lines). 1: new feasible region is larger than the original feasible region. 2: new feasible region is smaller than the original feasible region.

directly. This can be done by applying a modified version of the exchange algorithm used so far. This modified algorithm is presented in Section 3.2.2.2.

3.2.2.1 Approximation by a Convex Problem

In Section 2.3.1 it was shown that only the lower bounds on the magnitude error cause the feasible region of problems with magnitude and phase constraints to be non-convex. The upper magnitude bounds and the constraints on the phase error are convex if $\delta_\phi(\omega) \leq \pi/2$ holds. The latter requirement is, however, no restriction if phase approximation is desired at all. Hence, a convex optimization problem results if the lower magnitude bounds are replaced by an appropriate set of constraints. From Figure 3.2 it becomes obvious that the linear constraints used in Section 3.1.2 are in fact the best convex approximation to the non-convex lower magnitude bounds. Figure 3.4 illustrates the magnitude and phase constraints used in this section in the complex plane for one passband frequency point. Choosing the linear constraint denoted by '1' in Figure 3.4 results in the smallest convex region including the original non-convex feasible region. The linear constraint denoted by '2' in Figure 3.4 corresponds to the largest convex region contained in the original feasible region. The largest violation of the original lower magnitude constraints that may occur if constraint '1' is chosen has been given in Section 3.1.2:

$$\Delta_l(\omega) = \left[\left|D\left(e^{j\omega}\right)\right| - \delta_m(\omega)\right]\left[1 - \cos\delta_\phi(\omega)\right], \quad \omega \in \Omega^p. \tag{3.19}$$

This maximum violation is upper bounded by

$$\Delta_l(\omega) \leq \left[|D\left(e^{j\omega}\right)| - \delta_m(\omega)\right] \frac{\delta_\phi^2(\omega)}{2},$$

which shows that it decreases quadratically with decreasing $\delta_\phi(\omega)$. However, if there are passband regions where the functions $\delta_m(\omega)$ and $\delta_\phi(\omega)$ are chosen such that $\delta_\phi(\omega)/\text{rad} \gg \delta_m(\omega)$ holds, the maximum violation may become large relative to the magnitude constraint function $\delta_m(\omega)$. If the lower magnitude constraints are chosen according to '2' in Figure 3.4, no violations of the original constraints will occur. However, the errors introduced by this linearization will also be large whenever $\delta_\phi(\omega)/\text{rad} \gg \delta_m(\omega)$ is satisfied.

The set of infinitely many linear constraints describing the convex feasible region is given by

$$\begin{array}{rcll}
\text{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{j\alpha}\right] & \leq & |D\left(e^{j\omega}\right)| + \delta_m(\omega), & \omega \in \Omega_B,\ \alpha \in [0, 2\pi), \\
\text{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right] & \geq & L(\omega), & \omega \in \Omega^p \cap \Omega_B, \\
\text{Im}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right] & \leq & \tan\delta_\phi(\omega)\text{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right], & \omega \in \Omega^p \cap \Omega_B, \\
\text{Im}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right] & \geq & -\tan\delta_\phi(\omega)\text{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi_d(\omega)}\right], & \omega \in \Omega^p \cap \Omega_B,
\end{array} \tag{3.20}$$

where $L(\omega)$ is chosen as in (3.8) or (3.9), depending on whether the constraints denoted by '1' or '2' in Figure 3.4 are to be implemented. Note that the formulation of phase constraints in (3.20) assumes that $\delta_\phi(\omega) < \pi/2$ is satisfied.

The problem of minimizing the weighted least squared error subject to the constraints (3.20) can be solved by the exchange algorithm given in Section 3.2.1. The angles α in (3.20) for formulating new upper magnitude constraints are chosen as $\alpha := \alpha(\omega) = -\phi(\omega, \boldsymbol{h}^{(k)})$, where $\phi(\omega, \boldsymbol{h}) = \arg\{H\left(e^{j\omega}, \boldsymbol{h}\right)\}$ and $\boldsymbol{h}^{(k)}$ is the current iterate. Similar to the Taylor expansion of $|E_c(\omega, \boldsymbol{h})|$ used in Section 3.2.1, it can be shown that the first order Taylor series of $|H\left(e^{j\omega}, \boldsymbol{h}\right)|$ about $\boldsymbol{h}^{(k)}$ is given by

$$|H\left(e^{j\omega}, \boldsymbol{h}\right)| \approx \text{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}\right)e^{-j\phi(\omega, \boldsymbol{h}^{(k)})}\right] \tag{3.21}$$

for all frequencies satisfying $|H(e^{j\omega}, \boldsymbol{h}^{(k)})| > 0$. Hence, new upper magnitude constraints are formulated by using a Taylor linearization about the current iterate.

The exchange algorithm adapted to the problem under consideration works as follows:

0. $k = 0$. Solve the unconstrained quadratic minimization problem for $\boldsymbol{h}^{(0)}$.

1. Determine the local maxima of $E_m(\omega, \boldsymbol{h}^{(k)}) - \delta_m(\omega)$, $\omega \in \Omega_B$, and of $E_\phi(\omega, \boldsymbol{h}^{(k)}) - \delta_\phi(\omega)$, $\omega \in \Omega^p \cap \Omega_B$, and the local minima of $\text{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}^{(k)}\right)e^{-j\phi_d(\omega)}\right] - L(\omega)$, $\omega \in \Omega^p \cap \Omega_B$, and of $E_\phi(\omega, \boldsymbol{h}^{(k)}) + \delta_\phi(\omega)$, $\omega \in \Omega^p \cap \Omega_B$. If all constraints (3.20) are satisfied up to some specified tolerance, stop. Otherwise, go to 2.

2. For each of the functions considered in step 1, determine the sets of frequencies at which local maxima or minima violate the respective constraints in (3.20).

3. Formulate a new set of constraints by using the current active constraints and by imposing new constraints according to (3.20) at the respective sets of frequencies determined in step 2. Choose $\alpha := \alpha(\omega) = -\phi(\omega, \boldsymbol{h}^{(k)})$ in (3.20) for formulating new upper magnitude constraints.

4. $k := k + 1$. Compute $\boldsymbol{h}^{(k)}$ by solving the quadratic programming problem subject to the constraints determined in step 3. Go to 1.

For numerical implementation, it is again advantageous to solve the duals of the quadratic programming subproblems.

3.2.2.2 Exact Solution

The methods for solving the constrained least squares problem with magnitude and phase constraints presented so far approximate the non-convex feasible region by convex regions. These convex regions are described by a finite number of linear inequality constraints (Section 3.1.2) or by an infinite number of linear inequality constraints (Section 3.2.2.1), respectively. The errors introduced by these approximations increase with increasing phase errors (see (3.7) and (3.19)). Especially in cases where magnitude errors are considered to be much more undesirable than phase errors (i. e. $\delta_\phi(\omega)/\text{rad} \gg \delta_m(\omega)$), these convex regions represent the original non-convex feasible region only insufficiently.

For the derivation of a new algorithm that takes the non-convexity of the feasible region into account, it is important to understand why the exchange algorithm used in Sections (3.2.1) and (3.2.2.1) and all other standard algorithms for convex semi-infinite programming (see e. g. [42, 100]) cannot be used for problems with non-convex feasible regions. These algorithms solve a constrained optimization problem in every iteration step, where the set of constraints is composed of a part of the constraints used in the previous iteration step and new constraints determined by evaluating the original semi-infinite constraints at the current solution. Reusing a part of the old constraints is crucial for convergence. A nonlinear semi-infinite constraint can be written in the form

$$c(\omega, \boldsymbol{h}) \leq 0, \quad \forall \omega \in \Omega_B. \tag{3.22}$$

Each constraint used in a certain iteration step is derived from a first order Taylor expansion of $c(\omega, \boldsymbol{h})$ about some coefficient vector $\boldsymbol{h}_i$ evaluated at some frequency point ω:

$$c(\omega, \boldsymbol{h}_i) + (\boldsymbol{h} - \boldsymbol{h}_i)^T \nabla c(\omega, \boldsymbol{h}_i) \leq 0, \tag{3.23}$$

where $\nabla c(\omega, \boldsymbol{h})$ is the gradient vector of $c(\omega, \boldsymbol{h})$ with respect to $\boldsymbol{h}$. If $c(\omega, \boldsymbol{h})$ – and hence also the set defined by (3.22) – is convex, the inequality

$$c(\omega, \boldsymbol{h}_i) + (\boldsymbol{h} - \boldsymbol{h}_i)^T \nabla c(\omega, \boldsymbol{h}_i) \leq c(\omega, \boldsymbol{h}), \quad \forall \omega \in \Omega_B, \tag{3.24}$$

is satisfied for any $\boldsymbol{h}_i$ (see (A.15)). Hence, the constraint (3.23) can never be more restrictive than the original constraint (3.22). For this reason, the feasible regions of all subproblems always include the original feasible region. Constraints from previous iteration steps may become unnecessary and could be thrown away, but they will never cut away parts of the original feasible region. However, if $c(\omega, \boldsymbol{h})$ is not convex, the feasible set defined by (3.22) is non-convex in general and linear constraints as formulated by (3.23) might cut away parts of the original feasible region because (3.24) is not satisfied in general. Hence, old constraints must be removed because they might cut away the part of the feasible region that contains the optimum solution. Instead of reusing old constraints, some other constraints related to these old constraints must be used. The exchange algorithm presented in Section (3.2.1) reuses the active constraints of the previous iteration step. A logical extension is to formulate new constraints at the frequency points corresponding to active constraints of the previous iteration step. These constraints are used instead of the old active constraints. Hence, in every iteration step all constraints are formulated anew, and the propagation of old constraints cutting away parts of the feasible region is prevented.

Note that all these considerations only apply to non-convex constraint functions. In the problem under consideration, only the exchange rule for the lower bounds on the magnitude response must be adapted. The exchange rule for upper magnitude bounds and for phase constraints remains unchanged. The modified exchange algorithm for exactly solving the constrained least squares problem with magnitude and phase constraints works as follows:

0. $k = 0$. Solve the unconstrained quadratic minimization problem for $\boldsymbol{h}^{(0)}$.

1. Determine the local maxima of $E_m(\omega, \boldsymbol{h}^{(k)}) - \delta_m(\omega)$, $\omega \in \Omega_B$, and of $E_\phi(\omega, \boldsymbol{h}^{(k)}) - \delta_\phi(\omega)$, $\omega \in \Omega^p \cap \Omega_B$, and the local minima of $E_m(\omega, \boldsymbol{h}^{(k)}) + \delta_m(\omega)$, $\omega \in \Omega^p \cap \Omega_B$, and of $E_\phi(\omega, \boldsymbol{h}^{(k)}) + \delta_\phi(\omega)$, $\omega \in \Omega^p \cap \Omega_B$. If $|E_m(\omega, \boldsymbol{h}^{(k)})| \leq \delta_m(\omega)$, $\omega \in \Omega_B$, and $|E_\phi(\omega, \boldsymbol{h}^{(k)})| \leq \delta_\phi(\omega)$, $\omega \in \Omega^p \cap \Omega_B$, is satisfied up to some specified tolerance, stop. Otherwise, go to 2.

2. Determine the sets of frequencies at which local maxima or minima of the functions considered in step 1 violate the constraints.

3. Formulate a new set of constraints for the next iteration step:

- Reuse the current active upper magnitude bounds and active phase constraints.
- Impose new magnitude constraints at passband frequencies corresponding to active lower magnitude bounds in the current iteration step.
- Impose new magnitude and phase constraints at the respective sets of frequencies determined in step 2.

Formulate new magnitude constraints using a first order Taylor series of $E_m(\omega, \boldsymbol{h})$ about the current solution $\boldsymbol{h}^{(k)}$:

$$E_m(\omega, \boldsymbol{h}) \approx \operatorname{Re}\left[H\left(e^{j\omega}, \boldsymbol{h}\right) e^{-j\phi(\omega, \boldsymbol{h}^{(k)})}\right] - |D\left(e^{j\omega}\right)|.$$

4. $k := k + 1$. Compute $\boldsymbol{h}^{(k)}$ by solving the quadratic programming problem subject to the constraints determined in step 3. Go to 1.

To the best knowledge of the author of this thesis, no other algorithms for exactly solving the constrained least squares problem (2.77) with constraints on magnitude and phase errors have been published yet. The only algorithm that has been designed to solve a constrained least squares FIR filter design problem with a non-convex feasible region has been published by Sullivan and Adams [128, 129]. They consider magnitude and group delay constraints. In every iteration step, they use Taylor linearizations of the constraint functions. However, all functions are linearized about a coefficient vector that is a linear combination of the current and the previous coefficient vector. No old constraints are reused as is done for the convex constraints in the algorithm presented in this chapter. As can be seen from (3.21), a first order Taylor series of $|H\left(e^{j\omega}, \boldsymbol{h}\right)|$ about some coefficient vector $\boldsymbol{h}^{(k)}$ is only a good approximation to $H\left(e^{j\omega}, \boldsymbol{h}^{(k+1)}\right)$ if the phase $\phi(\omega, \boldsymbol{h}^{(k)})$ is a good approximation to the phase response $\phi(\omega, \boldsymbol{h}^{(k+1)})$ at the following iteration step. This is the case in the passbands because there a desired phase response is approximated. Hence, the passband phases will not change much from one iteration step to the next. However, in the stopbands only the magnitude response matters and the stopband phase response may change arbitrarily from one step to the next. For this reason, a Taylor linearization about some coefficient vector might be a very bad guess of the actual magnitude response in the stopbands. Hence, it is necessary to reuse old active stopband constraints. This has not been done in [128, 129] which is the probable cause for convergence problems occurring as soon as several stopband constraints become active as reported in [129]. In this case the algorithm published in [128, 129] converges "very slowly or not at all" [129]. As opposed to the algorithm by Sullivan and Adams, the algorithm presented in this section converged fast for all filter designs we tried. We have, however, no proof of convergence.

3.2.3 Design Example

In order to illustrate the properties of the exchange algorithms presented in this section, we design a chirp-lowpass filter according to the following specification:

$$D\left(e^{j\omega}\right) = \begin{cases} e^{j\phi_d(\omega)}, & 0 \leq \omega \leq \omega_p \\ 0, & \omega_s \leq \omega \leq \pi \end{cases}, \tag{3.25}$$

with the desired phase response $\phi_d(\omega)$ given by

$$\phi_d(\omega) = -\frac{N-1}{2}\omega - 8\pi\left(\frac{\omega}{\omega_p} - \frac{1}{2}\right)^2, \quad 0 \leq \omega \leq \omega_p, \tag{3.26}$$

where ω_p and ω_s are the passband and stopband edges, respectively, and N is the filter length. The desired phase response $\phi_d(\omega)$ corresponds to a linearly ascending desired group delay response. Its minimum and maximum values are given by

$$\begin{aligned} \tau_d(0) &= \frac{N-1}{2} - \frac{8\pi}{\omega_p}, \\ \tau_d(\omega_p) &= \frac{N-1}{2} + \frac{8\pi}{\omega_p}, \end{aligned}$$

respectively.

We choose $\omega_p = 0.2\pi$, $\omega_s = 0.225\pi$, and $N = 201$. With these choices, the desired group delay linearly ascends from 60 samples to 140 samples in the passband. We choose the constraints such that the maximum passband magnitude error is less than 0.007 and the minimum stopband attenuation is 45 dB. The passband phase error constraint is chosen as 0.007 radians. We design this filter once by constraining the magnitude of the complex error and a second time by independently constraining magnitude and phase errors. We choose the constraint functions $\delta_c(\omega) = \delta_m(\omega) = \delta_\phi(\omega) = 0.007$, $0 \leq \omega \leq \omega_p$, and $\delta_c(\omega) = \delta_m(\omega) = 10^{-45/20}$, $\omega_s \leq \omega \leq \pi$. We use the exchange algorithms presented in Sections 3.2.1 and 3.2.2.2. These algorithms are implemented by the Matlab programs `exch_ce` and `exch_mp2` given in Section 3.2.4. The filter specification is generated by the following commands:

```
N=201;
om=pi*[linspace(0,.2,800),linspace(.225,1,2800)];
omp=.2*pi;
Pd=-(N-1)/2*om-8*pi*(om/omp-.5).^2;
D=[exp(j*Pd(1:800)),zeros(1,2800)];
W=[ones(1,800),500*ones(1,2800)];
dc=[.007*ones(1,800),10^(-2.25)*ones(1,2800)]; dm=dc; dp=dc;
```

The filters are designed by the commands

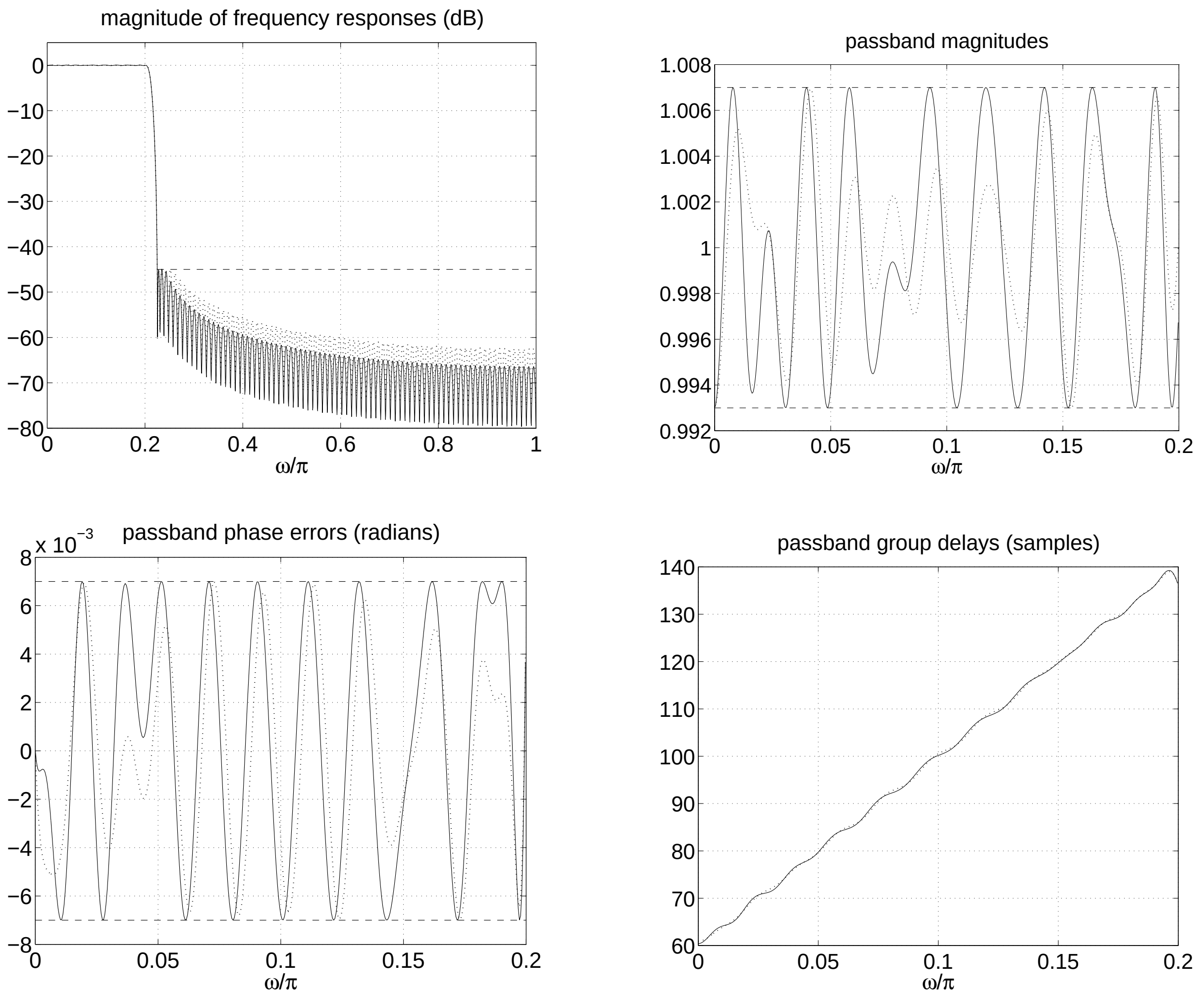


Figure 3.5: Constrained least squares designs of chirp-lowpass filter (length $N = 201$). Solid curves: design with independent constraints on magnitude and phase errors. Dotted curves: design with constraints on the complex error. Dashed lines: constraints.

```
    hc = exch_ce(N,om,D,W,dc);
    hmp = exch_mp2(N,om,D,W,dm,dp);
```

Figure 3.5 shows the results of these two designs. The stopband error energy is smaller for the filter designed with independent magnitude and phase constraints. This is due to the fact that for given maximum magnitude and phase errors, the corresponding constraints on the complex error function are more restrictive than the independent constraints on magnitude and phase errors. This can also be seen from the passband behavior of the magnitude responses and the phase errors shown in Figure 3.5. The solution computed by imposing magnitude and phase constraints (solid curves in Figure 3.5) touches the boundaries imposed by the passband constraints more often than the solution computed by imposing constraints on the complex error function.

We again consider the filter specification given by (3.25) and (3.26). We want to show the differences between solutions computed by the methods presented in Sections 3.2.2.1 and 3.2.2.2. Both methods solve the constrained least squares problem (2.77) with constraints on magnitude and phase errors. However, in Section 3.2.2.1 the non-convex feasible region is approximated by a convex region. Figure 3.4 illustrates two different sets of constraints that result in a convex feasible region. Both choices introduce errors. These errors either result in violations of the original magnitude constraints or in an increased objective function value due to a reduction of the feasible region. The errors are small as long as the phase constraint function $\delta_\phi(\omega)$ is small. The phase constraint function of the previous example is small enough such that the solutions computed by the algorithms of Sections 3.2.2.1 and 3.2.2.2 are virtually the same. However, if we trade passband magnitude error versus phase error by specifying $\delta_m(\omega) = 0.001,\ 0 \leq \omega \leq \omega_p$, and $\delta_\phi(\omega) = 0.02$ rad, $0 \leq \omega \leq \omega_p$, the difference between the solutions will become more pronounced. We design three filters according to the specifications (3.25) and (3.26) with the modified magnitude and phase constraint functions in the passband. Two filters are designed by the method of Section (3.2.2.1) with the constraints chosen according to '1' and '2' in Figure 3.4, respectively. The third filter is designed by the method presented in Section 3.2.2.2 which exactly solves the non-convex design problem. The algorithm presented in Section 3.2.2.1 is implemented by the Matlab program `exch_mp1` given in Section 3.2.4. The filters are computed by the following Matlab commands:

```
dm=[.001*ones(1,800),10^(-2.25)*ones(1,2800)];
dp=[.02*ones(1,800),-ones(1,2800)];
h11 = exch_mp1(N,om,D,W,dm,dp,1);
h12 = exch_mp1(N,om,D,W,dm,dp,2);
h2 = exch_mp2(N,om,D,W,dm,dp);
```

Figure 3.6 shows the passband magnitude responses and passband phase errors for the three designs. The solid curves correspond to the exact solution. The dotted and dashed curves correspond to approximate solutions computed with the method of Section 3.2.2.1. The solution shown by the dotted curves has been computed with constraints according to '1' in Figure 3.4. Hence, violations of lower magnitude bounds occur. These violations take on their maximum value in regions where the phase error is small (see Figure 3.6). The solution shown by the dashed curves has been computed with constraints according to '2' in Figure 3.4. No violations occur, but the constraints are too restrictive. In regions where the phase error is large, the lower magnitude constraints are most restrictive (see Figure 3.6).

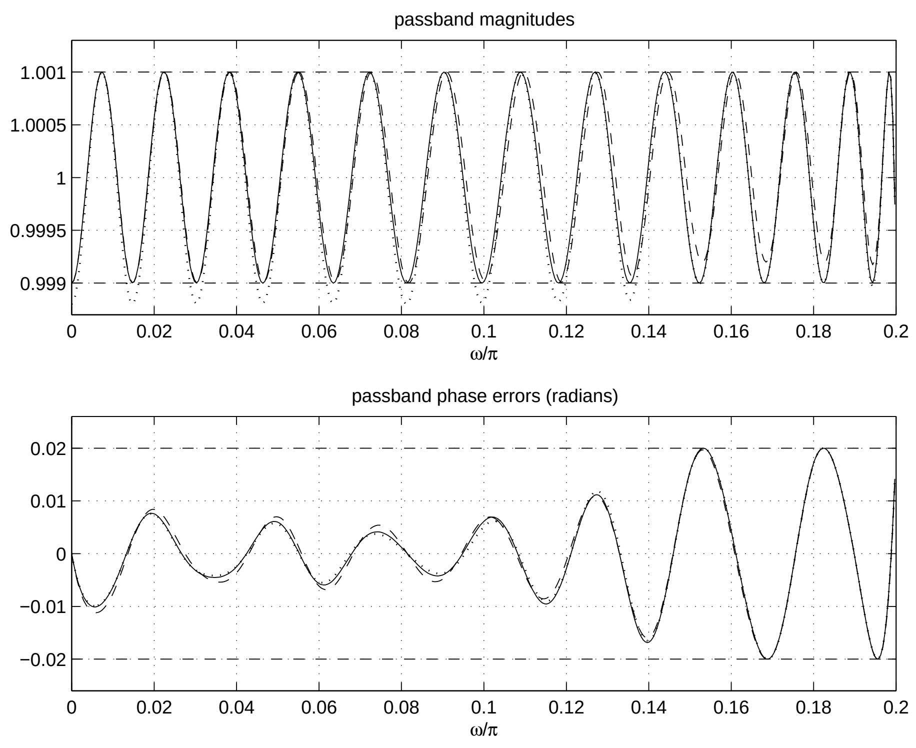


Figure 3.6: Constrained least squares designs of chirp-lowpass filter (length $N = 201$). Constraints on magnitude and phase errors. Solid curves: exact solution. Dotted curves: approximate solution with constraints '1' in Figure 3.4. Dashed curves: approximate solution with constraints '2' in Figure 3.4. Dashed lines: constraints.

3.2.4 Matlab Programs

The Matlab programs `exch_ce`, `exch_mp1`, and `exch_mp2` implement the methods described in Sections 3.2.1, 3.2.2.1, and 3.2.2.2, respectively. They all work with a grid of frequencies for specifying the desired response $D\left(e^{j\omega}\right)$, the weighting function $W(\omega)$, and the constraint functions $\delta_c(\omega)$, $\delta_m(\omega)$, and $\delta_\phi(\omega)$. Hence, no analytic expressions are necessary, and desired responses derived from measurements can directly be used. The input parameter `N` denotes the filter length. The vector `om` specifies the frequency grid on which the desired response, the weighting function, and the constraint functions are defined. The elements of `om` must be in the interval $[0, \pi]$. The desired frequency response, the weighting function, and the constraint functions are passed to the programs using the vectors `D`, `W`, `dc`, `dm`, and `dp`, respectively. These vectors define the respective functions at the grid points defined by the vector `om`. The program `exch_mp1` implementing the method described in Section 3.2.2.1 needs an additional scalar input argument `rsize` which takes on the values 1 or 2 depending on whether the linear lower magnitude constraints denoted by '1' or '2' in Figure 3.4 are to be used.

```
    function h = exch_ce(N,om,D,W,dc)
```

```
% EXCH_CE: least squares design of FIR filters with arbitrary magnitude
% and phase responses with constraints on the magnitude of the complex
% error function
% h = exch_ce(N,om,D,W,dc)
%
% h       filter impulse response
% N       filter length
% om      frequency grid (0 <= om <= pi)
% D       complex desired frequency response on the grid om
% W       positive weighting function on the grid om
% dc      positive constraint function on the grid om;
%         for intervals without constraints specify negative dc
%
% EXAMPLE:
% low-delay bandpass filter with passband and stopband constraints
% om=pi*[linspace(0,.34,750),linspace(.4,.6,500),linspace(.66,1,750)];
% D=[zeros(1,750),exp(-j*30*om(751:1250)),zeros(1,750)];
% W=[500*ones(1,750),ones(1,500),500*ones(1,750)];
% dc=[.001*ones(1,750),.01*ones(1,500),.001*ones(1,750)];
% h = exch_ce(100,om,D,W,dc);
%
% Author: Mathias C. Lang, Vienna University of Technology, Aug. 98

om=om(:); D=D(:); W=W(:); dc=dc(:); Lom=length(om);
tol = 1e-3; se = sqrt(eps);

% set up objective function: min h'*toeplitz(t)*h - 2*c'*h
t=zeros(N,1); c=t;
for i = 1:N,
   t(i) = W.'*cos((i-1)*om);
   c(i) = W.'*(real(D).*cos((i-1)*om) - imag(D).*sin((i-1)*om));
end
t=t/Lom; c=c/Lom;

% solve unconstrained L2 problem
[L,d] = invtoep(t);
for i=1:N, L(i:N,i)=L(i:N,i)*sqrt(d(i));end  % inv(toeplitz(t))=L*L'
h = L*(L'*c); h_uc = h;

% prepare for iteration
Aact=[]; bact=[];
IB = find(dc>=0); if ~length(IB), return; end
allconstr = 1; if length(IB) < length(dc), allconstr = 0; end
dc = dc(IB); Dm = abs(D(IB)); Dp = angle(D(IB)); om = om(IB);
evec = exp(-j*om);

% iterate
while 1,

   % compute error maxima
   E = polyval(h(N:-1:1),evec) - D(IB);
   Em = abs(E); Ep = angle(E);
   Em_dc = Em-dc;
   Imax = locmax(Em_dc);

   % find violating maxima
   Iviol = find(Em_dc(Imax)>0); Iviol = Imax(Iviol);
   if length(Iviol)==length(Imax) & allconstr,
      disp('There is no feasible solution. Relax the constraints.');
      break;
   end

   % make a plot
   plot(om/pi,Em_dc,om(Iviol)/pi,Em_dc(Iviol),'rx');
   title('constraint violation');
   xlabel('\omega/\pi'); grid, drawnow

   % check stopping criterion
   if all(Em_dc(Iviol)<=max(dc(Iviol)*tol,se)), break; end

   % formulate new constraints
   omviol = om(Iviol); Epviol = Ep(Iviol);
   Dmviol = Dm(Iviol); Dpviol = Dp(Iviol);
   Anew = cos(omviol*(0:N-1)+Epviol(:,ones(N,1)));
   bnew = Dmviol.*cos(Dpviol-Epviol) + dc(Iviol);

   % solve subproblem (dual formulation)
   A=[Aact;Anew]; b=[bact;bnew]; nb=length(b);
   AL = A*L; H = AL*AL'; f = b-A*h_uc;
   for i=1:nb, H(i,i) = H(i,i)+se; end
   l = nnqmin(H,f);
   h = h_uc-L*(AL'*l);

   % find active constraints
   act = find(l>se); Aact = A(act,:); bact = b(act);

end
```

```
function h = exch_mp1(N,om,D,W,dm,dp,rsize)
% EXCH_MP1: constrained least squares design of FIR filters with
% arbitrary magnitude and phase responses
%
% h = exch_mp1(N,om,D,W,dm,dp,rsize)
%
% h      filter impulse response
% N      filter length
% om     frequency grid (0 <= om <= pi)
% D      complex desired frequency response on the grid om
% W      positive weighting function on the grid om
% dm     positive magnitude constraint function on the grid om;
%        for intervals without constraints specify negative dm
% dp     positive phase constraint function on the grid om;
%        for intervals without constraints specify negative dp
% rsize  parameter specifying size of feasible region; for rsize=1,
%        the feasible region is approximated from the outside; for
%        rsize=2 it is approximated from the inside
%
% EXAMPLE:
% Low-delay bandpass filter with passband and stopband constraints
% om=pi*[linspace(0,.34,750),linspace(.4,.6,500),linspace(.66,1,750)];
% D=[zeros(1,750),exp(-j*30*om(751:1250)),zeros(1,750)];
% W=[500*ones(1,750),ones(1,500),500*ones(1,750)];
% dm=[.001*ones(1,750),.01*ones(1,500),.001*ones(1,750)]; dp=dm;
% h = exch_mp1(100,om,D,W,dm,dp,1);
%
% Author: Mathias C. Lang, Vienna University of Technology, Aug. 98

om=om(:); D=D(:); W=W(:); dm=dm(:); dp=dp(:);
tol = 1e-3; Dm = abs(D); Dp = angle(D); Lom = length(om);
se = sqrt(eps);

% set up objective function: min h'*toeplitz(t)*h - 2*c'*h
t=zeros(N,1); c=t;
for i = 1:N,
   t(i) = W.'*cos((i-1)*om);
   c(i) = W.'*(real(D).*cos((i-1)*om) - imag(D).*sin((i-1)*om));
end
t=t/Lom; c=c/Lom;

% solve unconstrained L2 problem
[L,d] = invtoep(t);
for i=1:N, L(i:N,i)=L(i:N,i)*sqrt(d(i));end  % inv(toeplitz(t))=L*L'
h = L*(L'*c); h_uc = h;

% prepare for iteration
Aact=[]; bact=[];
dp(find(dp<0 & dm>=0 & Dm))=pi/8;
IB = find(dm>=0 | (dp>=0 & Dm));
if ~length(IB), break; end
dm = dm(IB); dp = dp(IB); Dm = Dm(IB); Dp = Dp(IB); om = om(IB);
Im = find(dm>=0); Impass = find(dm>=0 & Dm); Ip = find(dp>=0 & Dm);
Implot = find(dm>0); Ipplot = find(dp>0 & Dm);
evec = exp(-j*om);
Lo = Dm(Impass)-dm(Impass); if rsize == 1, Lo=Lo.*cos(dp(Impass)); end

% iterate
while 1,

   % compute constraint maxima
```

```
H = polyval(h(N:-1:1),evec); Hm = abs(H); Hp = angle(H);
c_um = Hm(Im) - Dm(Im) - dm(Im);
c_lm = -real(H(Impass).*exp(-j*Dp(Impass))) + Lo;
c_up = princval(Hp(Ip) - Dp(Ip) - dp(Ip));
c_lp = princval(-Hp(Ip) + Dp(Ip) - dp(Ip));
Imax_um = locmax(c_um); Imax_lm = locmax(c_lm);
Imax_up = locmax(c_up); Imax_lp = locmax(c_lp);

% find violating maxima
Iv_um = find(c_um(Imax_um)>0); Iv_um = Imax_um(Iv_um);
Iv_lm = find(c_lm(Imax_lm)>0); Iv_lm = Imax_lm(Iv_lm);
Iv_up = find(c_up(Imax_up)>0); Iv_up = Imax_up(Iv_up);
Iv_lp = find(c_lp(Imax_lp)>0); Iv_lp = Imax_lp(Iv_lp);

% make a plot
if length(Implot) & length(Ipplot),
   subplot(2,1,1),
   plot(om(Implot)/pi,(Hm(Implot)-Dm(Implot))./dm(Implot));
   title('normalized magnitude error');
   xlabel('\omega/\pi'); grid;
   subplot(2,1,2),
   plot(om(Ipplot)/pi,princval(Hp(Ipplot)-Dp(Ipplot))./dp(Ipplot));
   title('normalized phase error');
   xlabel('\omega/\pi'); grid;
   drawnow
end

% check stopping criterion
if all([c_um(Iv_um)<=max(dm(Im(Iv_um))*tol,se);...
        c_lm(Iv_lm)<=max(dm(Impass(Iv_lm))*tol,se);...
        c_up(Iv_up)<=max(dp(Ip(Iv_up))*tol,se);...
        c_lp(Iv_lp)<=max(dp(Ip(Iv_lp))*tol,se)]),
break; end

% formulate new constraints
if length(Iv_um),
   I_um = Im(Iv_um);
   A_um_new = cos(om(I_um)*(0:N-1)+Hp(I_um,ones(N,1)));
   b_um_new = Dm(I_um) + dm(I_um);
else, A_um_new=[]; b_um_new=[]; end
if length(Iv_lm),
   I_lm = Impass(Iv_lm);
   A_lm_new = -cos(om(I_lm)*(0:N-1)+Dp(I_lm,ones(N,1)));
   b_lm_new = -Lo(Iv_lm);
else, A_lm_new=[]; b_lm_new=[]; end
if length(Iv_up),
   I_up = Ip(Iv_up);
   A_up_new = -sin(om(I_up)*(0:N-1)+Dp(I_up,ones(N,1)))-...
     tan(dp(I_up,ones(1,N))).*cos(om(I_up)*(0:N-1)+Dp(I_up,ones(N,1)));
   b_up_new = zeros(size(I_up));
```

```
      else, A_up_new=[]; b_up_new=[]; end
      if length(Iv_lp),
         I_lp = Ip(Iv_lp);
         A_lp_new = sin(om(I_lp)*(0:N-1)+Dp(I_lp,ones(N,1)))-...
           tan(dp(I_lp,ones(1,N))).*cos(om(I_lp)*(0:N-1)+Dp(I_lp,ones(N,1)));
         b_lp_new = zeros(size(I_lp));
      else, A_lp_new=[]; b_lp_new=[]; end

      % solve subproblem (dual formulation)
      A=[Aact;A_um_new;A_lm_new;A_up_new;A_lp_new];
      b=[bact;b_um_new;b_lm_new;b_up_new;b_lp_new]; nb=length(b);
      AL = A*L; H = AL*AL'; f = b-A*h_uc;
      for i=1:nb, H(i,i) = H(i,i)+se; end
      l = nnqmin(H,f);
      h = h_uc-L*(AL'*l);

      % check for infeasibility
      if max(A*h-b) > se,
         disp('There is no feasible solution. Relax the constraints.');
         break;
      end

      % find active constraints
      act = find(l>se); Aact = A(act,:); bact = b(act);

   end
```

```
   function h = exch_mp2(N,om,D,W,dm,dp)
   % EXCH_MP2: constrained least squares design of FIR filters with
   % arbitrary magnitude and phase responses
   %
   % h = exch_mp2(N,om,D,W,dm,dp)
   %
   % h      filter impulse response
   % N      filter length
   % om     frequency grid (0 <= om <= pi)
   % D      complex desired frequency response on the grid om
   % W      positive weighting function on the grid om
   % dm     positive magnitude constraint function on the grid om;
   %        for intervals without constraints specify negative dm
   % dp     positive phase constraint function on the grid om;
   %        for intervals without constraints specify negative dp
   %
   % EXAMPLE:
   % Low-delay bandpass filter with passband and stopband constraints
   % om=pi*[linspace(0,.34,750),linspace(.4,.6,500),linspace(.66,1,750)];
   % D=[zeros(1,750),exp(-j*30*om(751:1250)),zeros(1,750)];
   % W=[500*ones(1,750),ones(1,500),500*ones(1,750)];
   % dm=[.001*ones(1,750),.01*ones(1,500),.001*ones(1,750)]; dp=dm;
```

```
% h = exch_mp2(100,om,D,W,dm,dp);
%
% Author: Mathias C. Lang, Vienna University of Technology, Aug. 98

om=om(:); D=D(:); W=W(:); dm=dm(:); dp=dp(:);
tol = 1e-3; Dm = abs(D); Dp = angle(D); Lom = length(om);
se = sqrt(eps);

% set up objective function: min h'*toeplitz(t)*h - 2*c'*h
t=zeros(N,1); c=t;
for i = 1:N,
   t(i) = W.'*cos((i-1)*om);
   c(i) = W.'*(real(D).*cos((i-1)*om) - imag(D).*sin((i-1)*om));
end
t=t/Lom; c=c/Lom;

% solve unconstrained L2 problem
[L,d] = invtoep(t);
for i=1:N, L(i:N,i)=L(i:N,i)*sqrt(d(i));end  % inv(toeplitz(t))=L*L'
h = L*(L'*c); h_uc = h;

% prepare for iteration
Aact=[]; bact=[]; I_lm=[]; I_act_lm=[]; n_lm=0;
dp(find(dp<0 & dm>=0 & Dm))=pi/8;
IB = find(dm>=0 | (dp>=0 & Dm));
if ~length(IB), break; end
dm = dm(IB); dp = dp(IB); Dm = Dm(IB); Dp = Dp(IB); om = om(IB);
Im = find(dm>=0); Impass = find(dm>=0 & Dm); Ip = find(dp>=0 & Dm);
Implot = find(dm>0); Ipplot = find(dp>0 & Dm);
evec = exp(-j*om);

% iterate
while 1,

   % compute constraint maxima
   H = polyval(h(N:-1:1),evec); Hm = abs(H); Hp = angle(H);
   c_um = Hm(Im) - Dm(Im) - dm(Im);
   c_lm = -Hm(Impass) + Dm(Impass) - dm(Impass);
   c_up = princval(Hp(Ip) - Dp(Ip) - dp(Ip));
   c_lp = princval(-Hp(Ip) + Dp(Ip) - dp(Ip));
   Imax_um = locmax(c_um); Imax_lm = locmax(c_lm);
   Imax_up = locmax(c_up); Imax_lp = locmax(c_lp);

   % find violating maxima
   Iv_um = find(c_um(Imax_um)>0); Iv_um = Imax_um(Iv_um);
   Iv_lm = find(c_lm(Imax_lm)>0); Iv_lm = Imax_lm(Iv_lm);
   Iv_up = find(c_up(Imax_up)>0); Iv_up = Imax_up(Iv_up);
   Iv_lp = find(c_lp(Imax_lp)>0); Iv_lp = Imax_lp(Iv_lp);

   % make a plot
```

```
if length(Implot) & length(Ipplot),
   subplot(2,1,1),
   plot(om(Implot)/pi,(Hm(Implot)-Dm(Implot))./dm(Implot));
   title('normalized magnitude error');
   xlabel('\omega/\pi'); grid;
   subplot(2,1,2),
   plot(om(Ipplot)/pi,princval(Hp(Ipplot)-Dp(Ipplot))./dp(Ipplot));
   title('normalized phase error');
   xlabel('\omega/\pi'); grid;
   drawnow
end

% check stopping criterion
if all([c_um(Iv_um)<=max(dm(Im(Iv_um))*tol,se);...
        c_lm(Iv_lm)<=max(dm(Impass(Iv_lm))*tol,se);...
        c_up(Iv_up)<=max(dp(Ip(Iv_up))*tol,se);...
        c_lp(Iv_lp)<=max(dp(Ip(Iv_lp))*tol,se)]),
break; end

% formulate new constraints
if length(Iv_um),
   I_um = Im(Iv_um);
   A_um_new = cos(om(I_um)*(0:N-1)+Hp(I_um,ones(N,1)));
   b_um_new = Dm(I_um) + dm(I_um);
else, A_um_new=[]; b_um_new=[]; end
if length(Iv_lm) | length(I_act_lm),
   I_lm = [Impass(Iv_lm);I_act_lm]; n_lm = length(I_lm);
   A_lm_new = -cos(om(I_lm)*(0:N-1)+Hp(I_lm,ones(N,1)));
   b_lm_new = -Dm(I_lm) + dm(I_lm);
else, A_lm_new=[]; b_lm_new=[]; end
if length(Iv_up),
   I_up = Ip(Iv_up);
   A_up_new = -sin(om(I_up)*(0:N-1)+Dp(I_up,ones(N,1)))-...
     tan(dp(I_up,ones(1,N))).*cos(om(I_up)*(0:N-1)+Dp(I_up,ones(N,1)));
   b_up_new = zeros(size(I_up));
else, A_up_new=[]; b_up_new=[]; end
if length(Iv_lp),
   I_lp = Ip(Iv_lp);
   A_lp_new = sin(om(I_lp)*(0:N-1)+Dp(I_lp,ones(N,1)))-...
     tan(dp(I_lp,ones(1,N))).*cos(om(I_lp)*(0:N-1)+Dp(I_lp,ones(N,1)));
   b_lp_new = zeros(size(I_lp));
else, A_lp_new=[]; b_lp_new=[]; end

% solve subproblem (dual formulation)
A=[A_lm_new;Aact;A_um_new;A_up_new;A_lp_new];
b=[b_lm_new;bact;b_um_new;b_up_new;b_lp_new]; nb=length(b);
AL = A*L; H = AL*AL'; f = b-A*h_uc;
for i=1:nb, H(i,i) = H(i,i)+sqrt(eps); end
l = nnqmin(H,f);
h = h_uc-L*(AL'*l);
```

```
      % check for infeasibility
      if max(A*h-b) > sqrt(eps),
         disp('There is no feasible solution. Relax the constraints.');
         break;
      end

      % find active constraints
      act = find(l>sqrt(eps));
      I_act_lm = I_lm(act(find(act<=n_lm)));
      act = act(find(act>n_lm));
      Aact = A(act,:); bact = b(act);

   end
```

The program `locmax` for determining the local maxima of a real-valued vector has been given in Section 2.2.4. The function `nnqmin` for solving the duals of the quadratic programming subproblems is given in Appendix A. The function `invtoep` for computing the Cholesky factorization of the inverse of a positive definite Toeplitz matrix is given in Appendix B. Finally, the function `princval` used by the programs `exch_mp1` and `exch_mp2` computes the principal value of arbitrary phase values:

```
function p = princval(phi)
% function p = princval(phi)
% computes the principal value p [-pi,pi) of the arbitrary
% phase value phi (in rad); phi can be a matrix

p = rem(phi,2*pi);
gt = find(p >= pi); p(gt) = p(gt) - 2*pi;
le = find(p < -pi); p(le) = p(le) + 2*pi;
```

3.3 Iterative Reweighted Least Squares

In this section we present a generalization of Lawson's algorithm (see Section 2.2.2) that allows to solve the constrained least squares problem (2.76) and the constrained Chebyshev problem (2.78) with constraints on the complex error function. We exploit the fact that we can find the solutions to these constrained optimization problems by solving a linear weighted least squares problem. However, the appropriate weighting function is unknown. The algorithm finds this weighting function by solving a sequence of weighted least squares problems with different weighting functions. For this reason, this algorithm belongs to the class of *iterative reweighted least squares (IRLS)* algorithms. IRLS algorithms have been used for FIR filter design before [15, 21, 72], but with different and less flexible design criteria.

Similar to Lawson's algorithm, the proposed algorithm works with discrete frequency grids instead of continuous intervals. An extension to continuous intervals is possible, although it is of minor practical relevance. The number of grid points is not a critical parameter in the proposed algorithm. Hence, it can be made large enough in order to represent the continuous frequency intervals sufficiently well. The domain of approximation $\Omega = \Omega_{min} \cup \Omega_B$ is replaced by a grid of L $(L > N)$ frequency points ω_i. Define sets of indices I_{min} and I_B corresponding to the frequency sets Ω_{min} and Ω_B:

$$\omega_i \in \Omega_{min} \Leftrightarrow i \in I_{min}, \qquad \omega_i \in \Omega_B \Leftrightarrow i \in I_B.$$

The index set I corresponding to the frequency set Ω is the union of I_{min} and I_B:

$$I = I_{min} \cup I_B.$$

For convenience we repeat the problems under consideration, taking their discrete nature into account. The discrete constrained least squares problem reads

$$\begin{array}{ll} \underset{\boldsymbol{h}}{\text{minimize}} & \displaystyle\sum_{i \in I_{min}} W(\omega_i)|E_c(\omega_i, \boldsymbol{h})|^2 \\ \text{subject to} & |E_c(\omega_i, \boldsymbol{h})| \leq \delta_c(\omega_i), \quad \omega_i \in \Omega_B. \end{array} \tag{3.27}$$

The discrete constrained Chebyshev problem reads

$$\begin{array}{ll} \underset{\boldsymbol{h}}{\text{minimize}} & \displaystyle\max_{i \in I_{min}} W(\omega_i)|E_c(\omega_i, \boldsymbol{h})| \\ \text{subject to} & |E_c(\omega_i, \boldsymbol{h})| \leq \delta_c(\omega_i), \quad \omega_i \in \Omega_B. \end{array} \tag{3.28}$$

The formulation of the constrained Chebyshev problem (3.28) assumes that the sets Ω_{min} and Ω_B are disjoint. This is not necessarily the case for the constrained least squares problem (3.27).

We now show that the optimum solutions to problems (3.27) and (3.28) can be computed by solving a linear weighted least squares problem. We showed in Section 2.3.1 that the problems (3.27) and (3.28) are convex optimization problems. Hence, the Kuhn-Tucker conditions are necessary and sufficient optimality conditions (see Appendix A). Note that problem (3.27) is not changed by reformulating the constraints as

$$|E_c(\omega_i, \boldsymbol{h})|^2 \leq \delta_c^2(\omega_i), \quad \omega_i \in \Omega_B. \tag{3.29}$$

The Lagrangian function corresponding to problem (3.27) with the constraints formulated as in (3.29) is given by

$$L(\boldsymbol{h}, \boldsymbol{\lambda}) = \sum_{i \in I_{min}} W(\omega_i)|E_c(\omega_i, \boldsymbol{h})|^2 + \sum_{i \in I_B} \lambda_i \left[|E_c(\omega_i, \boldsymbol{h})|^2 - \delta_c^2(\omega_i)\right], \tag{3.30}$$

with the Lagrange multipliers λ_i contained in the vector $\boldsymbol{\lambda}$. Defining $W(\omega_i) = 0,\ \forall i \notin I_{min}$ and $\lambda_i = 0,\ \forall i \notin I_B$ the Lagrangian function (3.30) can be written as

$$L(\boldsymbol{h}, \boldsymbol{\lambda}) = \sum_{i \in I} [W(\omega_i) + \lambda_i] \, |E_c(\omega_i, \boldsymbol{h})|^2 - \sum_{i \in I} \lambda_i \delta_c^2(\omega_i). \tag{3.31}$$

Let us consider the general case where the vector of filter coefficients $\boldsymbol{h}$ is complex. For this case, the Kuhn-Tucker conditions (A.12) are given by

$$\frac{\partial L}{\partial \boldsymbol{h}^*} = \mathbf{0}, \qquad \frac{\partial L}{\partial \boldsymbol{\lambda}} \leq \mathbf{0}, \qquad \boldsymbol{\lambda} \geq \mathbf{0}, \qquad \boldsymbol{\lambda}^T \frac{\partial L}{\partial \boldsymbol{\lambda}} = 0, \tag{3.32}$$

where we define

$$\frac{\partial}{\partial \boldsymbol{h}^*} = \frac{1}{2} \left(\frac{\partial}{\partial \boldsymbol{h}_R} + j \frac{\partial}{\partial \boldsymbol{h}_I} \right), \tag{3.33}$$

with $\boldsymbol{h}_R = \text{Re}(\boldsymbol{h})$ and $\boldsymbol{h}_I = \text{Im}(\boldsymbol{h})$. The squared magnitude of the complex error function can be written as

$$\begin{aligned} |E_c(\omega, \boldsymbol{h})|^2 &= |\boldsymbol{e}^H(\omega)\boldsymbol{h} - D\left(e^{j\omega}\right)|^2 \\ &= \boldsymbol{h}^H \boldsymbol{e}(\omega)\boldsymbol{e}^H(\omega)\boldsymbol{h} - 2\text{Re}\left[D^*(e^{j\omega})\boldsymbol{e}^H(\omega)\boldsymbol{h}\right] + |D\left(e^{j\omega}\right)|^2, \end{aligned}$$

where the vector $\boldsymbol{e}(\omega)$ has been defined in (1.3). Using definition (3.33), its derivative with respect to $\boldsymbol{h}^*$ is given by

$$\begin{aligned} \frac{\partial |E_c(\omega, \boldsymbol{h})|^2}{\partial \boldsymbol{h}^*} &= \boldsymbol{e}(\omega)\boldsymbol{e}^H(\omega)\boldsymbol{h} - D\left(e^{j\omega}\right)\boldsymbol{e}(\omega) \\ &= E_c(\omega, \boldsymbol{h})\boldsymbol{e}(\omega). \end{aligned} \tag{3.34}$$

From (3.31) and (3.34), the first Kuhn-Tucker condition $\dfrac{\partial L}{\partial \boldsymbol{h}^*} = \mathbf{0}$ can be written as

$$\sum_{i \in I} [W(\omega_i) + \lambda_i] \, E_c(\omega_i, \boldsymbol{h})\boldsymbol{e}(\omega_i) = \mathbf{0}. \tag{3.35}$$

This condition can be expressed in matrix notation as

$$\boldsymbol{CWC}^H\boldsymbol{h} = \boldsymbol{CWd}, \tag{3.36}$$

where $\boldsymbol{C}$ is an $N \times L$ matrix with columns $\boldsymbol{e}(\omega_i),\ i \in I$, $\boldsymbol{W}$ is an $L \times L$ diagonal matrix with the elements $W(\omega_i) + \lambda_i,\ i \in I$, in its diagonal, and $\boldsymbol{d}$ is an $L \times 1$ vector with elements $D(e^{j\omega_i}),\ i \in I$. Note that the size of the $N \times N$ linear system (3.36) is independent of the number of frequency points. The system of linear equations (3.36) is equivalent to the normal equations (2.9) characterizing the solution to the discrete least squares problem

$$\underset{\boldsymbol{h}}{\text{minimize}} \ \sum_{i \in I} w_i |E_c(\omega_i, \boldsymbol{h})|^2, \tag{3.37}$$

with weights $w_i = W(\omega_i) + \lambda_i$. Hence, the solution to the constrained least squares problem (3.27) can be found by solving a weighted least squares problem, provided that the optimum multipliers λ_i are known. The IRLS algorithm must be designed to generate multipliers λ_i satisfying the remaining Kuhn-Tucker conditions in (3.32). The second condition in (3.32) merely states that the inequality constraints must be satisfied. The non-negativity constraint on the multipliers is easily satisfied. It is clear that the original weighting function $W(\omega)$ must be increased by adding multipliers $\lambda_i > 0$ at frequencies ω_i where the error constraints are violated. The fourth condition in (3.32) requires that multipliers corresponding to inactive constraints must vanish. Optimum multipliers λ_i may be positive only at frequencies satisfying $|E_c(\omega_i, \boldsymbol{h})| = \delta_c(\omega_i)$ at the optimum solution.

Let us now consider the Lagrangian function of the constrained Chebyshev problem (3.28). First note that problem (3.28) can equivalently be formulated as

$$\begin{array}{lrcll} \underset{\boldsymbol{h},\delta}{\text{minimize}} & \delta & & & \\ \text{subject to} & W^2(\omega_i)|E_c(\omega_i,\boldsymbol{h})|^2 & \leq & \delta, & \omega_i \in \Omega_{min}, \\ & |E_c(\omega_i,\boldsymbol{h})|^2 & \leq & \delta_c^2(\omega_i), & \omega_i \in \Omega_B. \end{array} \tag{3.38}$$

The Lagrangian function corresponding to problem (3.38) reads

$$L(\boldsymbol{h},\delta,\boldsymbol{\lambda},\boldsymbol{\mu}) = \delta + \sum_{i\in I_{min}} \mu_i \left[W^2(\omega_i)|E_c(\omega_i,\boldsymbol{h})|^2 - \delta\right] + \sum_{i\in I_B} \lambda_i \left[|E_c(\omega_i,\boldsymbol{h})|^2 - \delta_c^2(\omega_i)\right].$$

The multiplier vector $\boldsymbol{\lambda}$ corresponds to the original constraints on the error function, whereas the multiplier vector $\boldsymbol{\mu}$ corresponds to the artificial constraints introduced by reformulating the problem as in (3.38). If we define $W(\omega_i) = 0,\ \forall i \notin I_{min}$, $\lambda_i = 0,\ \forall i \notin I_B$, and $\mu_i = 0,\ \forall i \notin I_{min}$, the Lagrangian function can be written as

$$L(\boldsymbol{h},\delta,\boldsymbol{\lambda},\boldsymbol{\mu}) = \delta + \sum_{i\in I} \left[\mu_i W^2(\omega_i) + \lambda_i\right] |E_c(\omega_i,\boldsymbol{h})|^2 - \sum_{i\in I} \left[\mu_i\delta + \lambda_i\delta_c^2(\omega_i)\right]. \tag{3.39}$$

The Kuhn-Tucker conditions for problem (3.38) read

$$\begin{array}{lll} \dfrac{\partial L}{\partial \boldsymbol{h}^*} = \boldsymbol{0}, & \dfrac{\partial L}{\partial \delta} = 0, & \\ \dfrac{\partial L}{\partial \boldsymbol{\lambda}} \leq \boldsymbol{0}, & \boldsymbol{\lambda} \geq \boldsymbol{0}, & \boldsymbol{\lambda}^T \dfrac{\partial L}{\partial \boldsymbol{\lambda}} = 0, \\ \dfrac{\partial L}{\partial \boldsymbol{\mu}} \leq \boldsymbol{0}, & \boldsymbol{\mu} \geq \boldsymbol{0}, & \boldsymbol{\mu}^T \dfrac{\partial L}{\partial \boldsymbol{\mu}} = 0. \end{array} \tag{3.40}$$

From (3.34) and (3.39), the two conditions in the first row of (3.40) can be written as

$$\sum_{i\in I} \left[\mu_i W^2(\omega_i) + \lambda_i\right] E_c(\omega_i,\boldsymbol{h})\boldsymbol{e}(\omega_i) = \boldsymbol{0}, \tag{3.41}$$

and

$$\sum_{i \in I} \mu_i = 1. \tag{3.42}$$

Just like in the constrained least squares case, condition (3.41) is equivalent to the normal equations (3.36) with the diagonal weight matrix $\boldsymbol{W}$ containing the weights $\mu_i W^2(\omega_i) + \lambda_i,\ i \in I$. Note that the value of the normalization constant on the right-hand side of condition (3.42) is not relevant. For the case of unconstrained Chebyshev approximation, the multipliers λ_i vanish, and condition (3.41) is equivalent to the equations (2.49) given in Section 2.2.2. The equations (2.49) only take non-zero multipliers into account. In the unconstrained case, the normalization (3.42) is irrelevant. It merely suggests that the optimum weights are determined only up to a scale factor. In the constrained case, the normalization (3.42) – with an arbitrary but fixed constant on its right-hand side – is important because during the course of the IRLS algorithm an equilibrium between the multipliers λ_i and μ_i must be attained.

We have shown that the solutions to the constrained filter design problems (3.27) and (3.28) can be found by solving a linear weighted least squares problem (3.37) with weights $w_i = W(\omega_i) + \lambda_i$ in the constrained least squares case, and with weights $w_i = \mu_i W^2(\omega_i) + \lambda_i$ in the constrained Chebyshev case. The multipliers λ_i and μ_i must satisfy the Kuhn-Tucker conditions. We now formulate an IRLS algorithm based on Lawson's algorithm that finds appropriate multipliers by solving a sequence of linear weighted least squares problems. In the constrained least squares case, the weights of the least squares subproblems are given by

$$w_i^{(k)} = W(\omega_i) + \lambda_i^{(k)},$$

where k is the iteration index. The weights $w_i^{(k)}$ must be increased at frequency points ω_i where the constraints are violated and must be decreased at frequency points ω_i where the constraints are satisfied. However, their lower bound is given by the specified weighting function $W(\omega_i)$, where we assume $W(\omega_i) = 0$ for $\omega_i \notin \Omega_{min}$. A weight update that satisfies these requirements is

$$w_i^{(k+1)} = \max\left(w_i^{(k)} U_i^{(k)}, W(\omega_i)\right), \quad i \in I_B, \tag{3.43}$$

with

$$U_i^{(k)} = \left(\frac{|E_c(\omega_i, \boldsymbol{h}^{(k)})|}{\delta_c(\omega_i)}\right)^{\alpha}, \quad i \in I_B. \tag{3.44}$$

Useful values for the parameter α in (3.44) are $\alpha \in [1, 2]$, where $\alpha = 1$ corresponds to a direct generalization of Lawson's method. An update exponent $\alpha > 1$ has been proposed as a modification to Lawson's algorithm in [72, 102]. It may speed up convergence but may also cause the solutions of the subproblems to oscillate about the optimum solution if α is chosen too large. The multiplier update corresponding to (3.43) is

$$\lambda_i^{(k+1)} = \max\left(\lambda_i^{(k)} U_i^{(k)} + W(\omega_i)(U_i^{(k)} - 1), 0\right), \quad i \in I_B, \tag{3.45}$$

with $U_i^{(k)}$ given by (3.44). Note that $\lambda_i^{(k+1)} > 0$ is possible even if $\lambda_i^{(k)} = 0$. The initialization of the multipliers $\lambda_i^{(k)}$ should be done in such a way that the weights $w_i^{(0)} = W(\omega_i) + \lambda_i^{(0)}$ of the initial least squares problem do not vanish for any index $i \in I$. This is achieved by the choice

$$\lambda_i^{(0)} = \begin{cases} 1, & i \notin I_{min} \\ 0, & i \in I_{min} \end{cases}. \tag{3.46}$$

Let us now consider the constrained Chebyshev problem with weights

$$w_i^{(k)} = \mu_i^{(k)} W^2(\omega_i) + \lambda_i^{(k)}.$$

The multipliers $\mu_i^{(k)}$ are associated with frequency points $\omega_i \in \Omega_{min}$ where no error constraints are imposed. Hence, they are updated in the same manner as the weights in Lawson's algorithm. The multipliers $\lambda_i^{(k)}$ are associated with frequency points $\omega_i \in \Omega_B$ at which error constraints are imposed. Their update is performed similarly to the update (3.45), except for the fact that the least squares weighting function $W(\omega)$ in (3.45) is zero. Consequently, the update formulas for the multipliers λ_i and μ_i are given by

$$\begin{aligned} \lambda_i^{(k+1)} &= \lambda_i^{(k)} U_i^{(k)}, \quad i \in I_B, \\ \mu_i^{(k+1)} &= \mu_i^{(k)} U_i^{(k)}, \quad i \in I_{min}, \end{aligned} \tag{3.47}$$

with $U_i^{(k)}$ given by

$$U_i^{(k)} = \begin{cases} \left(\dfrac{|E_c(\omega_i, \boldsymbol{h}^{(k)})|}{\delta_c(\omega_i)}\right)^\alpha, & i \in I_B, \\[2ex] \dfrac{\left(W(\omega_i)|E_c(\omega_i, \boldsymbol{h}^{(k)})|\right)^\alpha}{\sum\limits_{i \in I_{min}} \mu_i^{(k)} \left(W(\omega_i)|E_c(\omega_i, \boldsymbol{h}^{(k)})|\right)^\alpha}, & i \in I_{min}, \end{cases} \tag{3.48}$$

where we assume the sets I_B and I_{min} to be disjoint. The parameter α should satisfy $1 \leq \alpha \leq 2$. Note that the update of μ_i according to (3.47) and (3.48) ensures that the normalization condition (3.42) is satisfied. The initialization of the multipliers λ_i and μ_i is done by setting

$$\lambda_i^{(0)} = \begin{cases} 1, & i \in I_B \\ 0, & i \in I_{min} \end{cases}, \qquad \mu_i^{(0)} = \begin{cases} 0, & i \in I_B \\ 1/|I_{min}|, & i \in I_{min} \end{cases}, \tag{3.49}$$

where $|I_{min}|$ denotes the number of elements in the set I_{min}.

The complete IRLS algorithm for solving constrained least squares problems (3.27) (constrained Chebyshev problems (3.28)) works as follows:

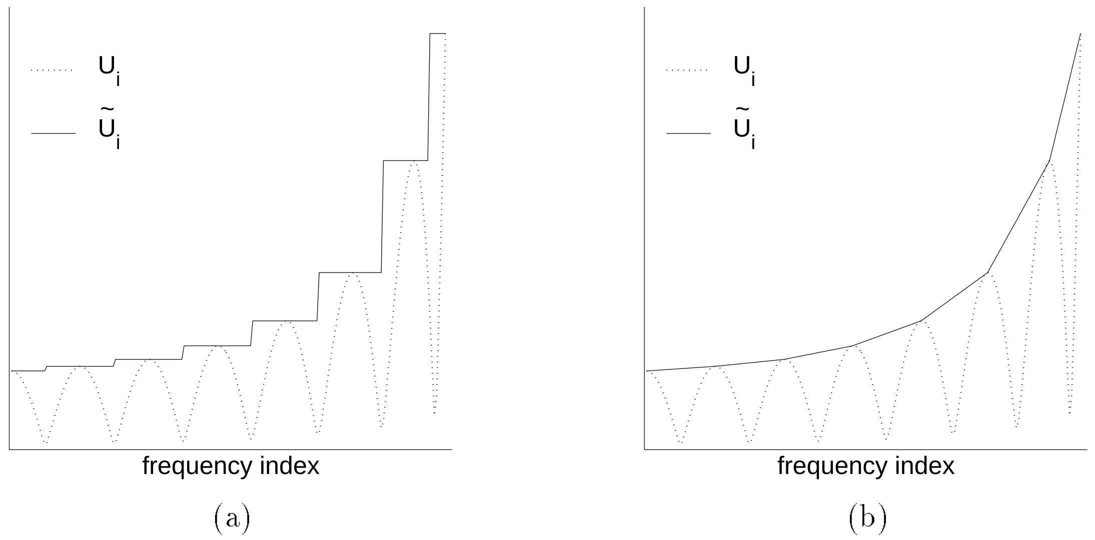


Figure 3.7: Update factors U_i and their envelopes $\tilde{U}_i$. (a) piecewise constant envelope. (b) piecewise linear envelope.

0. $k = 0$. Initialize the multipliers according to (3.46) ((3.49)).
1. Solve the weighted least squares problem (3.37) with weights $w_i = W(\omega_i) + \lambda_i^{(k)}$ ($w_i = \mu_i^{(k)} W^2(\omega_i) + \lambda_i^{(k)}$).
2. Update the multipliers according to (3.45) and (3.44) ((3.47) and (3.48)). $k := k+1$. Go to 1.

The iterations are stopped as soon as the mulipliers λ_i and μ_i satisfy the Kuhn-Tucker conditions (3.40) up to some specified tolerance. Non-negativity of λ_i and μ_i is guaranteed by the initialization and the way the updates are performed. What has to be checked is whether the inequality constraints are satisfied up to some specified tolerance, and whether the multipliers corresponding to inactive constraints are sufficiently small.

The algorithm presented so far is a generalization of Lawson's algorithm to constrained approximation problems. As a consequence, it suffers from the same drawbacks such as relatively slow convergence and propagation of zero weights due to the multiplicative multiplier update (3.47) in the case of constrained Chebyshev approximation. It should be noted, however, that zero weights rarely occur in practice. Both drawbacks can be mitigated by using update factors $\tilde{U}_i$ that are sampled envelopes of the update factors U_i defined by (3.44) and (3.48). This has been proposed in [72] for improving the convergence of Lawson's algorithm in the context FIR filter design by complex Chebyshev approximation. Envelope functions for updating the weights in an IRLS method have also been used in [21, 60]. We propose to use piecewise constant or piecewise linear envelopes. The construction of these envelopes is illustrated by Figure 3.7. Both types of envelopes yield similar results.

It should be noted that by using envelopes of the update factors, the resulting solutions are not optimal anymore. This is the case because the multipliers corresponding to inactive constraints will not vanish anymore, and hence the Kuhn-Tucker conditions are not satisfied. However, experiments show that the solutions obtained by this heuristic modification are very close to the optimum solutions.

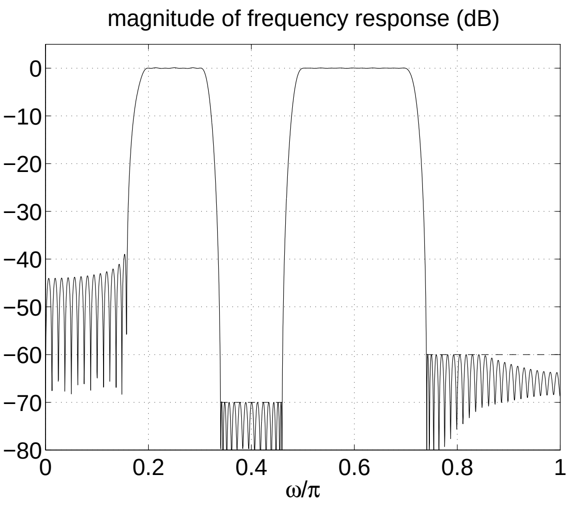
The computational complexity per iteration step of the proposed IRLS algorithm is $O(N^2)$, where N is the filter length. This is a direct consequence of the fact that Levinson's algorithm can be used to solve the least squares subproblems. Its memory requirement increases as $O(N)$ because the system matrix of the normal equations (3.36) is a Hermitian Toeplitz matrix and hence, only its first row or column needs to be stored. As a comparison, the computational complexity of exchange algorithms solving quadratic programming subproblems is $O(N^3)$ per iteration step and the memory requirement increases as $O(N^2)$. The main advantage of the proposed IRLS algorithm is its extremely low memory requirement because no matrices are stored. The computational effort is also low if the solutions are not required to be extremely accurate. Computing very accurate solutions is difficult because the generalized Lawson algorithm exhibits slow final convergence. If envelope functions are used, only approximations to the optimum solutions can be computed. For practical purposes, however, this accuracy problem is not severe and the differences to the true optimum solution are usually negligible.

3.3.1 Design Example

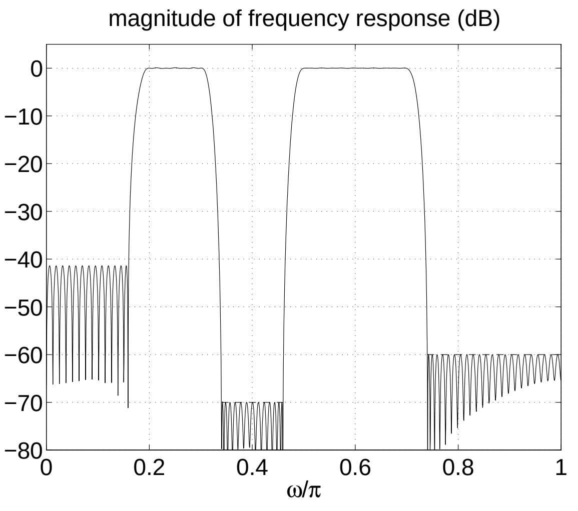
We design a multiband filter using the IRLS algorithm presented in this section. There are two passbands and three stopbands. The desired passband phase response is linear, but the desired constant group delays are different in both passbands. The complex desired frequency response is given by

$$D\left(e^{j\omega}\right) = \begin{cases} 0, & 0 \le \omega \le 0.16\pi, \\ e^{-j50\omega}, & 0.2\pi \le \omega \le 0.3\pi, \\ 0, & 0.34\pi \le \omega \le 0.46\pi, \\ e^{-j60\omega}, & 0.5\pi \le \omega \le 0.7\pi, \\ 0, & 0.74\pi \le \omega \le \pi. \end{cases} \tag{3.50}$$

We choose to impose constraints in all bands but the first stopband (counted from low to high frequencies). The minimum attenuations in stopbands 2 and 3 are required to be 70 dB and 60 dB, respectively. In passband 1, the magnitude of the complex error is required to be less than 0.01, whereas in passband 2 it is constrained to be less than 0.005. In both passbands and in stopband 2 a Chebyshev behavior is desired. This is achieved by choosing the weighting function as $W(\omega) = 0$ in these bands. Consequently, the filter's



(a)



(b)

Figure 3.8: Multiband filter design ($N = 161$). (a) Constrained least squares design. (b) Constrained Chebyshev design. Dashed lines: constraints.

frequency response in the passbands and in stopband 2 is only determined by the error constraints. We design two filters of length $N = 161$. One according to a constrained least squares criterion, and the other according to a constrained Chebyshev criterion. For the constrained least squares design, we minimize the error energy in stopbands 1 and 3 with a weighting of 1 and 5, respectively. For the constrained Chebyshev design, we minimize the maximum error in stopband 1. We use piecewise linear envelopes to compute the update factors.

Both filters are designed by the Matlab program `ccirls` given in Section 3.3.2. It implements the IRLS algorithm as described earlier in this section. The following Matlab commands generate the filter specification:

```
om=pi*[linspace(0,.16,320),linspace(.2,.3,200),linspace(.34,.46,240),...
   linspace(.5,.7,400),linspace(.74,1,520)];
omedge=pi*[0,.16,.2,.3,.34,.46,.5,.7,.74,1];
D=[zeros(1,320),exp(-j*om(321:520)*50),zeros(1,240),...
   exp(-j*om(761:1160)*60),zeros(1,520)];
dc=[-ones(1,320),.01*ones(1,200),10^(-3.5)*ones(1,240),...
   .005*ones(1,400),.001*ones(1,520)];
W=[ones(1,320),zeros(1,200),zeros(1,240),zeros(1,400),5*ones(1,520)];
```

The constrained least squares and constrained Chebyshev filters are designed by calling the program `ccirls`:

```
hl = ccirls(161,om,omedge,D,W,dc,'ls',2,100);
hc = ccirls(161,om,omedge,D,W,dc,'cheb',2,100);
```

Figure 3.8 shows the magnitudes of the frequency responses of both filters. The minimization of error energy can be seen in stopbands 1 and 3 in Figure 3.8 (a). The constrained

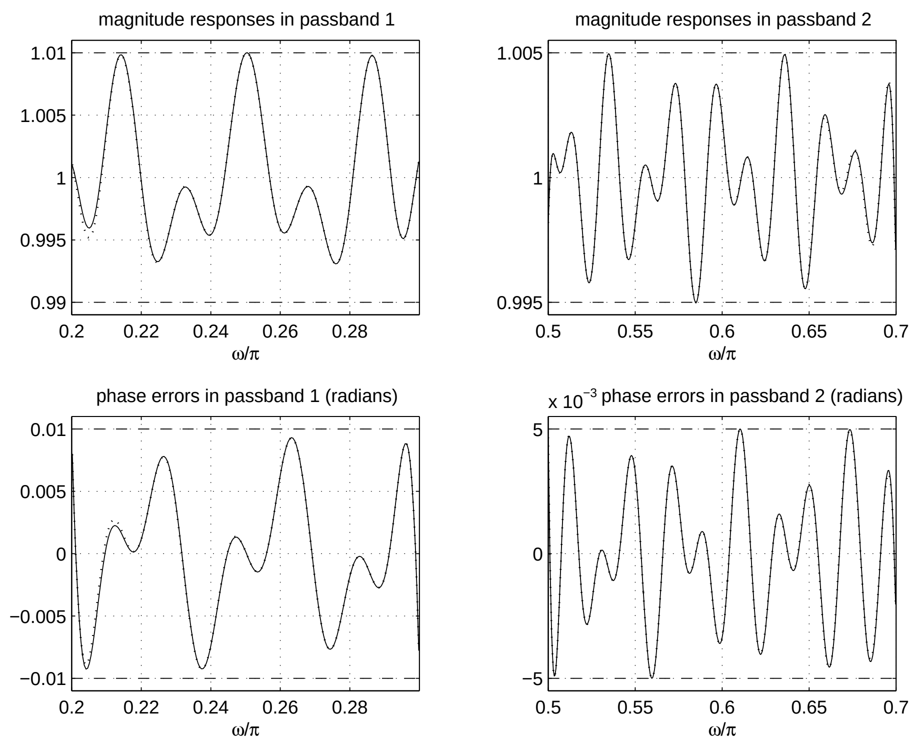


Figure 3.9: Passband magnitude and phase behavior of multiband filter ($N = 161$). Solid curves: constrained least squares design. Dotted curves: constrained Chebyshev design. Dashed lines: constraints. There is virtually no difference between both solutions in the passbands.

Chebyshev filter exhibits an equiripple error behavior in the stopbands (Figure 3.8 (b)). This filter is optimal according to the complex Chebyshev criterion. It could be designed by one of the algorithms discussed in Section 2.2. However, the weighting function that is necessary to satisfy the error constraints is not known in advance. Hence, several unconstrained complex Chebyshev approximations with different weighting functions have to be computed in order to find an appropriate weighting function by trial and error. This costly approach is avoided by the proposed IRLS algorithm.

Figure 3.9 shows the passband magnitude responses and phase errors for the constrained least squares and for the constrained Chebyshev designs. It can be seen that both filters show the same behavior in the passbands. This is the case because in the constrained least squares design, error energy is only minimized in stopbands 1 and 3. Consequently, the solution exhibits a Chebyshev behavior in the passbands and in stopband 2. Note that neither the magnitude error nor the phase error show an equiripple behavior in the passbands, despite the Chebyshev approximation in these bands. This is due to the complex nature of the approximation problem.

3.3.2 Matlab Programs

The Matlab program `ccirls` implements the IRLS algorithm for constrained filter design presented in the preceding section. It can solve constrained least squares problems as well as constrained Chebyshev problems. If desired, the multiplier update can be performed using envelopes. For this purpose, the function `envelope` is called by `ccirls`. The program `envelope` can generate piecewise constant and piecewise linear envelopes. The program `ccirls` only designs real-valued filters. The input parameter `N` is the desired filter length. The vector `om` is the frequency grid at which the desired function `D`, the weighting function `W`, and the constraint function `dc` are specified. The values contained in `om` must lie between 0 and π. If the constraint function `dc` contains negative values, the respective frequency points remain unconstrained. The string variable `crit` specifies the desired criterion. Choose `'cheb'` for constrained Chebyshev approximation and `'ls'` for constrained least squares approximation. The parameter `mode` chooses the way multiplier updates are performed. For `mode=0`, the update factors are computed according to (3.44) or (3.48) without the use of envelopes. Piecewise constant and piecewise linear envelopes are used if `mode=1` and `mode=2`, respectively. The value of α in (3.44) and (3.48) is chosen automatically. If no envelopes are used, $\alpha = 1$. If envelopes are used, $\alpha = 1.5$ for constrained Chebyshev problems, and $\alpha = 2$ for constrained least squares problems. These values have proved to be good choices for a large number of design problems. The parameter `maxit` determines the maximum number of iterations. The algorithm stops if the number of iterations exceeds `maxit` or if the difference between two successive solutions is smaller than some specified tolerance. Hence, the algorithm also stops if convergence becomes very slow, even if the Kuhn-Tucker conditions are not satisfied. The functions `lslevin` and `locmax` used by `ccirls` and by `envelope` have been given in Sections 2.1.4 and 2.2.4.

```
function [h,w] = ccirls(N,om,omedge,D,W,dc,crit,mode,maxit)
% CCIRLS: constrained FIR filter design in the complex domain
% by an iterative reweighted least squares method
% h = ccirls(N,om,omedge,D,W,dc,crit,mode,maxit)
%
% h      filter impulse response
% N      filter length
% om     frequency grid (0 <= om <= pi)
% omedge band edges; must be contained in om!
% D      complex desired frequency response on the grid om
% W      positive weighting function on the grid om
% dc     positive constraint function on the grid om;
%        for intervals without constraints specify negative dc
% crit   string variable defining the design criterion:
%        'cheb' for constrained Chebyshev design
%        'ls' for constrained least squares design
% mode   '0' for generalized Lawson (no envelopes)
%        '1' for piecewise constant envelopes
%        '2' for piecewise linear envelopes
% maxit  maximum number of iterations
%
% EXAMPLE (constrained least squares):
% Low-delay bandpass filter with passband and stopband constraints
% om=pi*[linspace(0,.34,750),linspace(.4,.6,500),linspace(.66,1,750)];
% omedge=pi*[0,.34,.4,.6,.66,1];
% D=[zeros(1,750),exp(-j*30*om(751:1250)),zeros(1,750)];
% W=[500*ones(1,750),ones(1,500),500*ones(1,750)];
% dc=[.001*ones(1,750),.01*ones(1,500),.001*ones(1,750)];
% h = ccirls(100,om,omedge,D,W,dc,'ls',2,50);
%
% Author: Mathias C. Lang, Vienna University of Technology, Aug. 98

om=om(:); omedge=omedge(:); D=D(:); W=W(:); dc=dc(:); L = length(om);
if strcmp(crit,'cheb')==1, cheb=1; else, cheb=0; end

% initialize, define
tol = 1e-5;
h = zeros(N,1); evec = exp(-j*om); IB = find(dc>0); IC = [];
if ~length(IB) & ~cheb, h = lslevin(N,om,D,W); return; end
allconstr = 1; if length(IB) < length(dc), allconstr = 0; end
alpha = 1+.5*(mode>0)*(1+~cheb);
if cheb,
   if length(IB), W(IB) = 0; end
   IC = find(W);
end

% initial multipliers
lam = ~W;
if cheb,
   mu = ~lam; mu = mu/sum(mu);
   w = mu.*(W.^2) + lam;
else
   w = W + lam;
end

% iterate
for i=1:maxit,

   % compute weighted l2 solution
   h0 = h;
   h = lslevin(N,om,D,w);

   % compute magnitude of complex error
   E = abs(polyval(h(N:-1:1),evec) - D);

   % stop if nothing changes
   dh = norm(h-h0)/norm(h);
   if dh < tol, break; end

   % update weights
   if length(IC),
      update = (W(IC).*E(IC)).^alpha;
      update = update/(mu(IC)'*update);
      plot(om(IC)/pi,update), hold on
      if mode,
         update = envelope(update,om(IC),omedge,mode);
         plot(om(IC)/pi,update,'r:')
      end
      mu(IC) = mu(IC).*update;
   end
   if length(IB),
      update = (E(IB)./dc(IB)).^alpha;
      % check for infeasibility
      if ~cheb & allconstr,
         Imax = locmax(update);
         if all(update(Imax) > 1),
            disp('There is no feasible solution. Relax the constraints.');
            break;
         end
      end
      plot(om(IB)/pi,update), hold on
      if mode,
         update = envelope(update,om(IB),omedge,mode);
         plot(om(IB)/pi,update,'r:')
      end
      if cheb,
         lam(IB) = lam(IB).*update;
      else
         lam(IB) = max(lam(IB).*update + W(IB).*(update-1),0);
      end
   end
   if cheb, w = mu.*(W.^2) + lam;
   else, w = W + lam; end

   xlabel('\omega/\pi');
   title('normalized magnitude of complex error');
   drawnow, hold off

end
```

```
function env = envelope(E,om,omedge,mode)
% env = envelope(E,om,omedge,mode)
% generates envelope of real-valued function E on the grid om
% in each band specified by the band edges in omedge;
% mode=1: envelope is piecewise constant
% mode=2: envelope is piecewise linear
%
% Author: Mathias C. Lang, Vienna University of Technology, Aug. 98

E=E(:); om=om(:); omedge=omedge(:); L = length(E); env = zeros(L,1);
del=[];
for i=1:length(omedge),
   if ~any(find(omedge(i)==om)), del=[del;i]; end
end
if length(del), omedge(del)=[]; end
Nbands = length(omedge)/2;
for k=1:Nbands,
   I = find(om>=omedge(2*k-1) & om<=omedge(2*k));
   Imax = locmax(E(I)); Imax = I(Imax);
   Emax = E(Imax); ommax = om(Imax);
   Id = locmax(-E(I)); Id = I(Id);
   if mode == 1,
      if Id(1)~=I(1), Id=[I(1);Id]; end
      if Id(end)~=I(end), Id=[Id;I(end)]; end
      for l=1:length(Id)-1,
         env(Id(l):Id(l+1)) = Emax(l);
      end
   elseif mode == 2,
      if Imax(1)~=I(1),
         Imax=[I(1);Imax]; Emax=[Emax(1);Emax];
         ommax=[om(I(1));ommax];
      end
      if Imax(end)~=I(end),
         Imax=[Imax;I(end)]; Emax=[Emax;Emax(end)];
         ommax=[ommax;om(I(end))];
      end
      dE_dom = diff(Emax)./diff(ommax);
      for l=1:length(Emax)-1,
         Il = Imax(l):Imax(l+1);
         env(Il) = dE_dom(l)*(om(Il)-ommax(l)) + Emax(l);
      end
   end
end
```

3.4 Summary

In this chapter, we have introduced new methods for solving constrained complex approximation problems arising in the design of digital FIR filters. We have focused on the constrained least squares problems (2.76) and (2.77) with constraints on the complex error function and on magnitude and phase errors, respectively. In Sections 3.1 and 3.3 we have also considered the constrained Chebyshev problems (2.78) and (2.79).

In Section 3.1, we have approximated the design problems by linear and quadratic programming problems. The advantage of this approach is that reliable and readily available software can be used for solving the design problems. However, errors are introduced by replacing the original problems by linear or quadratic programming problems. Some of these errors can be decreased by increasing the size of the programming problems. For the design of high-order filters ($N \geq 100$), however, the resulting programming problems become very large. This results in excessive computation times and very high memory requirements. Hence, the applicability of this approach is restricted to the design of filters with relatively low orders. We have provided Matlab programs implementing the proposed linear and quadratic programming methods in Section 3.1.4. The program `lqp_ce` solves problems with constraints on the complex error function, and the program `lqp_mp` solves problems with independent constraints on magnitude and phase errors. Both programs use the linear and quadratic programming routines provided by the Matlab Optimization Toolbox.

Section 3.2 has introduced exchange algorithms that solve a sequence of small subproblems in order to compute the optimum solution to the constrained filter design problem. This greatly reduces the computational effort and the memory requirements compared to the linear and quadratic programming approach. The exchange algorithm presented in Section 3.2.1 can solve convex quadratic programming problems with a large – or even infinite – number of constraints. We have applied it to the convex constrained least squares problem (2.76) with constraints on the complex error function. Section 3.2.2 considers the constrained least squares problem with magnitude and phase constraints. In Section 3.2.2.1, this non-convex problem has been approximated by a convex problem which can be solved by the exchange algorithm introduced in Section 3.2.1. The errors resulting from this approximation are small as long as the phase constraint function is small. The constrained least squares problem with magnitude and phase constraints can be solved exactly by the modified exchange algorithm presented in Section 3.2.2.2. It extends the algorithm of Section 3.2.1 to problems with non-convex feasible regions. Efficient Matlab implementations of the proposed exchange algorithms have been given in Section 3.2.4. The program `exch_ce` implements the multiple exchange algorithm presented in Section 3.2.1 for solving least squares problems with constraints on the complex error function. The algorithms for solving least squares problems with magnitude and phase constraints presented in Sections 3.2.2.1 and 3.2.2.2 are implemented by the programs `exch_mp1` and `exch_mp2`, respectively.

In Section 3.3, we have presented a different approach to solving constrained filter design problems. We have shown that the solutions to the constrained least squares and constrained Chebyshev problems (2.76, 2.78) with constraints on the complex error function can be computed by solving unconstrained linear weighted least squares problems. The optimum weights are computed by solving a sequence of weighted least squares problems with different weighting functions. This type of algorithms is called *iterative reweighted least squares (IRLS)* algorithms. The IRLS algorithm presented in Section 3.3 is based on Lawson's algorithm. Lawson's algorithm was originally developed for unconstrained Chebyshev problems. Here, it has been generalized to become applicable to constrained approximation problems. We have applied known extensions of Lawson's algorithm to the new generalized Lawson algorithm improving its convergence rate . The main advantage of this algorithm is its low memory requirement. No matrices are used, and the linear least squares subproblems are solved by Levinson's algorithm. The computational effort is low if the accuracy of the solutions is not demanded to be extremely high. The computation of very accurate solutions requires many iteration steps due the relatively slow final convergence of IRLS methods. The Matlab program `ccirls` implementing the proposed IRLS algorithm has been given in Section 3.3.2.

For solving complex constrained Chebyshev problems, we propose to use the IRLS algorithm presented in Section 3.3. Practical solutions with a reasonable accuracy can always be computed within fractions of the design times needed by other algorithms such as in [14, 90, 91]. Especially for the design of high-order filters ($N > 500$), the memory requirements are extremely low compared to the methods in [14, 90, 91].

Constrained least squares problems with constraints on the complex error function can be solved by the exchange algorithm presented in Section 3.2.1 and by the IRLS algorithm introduced in Section 3.3. The exchange algorithm converges faster and provides very accurate solutions more easily. Hence, if enough memory is available, the exchange algorithm should be preferred over the IRLS method. However, if the filter order is very high or if memory space is limited, the IRLS algorithm is more appropriate than the exchange algorithm.

For solving constrained least squares problems with constraints on magnitude and phase errors, we propose to use the exchange algorithms presented in Section 3.2.2. For many practical problems, replacing the original non-convex problem by the convex problem given in Section 3.2.2.1 will give good results. However, an exact solution to the original problem can only be computed by the modified exchange algorithm given in Section 3.2.2.2.

Table 3.1 shows an overview of the Matlab programs provided in Chapter 3. The linear and quadratic programming subroutines `lp` and `qp` are contained in the Matlab Optimization Toolbox. All other programs shown in Table 3.1 are given in this thesis.

program	**problem**	**method**	**remarks**	**subroutines**
`lqp_ce`	CLS and CC with constraints on the complex error (problems (2.76) and (2.78))	finite linear and quadratic programming (Section 3.1.1)	slow, high memory requirements, computes approximate solution*	`lp`, `qp`
`lqp_mp`	CLS and CC with constraints on magnitude and phase errors (problems (2.77) and (2.79))	finite linear and quadratic programming (Section 3.1.2)	slow, high memory requirements, computes approximate solution*	`lp`, `qp`
`exch_ce`	CLS with constraints on the complex error (problem (2.76))	multiple exchange algorithm based on quadratic programming (Section 3.2.1)	fast, memory requirements increase quadratically with filter length	`invtoep`, `locmax`, `nnqmin`
`exch_mp1`	CLS with constraints on magnitude and phase errors (problem (2.77))	multiple exchange algorithm based on quadratic programming (Section 3.2.2.1)	fast, memory requirements increase quadratically with filter length, computes approximate solution*	`invtoep`, `locmax`, `nnqmin`, `princval`
`exch_mp2`	CLS with constraints on magnitude and phase errors (problem (2.77))	multiple exchange algorithm based on quadratic programming (Section 3.2.2.2)	fast, memory requirements increase quadratically with filter length	`invtoep`, `locmax`, `nnqmin`, `princval`
`ccirls`	CLS and CC with constraints on the complex error (problems (2.76) and (2.78))	generalized Lawson (IRLS) based on Levinson's recursion (Section 3.3)	very low memory requirements, sometimes slow if highly accurate solutions are required	`lslevin`, `levin`, `envelope`, `locmax`

Table 3.1: Overview of Matlab programs provided in Chapter 3. CLS ... constrained least squares. CC ... constrained Chebyshev.
* Error bounds are known.

Chapter 4

FIR Filter Design Examples and Comparisons

The purpose of this chapter is to provide filter design examples according to specifications that have been given in the literature before. Using these examples, we compare previous results with results obtained by the algorithms presented in this thesis. Important issues are the usefulness of the design criterion, the accuracy of the solution, the memory requirements, and the computational effort. Making a fair comparison of different methods is complicated by the fact that some authors do not provide all necessary algorithmic details, or do not make their programs available.

All design times given in the following examples have been measured on a 166 MHz Pentium PC using Matlab 5.1.

Example 1: Low-Delay Lowpass Filter ($N = 250$) **[14]:**

We take the specifications of a low-delay lowpass filter with $N = 250$ coefficients from a paper on complex Chebyshev approximation by Burnside and Parks [14]. We have already used this example in Section 2.2.3 to compare different algorithms for complex Chebyshev approximation. The desired frequency response is given by (2.52). Figure 2.11 shows a plot of the optimum Chebyshev filter. From Table 2.2, the maximum error of the optimum solution is $\delta = 2.02 \cdot 10^{-4}$. Due to a weighting of 1 in the passband and 10 in the stopband, the magnitude of the complex error function is bounded by δ in the passband and by $\delta/10$ in the stopband. If we used a constrained least squares criterion and prescribed these values as bounds for the complex error function, we would obtain exactly the optimum Chebyshev filter. In this case, the filter is determined by the constraints only. There are no degrees of freedom left to minimize the error energy. The feasible region of the design problem is reduced to a single point in N-dimensional coefficient space. If the error bounds are chosen to be smaller than δ in the passband and

$\delta/10$ in the stopband, no solution exists because the feasible region is empty. However, if we allow a slightly increased peak error, we can decrease the error energy. In order to illustrate this effect, we design a filter according to the specifications given in [14] using a constrained least squares criterion. We use problem formulation (2.76) with constraints on the complex error function. We choose the constraint function according to

$$\delta_c(\omega) = \begin{cases} 2.1 \cdot 10^{-4}, & \omega \in \Omega^p, \\ 2.1 \cdot 10^{-5}, & \omega \in \Omega^s, \end{cases}$$

which corresponds to a $1.4 \cdot 10^{-4}$ dB increase of passband ripple and a 0.34 dB decrease of stopband attenuation compared to the optimum Chebyshev filter. In order to achieve a Chebyshev behavior in the passband, we choose the weighting to be 1 in the passband and 1000 in the stopband. We design the filter using the program `exch_ce` given in Section 3.2.4:

```
om=pi*[linspace(0,.46,1840),linspace(.5,1,2000)];
D=[exp(-j*om(1:1840)*100),zeros(1,2000)];
Wc=[ones(1,1840),1000*ones(1,2000)];
dc=[2.1e-4*ones(1,1840),2.1e-5*ones(1,2000)];
hc = exch_ce(250,om,D,Wc,dc);
```

The plots on the left-hand side of Figure 4.1 show the magnitude of the frequency response and the passband magnitude and phase errors of the designed filter. The stopband energy has been reduced by 4.2 dB compared to the optimum Chebyshev filter. This example shows that by slightly relaxing the requirements on peak errors, error energy can be reduced significantly. It took 38 seconds to compute this filter. In [14] it was reported that the design of the Chebyshev filter took 5 minutes on a comparable computer. According to [49], the authors of [14] used a Matlab program with the time consuming procedures written in C. This results in a considerable speed up compared to a standard Matlab program. Hence, the run time of the proposed algorithm favorably compares to the run time given in [14].

We can even do better by imposing independent constraints on magnitude and phase errors. From (2.59), the maximum magnitude error of the optimum Chebyshev filter is bounded by $\delta = 2.02 \cdot 10^{-4}$ in the passband and by $\delta/10 = 2.02 \cdot 10^{-5}$ in the stopband. From (2.60), the maximum phase error of the optimum Chebyshev filter is bounded by $\arcsin(\delta) \doteq \delta = 2.02 \cdot 10^{-4}$. These upper bounds for magnitude and phase errors are tight for the filter under consideration. By imposing these values as constraints for magnitude and phase errors, we can design a filter that satisfies the same tolerance scheme as the original Chebyshev filter. However, since independent constraints on magnitude and phase errors are less restrictive than corresponding constraints on the complex error function, there are still degrees of freedom left to minimize the error energy. We design

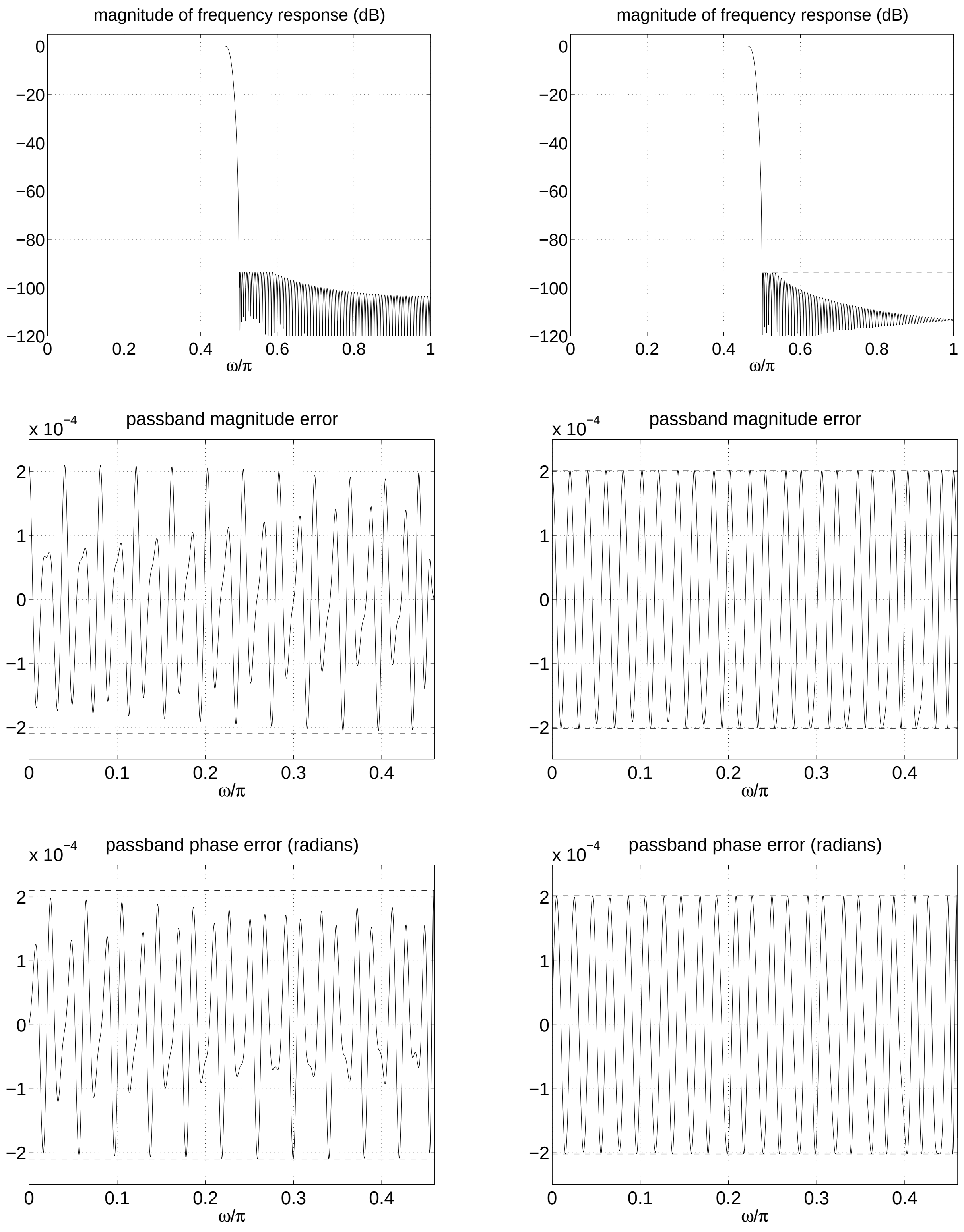


Figure 4.1: Constrained least squares designs of reduced delay lowpass filter (length $N = 250$). Left-hand side: constraints on the complex error function. Right-hand side: independent constraints on magnitude and phase errors. Dashed lines: constraints.

another filter according to a constrained least squares criterion with constraints on magnitude and phase errors. For this purpose, we use the algorithm presented in Section 3.2.2.2. This algorithm is implemented by the program `exch_mp2` given in Section 3.2.4. The following Matlab commands generate the specification and design the filter:

```
om=pi*[linspace(0,.46,1840),linspace(.5,1,2000)];
D=[exp(-j*om(1:1840)*100),zeros(1,2000)];
Wmp=[ones(1,1840),5000*ones(1,2000)];
dm=[2.02e-4*ones(1,1840),2.02e-5*ones(1,2000)];
dp=[2.02e-4*ones(1,1840),2.02e-5*ones(1,2000)];
hmp = exch_mp2(250,om,D,Wmp,dm,dp);
```

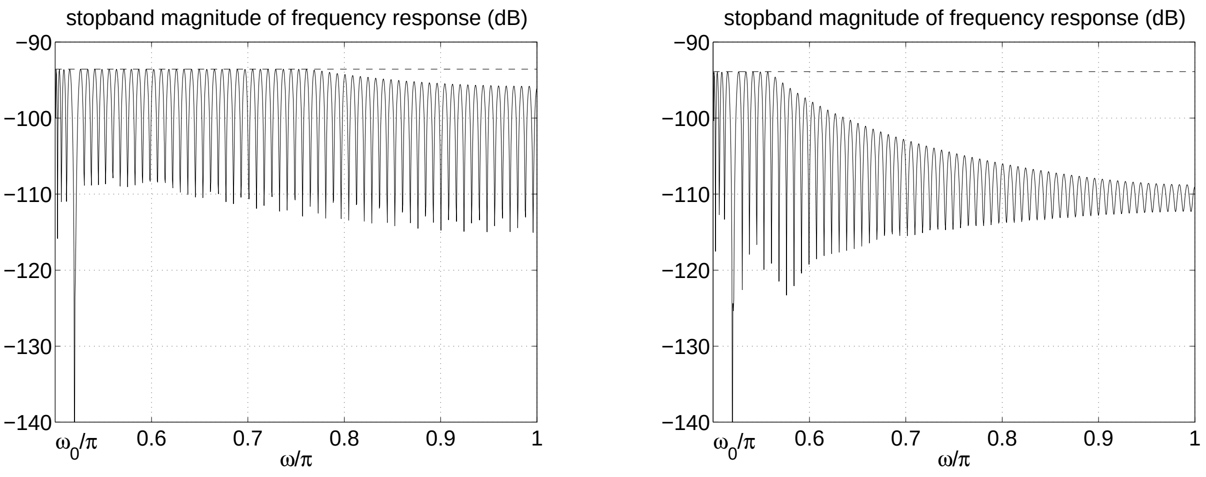
It took 70 seconds to compute this filter. The plots on the right-hand side of Figure 4.1 show the magnitude of the frequency response and the passband magnitude and phase errors of the designed filter. The difference in stopband energy of this filter and the optimum Chebyshev filter is 7.9 dB. Note that as opposed to the previous design, the maximum magnitude and phase errors are the same as those of the Chebyshev filter. Hence, unlike in the previous example, we get a considerably better stopband behavior without relaxing the original tolerance scheme.

We mention that the multiple exchange algorithm presented in Section 3.2.1 can incorporate a wide variety of other design constraints if they are convex. Examples of additional constraints that might be of practical interest are time domain constraints such as limits for the step response ripple [97]. It is easy to modify the programs given in Section 3.2.4 to incorporate other convex constraints. In Section 3.2.2.2 we also incorporated the non-convex lower bounds on the magnitude error. However, we do not know if the algorithm will work for general non-convex constraints.

Imposing point constraints in the frequency domain can be done by specifying zero values in the respective constraint function at the desired frequency points. We extend the two previous design examples by adding an additional point constraint in the stopband. We assume a strong sinusoidal interferer at the frequency $\omega_0 = 0.52\pi$. Hence, we shift a zero of the filter's transfer function to the unit circle at that frequency. This is achieved by choosing $\delta_c(\omega_0) = 0$ or $\delta_m(\omega_0) = 0$, respectively. The two filters with this point constraint are designed by the following Matlab commands (assuming all other necessary variables are still in the workspace):

```
dc0=dc; dc0(1921)=0; dm0=dm; dm0(1921)=0;
hc0 = exch_ce(250,om,D,Wc,dc0);
hmp0 = exch_mp2(250,om,D,Wmp,dm0,dp);
```

The passband magnitude and phase responses are virtually unaffected by the additional point constraint. The designs took 4 minutes and 86 seconds, respectively. Figure 4.2 shows the stopband behavior of both filters. Due to the additional constraint, the error



(a) (b)

Figure 4.2: Constrained least squares designs of reduced delay lowpass filter (length $N = 250$) with point constraints at $\omega_0 = 0.52\pi$. (a) constraints on the complex error function. (b) independent constraints on magnitude and phase errors. Dashed lines: constraints.

energy increases in both cases. For the filter designed with constraints on the complex error function (Figure 4.2 (a)), this almost results in a Chebyshev behavior. The reduction of stopband error energy compared to the Chebyshev filter is only 0.5 dB. However, the Chebyshev filter does not satisfy the point constraint. The filter designed with independent magnitude and phase constraints (Figure 4.2 (b)) still has 6.2 dB less stopband error energy than the optimum Chebyshev filter.

We mention that point constraints cannot be imposed by specifying a constraint function that vanishes at certain frequencies if the IRLS algorithm presented in Section 3.3 is used. This is due to the fact that the constraint function $\delta_c(\omega)$ appears in the denominator of the weight updates (3.44) and (3.48). However, point constraints can be imposed by specifying a very large weighting function $W(\omega)$ at frequency points where the actual frequency response is desired to interpolate the desired response.

Example 2: Digital Equalization and Anti-Aliasing Filter ($N = 51$) [95]:

We consider the cascade of an analog anti-aliasing filter followed by a sampler, a discrete-time filter, and a downsampler. The whole system acts as an optimum anti-aliasing filter. The advantage of this system is that a very simple analog filter is sufficient because the aliasing error can be further reduced by the digital filter. Furthermore, the digital filter can be used to equalize the passband magnitude and phase behavior of the analog filter. This combination has been proposed in [115]. A design of the digital filter according to the complex Chebyshev criterion has been presented in [95]. However, minimizing the energy of the aliasing error is more useful than minimizing its maximum value. Hence, a least squares behavior in the stopband of the cascaded system consisting of the analog prefilter and the digital filter to be designed is desirable [123]. A Chebyshev behavior is useful in the passband of the cascaded anti-aliasing filter [38]. A similar design example has been presented in [89].

We use the design specifications as given in [95]. The analog prefilter is chosen to be a third-order Chebyshev lowpass filter with an equiripple passband and a monotonic stopband response. The passband ripple is 0.5 dB, and the attenuation is 30 dB at $3\omega_p$, where ω_p is the passband edge. The transfer function of this filter is given in [95]. The oversampling factor is chosen to be 3. The complex frequency response of the cascaded system is desired to be

$$\tilde{D}(e^{j\omega}) = \begin{cases} e^{-j\omega\tau}, & \omega \in [0, \pi/16], \\ 0, & \omega \in [3\pi/16, \pi], \end{cases} \tag{4.1}$$

where the desired group delay is specified to be $\tau = 35$ samples. For comparison, we design two different digital filters. The first is optimum according to the complex Chebyshev criterion as the one given in [95]. The second filter is designed according to a constrained least squares criterion with constraints on the complex error function. We design this filter in such a way that the cascaded system has a Chebyshev passband and a least squares stopband. We consider this mixed design criterion to be more appropriate for this application than the complex Chebyshev criterion used in [95].

The unconstrained complex Chebyshev problem reads

$$\underset{\boldsymbol{h}}{\text{minimize}} \ \max_{\omega\in\Omega} \tilde{W}(\omega) \left| H_a(\omega/T) H\left(e^{j\omega}, \boldsymbol{h}\right) - \tilde{D}(e^{j\omega}) \right|, \tag{4.2}$$

where $\tilde{W}(\omega)$ is the weighting function referring to the approximation of the cascade, $H_a(\omega/T)$ is the frequency response of the analog filter, and T is the sampling period. Problem (4.2) is equivalent to the problem

$$\underset{\boldsymbol{h}}{\text{minimize}} \ \max_{\omega\in\Omega} \tilde{W}(\omega) |H_a(\omega/T)| \left| H\left(e^{j\omega}, \boldsymbol{h}\right) - \frac{\tilde{D}(e^{j\omega})}{H_a(\omega/T)} \right|, \tag{4.3}$$

where we exclude frequencies satisfying $H_a(\omega/T) = 0$ from the domain of approximation Ω. Hence, the desired frequency response for the digital filter to be designed is

$$D\left(e^{j\omega}\right) = \frac{\tilde{D}(e^{j\omega})}{H_a(\omega/T)}, \quad H_a(\omega/T) \neq 0. \tag{4.4}$$

The weighting function referring to the approximation of $D\left(e^{j\omega}\right)$ by $H\left(e^{j\omega}\right)$ is $W(\omega) = \tilde{W}(\omega)|H_a(\omega/T)|$.

The constrained least squares problem with constraints on the complex error function is given by

$$\begin{aligned}
&\underset{\boldsymbol{h}}{\text{minimize}} && \int_{\Omega_{min}} \tilde{W}(\omega)\left|H_a(\omega/T)H\left(e^{j\omega},\boldsymbol{h}\right)-\tilde{D}(e^{j\omega})\right|^2 d\omega \\
&\text{subject to} && \left|H_a(\omega/T)H\left(e^{j\omega},\boldsymbol{h}\right)-\tilde{D}(e^{j\omega})\right| \leq \tilde{\delta}_c(\omega), \quad \omega\in\Omega_B,
\end{aligned} \tag{4.5}$$

where $\tilde{\delta}_c(\omega)$ is the constraint function referring to the cascaded system. This problem is equivalent to

$$\begin{aligned}
&\underset{\boldsymbol{h}}{\text{minimize}} && \int_{\Omega_{min}} \tilde{W}(\omega)|H_a(\omega/T)|^2\left|H\left(e^{j\omega},\boldsymbol{h}\right)-\frac{\tilde{D}(e^{j\omega})}{H_a(\omega/T)}\right|^2 d\omega \\
&\text{subject to} && \left|H\left(e^{j\omega},\boldsymbol{h}\right)-\frac{\tilde{D}(e^{j\omega})}{H_a(\omega/T)}\right| \leq \frac{\tilde{\delta}_c(\omega)}{|H_a(\omega/T)|}, \quad \omega\in\Omega_B,
\end{aligned} \tag{4.6}$$

where we again exclude frequencies satisfying $H_a(\omega/T) = 0$ from the sets Ω_{min} and Ω_B. Problem (4.6) is the same as problem (2.76) with $D\left(e^{j\omega}\right)$ given by (4.4), $W(\omega) = \tilde{W}(\omega)|H_a(\omega/T)|^2$, and $\delta_c(\omega) = \tilde{\delta}_c(\omega)/|H_a(\omega/T)|$, $H_a(\omega/T) \neq 0$.

Using the specifications given in [95], we first design a filter according to the complex Chebyshev problem formulation (4.3). The weighting function $\tilde{W}(\omega)$ referring to the cascade is chosen to be 1 in the passband and 10 in the stopband. The desired overall response $\tilde{D}(e^{j\omega})$ is given by (4.1). For computing the optimum filter, we use the program `burnpark` given in Section 2.2.4. This program implements the algorithm proposed by Burnside and Parks [14]. The corresponding Matlab commands are

```
tau=35; N=51;
om=pi*[linspace(0,1/16,100),linspace(3/16,1,1300)];
Dt=[exp(-j*tau*om(1:100)),zeros(1,1300)];
Wt=[ones(1,100),10*ones(1,1300)];
z=[]; p=[-.6493,-.3246-j*1.0325,-.3246+j*1.0325]; k=.7606;
[num,den]=zp2tf(z,p,k);
Ha=freqs(num,den,om*16/pi);
D=Dt./Ha; W=Wt.*abs(Ha);
[hcc,delta] = burnpark(N,om,D,W);
```

The Chebyshev error of this filter is $2.68 \cdot 10^{-3}$ which is similar to the value $2.80 \cdot 10^{-3}$ given in [95]. The design took 76 seconds.

The filter solving the constrained least squares problem (4.6) is designed by the algorithm presented in Section 3.2.1 of this thesis. This algorithm is implemented by the program `exch_ce` given in Section 3.2.4. According to the result of the complex Chebyshev design, we specify the constraint function $\tilde{\delta}_c(\omega)$ in (4.6) to be $2.68 \cdot 10^{-3}$ in the passband. No constraint is imposed in the stopband. We choose the weighting function

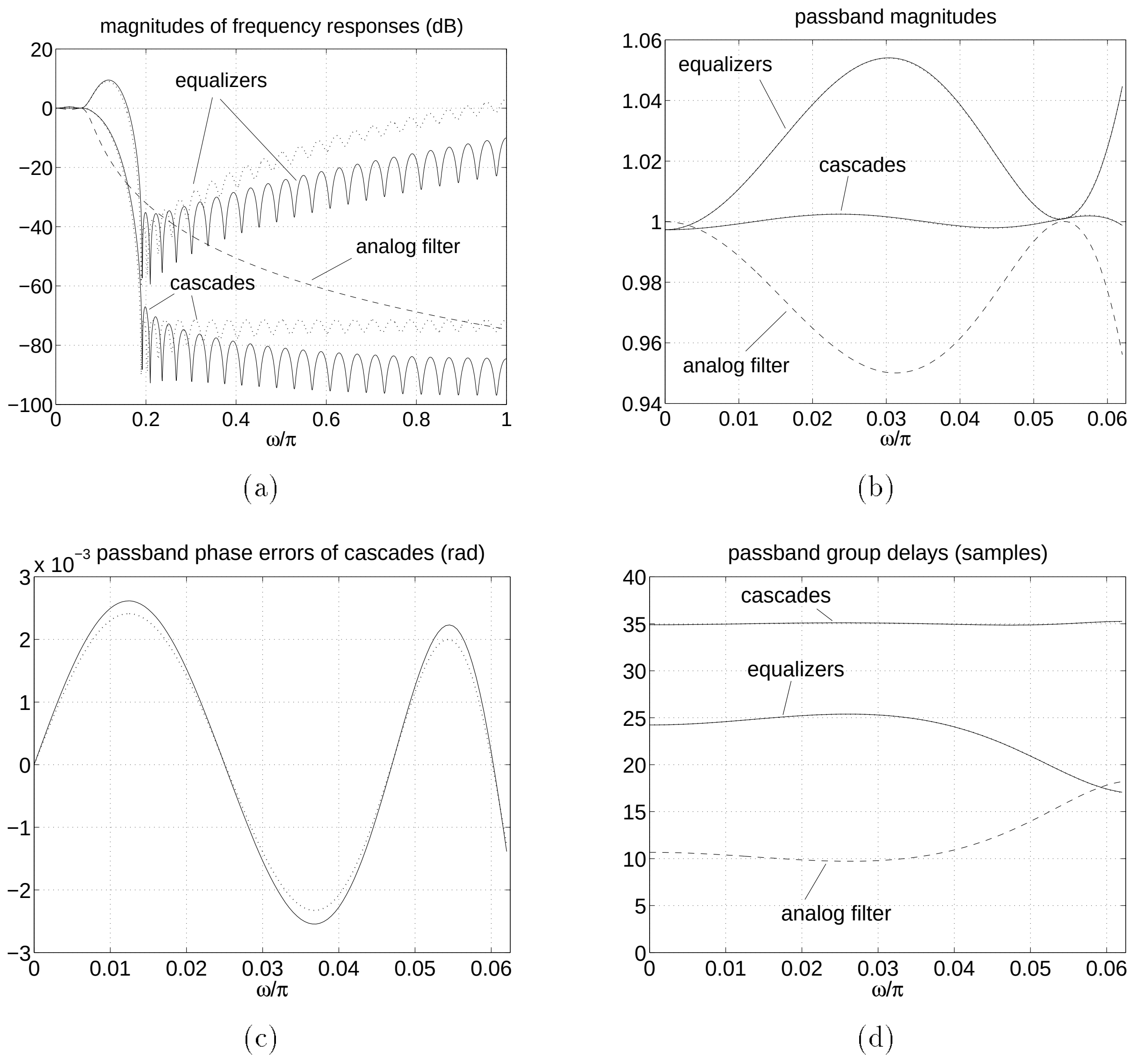


Figure 4.3: Digital equalization and anti-aliasing filters ($N = 51$). Dashed curves: analog prefilter. Solid curves: complex constrained least squares design. Dotted curves: complex Chebyshev design.

$\tilde{W}(\omega)$ to be 1 in the passband and 1000 in the stopband. This results in a Chebyshev behavior in the passband, and in a least squares behavior in the stopband. The filter is designed by adding the following Matlab commands to the previous ones:

```
Wt=[ones(1,100),1000*ones(1,1300)];
dct=[2.68e-3*ones(1,100),-ones(1,1300)];
W=Wt.*abs(Ha).^2;
dc=dct./abs(Ha);
hcls = exch_ce(N,om,D,W,dc);
```

This design took 0.8 seconds.

Figure 4.3 shows the design results for both designs. From Figure 4.3 (a) it can be seen that the cascade of the analog and digital anti-aliasing filters has a least squares stopband behavior for the constrained least squares design (solid curve), whereas it has an equiripple stopband behavior for the complex Chebyshev design (dotted curve). The passband responses of both filters are almost identical and cannot be distinguished in Figure 4.3 (b) and (d). The difference between the stopband energies of the two different cascaded anti-aliasing filters is 6.3 dB.

Example 3: Fractional Delay Lowpass Filter ($N = 95$) [6]:

This example has been used by John Adams and James Sullivan in [6] to illustrate their algorithm presented in [128, 129]. It is a lowpass filter with passband and stopband edges at $\omega_p = 0.125\pi$ and $\omega_s = 0.1608\pi$, respectively. The passband ripple should not exceed 1 dB and the stopband attenuation should be at least 45 dB. What is special about this filter is the desired fractional passband group delay specified to be 47.25 samples. Moreover, this desired group delay value is to be approximated very closely. Their algorithm allows the specification of bounds on the group delay error. The maximum deviation of the desired constant group delay value is specified to be 0.01 samples. Since Adams and Sullivan do not provide their program to the public, we have to rely on the figures given in [6]. The magnitude of the frequency response as well as the group delay response are equiripple in the passband.

In Section 2.3.1 we argued that constraining phase errors has more practical relevance than constraining group delay errors. Hence, we developed algorithms that allow to constrain magnitude and phase errors instead of magnitude and group delay errors. It is interesting to compare the results obtained by constraining phase and group delay errors alternatively. We specify the same tolerance scheme for the magnitude of the frequency response as used in [6]. However, we use constraints on the phase error instead of constraints on the group delay error. Since the phase approximation is desired to be very good we choose a phase constraint function $\delta_\phi(\omega) = 10^{-4}$, $\omega \in \Omega^p$. Because of this small phase constraint function, the linearization errors introduced by the method presented in Section 3.2.2.1 are negligible. We use the program `exch_mp1` given in Section 3.2.4 to design the desired filter. The following design specification reflects the fact that the maximum absolute frequency response value of Adams' filter is 0 dB:

```
om=pi*[linspace(0,.125,150),linspace(.1608,1,850)];
D=[.5*(1+10^(-.05))*exp(-j*om(1:150)*47.25),zeros(1,850)];
W=[ones(1,150),100*ones(1,850)];
dm=[.5*(1-10^(-.05))*ones(1,150),10^(-45/20)*ones(1,850)];
dp=[1e-4*ones(1,150),zeros(1,850)];
h = exch_mp1(95,om,D,W,dm,dp,1);
```

The design took 1.8 seconds. Figure 4.4 shows the design results. The magnitude of

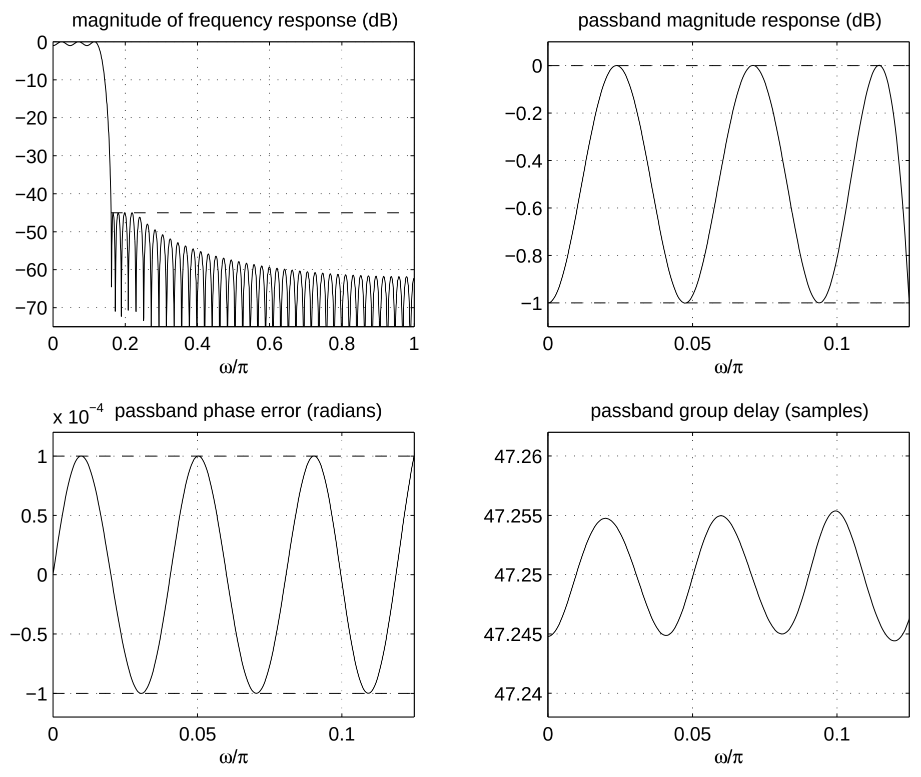


Figure 4.4: Constrained least squares design of fractional delay lowpass filter ($N = 95$) with magnitude and phase constraints. Dashed lines: constraints.

the frequency response looks identical to the one given in [6]. Note, however, that the maximum group delay error of this filter is only about half as large as the one of the filter shown in [6].

Since no design times are given in [6] we cannot compare the efficiency of our method to Adams' method. We can just state that the method presented in this thesis provides a result that favorably compares to the result given in [6]. Moreover, the computational effort to compute this filter with our algorithm is extremely small. We mention that the algorithm by Adams and Sullivan has convergence problems if many stopband constraints become active [129]. The algorithms presented in this thesis do not suffer from this problem.

Example 4: 'Fantasy' Design ($N = 50$) [55]:

This example is taken from a paper by Markus Lang and J. Bamberger on complex constrained least squares FIR filter design. They discuss the design problem with constraints on the complex error function as given by (2.76). They develop a direct generalization of Adams' multiple exchange algorithm for the constrained least squares design of linear phase filters [2]. The system of linear equations that must be solved in every iteration step of Adams' algorithm is replaced by a system of nonlinear equations. However, Adams' original algorithm presented in [2] is known to fail to converge for several design specifications other than lowpass [6, 119]. Hence, its complex version presented in [55] has the same problem. However, if the algorithm converges it is relatively fast. The authors of [55] are so kind to provide a Matlab implementation of their algorithm. We choose the following design example presented in [55]:

$$D\left(e^{j\omega}\right) = \begin{cases} \cos(\omega)e^{-j(20\omega+25\omega^2/3)}, & \omega \in [0, 0.3\pi], \\ 0, & \omega \in [0.4\pi, \pi]. \end{cases} \tag{4.7}$$

The constraint function is specified to be

$$\delta_c(\omega) = \begin{cases} \frac{0.05}{1+9\omega/\omega_p}, & \omega \in [0, 0.3\pi], \\ \frac{0.05}{1+9(\pi-\omega)/(\pi-\omega_s)}, & \omega \in [0.4\pi, \pi], \end{cases}$$

where $\omega_p = 0.3\pi$ and $\omega_s = 0.4\pi$ are the passband and stopband edges, respectively. The weighting function $W(\omega)$ is 1 in the passband and 1000 in the stopband. In [55] 200 equidistant frequency points are used to represent each band. We design two filters using the program provided by the authors of [55] and the program `exch_ce` given in Section 3.2.4. Both programs solve the same problem and consequently, they should yield the same results. The following commands design the filter using the program `exch_ce`:

```
om=pi*[linspace(0,.3,200),linspace(.4,1,200)];
D=[cos(om(1:200)).*exp(-j*(20*om(1:200)+25/3*om(1:200).^2)),zeros(1,200)];
W=[ones(1,200),1000*ones(1,200)];
dc=.05./[1+9*om(1:200)/(.3*pi),1+9*(pi-om(201:end))/(.6*pi)];
h1 = exch_ce(50,om,D,W,dc);
```

The design took 1 second. Using the program provided by the authors of [55], the design took 8 seconds. As could be expected, both designs are equivalent. Figure 4.5 (a) shows the magnitude of the normalized error function $|E_c(\omega)|/\delta_c(\omega)$ for this example.

It is relatively easy to find an example where the algorithm proposed in [55] does not converge. It will not converge for virtually all multiband filter design examples. However, divergence also occurs for several design examples with only two bands. If we change the desired frequency response given by (4.7) by replacing the cosine with a sine function, the problem becomes infeasible. This is detected by the program `exch_ce`. To make the problem feasible, we could increase the filter length, increase the transition band width, or choose a less restrictive constraint function. We increased the constraint function by a factor of 1.4 which was sufficient to ensure feasibility. The following commands design the optimum filter using the program `exch_ce`:

```
D=[sin(om(1:200)).*exp(-j*(20*om(1:200)+25/3*om(1:200).^2)),zeros(1,200)];
h2 = exch_ce(50,om,D,W,1.4*dc);
```

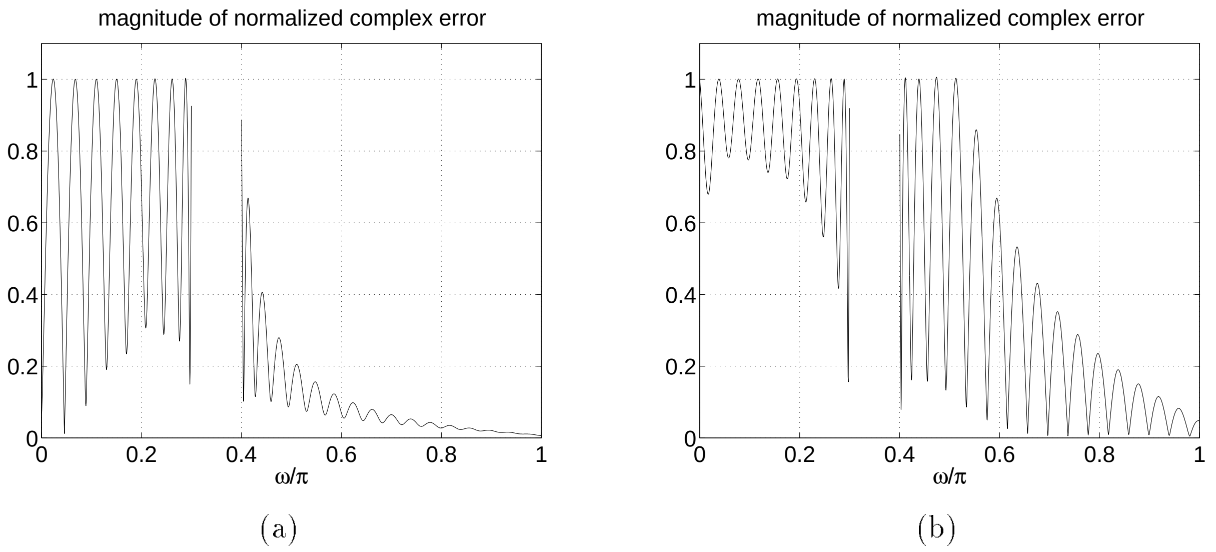


Figure 4.5: Constrained least squares designs with constraints on the complex error function (length $N = 50$). (a) specification according to [55]. (b) modified specification for which the algorithm of [55] does not converge.

This design took 2.5 seconds. The magnitude of the normalized error for this example is shown in Figure 4.5 (b). The algorithm of [55] did not converge for this modified design example.

From our experience, the algorithm proposed in [55] converges only for a small class of design problems. If the desired response is not lowpass, it will typically diverge. But even in the lowpass case numerical problem arise quite often. This is especially the case for longer filters ($N > 100$) and for high grid densities (more than 1000 points). We may conclude that the algorithm developed in [55] cannot compete with the multiple exchange method presented in Section 3.2.1 of this thesis.

Example 5: High-Order Low-Delay Lowpass Filter ($N = 800$) [90, 91]:

We take a design specification that has been used to show the capabilities of the methods proposed in [90, 91]. We have discussed these algorithms in Section 2.2.2. We are not aware of any other authors that published comparable high-order design examples for the complex approximation case. The authors of [90, 91] stress the high accuracy of their results. We consider numerical stability to be an important issue and the ability to design filters of very high degree without numerical problems is an appropriate indication that the respective algorithm is indeed stable. However, it does not seem worthwhile to spend several hours of computing just to gain tenths of dBs. In most practical cases these small improvements will be lost anyway due to coefficient quantization.

We design a filter according to the same specifications as given in [90, 91] using the iterative reweighted least squares (IRLS) algorithm presented in Section 3.3. The filter

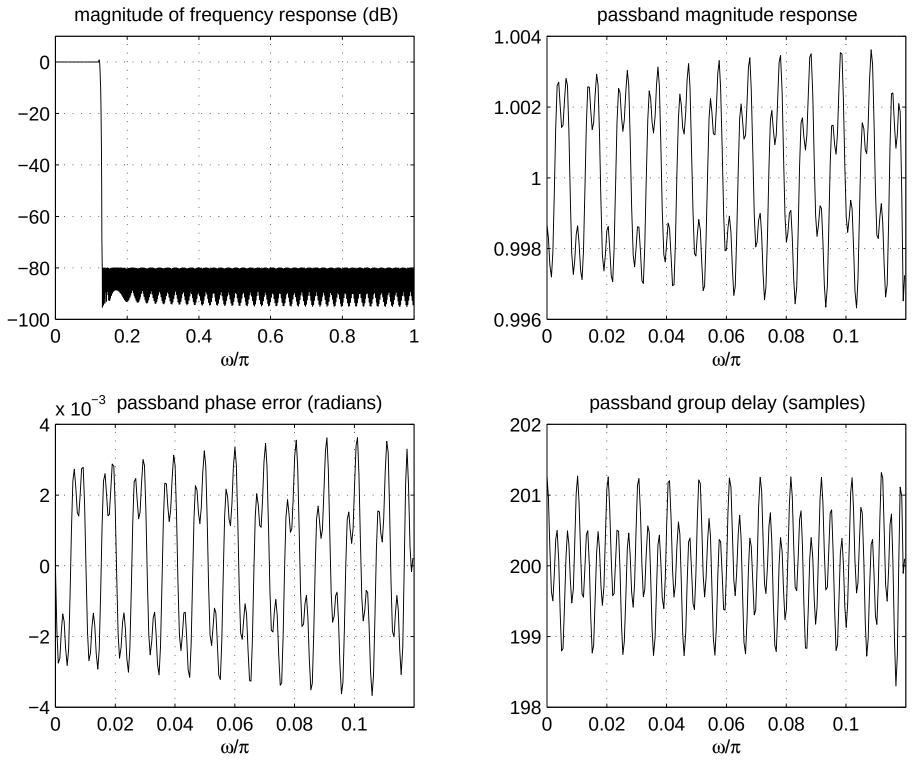


Figure 4.6: Constrained Chebyshev design of high-order low-delay lowpass filter ($N = 800$).

to be designed is a lowpass filter with passband edge $\omega_p = 0.12\pi$ and stopband edge $\omega_s = 0.13\pi$. The passband phase is desired to be linear with a group delay of 200 samples. This corresponds to a delay reduction of almost 50% compared to the delay of an exactly linear phase filter of the same degree. The filter is designed according to a constrained complex Chebyshev criterion. The minimum stopband attenuation is required to be 80 dB. We use a grid of 9900 points to represent the passband and the stopband. The following commands design the filter using the program `ccirls` given in Section 3.3.2:

```
om=pi*[linspace(0,.12,1200),linspace(.13,1,8700)];
omedge=pi*[0,.12,.13,1];
D=[exp(-j*om(1:1200)*200),zeros(1,8700)];
W=[ones(1,1200),zeros(1,8700)];
h = ccirls(800,om,omedge,D,W,dc,'cheb',2,50);
```

Figure 4.6 shows the design results. The design took 9 minutes. The algorithms of [90, 91] were coded in Fortran which should make them considerably faster than an equivalent Matlab implementation. For computing the optimum filters, the authors of [90, 91] used an IBM RS6000 system. Due to a different programming language and a different system, a fair comparison regarding computation times is hardly possible. In [91], the authors give computation times for some design examples. However, the maximum filter length used in these examples is 500. According to [91], the design of a lowpass filter with 500 coefficients took 305 and 140 minutes using the algorithms presented in [90] and [91], respectively. The design time for the length 800 lowpass filter under consideration has not been published in [90, 91].

Aside from computation times, the quality of the approximation is of course an important issue as well. For computing this example, we used the IRLS algorithm presented in Section 3.3 with a piecewise linear function to update the weights. As discussed in Section 3.3, this will result in slightly suboptimal solutions but helps to speed up convergence. Moreover, we only approximated the desired function on a grid of 9900 points instead of the continuous frequency intervals as has been done in [91]. In [90, 91], the maximum approximation error in the passband was reported to be $3.85\cdot 10^{-3}$. The 80 dB stopband constraint is satisfied with very high accuracy. Using the IRLS algorithm with piecewise linear update function on a grid of 9900 points we get a slightly less accurate approximation. However, the computation time and the memory requirement is only a small fraction of those needed by the algorithms proposed in [90, 91]. We have evaluated the complex error function of the filter designed by the IRLS algorithm on an FFT-grid with 2^{16} points. The maximum passband error on this dense grid is $3.95\cdot 10^{-3}$ and the minimum stopband attenuation is 79.93 dB. This corresponds to a $1.7\cdot 10^{-3}$ dB increase in passband ripple and a 0.07 dB decrease of stopband attenuation compared to the results obtained in [90, 91]. We consider these differences to be irrelevant from a practical point of view.

We also want to test the behavior of the multiple exchange algorithm presented in Section 3.2.1 for this high-order filter design example. For this purpose, we constrain the complex error in the passband to be smaller than $4\cdot 10^{-3}$ – a value slightly larger than the error of the optimum Chebyshev solution –, and require a minimum stopband attenuation of 80 dB. In order to obtain a Chebyshev behavior in the passband, we specify the weighting function to be 1 in the passband and 10^4 in the stopband. For computing the corresponding optimum constrained least squares filter, the following Matlab commands must be added to the previous ones:

```
W=[ones(1,1200),1e4*ones(1,8700)];
dc=[4e-3*ones(1,1200),1e-4*ones(1,8700)];
hcls1=exch_ce(800,om,D,W,dc);
```

This design took 7 minutes. Figure 4.7 shows the magnitude, phase, and group delay responses of this filter. Note that the approximation has again been computed on a frequency grid of 9900 points. We could have increased the grid density without considerably increasing the computational effort. Equivalently, the same algorithm could solve the continuous problem by modifying the computation of the error function to allow for

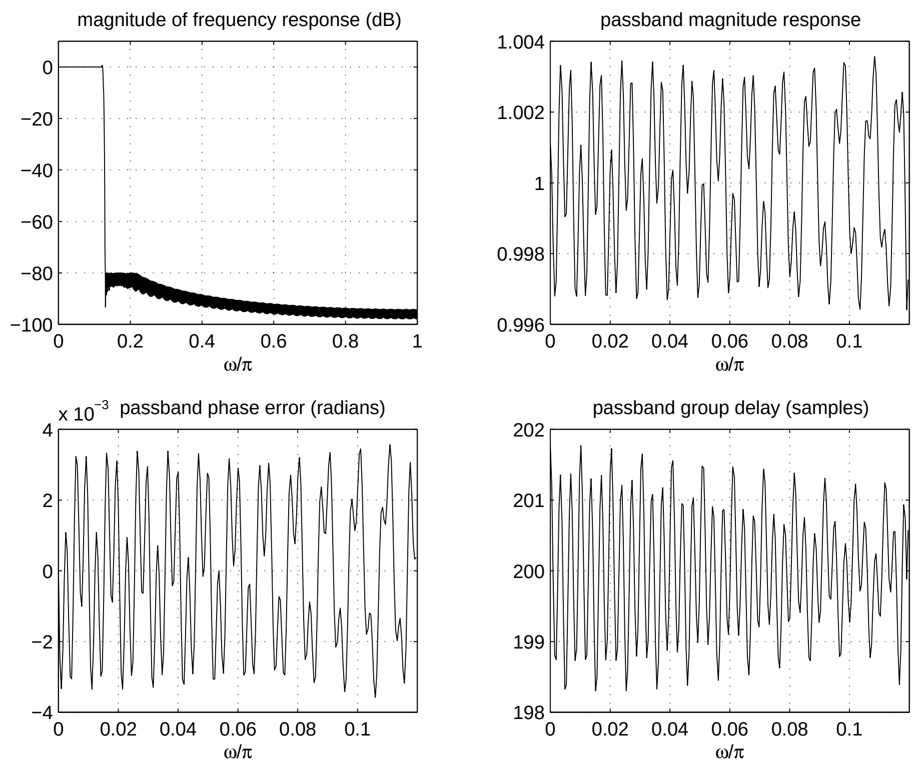


Figure 4.7: Constrained least squares design of high-order low-delay lowpass filter ($N = 800$).

continuous approximation intervals. This issue has been discussed in Section 3.2.1. However, errors introduced by frequency discretization are usually negligible if the number of equidistant grid points is larger than $10N$, where N is the filter length. We have evaluated the error functions of this filter on an FFT-grid with 2^{16} points. On this grid, the maximum violation of the passband constraints on the complex error function is 0.03 dB, and the maximum violation of the stopband constraint is .04 dB. Passband magnitude and phase errors are smaller than the constraint function. We conclude that the grid density has been chosen sufficiently large.

Note that by using a constrained least squared error criterion we were able to decrease the stopband energy by 6 dB compared to the constrained Chebyshev filter. The price for this is a 0.3 dB increase of the passband error.

Both filters designed so far have an undesirable peak of their magnitude response in the transition band. This transition band peak is shown in Figure 4.8. The dotted curve shows the response of the constrained Chebyshev filter designed by the IRLS method, and the dashed curve shows the response of the constrained least squares filter designed by the multiple exchange algorithm. This phenomenon occurs quite often for filters with almost linear phase and a passband group delay that is considerably smaller than $N/2$. Note that linear phase filters may also have peaks in their transition bands. However,

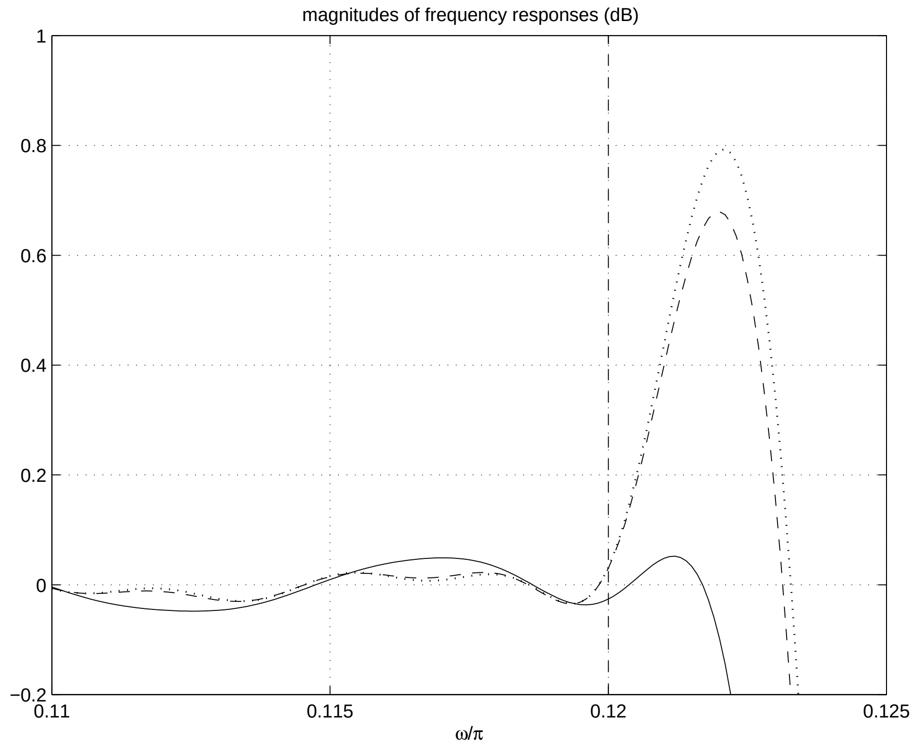


Figure 4.8: Transition band peaks of high-order low-delay lowpass filters ($N = 800$). Dotted curve: Chebyshev design. Dashed curve: constrained least squares design. Solid curve: constrained least squares design with transition band constraint. The dashed vertical line indicates the passband edge.

unlike in the complex approximation case, this will only happen if there are more than two bands.

This transition band peak has not been mentioned in [90, 91]. However, the plot of the magnitude error in Figure 2 of [90] shows that $|H(e^{j\omega})| > |D(e^{j\omega})|$ holds at the passband edge (indicated by the small cross) and that the derivative of the magnitude error is not zero at this point.[1] Hence, there is a local maximum of $|H(e^{j\omega})|$ in the transition band. However, its height is unknown.

We can avoid a large transition band peak if we impose an upper bound on the magnitude of the frequency response in the transition band. It is reasonable to use the same upper bound as in the passband. If a considerable reduction of stopband error energy compared to the Chebyshev filter is desired, we have to allow a larger passband error than in the previous example. This is due to the additional transition band constraint. We increase the maximum passband error from $4 \cdot 10^{-3}$ to $6 \cdot 10^{-3}$. The following Matlab commands design a constrained least squares filter with a transition

[1]Note that the definition of the magnitude error in [90] differs from its definition in this thesis by its sign (cf. page 200 of [90]).

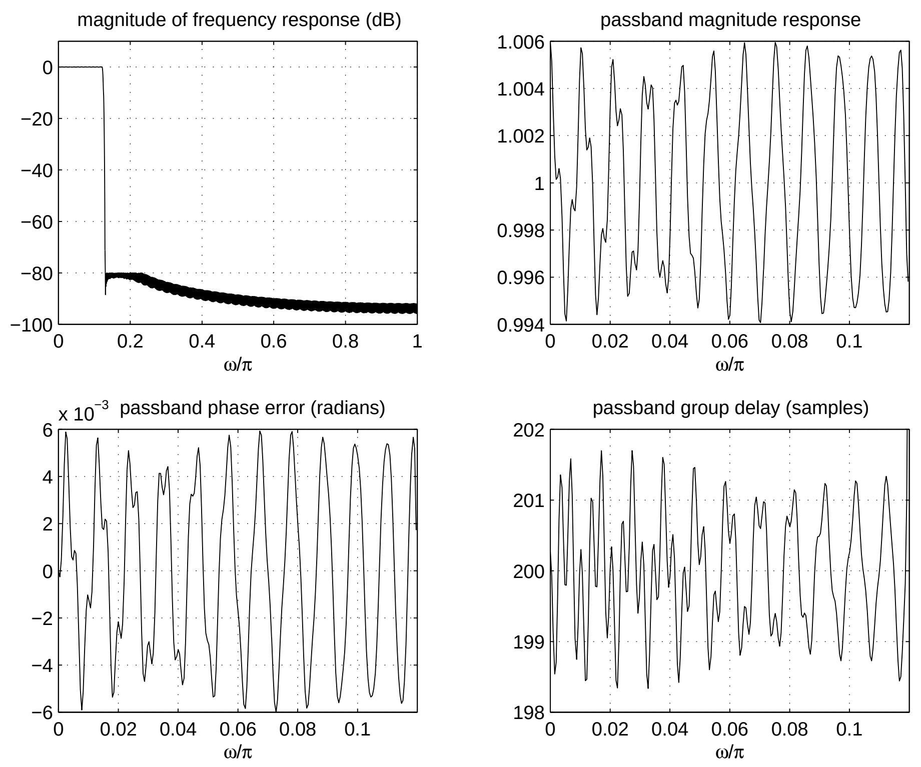


Figure 4.9: Constrained least squares design of high-order low-delay lowpass filter ($N = 800$) with transition band constraint.

band constraint:

```
om=pi*[linspace(0,.12,1200),linspace(.12,.13,100),linspace(.13,1,8700)];
D=[exp(-j*om(1:1200)*200),zeros(1,8800)];
W=[ones(1,1200),zeros(1,100),1e4*ones(1,8700)];
dc=[6e-3*ones(1,1200),1.006*ones(1,100),1e-4*ones(1,8700)];
hcls2=exch_ce(800,om,D,W,dc);
```

This design took 12 minutes. Figure 4.9 shows the magnitude, phase, and group delay responses of this filter. Its transition band behavior is shown by the solid curve in Figure 4.8.

We conclude that the algorithms presented in this thesis are equally well suited for the design of high-order filters as the ones proposed in [90, 91]. However, our algorithms are considerably faster than the ones in [90, 91]. In the case of the IRLS algorithm presented in Section 3.3, this comes at the expense of a slightly reduced accuracy of the design results. The multiple exchange algorithm for solving constrained least squares problems proposed in Section 3.2.1 can compute very accurate solutions, and is also very fast compared to the methods in [90, 91]. However, its memory requirements are comparable to those of the methods in [90, 91], and are much higher than the memory requirements of the IRLS algorithm. Note that the constrained least squares problem with magnitude and phase constraints solved by the method of Section 3.2.2.2 cannot be solved by the algorithms proposed in [90, 91] because it is a non-convex problem.

Chapter 5

Least Squares Design of Stable IIR Filters

5.1 Introduction

In this chapter, we present a method for designing recursive digital filters with infinite impulse responses (IIR) according to specifications on their magnitude and phase responses. A special feature of this method is that an arbitrary maximum pole radius can be prescribed. Hence, the stability problem is solved, and poles very close to the unit circle can be avoided.

There are several reasons for developing a special design algorithm instead of using standard nonlinear programming techniques. First, in order to apply standard methods, it is necessary to explicitly formulate the constraints on the pole radii as functions depending on the design parameters. This is only possible if special parameterizations of the transfer function are used. As will be discussed in Sections 5.2 and 5.3, these transfer function parameterizations correspond to very complicated and highly nonlinear objective functions to be minimized. Consequently, there are many different local minima, and the local solutions obtained by standard methods often correspond to filters with large approximation errors. The proposed method uses a parameterization of the transfer function where the constraints on the pole radii cannot be formulated explicitly in general. However, with this parameterization it is easier to find a good local minimum of the corresponding objective function, even if no appropriate starting guess is available. Another reason for using a specialized algorithm is to exploit the linear dependence of the transfer function on a large number of design parameters. This is achieved by the algorithm which solves the resulting subproblems given in Section 5.4.3.

The frequency domain synthesis of recursive digital filters is often accomplished by applying the bilinear transformation to the corresponding classical analog prototype filters for which closed form solutions exist. Classical filters are the Butterworth, Chebyshev I and II, and Cauer (elliptic) filters. These four approximations use Taylor series approximations and Chebyshev approximations in various combinations. This design approach is useful if the classical approximation criteria are appropriate, if the desired filter characteristic is one of the standard types (lowpass, highpass, bandpass, and bandstop), and if there is no specification on the phase response of the filter. In most other cases we have to use numerical methods to directly design the discrete-time filter. Many numerical methods for the frequency domain design of IIR filters have been proposed, e. g. [19, 23, 26, 30, 35, 45, 69, 72, 74, 76, 79, 106, 117, 122, 136, 138]. The motivations for developing these methods are mainly the following:

1. design filters with magnitude responses different from the standard ones,
2. use design criteria different from the ones used in classical analog filter design,
3. satisfy simultaneous specifications on magnitude and phase or group delay responses,
4. specialize to certain structures for implementation.

The work presented in this chapter is motivated by items 1 to 3. Our goal is to design IIR filters with arbitrary magnitude and phase responses using a weighted least squared error criterion. Moreover, we want to prescribe a maximum radius for the poles of the filter's transfer function.

IIR filters of this kind are interesting if high frequency selectivity as well as approximation of a phase response are required at the same time. FIR filters with equivalent frequency responses are of higher degree and need more memory and usually more computational complexity for their implementation. Furthermore, in many cases IIR filters can have a smaller group delay than equivalent FIR filters.

The choice of the design criterion is determined by the application. In many cases the least squared error criterion is useful in the stopbands of frequency selective filters. Minimizing the mean squared error in the stopbands minimizes the white noise gain in these bands. If an appropriate weighting function is used, the noise power of any colored noise process can be minimized provided its power density spectrum is known. Since the peak error is also a quantity of practical interest, we motivated the constrained least squares criterion in Section 2.3. However, in this chapter we focus on weighted least squared error filter design without frequency response constraints. This is a much more difficult task than in the FIR case due to the nonlinear interdependence of the filter coefficients and the filter's frequency response. Despite the lack of peak error constraints, the weighted least squared error criterion can be adapted to a broad range of practical problems by an appropriate choice of the weighting function.

Apart from the nonlinearity of the weighted least squares design problem in the IIR case, an additional difficulty is caused by the stability problem. A design is only useful if the corresponding filter is stable. A causal linear time-invariant discrete-time filter is stable if the poles of its transfer function are inside the unit circle of the complex plane. If only the magnitude response is of interest, any poles outside the unit circle can be reflected at the unit circle. Note, however, that in this case the poles may lie arbitrarily close to the unit circle. Reflecting poles at the unit circle will not change the magnitude response. It will, however, affect the filter's phase response. Hence, any numerical method for the design of stable IIR filters with specified magnitude and phase responses must constrain the poles to lie inside the unit circle. Because of finite wordlength effects occurring in practical implementations of the designed filter, not only stability of the ideal filter but also a certain stability margin is necessary. The poles of the transfer function to be implemented should not lie too close to the unit circle. It is known that the sensitivity of pole locations to coefficient quantization increases with decreasing distance from the unit circle. Poles close to the unit circle may also considerably enhance quantization noise and may increase the maximum amplitude of small scale limit cycles. Consequently, it is desirable to have control over the maximum pole radius when designing IIR filters. This can be achieved by the design algorithm presented in this chapter. For more details on implementation issues we refer to [47, 104, 112, 114].

5.2 Problem Formulation

The transfer function of a causal IIR filter is given by

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum\limits_{m=0}^{M} b_m z^{-m}}{\sum\limits_{n=0}^{N} a_n z^{-n}}, \qquad a_0 = 1, \tag{5.1}$$

where we assume the filter coefficients a_n and b_m to be real-valued. The generalization to complex coefficients is straightforward. For the design, the nonnegative integers M and N can be chosen arbitrarily and need not be equal. Note that we define the functions $A(z)$ and $B(z)$ as polynomials in z^{-1}. In the following, we will call $A(z)$ and $B(z)$ the denominator and numerator polynomials, respectively. Since M is the degree of the numerator polynomial in z^{-1}, we will call M the degree of the numerator. Likewise, we will call N the degree of the denominator. When we discuss the design algorithm in

Section 5.4, we only consider poles that are zeros of the denominator polynomial $A(z)$. Additional poles may occur at the origin of the complex plane if $M > N$.

We will show that for the design of frequency selective filters with arbitrary magnitude and phase responses and a constraint on the pole radii, it is often advantageous to choose $M > N$, i. e. to use more zeros than poles outside the origin of the complex plane. The reason for this is the fact that the poles ideally contribute only to the passbands of the filter, while zeros contribute to the passbands as well as to the stopbands. Zeros contributing to the passbands are necessary for approximating the desired passband phase response. Moreover, the zeros may lie anywhere in the complex plane whereas the poles of a stable filter are restricted to a circular region $|z| \leq \rho < 1$.

We want to design the frequency response $H\left(e^{j\omega}\right)$ in such a way that it optimally approximates a specified complex frequency response $D\left(e^{j\omega}\right)$ subject to a constraint on the pole radii. Assume the complex desired frequency response $D\left(e^{j\omega}\right)$ prescribing the desired magnitude and phase responses is specified on a grid of $L > M+N+1$ frequencies points ω_i, $i = 0, 1, \ldots, L-1$. Denote the pole radii – i. e. the magnitudes of the zeros of $A(z)$ – by r_n, $n = 1, 2, \ldots, N$. The weighted least squares design problem with constraints on the pole radii reads

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=0}^{L-1} W(\omega_i)\left|E(\omega_i)\right|^2, \\ \text{subject to} & r_n \leq \rho, \quad n = 1, 2, \ldots, N, \quad 0 < \rho < 1, \end{array} \tag{5.2}$$

where $E(\omega) = D\left(e^{j\omega}\right) - H\left(e^{j\omega}\right)$ is the complex error function, $W(\omega)$ is an arbitrary non-negative weighting function, and ρ is the desired maximum pole radius that must not be exceeded. Problem (5.2) is a constrained nonlinear optimization problem. A critical step in formulating the design problem is the choice of the optimization parameters. These parameters need not necessarily be the polynomial coefficients a_n and b_m introduced in (5.1). The polynomials $A(z)$ and $B(z)$ could be split into second order polynomials such that the optimization parameters are the coefficients of the cascade structure that is often used for implementing IIR filters [47, 104, 112, 113]. This parameterization of the transfer function $H(z)$ has been used for designing IIR filters e. g. in [26, 45, 75, 77]. The transfer function $H(z)$ is also represented by its poles and zeros and a scalar constant. Hence, the locations of poles and zeros can be used as optimization parameters for designing IIR filters. This has been done e. g. in [30], where the optimization parameters are the radii and angles of the poles and zeros. Yet another representation of the transfer function is given by its lattice coefficients [47, 113].

The design problem (5.2) is highly nonlinear and the objective function has several local minima that correspond to filters with different approximation qualities. Not all of them represent satisfactory solutions of the design problem. The properties of the objective function to be minimized strongly depend on the parameterization of the transfer function $H(z)$. It turns out that the use of the polynomial coefficients a_n and b_m as design parameters is more advantageous than the use of other parameterizations because in this case, satisfactory solutions can almost always be obtained without the knowledge of a good initial solution. Other parameterizations lead more often to local minima corresponding to unsatisfactory solutions unless the initial guess is already close to a good solution. However, in the general case of magnitude and phase approximation with a pole radii constraint, we cannot assume to know a good initial solution.

5.3 Stability

In this section, we discuss different strategies that have been proposed to guarantee the stability of a designed filter. These strategies can be divided in those using explicit constraints on the design variables, and those applying unconstrained optimization techniques.

We consider three different parameterizations of the denominator polynomial $A(z)$ that have been discussed briefly in the previous section:

1. polynomial coefficients:
$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N},$$
2. polynomial coefficients of second order factors:
$$A(z) = \begin{cases} \displaystyle\prod_{i=1}^{N/2} \left(1 + a_{1i} z^{-1} + a_{2i} z^{-2}\right), & N \text{ even}, \\ \left(1 + a_{10} z^{-1}\right) \displaystyle\prod_{i=1}^{(N-1)/2} \left(1 + a_{1i} z^{-1} + a_{2i} z^{-2}\right), & N \text{ odd}, \end{cases}$$
3. pole radii and angles:
$$A(z) = \prod_{i=1}^{N_1} \left(1 + q_i z^{-1}\right) \cdot \prod_{i=1}^{N_2} \left(1 - 2r_i \cos(\varphi_i) z^{-1} + r_i^2 z^{-2}\right), \quad N = N_1 + 2N_2.$$

We assume the filter to be real-valued, i. e. all parameters used in the above list are real-valued, and the poles and zeros lie either on the real axis or occur in complex conjugate pairs. Note that the second order factors in item 2 of the above list contain complex conjugate pole pairs as well as pairs of different real-valued poles. Hence, an additional factor $(1 + a_{10}z^{-1})$ is only necessary if N is odd. On the other hand, the second order factors in item 3 of the above list only contain complex conjugate pole pairs. Hence, if single real-valued poles are desired, first order factors $(1 + q_i z^{-1})$ must be included, no matter if N is even or odd.

5.3.1 Constraints on the Design Variables

In the previous section, we mentioned that it is advantageous to use the polynomial coefficients a_n and b_m as optimization parameters. This applies if we consider the minimization of the approximation error. However, it is not obvious how to handle the stability problem. We do not know any explicit formulation of constraints on the polynomial coefficients a_n that are necessary and sufficient for the stability of denominators $A(z)$ of arbitrary degrees N. The polynomial coefficients a_n and b_m have been used as design parameters e. g. in [19, 23, 72, 74, 76, 122, 136, 138]. In [72, 122, 136], the stability problem has not been addressed. A simple sufficient stability constraint that has been used in [23, 74, 76] is the requirement

$$\operatorname{Re}\{A(z)\} > 0, \quad |z| = 1. \tag{5.3}$$

A proof of the sufficiency of (5.3) has been given in [23]. The constraint (5.3) is linear with respect to the polynomial coefficients a_n which makes it especially attractive for the use in methods based on linear programming [23] or quadratic programming [74, 76]. It is, however, a semi-infinite constraint, because it has to be satisfied everywhere on the unit circle of the complex plane. Nevertheless, the examples given in [23, 74, 76] show that it is usually sufficient to satisfy this constraint on a dense grid of points equally distributed around the unit circle. It turns out that the constraint (5.3) is very restrictive. In many experiments we were able to find stable filters that do not satisfy the constraint (5.3) and have considerably smaller approximation errors than solutions satisfying (5.3). Examples will be given in Section 5.5. Note that condition (5.3) does not allow for the specification of a maximum pole radius. It can, however, be modified such that the poles of the filter are guaranteed to lie inside or on the circle $|z| = \rho$ with arbitrary $\rho > 0$. This modified condition and its proof are given in Section C.3 of the appendix.

Using the polynomial coefficients of second order factors of the denominator polynomial $A(z)$ as design parameters, a specified maximum pole radius ρ will not be exceeded if the coefficients satisfy

$$|a_{10}| \leq \rho, \qquad |a_{1i}| \leq \rho + \frac{a_{2i}}{\rho}, \qquad a_{2i} \leq \rho^2, \qquad i = 1, 2, \ldots, \left\lfloor \frac{N}{2} \right\rfloor. \tag{5.4}$$

The condition on a_{10} is obvious. The inequalities involving the coefficients a_{1i} and a_{2i}, $i \geq 1$, follow from a straightforward extension of the well-known stability triangle [47, 104, 113] as shown in Section C.3 of the appendix. The conditions (5.4) can be formulated as linear constraints on the design variables and can thus be handled by a large number of standard methods [34].

If the pole locations, i. e. the real-valued quantities q_i, r_i, and φ_i are used as optimization parameters, it is especially simple to formulate stability constraints. If we require a maximum pole radius ρ, the variables q_i and r_i must satisfy $0 \leq r_i \leq \rho$ and $-\rho \leq q_i \leq \rho$, respectively. These simple bound constraints are easily incorporated in an optimization algorithm [34].

5.3.2 Strategies Based on Unconstrained Optimization

As an alternative to the conditions (5.4) on the coefficients of first and second order factors of $A(z)$, a parameterization of all stable first and second order polynomials can be used to represent the factors of $A(z)$ [75, 77]. In this case, an algorithm for unconstrained optimization can be used to compute a stable locally optimal filter. However, the objective function becomes more complicated which makes it harder to find good solutions in general.

An interesting method for designing stable IIR filters with arbitrary magnitude and phase responses has been proposed in [138]. This method is based on the fact that in several applications the absolute delay of the filter is not important. Therefore, instead of imposing constraints on the design variables to guarantee stability, the authors propose to change the desired phase response by an additive linear phase term. The corresponding delay is chosen in such a way that the optimum filter is stable. This method is, however, very costly because several design problems with different desired phase responses must be solved.

Another strategy for designing stable filters has been used in [19, 30, 45]. Let the optimization parameters be contained in the vector $\boldsymbol{x}$. The methods proposed in [19, 30, 45] are iterative and generate a sequence of points, $\boldsymbol{x}^{(k)}$, $k = 1, 2, \ldots$, that ideally converges to a local optimum of the optimization problem. The update of a given point is performed by adding an appropriate update vector $\boldsymbol{\delta}$ scaled by a positive constant α:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha \boldsymbol{\delta}^{(k)}. \tag{5.5}$$

The update $\boldsymbol{\delta}$ is called *search direction* and the scalar α is the *step size*. The step size can either be fixed or it can be optimized in every iteration step. For designing stable filters, the poles are computed in every iteration step. If in some iteration step one or more poles are outside the unit circle – or outside any other circle with specified radius ρ –, then the new point $\boldsymbol{x}^{(k+1)}$ is discarded, and another point $\boldsymbol{x}^{(k+1)}$ is computed from the stable solution $\boldsymbol{x}^{(k)}$ by using the same search direction $\boldsymbol{\delta}^{(k)}$ but scaled by a new step size

α that is smaller than the previous one. Usually, the step sizes are decreased by factors of two. This process is repeated until $\boldsymbol{x}^{(k+1)}$ corresponds to a stable filter. It is interesting to note that this heuristic step size adaptation has also been applied in [30, 45] where the poles and zeros of the transfer function are used as design parameters and where simple bound constraints on the pole radii could have been imposed. One possible reason for this is the fact that several years ago methods for unconstrained optimization were more established and more easily available than methods incorporating constraints. The drawback of this step size adaptation approach is that the step size usually decreases to zero as soon as one or more poles approach the prescribed maximum pole radius although the solution might still be far from a constrained local minimum. This problem occurs because the search direction is optimized to decrease the cost function and does not take the stability constraint into account. Instead of just scaling this search direction, the search direction itself should be changed. An optimum approach should directly compute a search direction that decreases the approximation error but at the same time maintains stability. Such an optimum method for finding a search direction subject to a constraint on the pole radii will be discussed in the next section.

5.4 Design Algorithm

The algorithm presented in this section is based on the Gauss-Newton method which is a standard method for solving nonlinear least squares problems [34]. In Section 5.4.1, we discuss this method and its application to the unconstrained least squares IIR filter design problem. In Section 5.4.2, we will explain how to incorporate the required constraint on the pole radii. This modification of the original Gauss-Newton method is based on Rouché's theorem [131]. Section 5.4.3 presents a numerical algorithm solving the resulting subproblems of the modified Gauss-Newton method. Finally, in Section 5.4.4, the complete design algorithm is summarized.

5.4.1 The Gauss-Newton Method

We use the polynomial coefficients a_n and b_m to represent the transfer function $H(z)$ as shown in (5.1). Define a vector $\boldsymbol{x}$ containing the design variables:

$$\boldsymbol{x} = [a_1, a_2, \ldots, a_N, b_0, b_1, \ldots, b_M]^T. \tag{5.6}$$

In order to explicitly show the dependence of the frequency response $H(e^{j\omega})$ on the variables contained in $\boldsymbol{x}$, we will use the notation $H(e^{j\omega}, \boldsymbol{x})$ whenever necessary. Likewise, we will use the notation $E(\omega, \boldsymbol{x}) = D(e^{j\omega}) - H(e^{j\omega}, \boldsymbol{x})$. In the Gauss-Newton method, the residuals of the least squares problem are linearized in every iteration step such that a sequence of linear least squares problems is solved. In the IIR filter design problem (5.2), the residuals are $E(\omega_i, \boldsymbol{x})$, $i = 0, 1, \ldots, L-1$. At iteration step k the current iterate is $\boldsymbol{x}^{(k)}$. The first order Taylor series approximation of $H(e^{j\omega}, \boldsymbol{x})$ about $\boldsymbol{x}^{(k)}$ is given by

$$H\left(e^{j\omega}, \boldsymbol{x}\right) \approx H\left(e^{j\omega}, \boldsymbol{x}^{(k)}\right) + \nabla_{\boldsymbol{x}}^T H\left(e^{j\omega}, \boldsymbol{x}^{(k)}\right) \cdot \boldsymbol{\delta}, \qquad \boldsymbol{\delta} = \boldsymbol{x} - \boldsymbol{x}^{(k)}, \tag{5.7}$$

where $\nabla_{\boldsymbol{x}} H\left(e^{j\omega}, \boldsymbol{x}^{(k)}\right)$ is the gradient vector of $H\left(e^{j\omega}, \boldsymbol{x}\right)$ evaluated at $\boldsymbol{x} = \boldsymbol{x}^{(k)}$. The term on the right-hand side of (5.7) is the inner product of the gradient $\nabla_{\boldsymbol{x}} H\left(e^{j\omega}, \boldsymbol{x}^{(k)}\right)$ and the vector $\boldsymbol{\delta} = \boldsymbol{x} - \boldsymbol{x}^{(k)}$. From (5.7), the linearization of $E(\omega, \boldsymbol{x})$ at iteration step k is

$$\begin{aligned} E(\omega, \boldsymbol{x}) &\approx D\left(e^{j\omega}\right) - H\left(e^{j\omega}, \boldsymbol{x}^{(k)}\right) - \nabla_{\boldsymbol{x}}^T H\left(e^{j\omega}, \boldsymbol{x}^{(k)}\right) \cdot \boldsymbol{\delta} \\ &= E\left(\omega, \boldsymbol{x}^{(k)}\right) - \nabla_{\boldsymbol{x}}^T H\left(e^{j\omega}, \boldsymbol{x}^{(k)}\right) \cdot \boldsymbol{\delta}, \qquad \boldsymbol{\delta} = \boldsymbol{x} - \boldsymbol{x}^{(k)}. \end{aligned} \tag{5.8}$$

Replacing the complex error function $E(\omega, \boldsymbol{x})$ in problem (5.2) by its linearization (5.8), and using $\boldsymbol{\delta}$ as a variable vector to be optimized, we obtain the following linear least squares problem at iteration step k:

$$\underset{\boldsymbol{\delta}}{\text{minimize}} \quad \sum_{i=0}^{L-1} W(\omega_i) \left| E\left(\omega_i, \boldsymbol{x}^{(k)}\right) - \nabla_{\boldsymbol{x}}^T H\left(e^{j\omega_i}, \boldsymbol{x}^{(k)}\right) \cdot \boldsymbol{\delta} \right|^2. \tag{5.9}$$

Using the abbreviations

$$W_i := W(\omega_i), \quad E_i^{(k)} := E\left(\omega_i, \boldsymbol{x}^{(k)}\right), \quad \text{and} \quad \nabla_{\boldsymbol{x}} H_i^{(k)} := \nabla_{\boldsymbol{x}} H\left(e^{j\omega_i}, \boldsymbol{x}^{(k)}\right),$$

and noting that $\boldsymbol{\delta}$ is real-valued, the term to be minimized in (5.9) can be expanded as follows:

$$\begin{aligned} &\sum_{i=0}^{L-1} W_i |E_i^{(k)} - \nabla_{\boldsymbol{x}}^T H_i^{(k)} \cdot \boldsymbol{\delta}|^2 \\ &\quad = \sum_{i=0}^{L-1} W_i |E_i^{(k)}|^2 - 2\text{Re}\left\{ \sum_{i=0}^{L-1} (\nabla_{\boldsymbol{x}} H_i^{(k)})^* \, W_i E_i^{(k)} \right\}^T \boldsymbol{\delta} \\ &\qquad + \boldsymbol{\delta}^T \left\{ \sum_{i=0}^{L-1} (\nabla_{\boldsymbol{x}} H_i^{(k)})^* \, W_i \, (\nabla_{\boldsymbol{x}}^T H_i^{(k)}) \right\} \boldsymbol{\delta} \\ &\quad = \boldsymbol{e}^{(k)H} \boldsymbol{W} \boldsymbol{e}^{(k)} - \underbrace{2\text{Re}\left\{ \boldsymbol{J}^{(k)H} \boldsymbol{W} \boldsymbol{e}^{(k)} \right\}^T}_{\boldsymbol{f}^{(k)}} \boldsymbol{\delta} + \boldsymbol{\delta}^T \underbrace{\boldsymbol{J}^{(k)H} \boldsymbol{W} \boldsymbol{J}^{(k)}}_{\boldsymbol{H}^{(k)}} \boldsymbol{\delta}, \end{aligned} \tag{5.10}$$

with

$$\boldsymbol{e}^{(k)} \;=\; [E_0^{(k)}, E_1^{(k)}, \ldots, E_{L-1}^{(k)}]^T,$$

$$\begin{aligned}\boldsymbol{W} &= \begin{bmatrix} W_0 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & W_{L-1} \end{bmatrix}, \\ \boldsymbol{J}^{(k)} &= [\nabla_{\boldsymbol{x}} H_0^{(k)}, \nabla_{\boldsymbol{x}} H_1^{(k)}, \ldots, \nabla_{\boldsymbol{x}} H_{L-1}^{(k)}]^T. \end{aligned} \tag{5.11}$$

Since the term $\boldsymbol{e}^{(k)H}\boldsymbol{W}\boldsymbol{e}^{(k)}$ in (5.10) does not depend on $\boldsymbol{\delta}$, problem (5.9) can be written as

$$\underset{\boldsymbol{\delta}}{\text{minimize}} \quad \boldsymbol{\delta}^T \boldsymbol{H}^{(k)}\boldsymbol{\delta} - 2\boldsymbol{f}^{(k)T}\boldsymbol{\delta} \tag{5.12}$$

with

$$\boldsymbol{H}^{(k)} = \boldsymbol{J}^{(k)H}\boldsymbol{W}\boldsymbol{J}^{(k)} \quad \text{and} \quad \boldsymbol{f}^{(k)} = \text{Re}\left\{\boldsymbol{J}^{(k)H}\boldsymbol{W}\boldsymbol{e}^{(k)}\right\}. \tag{5.13}$$

Note that the matrices $\boldsymbol{H}^{(k)}$ are real-valued and positive semi-definite. Solving the minimization problem (5.12) is equivalent to computing the solution to the system of linear equations

$$\boldsymbol{H}^{(k)}\boldsymbol{\delta} = \boldsymbol{f}^{(k)}. \tag{5.14}$$

In every iteration step of the Gauss-Newton algorithm, the linear system (5.14) is solved for $\boldsymbol{\delta} = \boldsymbol{\delta}^{(k)}$. The new iterate is computed by setting

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha\boldsymbol{\delta}^{(k)}, \quad \alpha > 0. \tag{5.15}$$

A line search algorithm can be used to optimize the step size α in every iteration step [34]. However, the algorithm usually shows a good convergence behavior even with fixed step size.

Summarizing, the Gauss-Newton method works as follows:

0. Choose an initial guess $\boldsymbol{x}^{(0)}$. Set $k := 0$.
1. Compute $\boldsymbol{H}^{(k)}$ and $\boldsymbol{f}^{(k)}$ from (5.13).
2. Solve $\boldsymbol{H}^{(k)}\boldsymbol{\delta} = \boldsymbol{f}^{(k)}$ for $\boldsymbol{\delta} = \boldsymbol{\delta}^{(k)}$.
3. Set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha\boldsymbol{\delta}^{(k)}, \quad \alpha > 0$.
4. Set $k := k + 1$. Go to 1.

The iteration is stopped as soon as there is no significant change of $\boldsymbol{x}^{(k)}$ from one iteration step to the next. IIR filters designed by this algorithm may be stable or not, depending on the specification and on the orders of numerator and denominator. However, by appropriately changing the way the update $\boldsymbol{\delta}$ is computed, an arbitrary constraint on the pole radii can be satisfied provided that the initial guess $\boldsymbol{x}^{(0)}$ satisfies this constraint.

Such an initial guess is easily found by computing the FIR solution to the design problem. This is accomplished by solving a system of linear equations [87]. We will use the result of Rouché's theorem to appropriately modify the way the update $\boldsymbol{\delta}$ is computed such that the required constraint on the pole radii is met.

5.4.2 Rouché's Theorem

If $f(z)$ and $g(z)$ are analytic inside and on a closed contour C, and $|g(z)| < |f(z)|$ on C, then $f(z)$ and $f(z)+g(z)$ have the same number of zeros inside C.

A proof of this theorem is given in [131]. Let $\Delta(z)$ be defined by

$$\Delta(z) = \delta_1 z^{-1} + \delta_2 z^{-2} + \ldots + \delta_N z^{-N},$$

where δ_n, $n = 1,2,\ldots,N$, are the first N elements of the update vector $\boldsymbol{\delta}$ in (5.15). According to (5.6), these elements update the denominator coefficients a_n, $n = 1,2,\ldots,N$. Define

$$f(z) = z^N A(z) = z^N + a_1 z^{N-1} + \ldots + a_N$$

and

$$g(z) = z^N \Delta(z) = \delta_1 z^{N-1} + \delta_2 z^{N-2} + \ldots + \delta_N,$$

where $A(z)$ is the denominator polynomial of the filter's transfer function. The functions $f(z)$ and $g(z)$ are analytic for all finite z, and $f(z)$ has the same zeros as $A(z)$. Let the contour C be a circle with radius ρ centered at the origin of the complex plane. Furthermore, let the denominator polynomial $A^{(k)}(z)$ at iteration step k have all its zeros inside the circle C. According to (5.15), the new denominator polynomial is given by

$$A^{(k+1)}(z) = A^{(k)}(z) + \alpha \Delta^{(k)}(z), \quad \alpha > 0,$$

where $\Delta^{(k)}(z)$ is the denominator update at iteration step k. Assume we choose α from the open interval $(0,1)$. Then, according to Rouché's theorem, $A^{(k)}(z)$ and $A^{(k+1)}(z)$ have the same number of zeros inside C if the update $\Delta^{(k)}(z)$ satisfies

$$\left|\Delta^{(k)}(z)\right| \leq \left|A^{(k)}(z)\right|, \quad |z| = \rho, \tag{5.16}$$

because then $\alpha\left|\Delta^{(k)}(z)\right| < \left|A^{(k)}(z)\right|$ is satisfied on the circle C. Since $A^{(k)}(z)$ and $A^{(k+1)}(z)$ have the same total number of zeros and $A^{(k)}(z)$ has all its zeros inside C, $A^{(k+1)}(z)$ must also have all its zeros inside C. Therefore, if an initial polynomial $A^{(0)}(z)$ is chosen having all its zeros inside a circle with radius ρ, then all polynomials $A^{(k)}(z)$, $k = 1,2,\ldots$, have their zeros inside this circle if the updates $\Delta^{(k)}(z)$, $k = 0,1,\ldots$, satisfy (5.16).

In order to incorporate a constraint on the pole radii, the computation of the update vector $\boldsymbol{\delta}$ in the Gauss-Newton method must be changed such that the constraint (5.16) is satisfied. This is accomplished by solving the quadratic minimization problem (5.12) subject to a constraint on the first N elements of $\boldsymbol{\delta}$. The other $M+1$ elements of $\boldsymbol{\delta}$ update the numerator coefficients and have no influence on the poles. Hence, these coefficients need not be constrained. Since $\Delta(z)$ depends linearly on the update coefficients δ_n, $n = 1, 2, \ldots, N$, its magnitude $|\Delta(z)|$ is a convex function with respect to the coefficients δ_n. Hence, as discussed in Section A.3 of the appendix, the set defined by the constraint (5.16) is convex. The constraint (5.16) is a semi-infinite constraint because it must be satisfied everywhere on the circle $|z| = \rho$. Consequently, the subproblems are convex quadratic semi-infinite programming problems. They have the same structure as the constrained least squares FIR filter design problem with constraints on the complex error function given by (2.76). Therefore, they can be solved by the multiple exchange algorithm presented in Section 3.2.1.

5.4.3 Solving the Subproblems

In every iteration step of the proposed algorithm we compute an optimum update $\boldsymbol{\delta}^{(k)}$ to the actual coefficients $\boldsymbol{x}^{(k)}$. The update $\boldsymbol{\delta}^{(k)}$ is optimized in such a way that the requirement on the pole radii is met. Let $\boldsymbol{\delta}$ be partitioned in vectors $\boldsymbol{\delta}_a$ and $\boldsymbol{\delta}_b$, where $\boldsymbol{\delta}_a$ contains the first N elements of $\boldsymbol{\delta}$ which update the denominator coefficients a_n, $n = 1, 2, \ldots, N$, and $\boldsymbol{\delta}_b$ contains the other $M+1$ elements updating the numerator coefficients b_m, $m = 0, 1, \ldots, M$:

$$\boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\delta}_a \\ \boldsymbol{\delta}_b \end{bmatrix}. \tag{5.17}$$

We use the notation $\Delta(z, \boldsymbol{\delta}_a)$ to indicate the dependence of $\Delta(z)$ on $\boldsymbol{\delta}_a$. The optimization problem that must be solved in order to obtain the update $\boldsymbol{\delta}$ at iteration step k is the following:

$$\begin{aligned} &\underset{\boldsymbol{\delta}}{\text{minimize}} \quad \boldsymbol{\delta}^T \boldsymbol{H}^{(k)} \boldsymbol{\delta} - 2\boldsymbol{f}^{(k)T} \boldsymbol{\delta} \\ &\text{subject to} \quad \left|\Delta\left(\rho e^{j\theta}, \boldsymbol{\delta}_a\right)\right| \leq \left|A^{(k)}\left(\rho e^{j\theta}\right)\right|, \quad \theta \in [0, 2\pi), \end{aligned} \tag{5.18}$$

where $\boldsymbol{H}^{(k)}$ and $\boldsymbol{f}^{(k)}$ have been defined in (5.13). The constraint in (5.18) is just another formulation of the constraint (5.16) implied by Rouché's theorem. Note that only the first N of the $M + N + 1$ elements of the unknown update vector $\boldsymbol{\delta}$ are constrained.

Problem (5.18) can be solved by the multiple exchange algorithm developed in Section 3.2.1. For the problem under consideration, we have to implement the multiple exchange algorithm in such a way that the inequality in (5.18) is satisfied for all values $\theta \in [0, 2\pi)$. Otherwise, the requirements of Rouché's theorem are not met, and we cannot guarantee that the pole radii do not exceed the desired value ρ. Consequently, the determination of the local maxima of the constraint function in the multiple exchange algorithm must be carried out on a continuous interval instead of a grid. We do this by first computing the local maxima on a grid and then refining these approximate maxima by Newton's method. This approach for implementing constraints on continuous intervals has been proposed in [119] in the context of constrained least squares design of linear phase FIR filters. In order to apply Newton's method to compute the local maxima of the constraint function, it is advantageous to take the square of the inequality constraint in (5.18) for defining the constraint function. Let l denote the iteration index of the multiple exchange algorithm which solves problem (5.18) in every iteration step k of the modified Gauss-Newton method. In the following, we will denote the l^{th} multiple exchange step in iteration k by (k,l). In every iteration step (k,l) we compute the local maxima of the constraint function

$$c^{(k,l)}(\theta) = \left|\Delta\left(\rho e^{j\theta}, \boldsymbol{\delta}_a^{(k,l)}\right)\right|^2 - \left|A^{(k)}\left(\rho e^{j\theta}\right)\right|^2, \quad \theta \in [0, 2\pi), \tag{5.19}$$

where $A^{(k)}\left(\rho e^{j\theta}\right)$ is the actual denominator polynomial which is fixed at iteration step k. The vector $\boldsymbol{\delta}_a^{(k,l)}$ contains the first N update coefficients at iteration (k,l).

Note that we have used squared magnitudes for the definition of the constraint function $c^{(k,l)}(\theta)$ in (5.19). This simplifies the exact determination of local maxima. However, we still implement the constraint without squares as formulated in problem (5.18). This helps to obtain more accurate results because $\left|A^{(k)}(\rho e^{j\theta})\right|$ may become very small.

With these preliminaries, the multiple exchange algorithm solving the subproblem at iteration step k can be formulated as follows:

0. $l = 0$. Solve the unconstrained quadratic minimization problem (5.12) for $\boldsymbol{\delta}^{(k,0)}$.

1. Determine the local maxima of the constraint function $c^{(k,l)}(\theta)$ given by (5.19). If $c^{(k,l)}(\theta) \leq 0$ is satisfied for all $\theta \in [0, 2\pi)$, set $\boldsymbol{\delta}^{(k)} := \boldsymbol{\delta}^{(k,l)}$ and stop. Otherwise, go to 2.

2. Determine the set $\Theta_{viol}^{(k,l)}$ of local maximizers of $c^{(k,l)}(\theta)$ that satisfy $c^{(k,l)}(\theta) > 0$.

3. Formulate a new set of constraints by using the current active constraints and the new constraints

$$\operatorname{Re}\left\{\Delta\left(\rho e^{j\theta}, \boldsymbol{\delta}_a\right) e^{-j\phi_\Delta^{(k,l)}(\theta)}\right\} \leq \left|A^{(k)}\left(\rho e^{j\theta}\right)\right|, \quad \theta \in \Theta_{viol}^{(k,l)}.$$

4. $l := l + 1$. Compute $\boldsymbol{\delta}^{(k,l)}$ by solving the quadratic minimization problem (5.12) subject to the linear constraints determined in step 3. Go to 1.

The phase $\phi_{\Delta}^{(k,l)}(\theta)$ used in step 3 of the algorithm is defined by

$$\phi_{\Delta}^{(k,l)}(\theta) = \arg\left\{\Delta\left(\rho e^{j\theta}, \boldsymbol{\delta}_a^{(k,l)}\right)\right\}.$$

In step 1 of the multiple exchange algorithm, Newton's method is used to compute the local maxima of the constraint function $c^{(k,l)}(\theta)$. This results in a third iteration loop indexed by m. For the sake of clarity, we will drop the iteration indices k and l for describing the Newton subiterations. Let $\theta^{(m)}$ be one of the angles in the vicinity of which a local maximum of the constraint function $c(\theta)$ occurs. The angle $\theta^{(m)}$ is refined by the iteration

$$\theta^{(m+1)} = \theta^{(m)} - \frac{c'(\theta^{(m)})}{c''(\theta^{(m)})}, \quad m = 0, 1, 2, \ldots, \tag{5.20}$$

where $c'(\theta)$ and $c''(\theta)$ denote the first and second derivatives of the constraint function $c(\theta)$ with respect to the angle variable θ. The angle $\theta^{(0)}$ is the local maximum determined on a grid of angles. Define sequences $r_a(n)$ and $r_\delta(n)$ by

$$\begin{aligned} r_a(n) &= \sum_{i=0}^{N-n} a_i a_{i+n} \rho^{-2i-n}, \quad a_0 = 1, \quad n = 0, 1, \ldots, N, \\ r_\delta(n) &= \sum_{i=1}^{N-n} \delta_i \delta_{i+n} \rho^{-2i-n}, \quad n = 0, 1, \ldots, N-1, \end{aligned} \tag{5.21}$$

where a_i and δ_i are the fixed denominator coefficients and their fixed update coefficients in the respective iteration step (k, l). The constraint function $c(\theta)$ given by (5.19) can be written as

$$c(\theta) = r_\delta(0) + 2\sum_{n=1}^{N-1} r_\delta(n)\cos(n\theta) - r_a(0) - 2\sum_{n=1}^{N} r_a(n)\cos(n\theta).$$

Consequently, the derivatives $c'(\theta)$ and $c''(\theta)$ are given by

$$\begin{aligned} c'(\theta) &= -2\sum_{n=1}^{N-1} n r_\delta(n)\sin(n\theta) + 2\sum_{n=1}^{N} n r_a(n)\sin(n\theta) \\ c''(\theta) &= -2\sum_{n=1}^{N-1} n^2 r_\delta(n)\cos(n\theta) + 2\sum_{n=1}^{N} n^2 r_a(n)\cos(n\theta). \end{aligned}$$

In step 4 of the multiple exchange algorithm, a quadratic programming problem must be solved. In the implementation of the multiple exchange algorithm that we used for designing FIR filters, we solved the duals of the quadratic programming problems. One advantage of the dual is the fact that the number of dual variables is often smaller than the number of primal variables. Moreover, due to the simple bound constraints of the dual problem, a feasible initial solution of the dual quadratic programming problem is easily available. The first advantage is especially important for designing FIR filters of high orders. However, in the IIR case the number of variables is usually small. We will show that the number of variables of the quadratic programming problem is equal to the number of denominator coefficients and is independent of the numerator order. Moreover, the constraints are upper bound constraints on $\left|\Delta(\rho e^{j\theta})\right|$, and hence a feasible initial solution satisfying these constraints is easily found (e. g. $\delta_i = 0$, $i = 1, 2, \ldots, N$). Consequently, the two main advantages of the dual formulation are not significant for the problem under consideration. Hence, we choose to solve the primal quadratic programming problems directly. These problems have a special structure that can be exploited. At iteration step (k, l), the quadratic programming problem is given by

$$\begin{aligned} \underset{\boldsymbol{\delta}}{\text{minimize}} \quad & \boldsymbol{\delta}^T \boldsymbol{H}^{(k)} \boldsymbol{\delta} - 2\boldsymbol{f}^{(k)T}\boldsymbol{\delta} \\ \text{subject to} \quad & \boldsymbol{C}^{(k,l)}\boldsymbol{\delta}_a \leq \boldsymbol{d}^{(k,l)}, \end{aligned} \tag{5.22}$$

where the linear constraints determined in step 3 of the multiple exchange algorithm have been formulated in matrix notation using the matrix $\boldsymbol{C}^{(k,l)}$ and the vector $\boldsymbol{d}^{(k,l)}$. $\boldsymbol{C}^{(k,l)}$ and $\boldsymbol{d}^{(k,l)}$ consist of the rows of $\boldsymbol{C}^{(k,l-1)}$ and $\boldsymbol{d}^{(k,l-1)}$ corresponding to active constraints of problem (5.22) at step $(k, l-1)$ and of new elements given by

$$\begin{aligned} C_{mn}^{(k,l)} &= \rho^{-n} \cos\left[n\theta_m^{(k,l)} + \phi_\Delta^{(k,l)}(\theta_m^{(k,l)})\right], \quad n = 1, 2, \ldots, N \\ d_m^{(k,l)} &= \left|A^{(k)}\left(\rho e^{j\theta_m^{(k,l)}}\right)\right|. \end{aligned}$$

The number of columns of $\boldsymbol{C}^{(k,l)}$ is determined by the denominator degree N, whereas the number of rows of $\boldsymbol{C}^{(k,l)}$ and $\boldsymbol{d}^{(k,l)}$ varies according to the number of active constraints and the number of new angles $\theta_m^{(k,l)}$ determined in step 3 of the multiple exchange algorithm.

Note that the objective function to be minimized is fixed for all multiple exchange iteration steps l. Unlike in the FIR design problem, only a part of the design variables $\boldsymbol{\delta}$ is constrained. Due to this special structure, problem (5.22) can be formulated as a problem depending on only N instead of $M + N + 1$ variables. Let $\boldsymbol{\delta}$ be partitioned in vectors $\boldsymbol{\delta}_a$ and $\boldsymbol{\delta}_b$ as in (5.17). Furthermore, define partitions of $\boldsymbol{H}^{(k)}$ and $\boldsymbol{f}^{(k)}$ given by (5.13) according to

$$\boldsymbol{H}^{(k)} = \begin{bmatrix} \boldsymbol{H}_{11}^{(k)} & \boldsymbol{H}_{12}^{(k)} \\ \boldsymbol{H}_{12}^{(k)T} & \boldsymbol{H}_{22}^{(k)} \end{bmatrix}, \qquad \boldsymbol{f}^{(k)} = \begin{bmatrix} \boldsymbol{f}_1^{(k)} \\ \boldsymbol{f}_2^{(k)} \end{bmatrix}, \tag{5.23}$$

where $\boldsymbol{H}_{11}^{(k)}$ is $N \times N$, $\boldsymbol{H}_{12}^{(k)}$ is $N \times (M+1)$, $\boldsymbol{H}_{22}^{(k)}$ is $(M+1) \times (M+1)$, $\boldsymbol{f}_1^{(k)}$ is $N \times 1$, and $\boldsymbol{f}_2^{(k)}$ is $(M+1) \times 1$. Solving

$$\frac{\partial}{\partial \boldsymbol{\delta}_b}\left(\boldsymbol{\delta}^T \boldsymbol{H}^{(k)} \boldsymbol{\delta} - 2\boldsymbol{f}^{(k)T}\boldsymbol{\delta}\right) = \boldsymbol{0},$$

for $\boldsymbol{\delta}_b$ leads to

$$\boldsymbol{\delta}_b = \boldsymbol{H}_{22}^{-1(k)}\left(\boldsymbol{f}_2^{(k)} - \boldsymbol{H}_{12}^{(k)T}\boldsymbol{\delta}_a\right), \tag{5.24}$$

which can be used to eliminate the variable $\boldsymbol{\delta}_b$ from problem (5.22). The subproblem at iteration step (k,l) can then be formulated as

$$\begin{array}{ll} \underset{\boldsymbol{\delta}_a}{\text{minimize}} & \boldsymbol{\delta}_a^T\left(\boldsymbol{H}_{11}^{(k)} - \boldsymbol{H}_{12}^{(k)}\boldsymbol{H}_{22}^{-1(k)}\boldsymbol{H}_{12}^{(k)T}\right)\boldsymbol{\delta}_a + 2\left(\boldsymbol{H}_{12}^{(k)}\boldsymbol{H}_{22}^{-1(k)}\boldsymbol{f}_2^{(k)} - \boldsymbol{f}_1^{(k)}\right)^T\boldsymbol{\delta}_a \\ \text{subject to} & \boldsymbol{C}^{(k,l)}\boldsymbol{\delta}_a \leq \boldsymbol{d}^{(k,l)}. \end{array} \tag{5.25}$$

After the optimum $\boldsymbol{\delta}_a^{(k)}$ has been determined by the multiple exchange algorithm, the numerator update $\boldsymbol{\delta}_b^{(k)}$ is computed from (5.24). Note that the number of unknowns in each subproblem has been reduced from $M+N+1$ to N and that the matrices and vectors defining the objective function in (5.25) are fixed for all multiple exchange iterations l in iteration step k.

5.4.4 The Complete Algorithm

After having described the implementation of the multiple exchange algorithm that computes an optimum update vector $\boldsymbol{\delta}$ subject to a constraint on the pole radii, we can formulate the complete algorithm solving the least squares IIR filter design problem given by (5.2). Let the denominator and numerator coefficients be contained in the vector $\boldsymbol{x}$ as defined by (5.6). The algorithm works as follows:

0. Choose an initial guess $\boldsymbol{x}^{(0)}$ that satisfies the desired constraint on the pole radii. Set $k := 0$.

1. Compute $\boldsymbol{H}^{(k)}$ and $\boldsymbol{f}^{(k)}$ from (5.13) and compute the matrices and vectors defining the objective function of the subproblems (5.25) occurring in the multiple exchange algorithm.

2. Compute the optimum update $\boldsymbol{\delta}^{(k)}$ by solving problem (5.18) using the multiple exchange algorithm described in Section 5.4.3.

3. Set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha\boldsymbol{\delta}^{(k)}$ with $\alpha \in (0,1)$.

4. If $\frac{\|\boldsymbol{\delta}^{(k)}\|_2}{\|\boldsymbol{x}^{(k+1)}\|_2} < \epsilon$, stop. Otherwise, set $k := k+1$ and go to 1.

The choices $\alpha = 0.5$ and $\epsilon = 10^{-3}$ give satisfactory results in most cases. A more sophisticated version of the algorithm could incorporate a line search to optimize the step size α in every iteration step. Line search strategies are described in [34]. Note,

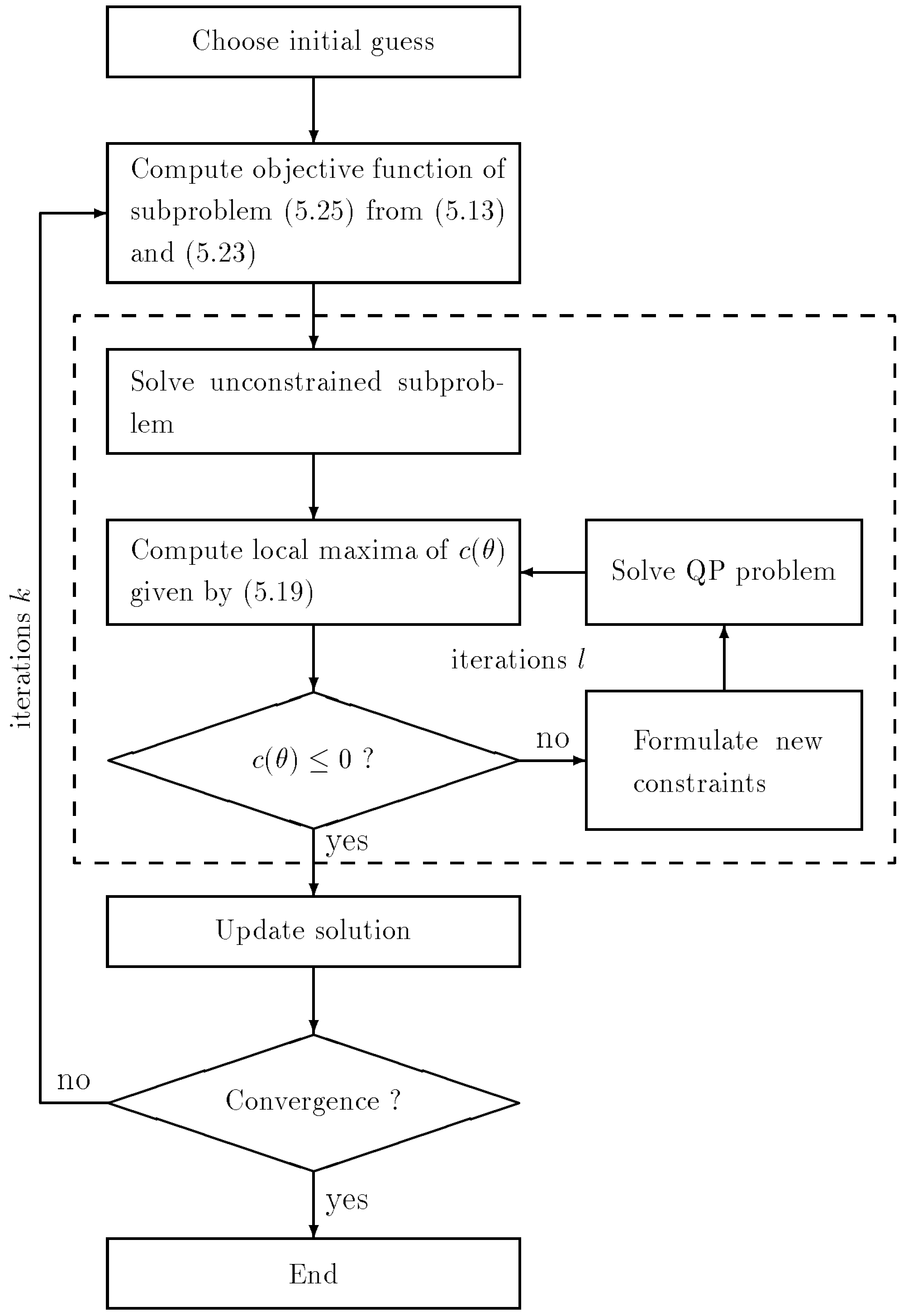

Choose initial guess
Compute objective function of subproblem (5.25) from (5.13) and (5.23)
Solve unconstrained subproblem
Compute local maxima of $c(\theta)$ given by (5.19)
Solve QP problem
iterations l
$c(\theta) \leq 0$?
no
Formulate new constraints
yes
Update solution
no
Convergence ?
yes
End
iterations k

Figure 5.1: Flowchart of the proposed IIR filter design algorithm. The dashed box contains the multiple exchange algorithm solving the subproblems.

however, that the step size must satisfy $|\alpha| < 1$. Otherwise, the requirement of Rouché's theorem is not met.

Figure 5.1 shows a flowchart of the proposed design algorithm. The dashed box indicates the multiple exchange algorithm for computing an optimum update according to Rouché's theorem.

5.5 Design Examples

In this section, we provide design examples to illustrate the performance of the design algorithm presented in this chapter. Some of these examples have already been given in the literature before. We use these examples to compare our method to previously published methods. We also compare the complexity required to implement the designed IIR filters to the complexity of other filter types, such as FIR filters, frequency selective IIR systems cascaded with phase equalizers, or coupled allpass systems. For comparing complexity, we compare the numbers of multipliers, adders, and delays required in canonical implementations of these filters (e. g. direct form, cascade form, or parallel form[47, 104, 113]). A canonical implementation of the transfer function $H(z)$ given by (5.1) requires $\max(M, N)$ delays, $M + N$ additions, and $M + N + 1$ multiplications per output sample. There are several special cases where the number of multiplications can be reduced while retaining the canonical property of the filter structure. This is due to symmetries in the filter coefficients. Some important special cases are

- linear phase FIR filters: $\lceil \frac{M+1}{2} \rceil$ multiplications[1],
- IIR filters with all their zeros on the unit circle or symmetric to the unit circle (e. g. Cauer filters): $\lceil \frac{M+1}{2} \rceil + N$ multiplications,
- IIR allpass filters: N multiplications [85].

The program `mpiir_l2` used in the following examples is a Matlab implementation of the proposed design algorithm and is given in Section 5.6. All design times have been measured on a 166 MHz Pentium PC using Matlab 5.1.

Example 1: Low-Order Lowpass Filter with Approximately Linear Passband Phase ($M = 4$, $N = 4$) [19]:

This example has been given in [19] where an algorithm for complex Chebyshev design of IIR filters has been proposed. The authors of [19] slightly modified the algorithm published by Ellacott and Williams [33] and applied it to the design of IIR filters. To

[1]The integer $n = \lceil x \rceil$ is the smallest integer satisfying $n > x$.

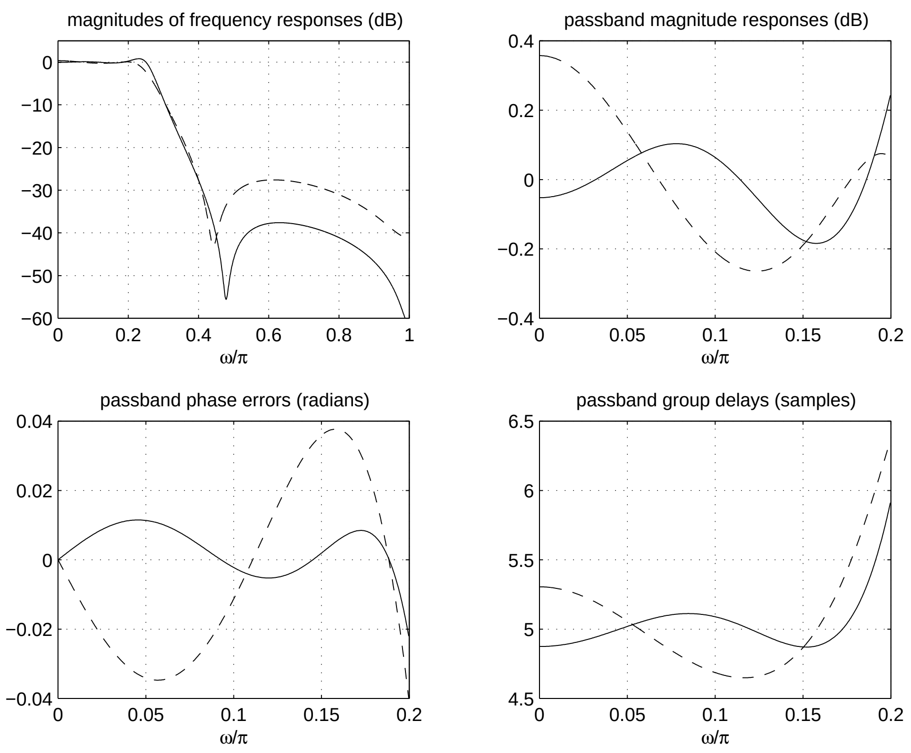


Figure 5.2: IIR lowpass filters with approximately linear passband phase responses ($M = 4$, $N = 4$). Solid curves: complex least squares design using the proposed algorithm. Dashed curves: complex Chebyshev design according to [19].

guarantee stability or to satisfy any desired maximum pole radius constraint, they use the heuristic step size adaptation described in Section 5.3.2. The pole locations are monitored and the step size is decreased until all poles are inside a circle with the desired radius. In this example, the maximum pole radius has been set to $\rho = 0.98$. The passband and stopband edges are $\omega_p = 0.2\pi$ and $\omega_s = 0.4\pi$, respectively. The desired passband phase response is linear with a group delay of five samples. The weighting is uniform over the passband and stopband. The following lines generate the specification and design the filter using the program given in Section 5.6.

```
om=pi*[linspace(0,.2,20),linspace(.4,1,60)];
D=[exp(-j*om(1:20)*5),zeros(1,60)];
W=ones(1,80);
[b,a,e] = mpiir_l2(4,4,om,D,W,.98);
```

The design took less than 3 seconds. In [19], the authors published the zero and pole locations of the filter designed by their algorithm. Hence, a comparison of the two filters is easily made. Figure 5.2 shows the magnitude responses, phase errors, and group delays of both filters. The solid curves correspond to the filter designed by the proposed algorithm and the dashed curves show the responses of the filter published in [19]. The filters are

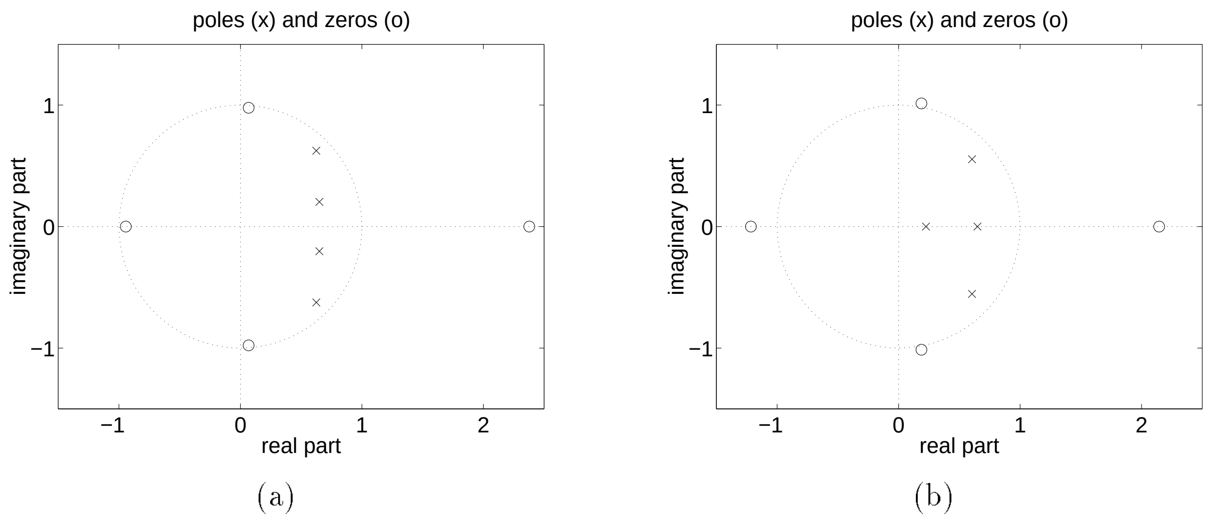


Figure 5.3: Pole-zero diagrams of IIR lowpass filters with approximately linear passband phase responses ($M = 4$, $N = 4$). (a) design by proposed algorithm. (b) design according to [19].

necessarily different because they have been designed according to different optimality criteria. However, the filter designed by the proposed algorithm is not only superior with respect to its mean squared approximation error, but also slightly superior with respect to its maximum error. The mean squared errors of the proposed filter and of the filter given in [19] are $1.3 \cdot 10^{-2}$ and $7.7 \cdot 10^{-2}$, respectively. Their maximum errors are $4.1 \cdot 10^{-2}$ and $4.2 \cdot 10^{-2}$, respectively.

Figure 5.3 shows the pole-zero diagrams for both filters. The plot on the left corresponds to the filter designed by the proposed algorithm, and the plot on the right corresponds to the filter published in [19]. Obviously, one of the four zeros is used to improve the passband phase response in both cases. The other three zeros contribute to the stopband. The stopband zeros of the proposed filter are closer to the unit circle than those of the other filter. The poles of the proposed filter are arranged more efficiently than the poles of the complex Chebyshev filter because the latter has one pole close to the origin.

It is interesting to compare IIR filters to FIR filters with comparable properties. An FIR filter approximating the same complex desired frequency response must be of order 16 to achieve a similar mean squared approximation error as the proposed IIR filter of order four. A canonical implementation of this IIR filter needs 4 delays, 9 multipliers, and 8 adders. The corresponding FIR filter needs 16 delays, 17 multipliers, and 16 adders. In order to achieve a similar Chebyshev error as the IIR filter proposed in [19], an FIR filter of order 13 is sufficient. It should be noted that the transition from passband to stopband is considerably steeper if IIR filters are used. This is not taken into account if only the approximation errors of IIR and FIR filters are compared because no approximation error is defined in the transition bands.

Example 2: Lowpass Filter with Approximately Linear Passband Phase and Magnitude Constraints ($M = 15,\ N = 15$) [74]:

In [74], an algorithm for designing stable IIR filters with arbitrary magnitude and phase responses subject to magnitude constraints has been proposed. We use the design example given in [74] to compare the results of our method to the approach of [74]. In [74], the polynomial coefficients of the designed filter have been published. Note that the algorithm proposed in this chapter does not incorporate any constraints other than a constraint on the pole radii. The algorithm proposed in [74] uses the linear stability constraint (5.3).

The passband and stopband edges of the lowpass filter are $\omega_p = 0.4\pi$ and $\omega_s = 0.56\pi$, respectively. The maximum passband ripple is specified as 0.2 dB, and the minimum stopband attenuation is required to be 30 dB. The desired phase response is linear in the passband with a group delay of 15 samples. The error weighting is 1 in the passband and 100 in the stopband. The orders of numerator and denominator are chosen as $M = N = 15$. Using the proposed algorithm, we cannot specify the required constraints on the magnitude response. Nevertheless, we choose the same values for M and N and compare the results. The maximum pole radius of the filter proposed in [74] is 0.8263. Note that this value had not been specified but just resulted from the design process. In order to make a fair comparison, we use this value as a constraint on the pole radii for computing a solution by our algorithm. The following Matlab commands design the filter by the proposed algorithm:

```
om=pi*[linspace(0,.4,40),linspace(.56,1,44)];
D=[exp(-j*om(1:40)*15),zeros(1,44)];
W=[ones(1,40),100*ones(1,44)];
[b,a,e] = mpiir_l2(15,15,om,D,W,.8263);
```

This design took six minutes. Figure 5.4 shows the design results for both filters. The solid curves correspond to the filter designed by the proposed algorithm, and the dashed curves correspond to the filter published in [74]. The filter designed by the proposed algorithm is considerably better than the one proposed in [74]. The maximum passband ripple of the filter published in [74] is 0.19 dB, and the maximum passband ripple of the proposed filter is 0.05 dB. The minimum stopband attenuations of the filters are 23 dB and 64 dB, respectively. These minimum attenuations occur at the stopband edge $\omega_s = 0.56\pi$. The maximum passband phase error of the proposed filter is smaller than the one of the filter in [74]. Only the maximum group delay error occurring at the passband edge is larger for the proposed filter. However, as discussed in Section 2.3.1, the group delay error is not an important quantity if the phase approximation is good.

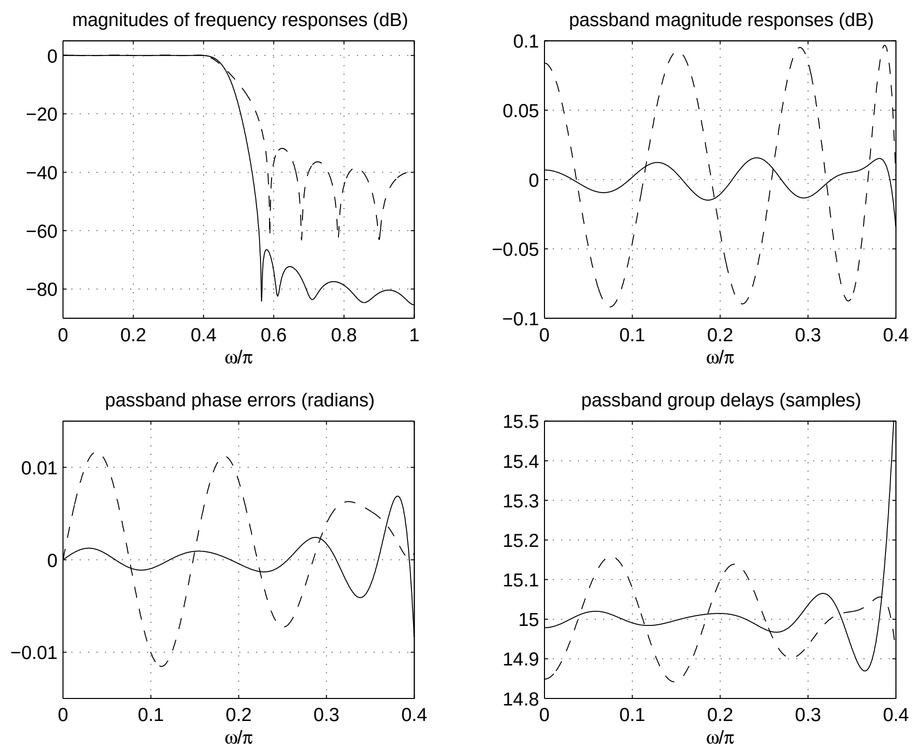
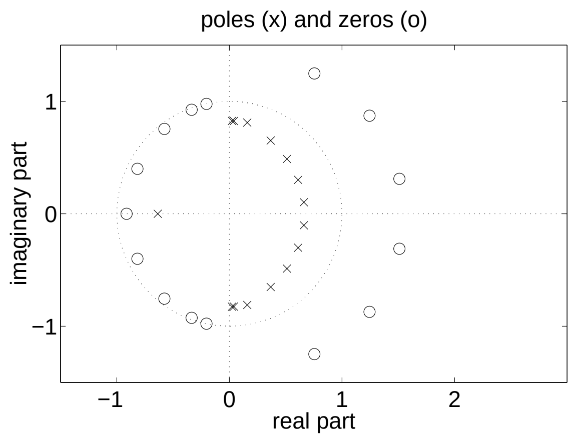
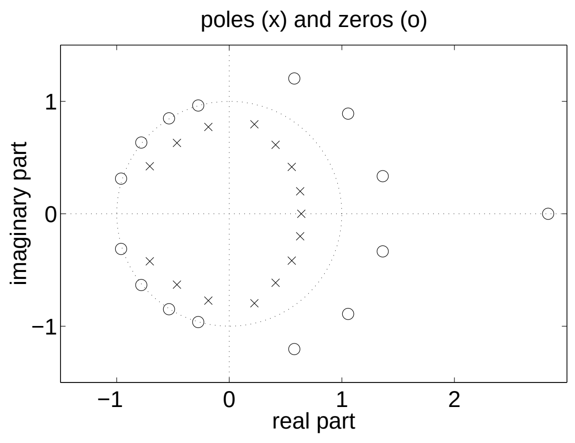


Figure 5.4: IIR lowpass filters with approximately linear passband phase responses ($M = 15$, $N = 15$). Solid curves: complex least squares design using the proposed algorithm. Dashed curves: constrained complex least squares design according to [74].

Figure 5.5 shows the pole-zero diagrams of both filters. The plot on the left shows the pole-zero diagram of the proposed filter. Six zeros contribute to the passband to fix the phase response. The nine remaining zeros contribute to the stopband. They are, however, not exactly on the unit circle. Note that minimizing the mean squared error not necessarily results in stopband zeros on the unit circle. In this example, if the stopband zeros are shifted to the unit circle, leaving their angles unchanged, and scaling the new numerator polynomial such that the approximation error is minimized, the error is increased from $4.2 \cdot 10^{-4}$ to 0.39. The filter published in [74] has seven zeros at angles associated with the passband. However, the influence of the real-valued zero is relatively small due to its large magnitude (see Figure 5.5 (b)). The eight remaining zeros are used for the stopband. They are very close to the unit circle. The poles make most of the difference between the two filters. All poles of the proposed filter except for one have angles corresponding to the passband. There might be a better solution which has all of its poles at angles corresponding to the passband. It is, however, very difficult to find such a solution, especially if the denominator degree is high. Only nine poles of the filter proposed in [74] contribute to the passband. The six remaining poles have angles corresponding to the stopband. This is the main reason why this filter behaves



(a)



(b)

Figure 5.5: Pole-zero diagrams of IIR lowpass filters with approximately linear passband phase responses ($M = 15$, $N = 15$). (a) design by proposed algorithm. (b) design according to [74].

considerably worse than the one designed by the proposed algorithm.

A probable reason why the two designs are so different is the fact that the stability constraints used in the two design algorithms are different. The constraint (5.3) used in [74] seems to be much more restrictive than the constraint imposed by satisfying Rouché's theorem in every iteration step. We mention that the filter designed by the proposed algorithm does not satisfy the sufficient stability constraint (5.3).

Example 3: Highpass Filter with Approximately Linear Passband Phase:

This example is taken from [76] where an algorithm for solving the weighted least squares IIR filter design problem has been proposed. The method does not offer the possibility to prescribe a maximum pole radius, but the filters are guaranteed to be stable. This is achieved by imposing the linear stability constraint (5.3). The coefficients of the designed filter have been published in [76].

The filter under consideration is a highpass filter with a linear desired passband phase response. The authors of [76] chose the orders of numerator and denominator polynomials as $M = N = 14$. The passband and stopband edges are $\omega_p = 0.475\pi$ and $\omega_s = 0.525\pi$, respectively. There is no weighting of the approximation error. The desired passband group delay is 12 samples. The maximum pole radius of the filter presented in [76] is 0.9276. We use this value as a constraint on the pole radii for designing the filter by the proposed algorithm. We use the same numerator degree $M = 14$, but we choose the reduce the denominator degree from 14 to $N = 6$. This decreases the power of round-off noise in fixed-point implementations [46]. Moreover, the number of multipliers and adders for implementing the filter is decreased as well.

The filter is designed by the following commands:

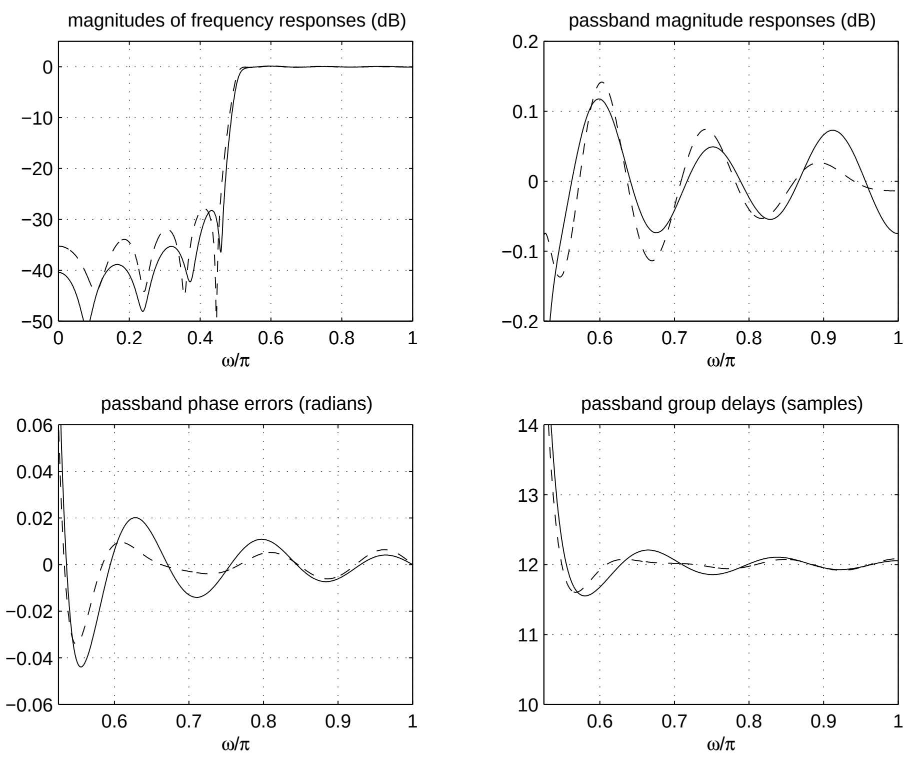
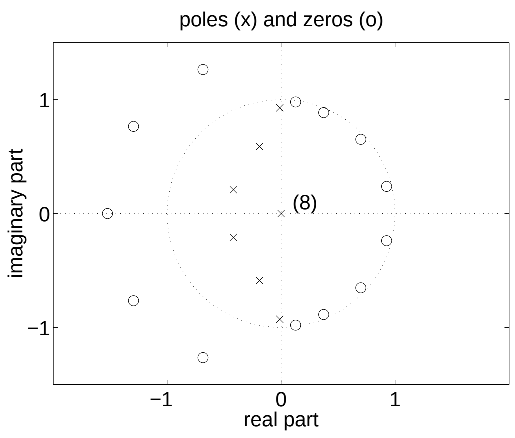


Figure 5.6: IIR highpass filters with approximately linear passband phase responses. Solid curves: design using the proposed algorithm with $M = 14$ and $N = 6$. Dashed curves: design according to [76] with $M = 14$ and $N = 14$.

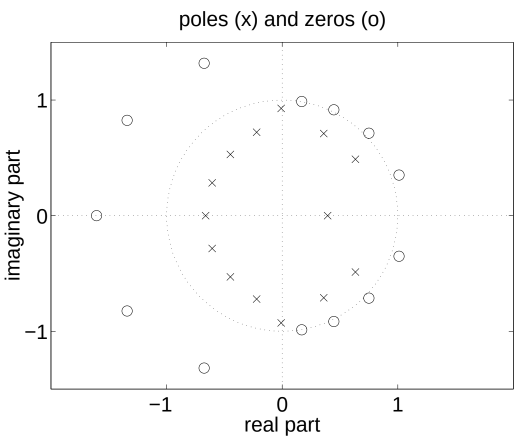
```
om=pi*[linspace(0,.475,50),linspace(.525,1,50)];
D=[zeros(1,50),exp(-j*om(51:100)*12)];
W=ones(1,100);
[b,a]=mpiir_l2(14,6,om,D,W,.9276);
```

This design took 9 seconds. Figure 5.6 shows the magnitude, phase, and group delay responses of both filters. The solid curves correspond to the design by the proposed algorithm, and the dashed curves correspond to the design presented in [76]. Despite the lower order of the denominator polynomial, the proposed design achieves a lower mean squared approximation error of $4.6 \cdot 10^{-2}$ compared to an error of $7.9 \cdot 10^{-2}$ achieved by the filter in [76].

Figure 5.7 shows the corresponding pole-zero diagrams. The plot on the left shows the poles and zeros of the proposed filter. Both filters have one real-valued zero outside the range of the plots. These zeros are at 63.7 for the proposed filters, and at -5.7 for the filter given in [76]. All six non-trivial poles of the proposed filter have angles corresponding to the passband. Five of its zeros contribute to the passband. This is necessary for obtaining a good phase approximation. Eight of the nine remaining zeros are close to the unit circle and contribute to the stopband. One zero (not shown on the plot) is not used efficiently.



(a)



(b)

Figure 5.7: Pole-zero diagrams of IIR highpass filters with approximately linear passband phase responses. (a) design by proposed algorithm ($M = 14$, $N = 6$). (b) design according to [76] ($M = 14$, $N = 14$).

Figure 5.7 (b) shows the pole-zero diagram of the filter proposed in [76]. Five poles have angles that correspond to the stopband. These pole locations are not useful for achieving a good frequency response. The zeros of this filter have similar locations as the zeros of our filter. Despite the fact that the filter designed by the algorithm proposed in [76] uses 14 non-trivial poles instead of 6, it does not achieve a better frequency response and a smaller approximation error than the proposed filter.

From what we have shown so far, it seems that by increasing the degree of the denominator from six to 14, the proposed algorithm could design a filter that is considerably better than the one designed in [76]. This is, however, not the case. We investigated the behavior of filters with fixed numerator degrees $M = 14$ and varying degrees of the denominator. We designed filters according to the above specifications for denominator degrees ranging from $N = 0$ to $N = 14$. The filters were designed by the proposed algorithm with a maximum pole radius of 0.9276 as used before, and by the algorithm described in [76]. In both cases, we used the respective solution for one value of N as starting guess for the following problem with higher N. This makes the approximation error decrease monotonically with increasing N. If all designs are started with the same initial guess, the approximation error need not necessarily decrease monotonically with increasing denominator order N. This is the case because there are different local solutions and for any N, there may be local solutions that are worse than a local solution for a smaller value of N.

The approximation errors of the designed filters are shown in Figure 5.8. The errors considerably decrease for increasing denominator orders N in the range $0 \leq N \leq 6$, but then the curves start to become flat and not much is gained by further increasing the

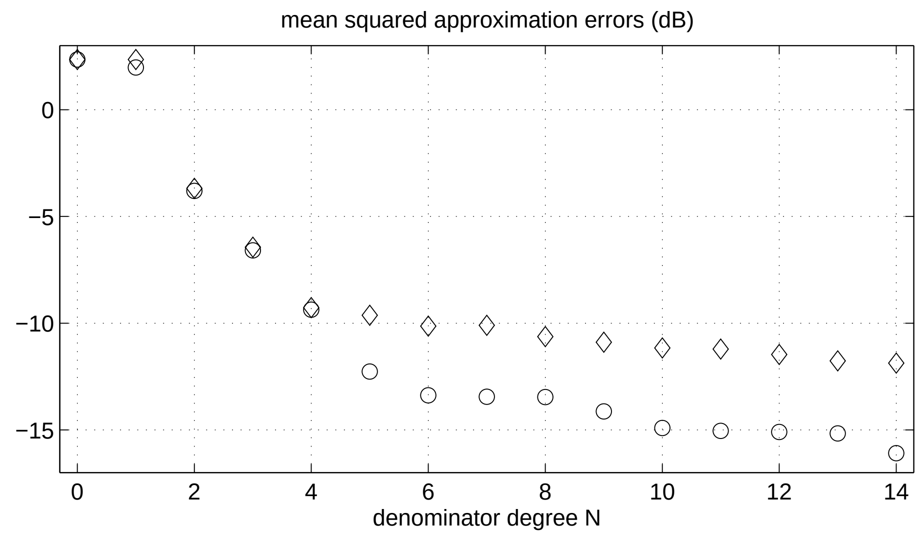


Figure 5.8: Approximation errors of highpass filters with fixed numerator degrees $M = 14$ and varying denominator degrees $N = 0 \ldots 14$. The designs were computed by the proposed algorithm ($\circ$) and by the algorithm given in [76] ($\diamond$).

order of the denominator. This turns out to be a fairly general result for IIR filter design problems with magnitude and phase specifications and a constraint on the pole radii. From Figure 5.8, this behavior occurs for two different design algorithms, and we found this observation to be true for many different design specifications. For high denominator degrees, pole-zero cancellations are more likely to occur, and often some of the poles are placed at angles that correspond to the stopbands instead of the passbands. Note that for this example, the results obtained by the proposed algorithm are only slightly better than those obtained by the method described in [76]. Example 4 shows a design where the difference between the solutions obtained by the two methods is much more pronounced.

We conclude that it is useful to limit the number of poles outside the origin for IIR design problems with specifications on magnitude and phase responses and a constraint on the pole radii. The number of poles that can be placed efficiently increases with increasing passband widths and with increasing number of passbands separated by stopbands. In most cases not much is gained if more than six to ten poles are used. If the approximation error is still too large, the numerator degree should be increased. It should be noted that using only a few poles is also advantageous for the implementation of the filter.

Example 4: Narrow Bandpass Filter with Approximately Linear Passband Phase ($M = 20,\ N = 8$):

In this example, we compare six different methods for solving the least squares IIR filter design problem with specified magnitude and phase responses:

1. the algorithm presented in this chapter,
2. the method proposed by W.-S. Lu et al. [76],
3. a sequential quadratic (SQP) programming method using second order factors of the denominator polynomial $A(z)$,
4. an SQP method using the pole radii and angles as design parameters,
5. the program `invfreqz` provided in the *Matlab Signal Processing Toolbox 4.0.1*,
6. an unconstrained nonlinear least squares method (Levenberg-Marquard).

We have already discussed the properties of the method proposed in [76] in Example 3. For the designs by methods 3 and 4 in the above list, we used the program `constr` contained in the Matlab Optimization Toolbox 1.5.1. It implements an SQP method for solving constrained nonlinear programming problems. In method 3, the constraints (5.4) are imposed on the design parameters to prescribe a maximum pole radius. Method 4 directly constrains the pole radii which are used as design parameters. The Matlab program `invfreqz` (method 5) contained in the Matlab Signal Processing Toolbox implements a Gauss-Newton method incorporating a heuristic approach for ensuring stability. This approach is based on reflecting poles outside the unit circle in every iteration step. The program `invfreqz` can also be used in a non-iterative mode. In this case, the equation error method [87] is implemented. However, for this example we use the more sophisticated iterative mode. Method 6 in the above list solves the nonlinear least squares approximation problem using a Levenberg-Marquard method [34] which is implemented by the Matlab program `leastsq` contained in the Optimization Toolbox. In this case, no stability constraint is imposed.

The passband and stopband edge frequencies of the bandpass filter to be designed are $\omega_{p1} = 0.4\pi$, $\omega_{p2} = 0.5\pi$, $\omega_{s1} = 0.36\pi$, and $\omega_{s2} = 0.54\pi$, respectively. We choose a linear desired passband phase with a group delay of 20 samples. The weighting function is 1 in the passband and 100 in the stopbands. We choose the numerator degree $M = 20$ and the denominator degree $N = 8$. Methods 1, 3, and 4 in the above list can prescribe a maximum pole radius. We use the value $\rho = 0.98$. All methods except for `invfreqz` use the FIR solution as initial guess. The program `invfreqz` uses the equation error

method	error	max. pole radius
6	0.0425	0.9974
1	0.0957	0.9800
3	1.354	0.7576
2	1.838	0.8937
5	7.573	0.9998
4	9.062	0.0116

Table 5.1: Approximation errors and maximum pole radii for different design methods. The numbering of the methods refers to the list on page 159.

method [87] to compute an initial guess. The following Matlab lines design the filter by the proposed algorithm:

```
om=pi*[linspace(0,.36,36),linspace(.4,.5,10),linspace(.54,1,46)];
D=[zeros(1,36),exp(-j*om(37:46)*20),zeros(1,46)];
W=[100*ones(1,36),ones(1,10),100*ones(1,46)];
[b,a] = mpiir_l2(20,8,om,D,W,.98);
```

This design took 30 seconds. Table 5.1 shows the approximation errors and the maximum pole radii of the filters designed by the six methods listed above. As expected, the smallest approximation error is achieved by method 6 which performs unconstrained minimization of the approximation error. Note that the corresponding filter is stable. However, two pole-pairs are very close to the unit circle which results in an undesired behavior of the frequency response in the transition bands as shown by the dashed curve in Figure 5.9. All other methods incorporate some stability constraint. The method presented in this chapter designs a filter with an approximation error slightly larger than the one of method 6. However, it satisfies the constraint on the pole radii and can easily be implemented using fixed-point arithmetic. Moreover, it does not have undesired transition band excursions. Its frequency response is shown by the solid curve in Figure 5.9. All other methods provide inferior filters with large approximation errors. The filter designed by the program `invfreqz` has one pole pair that is virtually on the unit circle. This is probably due to numerical effects occurring if poles that are close the unit circle are reflected several times. Note that the maximum pole radius of the filter designed by method 4 is very small (0.0116). Obviously, the algorithm converged to a local minimum which is very close to the FIR solution used as an initial guess. Figure 5.9 shows the frequency responses of the filters designed by the proposed method (solid curves), by unconstrained approximation (method 6, dashed curves), and by method 3 which uses a second order factorization of the denominator polynomial (dotted curves).

An FIR filter approximating the same complex frequency response must be of order 96 to achieve a similar approximation error as the IIR filter designed by the proposed

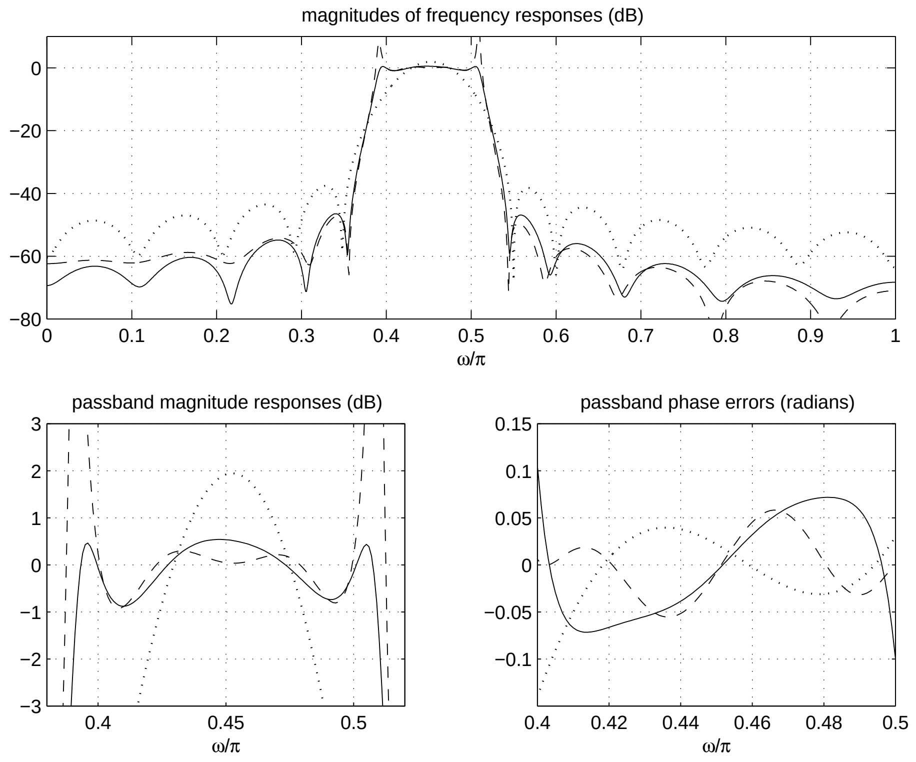


Figure 5.9: IIR bandpass filters with approximately linear passband phase responses ($M = 20$, $N = 8$). Solid curves: design using the proposed algorithm. Dashed curves: design without constraints on the pole radii. Dotted curves: design by a sequential quadratic programming method using a second order factorization of the denominator polynomial.

algorithm. For its implementation, 96 delays, 97 multipliers, and 96 adders are required. The corresponding IIR filter with $M = 20$ and $N = 8$ needs 20 delays, 29 multipliers, and 28 adders.

Similar to the previous example, we used the proposed algorithm and the algorithm published in [76] to design filters with fixed numerator degrees and varying denominator degrees approximating the same desired frequency response. Figure 5.10 shows the approximation errors versus denominator degree N. The numerator degree is $M = 20$ for all designs. The approximation errors obtained by the proposed algorithm are shown as circles, whereas the diamonds correspond to the results obtained by the method presented in [76]. Note that the difference between the approximation errors obtained by the two methods is considerably larger for this specification than for the one used in Example 3. Here, the flattening of the curves is also more pronounced. This is partly due to numerical problems for denominator degrees greater than 15. The occurrence of filter pairs with very similar approximation errors is due to the fact that a real-valued pole cannot considerably improve the approximation quality for a bandpass filter specification. Therefore, increasing an even denominator degree by one only negligibly decreases

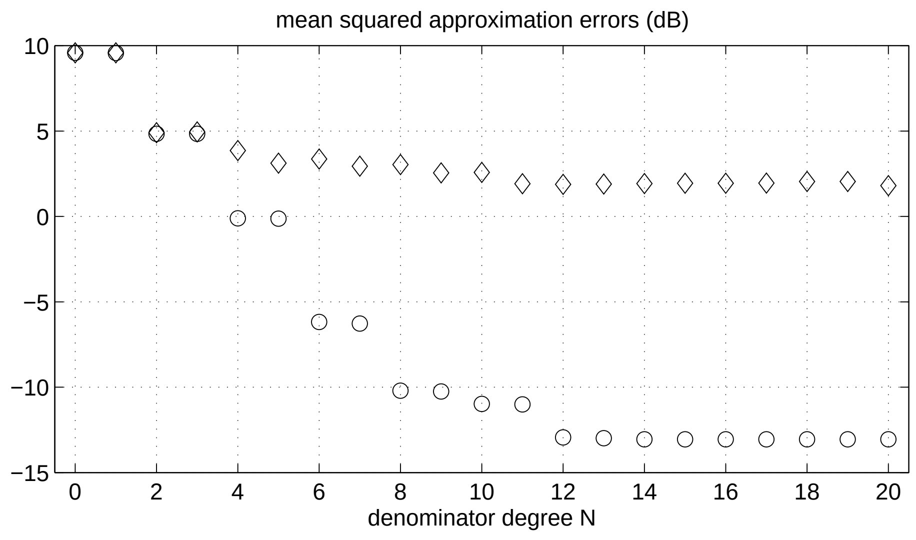


Figure 5.10: Approximation errors of bandpass filters with fixed numerator degrees $M = 20$ and varying denominator degrees $N = 0 \ldots 20$. The designs were computed by the proposed algorithm ($\circ$) and by the algorithm given in [76] ($\diamond$).

the approximation error. From Figure 5.10 it is obvious that for this example it does not make sense to use more than 8 to 12 non-trivial poles. Instead of increasing the denominator degree at this point, it is much more efficient to increase the degree of the numerator if a smaller approximation error is desired.

Example 5: IIR Equalization and Anti-Aliasing Filter:

We use the same specification as in Example 2 presented in Chapter 4 where we designed an FIR equalization and anti-aliasing filter of order 50. We want to design an IIR equalization filter that achieves an equivalent performance as this FIR equalizer of order 50. It turns out that an IIR filter with four non-trivial poles and 20 zeros is sufficient if we specify a maximum pole radius $\rho = 0.97$. The following commands generate the specification and design the IIR filter by the proposed algorithm:

```
tau=35; M=20; N=4; r=0.97;
om=pi*[linspace(0,1/16,20),linspace(3/16,1,260)];
Dt=[exp(-j*tau*om(1:20)),zeros(1,260)];
Wt=[ones(1,20),50*ones(1,260)];
z=[]; p=[-.6493,-.3246-j*1.0325,-.3246+j*1.0325]; k=.7606;
[num,den]=zp2tf(z,p,k);
Ha=freqs(num,den,om*16/pi);
D=Dt./Ha; W=Wt.*abs(Ha).^2;
```

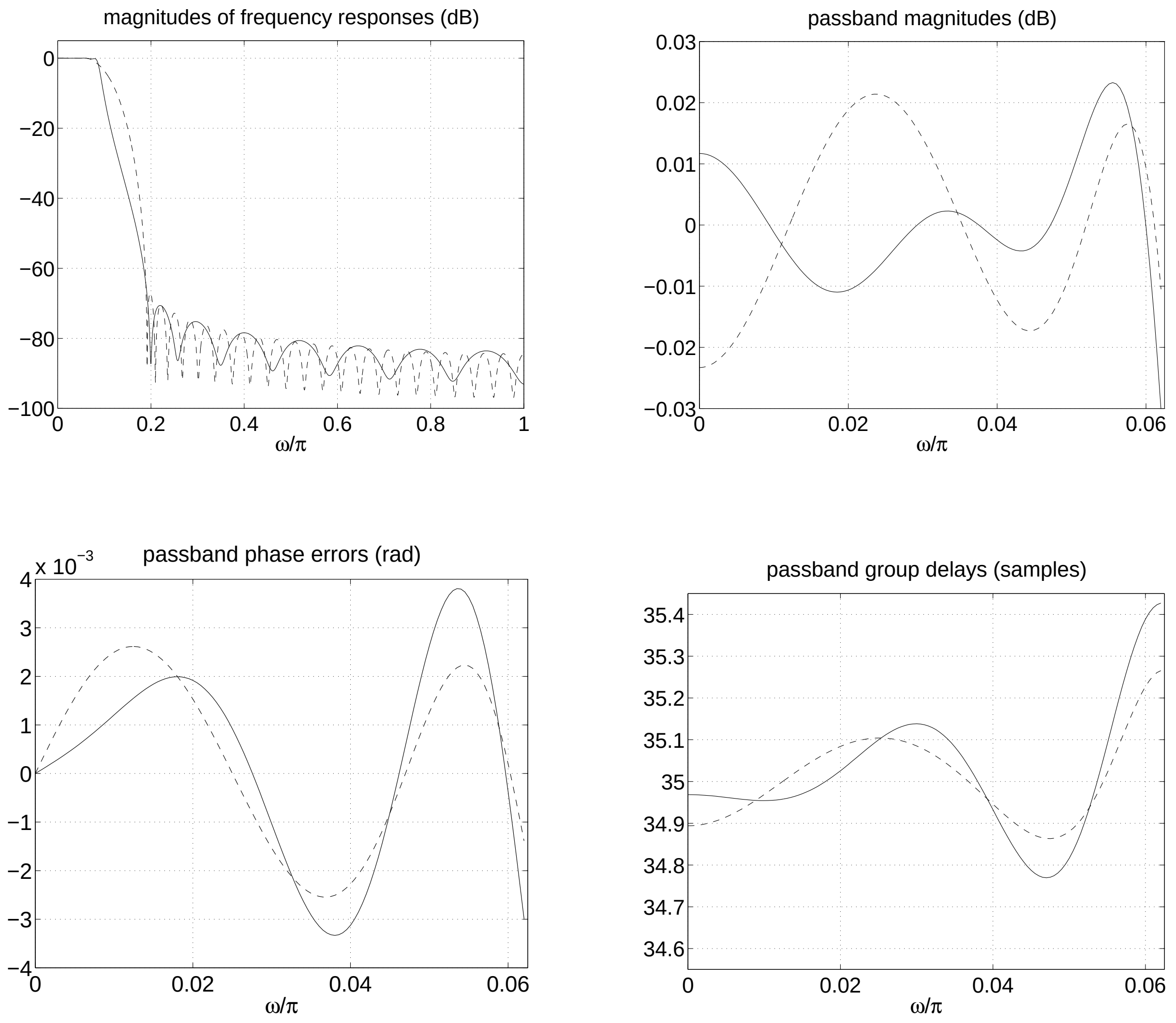


Figure 5.11: Digital equalization and anti-aliasing filters. Solid curves: analog prefilter cascaded with IIR equalizer ($M = 20$, $N = 4$). Dashed curves: analog prefilter cascaded with FIR equalizer of order 50 (see Example 2 in Chapter 4).

```
[b,a] = mpiir_l2(M,N,om,D,W,r);
```

This design took 20 seconds. Figure 5.11 shows the frequency responses of the cascades of analog prefilter and digital equalizers for the IIR equalizer (solid curves) as well as for the FIR equalizer presented in Chapter 4 (dashed curves). Obviously, the equalizers behave similarly in the passband and in the stopband with respect to magnitude and phase responses. Note, however, that the transition from the passband to the stopband is considerably steeper if the IIR equalizer is used.

The IIR equalizer can be implemented with 20 delays, 25 multipliers, and 24 adders. For implementing the equivalent FIR filter, 50 delays, 51 multipliers, and 50 adders are necessary.

Example 6: Comparison to a Cascade of Cauer Lowpass Filter and Phase Equalizer [29]:

In [29, 30], Deczky gave an example of an allpass filter equalizing the group delay of a Cauer (elliptic) lowpass filter. He designed the allpass filter by minimizing the L_p-norm of the group delay approximation error using the Fletcher-Powell algorithm. Not only the filter coefficients but also the total delay of the cascade were free parameters to optimize. The order of the Cauer filter is 4, and the allpass filter is of order 8. For this example, Deczky chose an L_{10}-norm such that the group delay approximation is close to a Chebyshev approximation. Due to the Cauer lowpass, the magnitude of the desired lowpass frequency response is approximated in a Chebyshev sense as well. Despite the different design criteria, we try to obtain a comparable filter by the design algorithm developed in this chapter. We choose $M = 10$ and $N = 6$. We choose the desired passband group delay as 8.5 samples which is less than the average passband group delay of 15.9 samples of the cascade. The maximum pole radius is chosen as $\rho = 0.92$. The following commands generate the required specification and design the filter by the program `mpiir_l2`:

```
om=pi*[linspace(0,.5,50),linspace(.6,1,40)];
D=.5*(1+10^(-.5/20))*[exp(-j*om(1:50)*8.5),zeros(1,40)];
W=[ones(1,50),10*ones(1,40)];
[b,a] = mpiir_l2(10,6,om,D,W,.92);
```

The desired frequency response $D\left(e^{j\omega}\right)$ has been scaled such that the designed filter approximates the same passband magnitude value as the Cauer filter whose magnitude response is upper bounded by 1. The design took 7 seconds.

Figure 5.12 shows the magnitude, phase, and group delay responses of the proposed filter and of the design given in [29]. The magnitude response of the proposed filter is better than the one of the Cauer filter not only with respect to a weighted least squares criterion but also with respect to a Chebyshev criterion. The passband phase response of the proposed filter is better than the one of the cascade on more than 90% of the passband. Note that the average passband group delay of the proposed filter is only about half as large as the one of the cascade.

Figure 5.13 shows the pole-zero diagrams of both filters. The plot on the left corresponds to the proposed filter. It achieves a better performance than the cascaded system using only half as many poles outside the origin of the complex plane. Due to the small number of non-trivial poles, fewer zeros outside the unit circle are necessary to compensate for the phase distortion. More zeros can be used for the stopband while still keeping the total number of zeros smaller than the number of zeros of the cascade (10 instead of

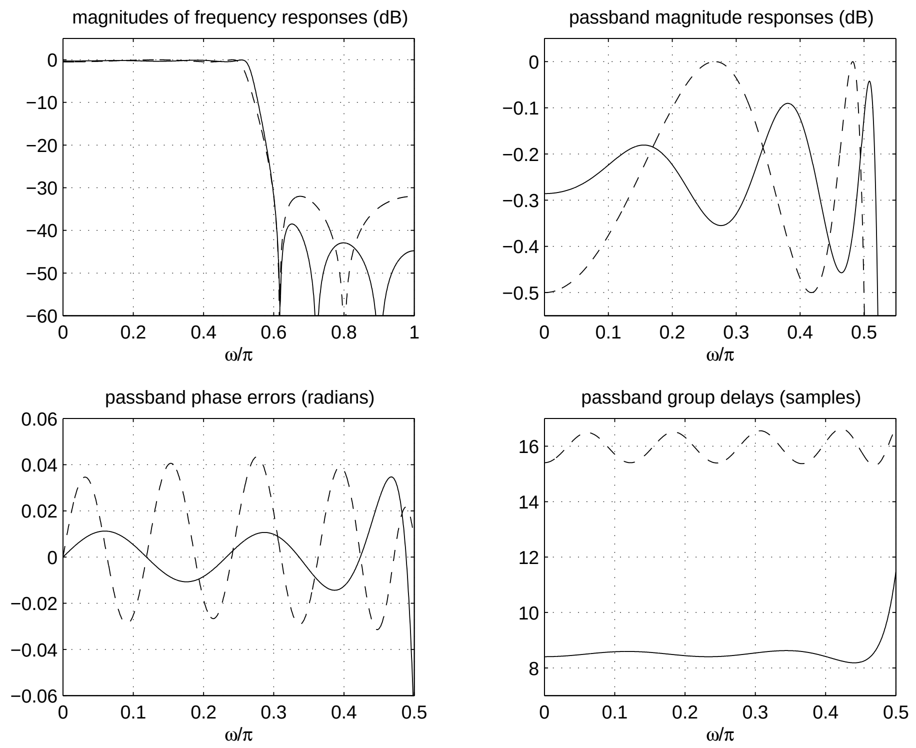


Figure 5.12: IIR lowpass filters with approximately linear passband phase responses. Solid curves: complex least squares design using the proposed algorithm ($M = 10$, $N = 6$). Dashed curves: cascade of Cauer filter ($M = N = 4$) and phase equalizer ($M = N = 8$) [29].

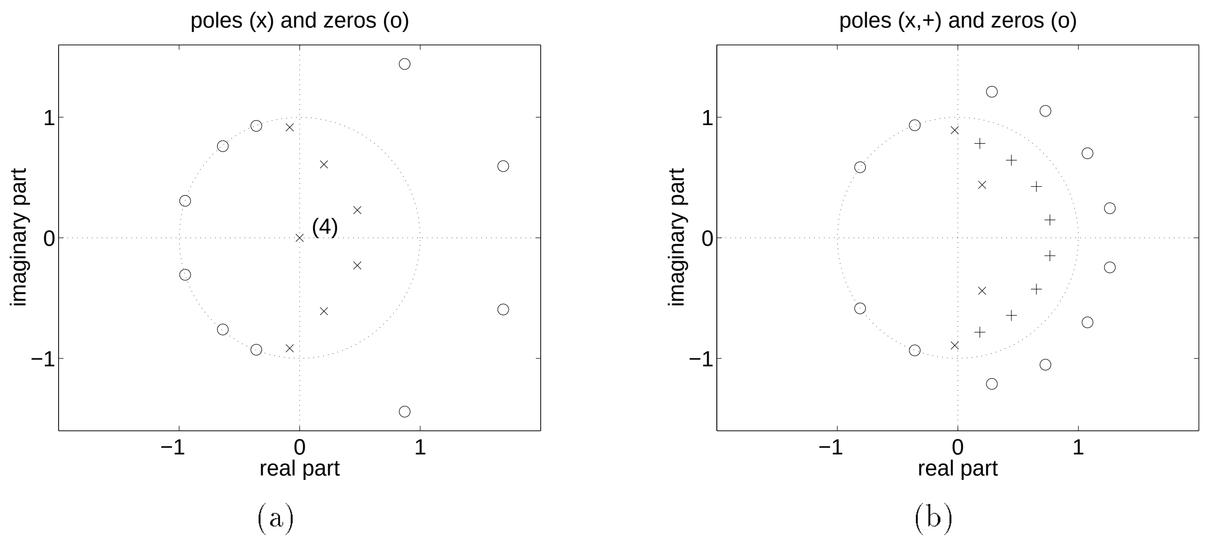


Figure 5.13: Pole-zero diagrams of IIR lowpass filters with approximately linear passband phase responses. (a) design by the proposed algorithm ($M = 10$, $N = 6$). (b) cascade of Cauer filter ($M = N = 4$) and phase equalizer ($M = N = 8$) [29]. The symbols $\times$ indicate the poles of the Cauer filter, and the symbols $+$ indicate the poles of the allpass equalizer. The zeros of the Cauer filter are on the unit circle, and the zeros of the allpass filter are outside the unit circle.

12).

A linear phase FIR filter with a mean squared approximation error that is similar to the one of the proposed IIR filter must be of order 30. Table 5.2 shows the complexities and the average values τ_{av} of the passband group delays of the three filter types solving the problem under consideration. The numbers of multiplications are valid if the implementations make use of all symmetries concerning the coefficients of the FIR filter, of the Cauer filter, and of the allpass filter. If the number of delays, the total number of additions and multiplications, and the average passband group delay are considered to be relevant quantities, the proposed IIR filter is superior to the other two filter implementations. From Figure 5.13, we see that the stopband zeros of the proposed filter lie virtually on the unit circle. Using this fact, the number of multiplications per output sample can be decreased from 17 to 14 which is shown in Table 5.2 by the value in parentheses.

In [98], a comparison between linear phase FIR filters and group delay equalized IIR filters is made. Similar to this design example, the authors of [98] used Cauer filters and designed the phase equalizers by Deczky's method given in [29]. The basis of the comparison exclusively was the number of multiplications required per output sample. Using this number as the only measure of complexity, we see from Table 5.2 that it is not worthwhile replacing the linear phase FIR filter by an equalized Cauer filter. Neither the complexity nor the passband group delay is decreased by a reasonable amount. Moreover, unlike the FIR filter, the cascade combination of the Cauer and the allpass filter has a phase error. A similar conclusion was drawn in [98]. If we still consider the number of multiplications exclusively, the proposed IIR filter might be preferable in situations where the group delay is desired to be as small as possible. However, in modern digital signal processors, additions and multiplications take the same time. Considering this fact and taking the number of delays into account as well, the equalized Cauer filter might be preferred over the FIR filter in some cases and the proposed IIR filter is preferable in most cases. Another important point that should be noted is the following. In [98], all design specifications can be satisfied by FIR filters of orders less than 45. This is also true for the specifications given in [29] which we used in the example given here. For more

filter type	delays	mults	adds	τ_{av} (samples)
proposed IIR	10	17 (14)	16	8.5
Cauer+EQ	12	15	24	15.9
FIR	30	16	30	16.0

Table 5.2: Complexity and average passband group delays of canonical implementations of equivalent lowpass filters with (approximately) linear passband phase responses. The number in parentheses is valid if the implementation makes use of stopband zeros on the unit circle.

demanding specifications, the relative efficiency of the IIR solution over the FIR solution is strongly increased. Example 4 in this section confirms this observation. There, an FIR filter of order 96 was equivalent to an IIR filter with $M = 20$ and $N = 8$. However, this effect does not occur in the case of cascades of Cauer filters and phase equalizing allpass filters. Due to the large group delay peak of high-order Cauer filters ($N > 8$), the order required for the equalizer becomes extremely large. Therefore, the cascade solution is not practical in these cases.

Example 7: Comparison to Coupled Allpass Filters:

Recursive digital filters implemented as a sum of two allpass filters have several desirable properties such as low passband sensitivity to coefficient quantization [137]. These properties are based on the fact that there are structurally lossless implementations of allpass filters where the allpass property is preserved in spite of multiplier quantization [85, 101]. The conditions for a transfer function of a digital filter to be implementable as a sum of two allpass filters have been given in [107]. An important class of filters satisfying these conditions are odd-order classical lowpass filters derived from the corresponding analog filters via the bilinear transformation.

If we restrict ourselves to the class of filters that can be implemented as a sum of two allpass filters, we can design these filters using methods for the design of allpass filters. A vast amount of such methods has been proposed, e. g. [35, 56, 54, 57, 116]. A special class of coupled allpass systems are those where one of the two parallel allpass filters is a cascade of delay elements. In the passbands of these systems, the non-trivial allpass filter approximates the linear phase of the delays. Hence, these systems have a passband phase response that is approximately linear. In the stopbands, a phase shift of π between the phase of the two allpass filters is approximated.

We design two lowpass filters with approximately linear passband phase responses. One design uses the coupled allpass structure where one of the allpass filters is a cascade of delays. The other allpass filter is designed by the method proposed in [56] where the desired phase response is approximated in a least squares sense.[2] The second design is computed by the algorithm presented in this chapter. We choose the passband and stopband edges as $\omega_p = 0.2\pi$ and $\omega_s = 0.23\pi$, respectively. The maximum passband ripple is required to be 1 dB, and the minimum stopband attenuation should be 50 dB. The mean squared approximation error is to be minimized. Since neither the proposed method nor the method for the design of the allpass filter in [56] can impose peak error constraints, some trials are necessary to satisfy the required magnitude constraints.

[2] I am indebted to Professor H. W. Schüßler who provided me with his Matlab implementation of the algorithm proposed in [56].

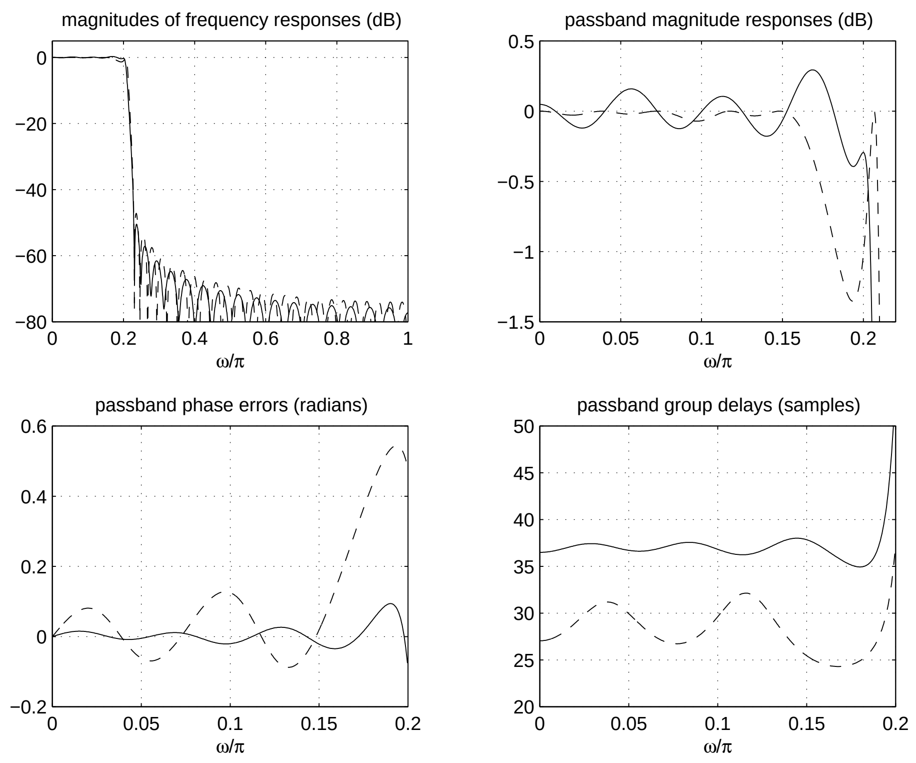


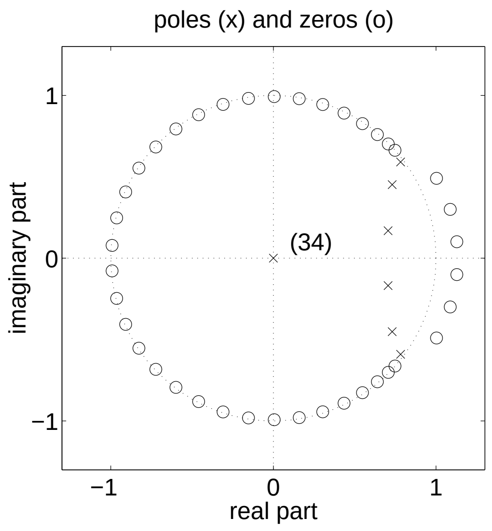
Figure 5.14: IIR lowpass filters with approximately linear passband phase responses. Solid curves: complex least squares design using the proposed algorithm ($M = 40$, $N = 6$). Dashed curves: allpass of order 30 coupled with a delay z^{-29}. The allpass was designed by the method proposed in [56].

In order to satisfy the specifications, an allpass filter of order $N_A = 30$ is required. This allpass filter is combined with a delay $z^{-(N_A-1)} = z^{-29}$. The transfer function of the total system which is the sum of the two allpass filters has a numerator of degree $M = 2N_A - 1$ and a denominator degree $N = N_A$. The passband phase that is approximated by the system is $-(N_A - 1)\omega$. For the design by the proposed algorithm, we choose $M = 40$ and $N = 6$. We choose the desired passband group delay as 37 samples and the maximum pole radius as $\rho = 0.98$. The following lines generate the specification and design the filter by the algorithm proposed in this chapter:

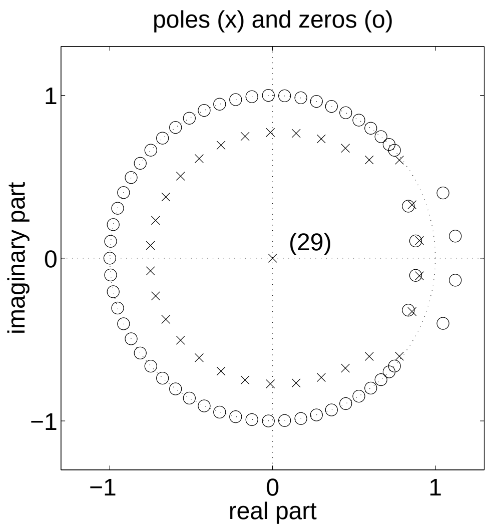
```
tau=37; r=.98;
om=pi*[linspace(0,.2,20),linspace(.23,1,77)];
D=[exp(-j*om(1:20)*tau),zeros(1,77)];
W=[ones(1,20),1000*ones(1,77)];
[b,a]=mpiir_l2(40,6,om,D,W,r);
```

This design took 19 seconds.

Figure 5.14 shows the design results for both filters. The solid curves correspond to the filter designed by the proposed algorithm, and the dashed curves correspond to



(a)



(b)

Figure 5.15: Pole-zero diagrams of IIR lowpass filters with approximately linear passband phase responses. (a) Design by the proposed algorithm ($M = 40$, $N = 6$). (b) allpass of order 30 coupled with a delay z^{-29}. The allpass was designed by the method proposed in [56].

the filter designed as a sum of two allpass filters. Note that the magnitude response of the allpass sum is upper bounded by 1, whereas the magnitude of the proposed filter approximates 1 in the passband. The magnitude response of the filter designed by the proposed algorithm is slightly better than the one of the allpass sum. The phase error of the allpass sum is considerably larger. If a smaller phase error is required, its passband magnitude error must be reduced at the cost of a reduced stopband attenuation. Note, however, that the average passband group delay of the allpass sum is smaller than the one of the proposed filter (29 samples compared to 37 samples).

Figure 5.15 shows the pole-zero diagrams of both filters. Figure 5.15 (a) shows that 6 zeros of the proposed filter are used to approximate the linear passband phase, whereas the remaining 34 zeros contribute to the stopband. The pole-zero diagram of the allpass sum is shown in Figure 5.15 (b). There are 30 non-trivial poles. Most of them lie approximately on a circle. There are two complex conjugate poles at the band edges. The radius of these poles is 0.985 which is larger than the maximum pole radius $\rho = 0.98$ of the proposed filter. Since the numerator polynomial of the allpass sum is a mirror-image polynomial [107, 116], the zeros of the allpass sum are either on the unit circle or symmetric to the unit circle. Note that the 4 zeros inside the unit circle almost cancel 4 poles.

From Figure 5.15, it seems that the poles and zeros of the allpass sum are not placed in an efficient way. Hence, using this structure exclusively for the design of transfer functions is not attractive. On the other hand, if the filter is not only designed but

filter type	delays	mults	adds	τ_{av} (samples)
proposed IIR	40	47 (30)	46	37.0
allpass sum	59	30	61	29.0
FIR	117	59	117	58.5

Table 5.3: Complexity and average passband group delays of canonical implementations of equivalent lowpass filters with (approximately) linear passband phase responses. The number in parentheses is valid if the implementation makes use of stopband zeros on the unit circle.

also implemented as an allpass sum, it becomes competitive to other designs regarding computational complexity. Table 5.3 shows the complexity and the average passband group delays of the general IIR filter designed by the proposed algorithm, of the filter designed and implemented as an allpass sum, and of a linear phase FIR filter with an equivalent performance. The degree of the FIR filter is 117. The numbers given in Table 5.3 refer to canonical implementations with a minimum number of multipliers which make use of the symmetry of the FIR filter coefficients and of the equivalence of the numerator and denominator coefficients of the allpass filter. This is also necessary to obtain a structurally lossless implementation of the allpass filter. For the general IIR filter designed by the proposed design algorithm, the number of multiplications in parentheses is valid if we make use of the fact that the stopband zeros lie virtually on the unit circle.

From Table 5.3, it can be seen that the FIR filter has the largest passband group delay and needs the maximum complexity for its canonical implementation. However, there is no phase error. The proposed IIR filter has the smallest complexity if the implementation takes into account that the stopband zeros lie on the unit circle. Its average passband group delay is larger than the one of the allpass sum. On the other hand, its phase error is considerably smaller than the phase error of the allpass sum (see Figure 5.14). We conclude that for this example, the proposed IIR filter is superior to the other two implementations of equivalent filters.

5.6 Matlab Programs

In this section, we provide Matlab programs implementing the design algorithm presented in Section 5.4. The program `mpiir_l2` implements the main algorithm and calls the program `update` which computes the optimum solution update based on Rouché's theorem in every iteration step. The program `update` implements the multiple exchange algorithm as described in Section 5.4.3. The program `mpiir_l2` returns the coefficients of the numerator and denominator polynomials of the optimum filter and stores them in the output vectors `b` and `a`, respectively. Additionally, it computes the mean squared approximation error. Its inputs are the desired numerator and denominator orders `M` and `N`, a frequency vector `om` specified on $[0, \pi]$ defining the grid on which the complex desired response vector `D` and the weighting function vector `W` are specified. The maximum pole radius is passed to `mpiir_l2` by the scalar `r`. An optional initial guess of the denominator polynomial coefficients can be supplied by the vector `a0`. This vector is discarded if its size does not match the specified denominator degree `N`, or if the zeros of the corresponding polynomial have radii larger than the specified value `r`.

```
function [b,a,l2error] = mpiir_l2(M,N,om,D,W,r,a0)
% MPIIR_L2: [b,a,l2error] = mpiir_l2(M,N,om,D,W,r,a0)
% Least Squares Digital IIR Filter Design with Arbitrary Magnitude
% and Phase Responses and Specified Maximum Pole Radius
%
% OUTPUT:
% b          numerator polynomial coefficients
% a          denominator polynomial coefficients
% l2error    approximation error
%
% INPUT:
% M          order of numerator polynomial
% N          order of denominator polynomial
% om         frequency grid, 0<=om<=pi
% D          complex desired frequency response on the grid om
% W          positive weighting function on the grid om
% r          maximum pole radius
% a0         initial denominator guess; optional
%
% EXAMPLE:
% Lowpass filter with approximately linear passband phase
% (passband group delay = 19 samples, maximum pole radius = 0.97)
% om=pi*[linspace(0,.2,20),linspace(.25,1,75)];
% D=[exp(-j*om(1:20)*19),zeros(1,75)];
% W=[ones(1,20),100*ones(1,75)];
% [b,a,e]=mpiir_l2(16,6,om,D,W,.97);
%
% Author: Mathias C. Lang, Vienna University of Technology, Oct. 98

om=om(:); D=D(:); W=W(:); srW = sqrt(W);
EM = exp(-j*om*(0:max([M,N]))); tol = 1e-4; alpha = 0.5;
disp('   ERROR    MAX.RADIUS    STEP       SLOPE');

% initial solution
ini=0;
if nargin == 7,
   a=a0(:);
   if length(a)~=N+1, ini=1;
   elseif a(1)==0, ini=1;
   elseif max(abs(roots(a))) > r, ini=1;
   else
```

```
      if a(1)~=1, a=a/a(1); end
      A = freqz(a,1,om); b = lslevin(M+1,om,A.*D,W./(abs(A).^2));
   end
else, ini=1;
end
if ini,     % compute FIR solution
   a = [1;zeros(N,1)]; b = lslevin(M+1,om,D,W);
end
x = [a(2:N+1);b]; delta = [];

% iterate (outer loop)
while 1,

   % compute complex error, Jacobian, and objective function value
   A = EM(:,1:N+1)*a; B = EM(:,1:M+1)*b; H = B./A;
   E = srW.*(D - H); l2error = E'*E; if N<1, break; end
   vec1 = srW./A; vec2 = -vec1.*H;
   J = [vec2(:,ones(1,N)).*EM(:,2:N+1),vec1(:,ones(1,M+1)).*EM(:,1:M+1)];

   % compute search direction
   delta0 = delta;
   [delta,how] = update(J,E,a,r,M,N);
   if ~strcmp(how,'ok'), delta=delta0; break; end

   % update solution
   x = x + alpha*delta; a = [1;x(1:N)]; b = x(N+1:M+N+1);

   % display results
   step = norm(delta)/norm(x); pr = max(abs(roots(a)));
   slope = -2*real(E'*J)*delta/l2error;
   disp(sprintf('\b \b %5.3e %5.3e %5.3e %5.3e',l2error,pr,step,slope))

   % check stopping criterion
   if (step < tol & abs(slope) < tol) | pr > r, break; end

end   % outer loop
```

```
function [delta,how] = update(J,E,a,r,M,N)
% subroutine for mpiir_l2.m (IIR filter design)
% computes solution update subject to constraint on pole radii;
% applies Rouche's Theorem
% J      Jacobian of actual frequency response
% E      complex error
% a      actual denominator coefficients
% r      maximum pole radius
% M      order of numerator polynomial
% N      order of denominator polynomial
%
% Author: Mathias C. Lang, Vienna University of Technology, Oct. 98
```

```
tol = 1e-6; se = sqrt(eps); itmax = 10; cnt = 0; how = 'ok';
nfft = 2^10; om = 2*pi*(0:nfft/2)'/nfft;
a = a.*(r.^-(0:N)'); A = abs(fft(a,nfft)); A = A(1:nfft/2+1); A=A(:);
ra = conv(a,a(N+1:-1:1)); ra = ra(N+1:2*N+1);
nra = (1:N)'.*ra(2:N+1); n2ra = (1:N)'.*nra;
R = r.^-(1:N); Bact=[]; cact=[];

% solve unconstrained problem
J = [real(J);imag(J)]; E = [real(E);imag(E)];
H=J'*J; H=H+se*eye(size(H)); f=J'*E; delta=H\f;

% compute matrices and vectors for qp-subproblems
H11=H(1:N,1:N); H12=H(1:N,N+1:M+N+1); H22=H(N+1:M+N+1,N+1:M+N+1);
Hh=H12/H22; H=H11-Hh*H12'; H=.5*(H+H');
f1=f(1:N); f2=f(N+1:M+N+1); f=Hh*f2-f1;

% iterate
while cnt<50,

   % compute error maxima on a grid
   d = delta(1:N); d = d.*(r.^-(1:N)');
   D = abs(fft(d,nfft)); D = D(1:nfft/2+1); D=D(:);
   Imax = locmax(D.^2-A.^2); ommax = om(Imax);
   if ~length(Imax), break; end

   % refine maxima using Newton's method
   if N>1,
      rd = conv(d,d(N:-1:1)); rd = rd(N:2*N-1);
      nrd = (1:N-1)'.*rd(2:N); n2rd = (1:N-1)'.*nrd;
   end
   for i=1:itmax,
      Mc = cos(ommax*(1:N)); Ms = sin(ommax*(1:N));
      gp = -Ms*nra; gpp = -Mc*n2ra;
      if N>1, gp = gp + Ms(:,1:N-1)*nrd; gpp = gpp + Mc(:,1:N-1)*n2rd; end
      Ipp = find(gpp); if ~length(Ipp), break; end
      ommax(Ipp) = ommax(Ipp)- gp(Ipp)./gpp(Ipp);
   end

   % find violating maxima
   Dmax = exp(-j*ommax*(1:N))*d; Amax = exp(-j*ommax*(0:N))*a;
   Iviol = find(abs(Dmax)>abs(Amax));
   omviol = ommax(Iviol); nviol = length(Iviol);
   Dviol = Dmax(Iviol); Aviol = Amax(Iviol);

   % check stopping criterion
   if ~nviol | all(abs(Dmax)-abs(Amax)<=max(abs(Amax)*tol,se)), break; end
   cnt = cnt+1;

   % formulate new constraints
```

```
        PDviol = angle(Dviol);
        B = R(ones(nviol,1),:).*cos(omviol*(1:N)+PDviol(:,ones(N,1)));
        c = abs(Aviol);

        % solve subproblem
        B=[Bact;B]; c=[cact;c];
        [delta,lam,how]=qp(H,f,B,c);
        if ~strcmp(how,'ok'), break; end

        % find active constraints
        act = find(lam>0); Bact = B(act,:); cact = c(act);

    end

    % add numerator coefficient update
    if length(delta)==N, delta=[delta;H22\f2-Hh'*delta]; end
```

5.7 Summary

In this chapter, we have presented a method for designing IIR digital filters satisfying specifications on their magnitude and phase responses according to a weighted least squared error criterion. The proposed method allows for a prescription of a maximum radius that must not be exceeded by the poles of the designed transfer function. This feature is not only useful to guarantee the stability of the designed filters, but can also be used to achieve a certain stability margin. Filters with poles having a prescribed distance from the unit circle can be implemented more easily using fixed-point arithmetic.

The proposed algorithm is based on the result of Rouché's theorem. Most optimization methods generate a sequence of parameter sets by adding appropriate updates to the actual parameters at every iteration step. Rouché's theorem states a condition on these updates such that the transfer function corresponding to the new set of parameters has the same number of zeros as the old transfer function inside a given contour in the complex plane. Using the circle $|z| = \rho$, $0 < \rho < 1$, as a contour, a condition that guarantees a specified stability margin results. This basic idea is fairly general and can be incorporated in all optimization methods that are based on additive updates of preliminary solutions. Consequently, this idea can be used regardless of the desired design criterion.

We have chosen a weighted least squared error design criterion which is useful for many applications. We have extended the Gauss-Newton algorithm by incorporating the constraint required by Rouché's theorem. This results in subproblems that are convex semi-infinite quadratic programming problems. Problems of this kind have already been investigated in Chapter 2 (Section 2.3). We have proposed algorithms for solving these problems in Chapter 3. The semi-infinite property of the subproblems is caused by the requirements of Rouché's theorem that guarantee a prescribed stability margin. The FIR filter design problems considered in Chapters 2 and 3 were semi-infinite because we required constraints on the filter's frequency response to be satisfied on continuous frequency intervals. In this case, replacing the frequency intervals by dense frequency grids does not make much difference if tenths of a dB are considered to be negligible. On the other hand, the semi-infinite constraint imposed by requiring a certain stability margin cannot be replaced by a finite number of constraints because then, the pole radii may attain arbitrary values. The corresponding filters will be unstable in many cases and are useless. Therefore, in this case it is really necessary to take the semi-infinite nature of the subproblems into account. The only algorithm proposed in Chapter 3 that can handle semi-infinite constraints exactly is the multiple exchange algorithm presented in Section 3.2.1. The complete IIR design algorithm uses the Gauss-Newton iteration with modified subproblems that are solved by the multiple exchange algorithm of Section 3.2.1. Due to the special structure of the subproblems it is possible to reduce the number of variables down to the specified degree of the denominator polynomial. The total number of variables in the subproblems does not depend on the numerator degree.

This is a desirable property because we have shown that for IIR filter design problems with magnitude and phase specifications and a prescribed maximum pole radius, it does not make sense to choose a high denominator degree. For all design problems we have examined, the approximation error does not considerably decrease if the denominator degree is increased above values of eight to ten. For higher denominator degrees, pole-zero cancellations are more likely to occur, and poles are placed more often at angles corresponding to stopbands. If neither of these effects occur, the poles usually crowd at angles corresponding to the transition bands and have radii close or equal to the specified maximum radius. This crowding of poles is an undesirable behavior because the sensitivity of the pole locations with respect to coefficient quantization may become very large. Moreover, filters with crowded poles usually have more serious quantization noise problems.

In Section 5.5, we have compared the proposed approach to several other methods published previously. It has turned out that the proposed algorithm outperforms most other methods. This is partly due to the incorporation of Rouché's theorem instead of other sufficient stability constraints that are much more restrictive (e. g. the linear constraint (5.3)). Another reason for its good performance is the fact that we use polynomial coefficients as design parameters instead of coefficients of second order factors or pole radii and pole angles. This makes it easier to find a satisfactory local solution to the design problem, even if the initial guess is bad. We typically use the FIR solution to the design problem as initial guess. This solution is usually far away from a good local solution. Most other parameterizations of the design problem require a more sophisticated initial guess to find a good solution.

In Section 5.6, we have provided the Matlab program `mpiir_l2` implementing the proposed design algorithm. The computational burden mainly depends on the desired denominator degree. The program is reasonably fast for denominator degrees $N \leq 10$. It uses the subroutines `update`, `lslevin`, `levin`, and `locmax` provided in this thesis, and the program `qp` contained in the Matlab Optimization Toolbox.

Chapter 6

Conclusions and Open Problems

6.1 Conclusions

In this dissertation, we have presented several new algorithms for designing FIR and IIR digital filters. The proposed algorithms can solve design problems where simultaneous magnitude and phase approximation is desired. Most standard methods concentrate only on the magnitude (minimum phase or linear phase filters), or only on the phase or group delay (allpass filters). For the design of FIR filters, we have used complex least squares and complex Chebyshev optimality criteria. In addition, we have incorporated peak constraints on magnitude and phase approximation errors. This allows to design a broad range of filters that cannot be designed using standard methods. We have shown that the proposed filters are better suited to many practical problems than approximations according to conventional criteria. In the case of IIR filters, we have used the complex least squares optimality criterion in combination with a constraint on the pole radii of the designed transfer function. This constraint is important to guarantee the stability of the designed filter. Moreover, the fact that the minimum distance of the poles to the unit circle can be prescribed is advantageous for implementing the designed filter using fixed-point arithmetic. Poles very close to the unit circle are very sensitive to coefficient quantization, may increase the maximum amplitude of small scale limit cycles, and may enhance quantization noise considerably. All these undesirable effects can be controlled by bounding the maximum pole radius by a reasonable value. We provide computationally efficient Matlab programs implementing all design algorithms presented in this thesis. We consider this to be an important issue because only a good implementation makes an algorithm useful in practice.

6.2 Open Problems

There remain several open problems related to the work we have presented in this thesis. At this point, we shall discuss two important problems that seem especially interesting.

Throughout this thesis we have considered the approximation of a fixed complex desired frequency response $D(e^{j\omega})$. However, in several applications the total delay of the designed system is of minor interest if only the approximation error is small. Therefore, it would be useful to specify a desired frequency response with a variable delay. The algorithm should find the optimum delay and the corresponding optimum filter coefficients such that the approximation error is minimized. This approach has been investigated for the design of allpass filters in [56]. For the general IIR filter case, it seems that no satisfactory solution to this problem has been found yet.

In Chapter 5 we have considered the complex approximation problem for IIR filters. Due to the nonlinear dependence of the filter coefficients on the transfer function, this problem is considerably harder to solve than corresponding FIR filter design problems. The stability problem poses an additional difficulty. For the design of IIR filters, we have only considered a least squared error criterion with constraints on the pole radii. It would, however, be interesting to use more sophisticated criteria incorporating frequency response constraints as discussed in Chapter 2. This has been tried e. g. in [74]. We have shown in Example 2 of Section 5.5 that the filter proposed in [74] is considerably worse than the solution computed by the proposed design algorithm without frequency response constraints. Apparently, the problem of designing stable IIR filters with arbitrary magnitude and phase responses subject to frequency response constraints has not been solved satisfactorily yet.

Appendix A

Convex Programming

This appendix provides some basic results of optimization theory (mathematical programming) that are necessary for understanding the concepts and algorithms presented in this thesis. After formulating the optimization problem and introducing some important terms, we specialize to convex programming problems. The discussion of Lagrange multipliers and of the Lagrangian function leads to the important concept of duality. It provides an alternative formulation of a convex programming problem which might be computationally easier to solve. An introduction to a certain class of algorithms solving linearly constrained programming problems concludes this section.

A.1 Introduction

The optimization problems discussed here have the following structure:

$$\begin{array}{ll} \underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{minimize}} & f(\boldsymbol{x}) \\ \text{subject to} & c_i(\boldsymbol{x}) \leq 0, \quad i \in I, \\ & c_i(\boldsymbol{x}) = 0, \quad i \in E. \end{array} \tag{A.1}$$

The function $f(\boldsymbol{x})$ is the *objective function* and the functions $c_i(\boldsymbol{x})$, $i = 1, 2, \ldots, m$, are the *constraint functions*. The sets I and E are the sets of inequality constraints and equality constraints, respectively. The constraints define the *feasible region* $K = \{\boldsymbol{x} \in \mathbb{R}^n |\ c_i(\boldsymbol{x}) \leq 0,\ i \in I;\ c_i(\boldsymbol{x}) = 0,\ i \in E\}$ on which the objective function is minimized. A point $\boldsymbol{x}'$ that satisfies all constraints is said to be *feasible*. Otherwise the point is *infeasible*. An *infeasible* problem is a problem with inconsistent constraints such that the feasible region K is empty. At a feasible point $\boldsymbol{x}'$ the inequality constraint $c_i(\boldsymbol{x}')$, $i \in I$, is said to be *active* if $c_i(\boldsymbol{x}') = 0$. Otherwise it is termed *inactive*.

Define the gradient vector of the objective function by

$$\boldsymbol{g}(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \left[\frac{\partial f(\boldsymbol{x})}{\partial x_1}, \ldots, \frac{\partial f(\boldsymbol{x})}{\partial x_n}\right]^T,$$

and define the Hessian matrix of the objective function by

$$\boldsymbol{H}(\boldsymbol{x}) = \left[\frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial x_1}, \ldots, \frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial x_n}\right].$$

The Jacobian matrix consists of the gradients of the constraint functions:

$$\boldsymbol{A}(\boldsymbol{x}) = \left[\frac{\partial c_1(\boldsymbol{x})}{\partial \boldsymbol{x}}, \ldots, \frac{\partial c_m(\boldsymbol{x})}{\partial \boldsymbol{x}}\right].$$

The columns $\boldsymbol{a}_i(\boldsymbol{x})$ of $\boldsymbol{A}(\boldsymbol{x})$ are orthogonal to the contours $c_i(\boldsymbol{x}) = \mathit{const}$. Those vectors $\boldsymbol{a}_i(\boldsymbol{x})$ associated with inequality constraints $c_i(\boldsymbol{x}) \leq 0$, $i \in I$, point to the outside of the feasible region since they are the directions of greatest increase of $c_i(\boldsymbol{x})$.

A.2 Lagrange Multipliers and Kuhn-Tucker Conditions

This section introduces the concept of Lagrange multipliers and derives necessary conditions for a point $\boldsymbol{x}_o$ to be a local solution of problem (A.1).

Assume $I = \emptyset$, i. e. there are only equality constraints. Let $\boldsymbol{x}_o \in K$ be a local solution to problem (A.1). A feasible direction[1] $\boldsymbol{s}$ at the point $\boldsymbol{x}_o$ must satisfy the condition

$$\boldsymbol{s}^T \boldsymbol{A}(\boldsymbol{x}_o) = \boldsymbol{0}. \tag{A.2}$$

In general, condition (A.2) is necessary but not sufficient for $\boldsymbol{s}$ to be a feasible direction. It becomes sufficient if an assumption referred to as *constraint qualification* (see e. g. [34]) is satisfied at $\boldsymbol{x}_o$. In practical situations this assumption is usually valid. Sufficient conditions for this assumption to be valid are e. g. the linearity of the constraint functions $c_i(\boldsymbol{x})$ or a full rank of the Jacobian matrix $\boldsymbol{A}(\boldsymbol{x}_o)$. For all further discussions we assume the constraint qualification to be satisfied, i. e. condition (A.2) is necessary and sufficient for $\boldsymbol{s}$ to be a feasible direction.

By Taylor's formula

$$f(\boldsymbol{x}_o + \alpha \boldsymbol{s}) = f(\boldsymbol{x}_o) + \alpha \boldsymbol{s}^T \boldsymbol{g}(\boldsymbol{x}_o) + \tfrac{1}{2}\alpha^2 \boldsymbol{s}^T \boldsymbol{H}(\boldsymbol{x}_o + \eta\alpha\boldsymbol{s})\boldsymbol{s}, \quad 0 < \eta < 1, \tag{A.3}$$

[1]A feasible direction $\boldsymbol{s}$ at the point $\boldsymbol{x}_o \in K$ is defined by a limit process corresponding to the notion that an incremental step from $\boldsymbol{x}_o$ along $\boldsymbol{s}$ yields another feasible point [34].

for any real-valued scalar α. For $\boldsymbol{x}_o$ to be a stationary point, the condition

$$\boldsymbol{s}^T \boldsymbol{g}(\boldsymbol{x}_o) = 0 \tag{A.4}$$

must be satisfied. In the unconstrained case condition (A.4) must be satisfied for all $\boldsymbol{s}$, and hence $\boldsymbol{g}(\boldsymbol{x}_o) = \mathbf{0}$ follows. However, in the constrained case, condition (A.4) need only be satisfied for all feasible directions $\boldsymbol{s}$. This is the case if the gradient vector at $\boldsymbol{x}_o$ is a linear combination of the gradients of the constraint functions at $\boldsymbol{x}_o$:

$$\boldsymbol{g}(\boldsymbol{x}_o) + \boldsymbol{A}(\boldsymbol{x}_o)\boldsymbol{\lambda}_o = \mathbf{0}. \tag{A.5}$$

Then, according to (A.2), feasible directions $\boldsymbol{s}$ satisfy

$$\boldsymbol{s}^T \boldsymbol{g}(\boldsymbol{x}_o) = -\boldsymbol{s}^T \boldsymbol{A}(\boldsymbol{x}_o)\boldsymbol{\lambda}_o = 0.$$

The elements of the vector $\boldsymbol{\lambda}_o$ are called *Lagrange multipliers*. Each multiplier is associated with a constraint function. The necessary optimality conditions for the equality constrained optimization problem can be written in a very compact form using the *Lagrangian function*

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{c}(\boldsymbol{x}), \tag{A.6}$$

where the column vector $\boldsymbol{c}(\boldsymbol{x})$ contains all constraint functions $c_i(\boldsymbol{x})$, $i \in E$. The necessary condition (A.5) for a constrained stationary point $\boldsymbol{x}_o$ and the feasibility condition $\boldsymbol{c}(\boldsymbol{x}_o) = \mathbf{0}$ can be expressed as

$$\frac{\partial L}{\partial \boldsymbol{x}} = \mathbf{0}, \quad \frac{\partial L}{\partial \boldsymbol{\lambda}} = \mathbf{0}. \tag{A.7}$$

Note that a vector of Lagrange multipliers $\boldsymbol{\lambda}_o$ is associated with every point $\boldsymbol{x}_o$ satisfying (A.7). This vector is unique if the Jacobian matrix $\boldsymbol{A}(\boldsymbol{x}_o)$ has full rank. From (A.7) it follows that a necessary condition for a local optimum is that the point $(\boldsymbol{x}_o, \boldsymbol{\lambda}_o)$ is a stationary point of the Lagrangian function.

The *method of Lagrange multipliers* for equality constrained problems directly solves the equations (A.7). However, in general the system (A.7) is nonlinear and moreover, it represents only a necessary condition for optimality.

Most commercial optimization codes not only provide the optimum point $\boldsymbol{x}_o$ but also the Lagrange multipliers $\boldsymbol{\lambda}_o$ associated with it. So far it seems that the Lagrange multipliers are only an unnecessary side product. However, they have an important interpretation. Consider the perturbed equality constrained problem with constraints $\boldsymbol{c}(\boldsymbol{x}) + \boldsymbol{\epsilon} = \mathbf{0}$. The solution and its multipliers are now functions of the perturbation vector $\boldsymbol{\epsilon}$, i. e. $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\epsilon})$ and $\boldsymbol{\lambda} = \boldsymbol{\lambda}(\boldsymbol{\epsilon})$. The Lagrangian function of the perturbed problem is

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\epsilon}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \left[\boldsymbol{c}(\boldsymbol{x}) + \boldsymbol{\epsilon}\right].$$

The change of the objective function $f(\boldsymbol{x}(\boldsymbol{\epsilon}))$ with respect to the perturbation $\boldsymbol{\epsilon}$ is given by

$$\frac{df}{d\boldsymbol{\epsilon}} = \frac{dL}{d\boldsymbol{\epsilon}} = \frac{\partial \boldsymbol{x}^T}{\partial \boldsymbol{\epsilon}}\frac{\partial L}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{\lambda}^T}{\partial \boldsymbol{\epsilon}}\frac{\partial L}{\partial \boldsymbol{\lambda}} + \frac{\partial L}{\partial \boldsymbol{\epsilon}} = \frac{\partial L}{\partial \boldsymbol{\epsilon}} = \boldsymbol{\lambda}, \tag{A.8}$$

where the first equality follows from the fact that $\boldsymbol{x}(\boldsymbol{\epsilon})$ is optimum and hence necessarily feasible, and the second but last equality follows from condition (A.7). Hence, each Lagrange multiplier determines the sensitivity of the objective function with respect to changes of the corresponding constraint.

Let us now consider the case $E = \emptyset$, i. e. only inequality constraints are present in problem (A.1). It is important to note that at a solution $\boldsymbol{x}_o$ only the active constraints influence all possible feasible directions at that point. Let $\tilde{\boldsymbol{A}}(\boldsymbol{x}_o)$ be the part of the Jacobian matrix that contains only the gradients of the active constraints at the point $\boldsymbol{x}_o$. Assuming constraint qualification, the feasible directions are defined by

$$\boldsymbol{s}^T\tilde{\boldsymbol{A}}(\boldsymbol{x}_o) \leq \boldsymbol{0}. \tag{A.9}$$

For $\boldsymbol{x}_o$ to be a local minimum, moving from $\boldsymbol{x}_o$ along any feasible direction $\boldsymbol{s}$ must not decrease the objective function $f(\boldsymbol{x})$, i. e.

$$\boldsymbol{s}^T\boldsymbol{g}(\boldsymbol{x}_o) \geq 0 \tag{A.10}$$

is required for all feasible directions $\boldsymbol{s}$. Condition (A.10) is satisfied if

$$\boldsymbol{g}(\boldsymbol{x}_o) + \tilde{\boldsymbol{A}}(\boldsymbol{x}_o)\boldsymbol{\lambda}_o = \boldsymbol{0}, \quad \boldsymbol{\lambda}_o \geq \boldsymbol{0}, \tag{A.11}$$

because in this case

$$\boldsymbol{s}^T\boldsymbol{g}(\boldsymbol{x}_o) = -\boldsymbol{s}^T\tilde{\boldsymbol{A}}(\boldsymbol{x}_o)\boldsymbol{\lambda}_o \geq 0$$

holds for all feasible directions $\boldsymbol{s}$. Sufficiency of condition (A.11) for (A.10) to hold is obvious. The fact that (A.11) is also necessary results from Farkas' lemma (see e. g. [34]).

Figure A.1 illustrates condition (A.11) for a two-dimensional problem with three linear inequality constraints. The point $\boldsymbol{x}_u$ is the unconstrained minimum of the objective function $f(\boldsymbol{x})$. However, $\boldsymbol{x}_u$ is infeasible. The point $\boldsymbol{x}'$ cannot be a local optimum since the gradient $\boldsymbol{g}(\boldsymbol{x}')$ is not collinear with the gradient $\boldsymbol{a}_3$ of the active constraint. The gradient $\boldsymbol{g}(\boldsymbol{x}'')$ at $\boldsymbol{x}''$ satisfies $\boldsymbol{g}(\boldsymbol{x}'') + \lambda\boldsymbol{a}_2 = \boldsymbol{0}$, but $\lambda < 0$ and hence, the point $\boldsymbol{x}''$ cannot be a local minimizer. The optimum point $\boldsymbol{x}_o$ satisfies $\boldsymbol{g}(\boldsymbol{x}_o) + \lambda\boldsymbol{a}_1 = \boldsymbol{0}$ with $\lambda > 0$.

If all inactive constraints are associated with zero Lagrange multipliers, condition (A.11) and the feasibility condition can be formulated using the Lagrangian function (A.6):

$$\frac{\partial L}{\partial \boldsymbol{x}} = \boldsymbol{0}, \qquad \frac{\partial L}{\partial \boldsymbol{\lambda}} \leq \boldsymbol{0}, \qquad \boldsymbol{\lambda} \geq \boldsymbol{0}, \qquad \boldsymbol{\lambda}^T\frac{\partial L}{\partial \boldsymbol{\lambda}} = 0. \tag{A.12}$$

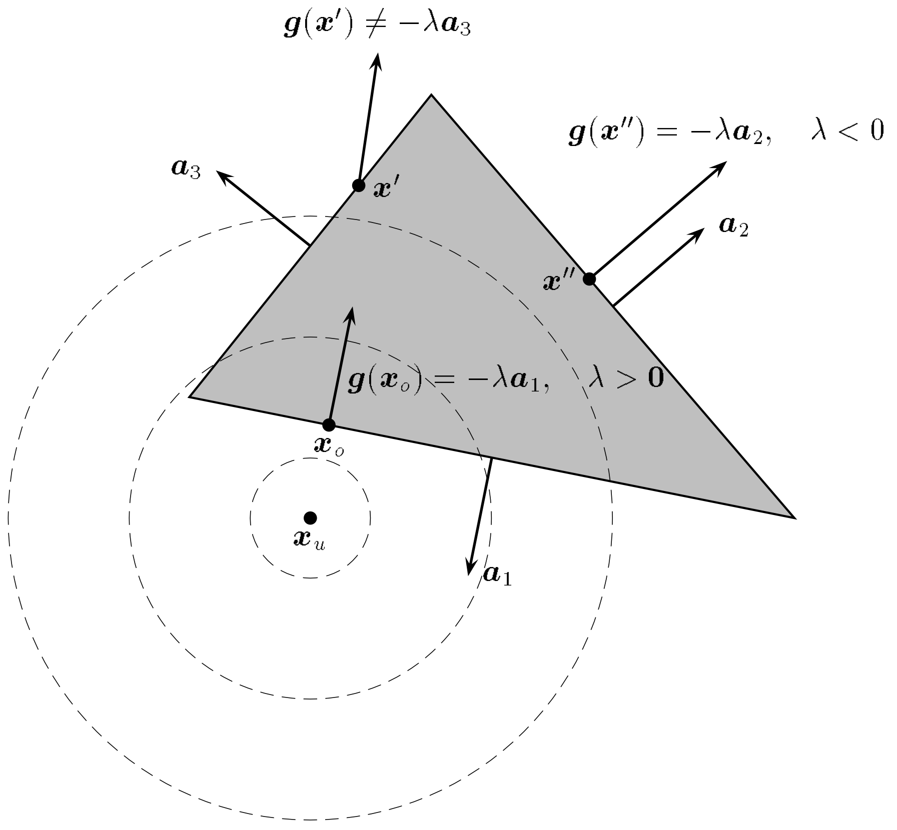

$\boldsymbol{g}(\boldsymbol{x}') \neq -\lambda \boldsymbol{a}_3$
$\boldsymbol{g}(\boldsymbol{x}'') = -\lambda \boldsymbol{a}_2, \quad \lambda < 0$
$\boldsymbol{a}_3$
$\boldsymbol{x}'$
$\boldsymbol{a}_2$
$\boldsymbol{x}''$
$\boldsymbol{g}(\boldsymbol{x}_o) = -\lambda \boldsymbol{a}_1, \quad \lambda > 0$
$\boldsymbol{x}_o$
$\boldsymbol{x}_u$
$\boldsymbol{a}_1$

Figure A.1: Optimization problem with linear constraints. Dashed circles: contours of the objective function $f(\boldsymbol{x})$. Solid lines: borders of the feasible region (shaded) defined by linear inequality constraints. The point $\boldsymbol{x}_u$ is the unconstrained minimum of $f(\boldsymbol{x})$, the feasible points $\boldsymbol{x}'$ and $\boldsymbol{x}''$ do not satisfy the optimality condition (A.11), and the point $\boldsymbol{x}_o$ is the optimum solution. The vectors $\boldsymbol{a}_i$, $1 \leq i \leq 3$, are the gradients of the constraint functions, and $\boldsymbol{g}(\boldsymbol{x})$ is the gradient of $f(\boldsymbol{x})$.

The additional requirement $\boldsymbol{\lambda}^T \partial L / \partial \boldsymbol{\lambda} = 0$ in (A.12) guarantees that inactive constraints are associated with zero Lagrange multipliers. The conditions (A.12) are referred to as *Kuhn-Tucker conditions.* Under a regularity assumption[2] these conditions are necessary for a point to be a local minimizer subject to inequality constraints. Note that each equality constraint $c_i(\boldsymbol{x}) = 0$, $i \in E$, can be written as $c_i(\boldsymbol{x}) \leq 0$ and $-c_i(\boldsymbol{x}) \leq 0$. Hence, conditions (A.12) include the case $I \neq 0$ and $E \neq 0$. However, in practice equality constraints are handled separately by considering them as active constraints at every feasible point. If equality constraints are handled separately, the Kuhn-Tucker conditions (A.12) remain unchanged except for the fact that there are no sign restrictions

[2]The regularity property is weaker than the constraint qualification. Hence, since we assume the constraint qualification assumption to be valid, regularity is implied. See e. g. [34] for more details.

for Lagrange multipliers associated with equality constraints.

So far we have derived necessary optimality conditions by considering first derivatives. By considering the Hessian of the Lagrangian function, sufficient conditions for local solutions can be derived [34]. However, we will show that for convex programming problems first order conditions are sufficient.

A.3 Convexity

Convexity of a mathematical programming problem is a very desirable property since it implies the globality of solutions and the sufficiency of first order optimality conditions.

A set K in $\mathbb{R}^n$ is convex if for $\boldsymbol{x}_1, \boldsymbol{x}_2 \in K$ $\boldsymbol{x}(\alpha) \in K$ follows, where

$$\boldsymbol{x}(\alpha) = \alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2, \quad \forall \alpha \in [0,1]. \tag{A.13}$$

Examples of convex sets are hyperplanes $\boldsymbol{a}^T\boldsymbol{x} = b$ and half spaces $\boldsymbol{a}^T\boldsymbol{x} \leq b$ defined by linear equalities and inequalities, respectively. Note that any intersection of convex sets is a convex set.

A continuous functions $f(\boldsymbol{x})$ defined on a convex set K is convex if for $\boldsymbol{x}_1, \boldsymbol{x}_2 \in K$

$$f(\boldsymbol{x}(\alpha)) \leq \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2), \quad \forall \alpha \in [0,1] \tag{A.14}$$

follows, where $\boldsymbol{x}(\alpha)$ is defined by (A.13). Assuming differentiability of $f(\boldsymbol{x})$, the condition

$$f(\boldsymbol{x}_2) \geq f(\boldsymbol{x}_1) + (\boldsymbol{x}_2 - \boldsymbol{x}_1)^T \boldsymbol{g}(\boldsymbol{x}_1) \tag{A.15}$$

is equivalent to condition (A.14). Another equivalent condition for twice differentiable functions is

$$\boldsymbol{y}^T \boldsymbol{H}(\boldsymbol{x}) \boldsymbol{y} \geq 0, \quad \forall \boldsymbol{x} \in K, \quad \forall \boldsymbol{y} \in \mathbb{R}^n \tag{A.16}$$

where $\boldsymbol{H}(\boldsymbol{x})$ is the Hessian matrix of $f(\boldsymbol{x})$. For a function $f(\boldsymbol{x})$ to be *strictly convex*, the inequalities in conditions (A.14) and (A.15) must be strict for $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ and $\alpha \in (0,1)$. A positive definite Hessian matrix $\boldsymbol{H}(\boldsymbol{x})$ is sufficient but in general not necessary for $f(\boldsymbol{x})$ to be strictly convex. However, quadratic functions $f(\boldsymbol{x})$ are strictly convex if and only if their Hessian matrix $\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{H}$ is positive definite.

Note that a weighted sum of convex functions $f_i(\boldsymbol{x})$, $i = 1, 2, \ldots, m$, defined on a convex set K is a convex function on K if the weights are non-negative, i. e.

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}), \quad \lambda_i \geq 0, \quad i = 1, 2, \ldots, m, \tag{A.17}$$

is convex.

An important relationship between convex sets and convex functions is the following. If $c(\boldsymbol{x})$ is a convex function on $\mathbb{R}^n$ then the set $K = \{\boldsymbol{x} | \ c(\boldsymbol{x}) \leq 0\}$ is convex. Moreover, since the intersection of convex sets is a convex set, also the set $K = \{\boldsymbol{x} | \ \boldsymbol{c}(\boldsymbol{x}) \leq 0\}$ is convex, where $\boldsymbol{c}(\boldsymbol{x})$ is a vector of convex functions. Note that convexity of $K = \{\boldsymbol{x} | \ c(\boldsymbol{x}) \leq 0\}$ does not imply convexity of the function $c(\boldsymbol{x})$. However, if K is non-convex, $c(\boldsymbol{x})$ must be a non-convex function.

Problem (A.1) is a *convex programming problem* if $f(\boldsymbol{x})$ and $c_i(\boldsymbol{x})$, $i \in I$, are convex functions. Since an equality constraint $c(\boldsymbol{x}) = 0$ is equivalent to $c(\boldsymbol{x}) \leq 0$ and $-c(\boldsymbol{x}) \leq 0$, the functions $c_i(\boldsymbol{x})$, $i \in E$, must be linear (affine) for problem (A.1) to be convex.

An important property of convex programming problems is that every local solution is a global solution. This can be seen as follows. Assume $\boldsymbol{x}_1$ is a local but not a global solution to problem (A.1), i. e. there is a point $\boldsymbol{x}_2 \in K$ such that $f(\boldsymbol{x}_2) < f(\boldsymbol{x}_1)$. By convexity we have $f(\boldsymbol{x}(\alpha)) \leq \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2) < f(\boldsymbol{x}_1)$, $\forall \alpha \in [0, 1)$. This implies that there is no neighborhood of $\boldsymbol{x}_1$ in which $f(\boldsymbol{x}) \geq f(\boldsymbol{x}_1)$ holds. This contradicts the local solution property of $\boldsymbol{x}_1$.

Let now $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ be global solutions. This fact and convexity of $f(\boldsymbol{x})$ imply $f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2) \leq f(\boldsymbol{x}(\alpha)) \leq \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2) = f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2)$, $\forall \alpha \in [0, 1]$. Hence, every $\boldsymbol{x}(\alpha)$, $\alpha \in [0, 1]$ is also a global solution and the set of global solutions is convex.

If the objective function $f(\boldsymbol{x})$ is strictly convex on the feasible region K, then the global solution is unique. This is also shown by contradiction. Assume $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are global solutions ($\boldsymbol{x}_1 \neq \boldsymbol{x}_2$). Then from strict convexity of $f(\boldsymbol{x})$

$$f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2) \leq f(\boldsymbol{x}(\alpha)) < \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2) = f(\boldsymbol{x}_1) = f(\boldsymbol{x}_2)$$

follows, which contradicts the assumption that there are two different global solutions if $f(\boldsymbol{x})$ is strictly convex.

Finally, we show the sufficiency of the Kuhn-Tucker optimality conditions (A.12) for convex programming problems. Let $\boldsymbol{x}_o$ be an optimum solution and let $\boldsymbol{\lambda}_o$ be its associated vector of Lagrange multipliers. For any feasible $\boldsymbol{x}$ the Kuhn-Tucker conditions imply

$$\begin{aligned} f(\boldsymbol{x}) &\geq f(\boldsymbol{x}) + \boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}) \\ &\geq f(\boldsymbol{x}_o) + (\boldsymbol{x} - \boldsymbol{x}_o)^T \boldsymbol{g}(\boldsymbol{x}_o) + \boldsymbol{\lambda}_o^T \left[\boldsymbol{c}(\boldsymbol{x}_o) + \boldsymbol{A}^T(\boldsymbol{x}_o)(\boldsymbol{x} - \boldsymbol{x}_o)\right] \\ &= f(\boldsymbol{x}_o) + (\boldsymbol{x} - \boldsymbol{x}_o)^T \left[\boldsymbol{g}(\boldsymbol{x}_o) + \boldsymbol{A}(\boldsymbol{x}_o)\boldsymbol{\lambda}_o\right] + \boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}_o) \\ &= f(\boldsymbol{x}_o), \quad \boldsymbol{x} \in K. \end{aligned} \tag{A.18}$$

The first inequality in (A.18) follows from $\boldsymbol{\lambda}_o \geq \boldsymbol{0}$ and from feasibility of $\boldsymbol{x}$ ($\boldsymbol{c}(\boldsymbol{x}) \leq \boldsymbol{0}$). The second inequality follows from convexity of $f(\boldsymbol{x})$ and $\boldsymbol{c}(\boldsymbol{x})$ using condition (A.15). The last equality is implied by the Kuhn-Tucker conditions. Consequently, for convex programming problems the Kuhn-Tucker conditions (A.12) are necessary (under a regularity assumption) and sufficient optimality conditions. Linearity of the constraint functions $c_i(\boldsymbol{x})$, $i = 1, 2, \ldots, m$, is sufficient for the regularity assumption to be valid.

A.4 Duality

Duality is an important concept in the theory of mathematical programming. If the original optimization problem – referred to as the *primal* – is convex, its solution can be computed by solving a transformed problem – termed the *dual* – which may in some cases be more convenient to solve. Saddlepoints play an important role in the following discussion of duality theory.

We define the point $(\boldsymbol{x}_o, \boldsymbol{\lambda}_o)$ to be a saddlepoint of $L(\boldsymbol{x}, \boldsymbol{\lambda})$ with respect to minimizing over $\boldsymbol{x} \in \mathbb{R}^n$ and maximizing over $\boldsymbol{\lambda} \in \mathbb{R}^m_+$ if it satisfies

$$L(\boldsymbol{x}_o, \boldsymbol{\lambda}) \le L(\boldsymbol{x}_o, \boldsymbol{\lambda}_o) \le L(\boldsymbol{x}, \boldsymbol{\lambda}_o), \quad \forall (\boldsymbol{x}, \boldsymbol{\lambda}) \in \mathbb{R}^n \times \mathbb{R}^m_+. \tag{A.19}$$

We show that if $(\boldsymbol{x}_o, \boldsymbol{\lambda}_o)$ is a saddlepoint of the Lagrangian function $L(\boldsymbol{x}, \boldsymbol{\lambda})$, it is an optimal solution to problem (A.1). Using the definition of the Lagrangian function (A.6), the left-hand side inequality in (A.19) reads

$$f(\boldsymbol{x}_o) + \boldsymbol{\lambda}^T \boldsymbol{c}(\boldsymbol{x}_o) \le f(\boldsymbol{x}_o) + \boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}_o), \quad \forall \boldsymbol{\lambda} \in \mathbb{R}^m_+,$$

which implies

$$\boldsymbol{\lambda}^T \boldsymbol{c}(\boldsymbol{x}_o) \le \boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}_o), \quad \forall \boldsymbol{\lambda} \in \mathbb{R}^m_+. \tag{A.20}$$

Inequality (A.20) can only hold for all $\boldsymbol{\lambda} \ge \boldsymbol{0}$ if $\boldsymbol{c}(\boldsymbol{x}_o) \le \boldsymbol{0}$. Hence, $\boldsymbol{x}_o$ must be feasible. Setting $\boldsymbol{\lambda} = \boldsymbol{0}$ in (A.20) we get $0 \le \boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}_o)$. However, since $\boldsymbol{\lambda}_o \ge \boldsymbol{0}$ and $\boldsymbol{c}(\boldsymbol{x}_o) \le \boldsymbol{0}$ hold, $\boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}_o) \le 0$ follows. Consequently, $\boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}_o) = 0$ holds.

The right-hand side inequality in (A.19) reads

$$f(\boldsymbol{x}_o) + \boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}_o) \le f(\boldsymbol{x}) + \boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \tag{A.21}$$

Feasible $\boldsymbol{x}$ must satisfy $\boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}) \le 0$. Taking into account that $\boldsymbol{\lambda}_o^T \boldsymbol{c}(\boldsymbol{x}_o) = 0$ holds, inequality (A.21) implies

$$f(\boldsymbol{x}_o) \le f(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in K.$$

Hence, the fact that $(\boldsymbol{x}_o, \boldsymbol{\lambda}_o)$ is a saddlepoint of $L(\boldsymbol{x}, \boldsymbol{\lambda})$ under the restriction $\boldsymbol{\lambda} \ge \boldsymbol{0}$ is a sufficient condition for $\boldsymbol{x}_o$ being an optimal solution to problem (A.1). Note that we did not assume convexity of the objective or the constraint functions.

We now specialize to convex programming problems and show that in this case a point $(\boldsymbol{x}_o, \boldsymbol{\lambda}_o)$ satisfying the Kuhn-Tucker conditions (A.12) must be a saddlepoint of the Lagrangian function under the restriction $\boldsymbol{\lambda} \ge \boldsymbol{0}$. For convenience we repeat the formulation of the Kuhn-Tucker conditions (A.12) without using of the Lagrangian function:

$$\boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{A}(\boldsymbol{x})\boldsymbol{\lambda} = \boldsymbol{0}, \qquad \boldsymbol{c}(\boldsymbol{x}) \leq \boldsymbol{0}, \qquad \boldsymbol{\lambda} \geq \boldsymbol{0}, \qquad \boldsymbol{\lambda}^T\boldsymbol{c}(\boldsymbol{x}) = 0, \tag{A.22}$$

where $\boldsymbol{g}(\boldsymbol{x})$ is the gradient of the objective function $f(\boldsymbol{x})$, and $\boldsymbol{A}(\boldsymbol{x})$ is the Jacobian matrix the columns of which are the gradients of the constraint functions. Note that the vector $\boldsymbol{c}(\boldsymbol{x})$ contains all inequality constraint functions. If there are equality constraints, they can be included by representing each of them by two inequality constraints. We now show that if $(\boldsymbol{x}_o, \boldsymbol{\lambda}_o)$ is a Kuhn-Tucker point satisfying (A.22) then the saddlepoint condition (A.19) follows under the convexity assumption.

The left-hand side inequality in (A.19) immediately follows from feasibility of $\boldsymbol{x}_o$ ($\boldsymbol{c}(\boldsymbol{x}_o) \leq \boldsymbol{0}$) and from $\boldsymbol{\lambda}_o^T\boldsymbol{c}(\boldsymbol{x}_o) = 0$:

$$L(\boldsymbol{x}_o, \boldsymbol{\lambda}) = f(\boldsymbol{x}_o) + \boldsymbol{\lambda}^T\boldsymbol{c}(\boldsymbol{x}_o) \leq f(\boldsymbol{x}_o) = f(\boldsymbol{x}_o) + \boldsymbol{\lambda}_o^T\boldsymbol{c}(\boldsymbol{x}_o) = L(\boldsymbol{x}_o, \boldsymbol{\lambda}_o), \quad \forall\boldsymbol{\lambda} \geq \boldsymbol{0}.$$

The right-hand side inequality in (A.19) follows from convexity of $f(\boldsymbol{x})$ and of the constraint functions $c_i(\boldsymbol{x})$, $i = 1, 2, \ldots, m$, (condition (A.15)) and from the Kuhn-Tucker conditions (A.22):

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\lambda}_o) &= f(\boldsymbol{x}) + \boldsymbol{\lambda}_o^T\boldsymbol{c}(\boldsymbol{x}) \\ &\geq f(\boldsymbol{x}_o) + (\boldsymbol{x} - \boldsymbol{x}_o)^T\boldsymbol{g}(\boldsymbol{x}_o) + \boldsymbol{\lambda}_o^T\left[\boldsymbol{c}(\boldsymbol{x}_o) + \boldsymbol{A}^T(\boldsymbol{x}_o)(\boldsymbol{x} - \boldsymbol{x}_o)\right] \\ &= f(\boldsymbol{x}_o) + \boldsymbol{\lambda}_o^T\boldsymbol{c}(\boldsymbol{x}_o) + (\boldsymbol{x} - \boldsymbol{x}_o)^T\left[\boldsymbol{g}(\boldsymbol{x}_o) + \boldsymbol{A}(\boldsymbol{x}_o)\boldsymbol{\lambda}_o\right] \\ &= f(\boldsymbol{x}_o) + \boldsymbol{\lambda}_o^T\boldsymbol{c}(\boldsymbol{x}_o) \\ &= L(\boldsymbol{x}_o, \boldsymbol{\lambda}_o). \end{aligned}$$

Hence, the saddlepoint condition (A.19) is implied by the Kuhn-Tucker conditions (A.22) if the problem is convex. It follows that for convex programming problems the saddlepoint condition is not only sufficient but also necessary (regularity assumed) for optimality. For convex programming problems the Kuhn-Tucker conditions and the saddlepoint condition are equivalent and both are necessary and sufficient for a point $(\boldsymbol{x}_o, \boldsymbol{\lambda}_o)$ to be a global optimum solution. For general programming problems, the Kuhn-Tucker conditions are necessary (regularity assumed) and the saddlepoint condition is sufficient for local optimality.

Consequently, convex programming problems can be solved by computing a saddlepoint of the Lagrangian function under the restriction $\boldsymbol{\lambda} \geq \boldsymbol{0}$. If $(\boldsymbol{x}_o, \boldsymbol{\lambda}_o)$ is a saddlepoint of $L(\boldsymbol{x}, \boldsymbol{\lambda})$ under the restriction $\boldsymbol{\lambda} \geq \boldsymbol{0}$, then

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda} \geq \boldsymbol{0}} L(\boldsymbol{x}, \boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda} \geq \boldsymbol{0}} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}) = L(\boldsymbol{x}_o, \boldsymbol{\lambda}_o) \tag{A.23}$$

is satisfied. To show this, note that

$$\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}) \leq L(\boldsymbol{x}, \boldsymbol{\lambda}), \quad \forall\boldsymbol{x}, \quad \forall\boldsymbol{\lambda} \geq \boldsymbol{0},$$

and

$$\max_{\boldsymbol{\lambda}\geq\mathbf{0}} L(\boldsymbol{x},\boldsymbol{\lambda}) \geq L(\boldsymbol{x},\boldsymbol{\lambda}), \quad \forall \boldsymbol{x}, \quad \forall \boldsymbol{\lambda}\geq\mathbf{0},$$

and hence,

$$\max_{\boldsymbol{\lambda}\geq\mathbf{0}} \min_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\lambda}) \leq \min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}\geq\mathbf{0}} L(\boldsymbol{x},\boldsymbol{\lambda}) \tag{A.24}$$

holds in general. Now if $(\boldsymbol{x}_o,\boldsymbol{\lambda}_o)$ is a saddlepoint of $L(\boldsymbol{x},\boldsymbol{\lambda})$ under the restriction $\boldsymbol{\lambda}\geq\mathbf{0}$, then

$$\begin{aligned} L(\boldsymbol{x}_o,\boldsymbol{\lambda}_o) &= \min_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\lambda}_o) \\ &\leq \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \min_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\lambda}) \leq \min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}\geq\mathbf{0}} L(\boldsymbol{x},\boldsymbol{\lambda}) \\ &\leq \max_{\boldsymbol{\lambda}\geq\mathbf{0}} L(\boldsymbol{x}_o,\boldsymbol{\lambda}) \\ &= L(\boldsymbol{x}_o,\boldsymbol{\lambda}_o) \end{aligned}$$

follows, which obviously implies that (A.24) holds with equality and (A.23) is satisfied. Consequently, a saddlepoint $(\boldsymbol{x}_o,\boldsymbol{\lambda}_o)$ of the Lagrangian function corresponding to a solution of an associated convex programming problem can be computed either by solving

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \max_{\boldsymbol{\lambda}\geq\mathbf{0}} L(\boldsymbol{x},\boldsymbol{\lambda}) \tag{A.25}$$

or by solving

$$\underset{\boldsymbol{\lambda}\geq\mathbf{0}}{\text{maximize}} \ \min_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\lambda}). \tag{A.26}$$

We refer to problem (A.25) as the *primal* and to problem (A.26) as the *dual*. In the case of convex programming problems, either of them can be solved in order to compute a global optimum.

We mentioned in Section A.3 that a weighted sum of convex functions (A.17) is itself a convex function if the weights are non-negative. It follows that the Lagrangian function associated with a convex programming problem is a convex function with respect to $\boldsymbol{x}$ if $\boldsymbol{\lambda}\geq\mathbf{0}$ holds. Hence, assuming differentiability of $f(\boldsymbol{x})$ and $\boldsymbol{c}(\boldsymbol{x})$, the dual (A.26) of a convex programming problem can also be written as

$$\begin{aligned} &\underset{\boldsymbol{x},\boldsymbol{\lambda}}{\text{maximize}} && L(\boldsymbol{x},\boldsymbol{\lambda}) \\ &\text{subject to} && \frac{\partial L}{\partial \boldsymbol{x}} = \mathbf{0}, \quad \boldsymbol{\lambda}\geq\mathbf{0}. \end{aligned} \tag{A.27}$$

This form of the dual is referred to as the *Wolfe dual* [34]. The fact that the maximization in (A.26) is only performed over $\boldsymbol{\lambda}\geq\mathbf{0}$ and over minima of $L(\boldsymbol{x},\boldsymbol{\lambda})$ with respect to $\boldsymbol{x}$ is taken into account by the constraints in (A.27). Note that this is possible due to the convexity of $L(\boldsymbol{x}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{x}$ for $\boldsymbol{\lambda} \geq \mathbf{0}$. Solving problem (A.27) is equivalent to solving the primal (A.1) if the problem is convex and if the regularity assumption holds.

We now derive the dual problems for a linear programming problem and for a convex quadratic programming problem with linear constraints. The regularity assumption is valid for both problems due to the linearity of the constraint functions. Let a linear programming problem be given by

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{a}^T \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}^T \boldsymbol{x} \leq \boldsymbol{b}. \end{aligned} \tag{A.28}$$

Due to the linearity of all functions the problem is obviously convex. The associated Lagrangian function is

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{a}^T \boldsymbol{x} + \boldsymbol{\lambda}^T (\boldsymbol{A}^T \boldsymbol{x} - \boldsymbol{b}).$$

Its derivative with respect to $\boldsymbol{x}$ is

$$\frac{\partial L}{\partial \boldsymbol{x}} = \boldsymbol{a} + \boldsymbol{A}\boldsymbol{\lambda}.$$

The constraint $\partial L / \partial \boldsymbol{x} = \mathbf{0}$ of the dual (A.27) becomes

$$\boldsymbol{A}\boldsymbol{\lambda} = -\boldsymbol{a}.$$

Substituting for $\boldsymbol{a}$ in the Lagrangian function, the dual problem can be formulated as

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{maximize}} \quad & -\boldsymbol{b}^T \boldsymbol{\lambda} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{\lambda} = -\boldsymbol{a}, \quad \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \tag{A.29}$$

Once the dual (A.29) has been solved, the solution to the primal can be computed by solving the reduced square system $\tilde{\boldsymbol{A}}^T \boldsymbol{x} = \tilde{\boldsymbol{b}}$, where the rows corresponding to inactive constraints have been eliminated. These rows are identified by their associated zero Lagrange multipliers.

Let a quadratic programming problem be given by

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \tfrac{1}{2} \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{f}^T \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}^T \boldsymbol{x} \leq \boldsymbol{b}, \end{aligned} \tag{A.30}$$

where $\boldsymbol{H}$ is a symmetric positive definite matrix. Since $\boldsymbol{H}$ is the Hessian matrix of the objective function, problem (A.30) is convex. The corresponding Lagrangian function is

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \tfrac{1}{2} \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{f}^T \boldsymbol{x} + \boldsymbol{\lambda}^T (\boldsymbol{A}^T \boldsymbol{x} - \boldsymbol{b}).$$

Its derivative with respect to $\boldsymbol{x}$ is

$$\frac{\partial L}{\partial \boldsymbol{x}} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f} + \boldsymbol{A}\boldsymbol{\lambda}.$$

By setting this derivative to zero according to (A.27) and using the fact that $\boldsymbol{H}$ is nonsingular, the variable $\boldsymbol{x}$ can be expressed as

$$\boldsymbol{x} = -\boldsymbol{H}^{-1}(\boldsymbol{f} + \boldsymbol{A}\boldsymbol{\lambda}). \tag{A.31}$$

Using this expression to eliminate $\boldsymbol{x}$ from the Lagrangian function, the dual of problem (A.30) can be written as

$$\begin{array}{ll} \underset{\boldsymbol{\lambda}}{\text{maximize}} & -\frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{A}^T\boldsymbol{H}^{-1}\boldsymbol{A}\boldsymbol{\lambda} - (\boldsymbol{A}^T\boldsymbol{H}^{-1}\boldsymbol{f} + \boldsymbol{b})^T\boldsymbol{\lambda} \\ \text{subject to} & \boldsymbol{\lambda} \geq \boldsymbol{0}. \end{array} \tag{A.32}$$

Since there are no general constraints but only simple bound constraints on the variables $\boldsymbol{\lambda}$, this problem may be easier to solve than the primal (A.30). Once the optimum solution to (A.32) has been found, $\boldsymbol{x}$ can be computed from (A.31).

A.5 Algorithms

This section gives a brief introduction to methods used for solving optimization problems with linear constraint functions. Problems with nonlinear constraint functions can be solved by solving a sequence of linearly constrained subproblems [34, 36]. We will focus the discussion on linear and quadratic programming problems and on active set methods. A Matlab program is provided that solves a special convex quadratic programming problem, where the only constraints are non-negativity constraints on the variables. The dual quadratic programming problem (A.32) is of this type.

The oldest method for solving linear programming problems is the *simplex algorithm* due to Dantzig (see e. g. [28]). It has become a standard method for which many reliable software implementations are available. Another class of algorithms that can not only solve linear but also arbitrary convex programming problems is called *interior point methods*. One of the most well-known interior point methods is due to Karmarkar [51]. The theoretical advantage of interior point methods is that their computational complexity is polynomial in the number of variables n. This is to be compared to the worst case complexity of the simplex algorithm that may increase exponentially in the number of variables. However, this exponential complexity behavior rarely occurs in practice. A different approach to solving linear, quadratic, and general linearly constrained optimization problems is the *active set method*. It has become the standard method for solving quadratic programming problems, and most existing quadratic programming software is based on an active set method. In the case of linear programming, it can be shown that the active set method is equivalent to the simplex method [34]. However, the simplex and the active set methods differ in the dimension of the matrices they store and update in every iteration step. This is an important point, and which method is best depends on the structure of the problem under consideration. The (revised) simplex method stores and updates an $m \times m$ matrix, where m is the number of constraints in the problem. The active set method stores and updates an $n \times n$ matrix, where n is the number of variables. Hence, for problems with $n \ll m$ the active set method is preferable and vice versa. Note, however, that by the duality transformation the roles of m and n are interchanged. Hence, if a certain software implementation (simplex or active set method) is available, the problem can always be formulated in such a way that it is optimally suited to this software.

All methods mentioned so far compute a sequence of feasible points and they need a feasible guess to start with. It can be shown that the problem of finding a feasible solution satisfying linear equality and inequality constraints can itself be formulated as a linear programming problem [34]. Finding a feasible initial guess for this new problem is trivial due to its special structure. The computation of an initial feasible guess is usually referred to as *phase I* of the optimization procedure, while the actual computation of the optimal solution is termed *phase II*. From now on we will assume that the *phase I* problem has been solved and that an initial feasible guess is available.

Active set methods are based on the fact that every problem with inequality constraints could be treated as an equality constrained problem if the active constraints at the optimal solution were known. In every iteration step of an active set method certain constraints – the active set – are regarded as equality constraints and the other constraints are disregarded. The active set is iteratively adjusted until it contains the correct active constraints at the solution.

Consider the linear programming problem given by (A.28). Additional equality constraints are easily taken into account by regarding them as a fixed part of the active set in every iteration step. Due to the linearity of the objective function, the optimal solution occurs at a vertex of the feasible region, i. e. at an extremal point where n constraints are active. Assume an initial guess $\boldsymbol{x}$ being a feasible vertex of problem (A.28) is given. This point satisfies

$$\tilde{\boldsymbol{A}}^T \boldsymbol{x} = \tilde{\boldsymbol{b}},$$

where $\tilde{\boldsymbol{A}}^T$ and $\tilde{\boldsymbol{b}}$ are those parts of $\boldsymbol{A}^T$ and $\boldsymbol{b}$ in (A.28) corresponding to the n active constraints at the vertex $\boldsymbol{x}$. From (A.5) the Lagrange multipliers are related to $\boldsymbol{x}$ by

$$\tilde{\boldsymbol{A}} \boldsymbol{\lambda} = -\boldsymbol{g}(\boldsymbol{x}),$$

where the gradient of the objective function in (A.28) is $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{a}$. Hence, the Lagrange multipliers are given by

$$\boldsymbol{\lambda} = -\tilde{\boldsymbol{A}}^{-1}\boldsymbol{a}. \tag{A.33}$$

Note that the expression $\boldsymbol{s}^T\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{s}^T\boldsymbol{a}$ is the change of the objective function due to a move along the direction $\boldsymbol{s}$. Consequently, according to (A.33) the rows $\boldsymbol{s}_i^T$, $i = 1, 2, \dots, n$, of $-\tilde{\boldsymbol{A}}^{-1}$ are downhill directions if their associated Lagrange multipliers are negative. Moreover, they are feasible since they satisfy $\boldsymbol{s}_i^T\tilde{\boldsymbol{A}} \leq \boldsymbol{0}$, $i = 1, 2, \dots, n$. In fact, by moving along $\boldsymbol{s}_q$, $1 \leq q \leq n$, all constraints except for the q^{th} remain active, corresponding to a move along an edge of the feasible region towards the next vertex. If at some iteration step $\boldsymbol{\lambda} \geq \boldsymbol{0}$ is satisfied, no feasible downhill direction exists, and the solution corresponding to the actual vertex is optimal. If $\boldsymbol{\lambda} \geq \boldsymbol{0}$ is not satisfied, the row $\boldsymbol{s}_q^T$ of $-\tilde{\boldsymbol{A}}^{-1}$ corresponding to the most negative multiplier is chosen as a search direction for computing the next iterate:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha\boldsymbol{s}_q, \quad \alpha \geq 0, \tag{A.34}$$

where k denotes the iteration index. This search corresponds to a move along a downhill edge of the feasible region. The step size α in (A.34) is determined by the first inactive inequality constraint to become active. As soon as this constraint becomes active, the next vertex has been reached. The new active set is obtained from the old active set by adding this new active inequality constraint and by removing the q^{th} constraint corresponding to the most negative Lagrange multiplier. At this point the process is repeated until all multipliers are non-negative and the optimum solution has been found. The exchange of one constraint in every iteration step corresponds to the exchange of one column in $\tilde{\boldsymbol{A}}$, and hence $\tilde{\boldsymbol{A}}^{-1}$ need not be recomputed but can be updated e. g. using the Sherman-Morrison formula [34].

Consider now the quadratic programming problem (A.30). Additional equality constraints can easily be handled as explained in the context of linear programming. In the case of a quadratic objective function, the optimum solution need not necessarily be a vertex of the feasible region. We assume a feasible initial point to be known, and reformulate problem (A.30) to optimize the update vector $\boldsymbol{s}$ from a given point $\boldsymbol{x}^{(k)}$: $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{s}$, where k is the iteration index. The transformed problem reads

$$\begin{aligned} &\underset{\boldsymbol{s}}{\text{minimize}} && \tfrac{1}{2}\boldsymbol{s}^T\boldsymbol{H}\boldsymbol{s} + \boldsymbol{g}^{(k)T}\boldsymbol{s} \\ &\text{subject to} && \boldsymbol{A}^T\boldsymbol{s} \leq \boldsymbol{b} - \boldsymbol{A}^T\boldsymbol{x}^{(k)}, \end{aligned} \tag{A.35}$$

where $\boldsymbol{g}^{(k)}$ is the gradient of the objective function evaluated at $\boldsymbol{x}^{(k)}$:

$$\boldsymbol{g}^{(k)} = \boldsymbol{g}(\boldsymbol{x}^{(k)}) = \boldsymbol{H}\boldsymbol{x}^{(k)} + \boldsymbol{f}.$$

The active set contains all equality constraints and those inequality constraints that are satisfied with equality at the point $\boldsymbol{x}^{(k)}$. Let $\tilde{\boldsymbol{A}}$ be the part of matrix $\boldsymbol{A}$ that contains all columns corresponding to the active set. The active set method treats those constraints belonging to the active set as equality constraints and disregards all other constraints. Hence, the subproblem reads

$$\begin{aligned} \underset{\boldsymbol{s}}{\text{minimize}} \quad & \tfrac{1}{2}\boldsymbol{s}^T\boldsymbol{H}\boldsymbol{s} + \boldsymbol{g}^{(k)T}\boldsymbol{s} \\ \text{subject to} \quad & \tilde{\boldsymbol{A}}^T\boldsymbol{s} = \boldsymbol{0}. \end{aligned} \tag{A.36}$$

This subproblem contains only equality constraints and can be solved for $\boldsymbol{s}$ either by the method of Lagrange multipliers or by a generalized elimination method [34]. The next iterate is chosen as

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha\boldsymbol{s}, \quad \alpha \geq 0. \tag{A.37}$$

If $\boldsymbol{x}^{(k+1)}$ were infeasible for $\alpha = 1$, then $\alpha < 1$ is chosen such that one inequality constraint not belonging to the current active set becomes active. This new constraint is added to the existing active set and the equality constrained problem (A.36) is solved using the new active set. In this manner a sequence of feasible iterates $\boldsymbol{x}^{(k)}$ is generated that satisfy an increasing number of constraints with equality. This process is repeated until the new iterate is feasible for $\alpha = 1$. This means that in the next iteration step the optimum solution to problem (A.36) would be $\boldsymbol{s} = \boldsymbol{0}$. Now the Lagrange multipliers corresponding to the actual solution are checked. If $\boldsymbol{\lambda} \geq \boldsymbol{0}$ holds, the optimum solution has been found. If not, the constraint corresponding to the most negative multiplier is removed from the active set, and the whole process starts anew.

The Matlab programs `lp` and `qp` contained in the Matlab Optimization Toolbox implement active set methods for solving linear and quadratic programming problems, respectively. However, applying `qp` to the dual quadratic programming problem given by (A.32) is very inefficient because the fact that the inequality constraints are only simple bound constraints on the variables (and hence $\boldsymbol{A} = -\boldsymbol{I}$ and $\boldsymbol{b} = \boldsymbol{0}$ in (A.30)) is not taken into account in `qp`. The following Matlab program `nnqmin` efficiently solves a convex quadratic programming problem with non-negativity constraints on the variables. It provides the solution to the problem and the associated vector of Lagrange multipliers. A feasible initial guess can be supplied by the user but is not necessary.

```
function [x,l] = nnqmin(H,f,x0)
% NNQMIN: quadratic function minimization subject to non-negativity
% constraints on the variables:
% min 0.5*x'*H*x + f'*x, s.t. x >= 0
% [x,l] = nnqmin(H,f,x0) returns solution x and corresponding
% Lagrange multipliers l; H is assumed to be positive definite
% x0: optional starting guess
```

```
%
% Author: Mathias C. Lang, Vienna University of Technology, July '98

f = f(:); n = length(f);
if nargin == 3,
   x=x0(:);
   if length(x) ~= n, error('x0 has wrong length!'); end
   Ix = find(x < 0);
   if ~isempty(Ix), x(Ix)=0; end
else
   x = zeros(n,1);
end

I = find(x>0);    % indices of inactive constraints
xI = x(I);

while 1,

   if ~isempty(I), alpha = 0;
   else, alpha = 1; end

   % solve equality constrained problem
   while alpha < 1,

      s = -(xI + H(I,I)\f(I));      % search direction
      xnew = xI + s;
      if min(xnew) < 0,             % new constraint becomes active
         Is = find(s < 0);
         [alpha,p]=min(-xI(Is)./s(Is)); p=Is(p);
         I(p) = [];
         xI = xI + alpha*s; xI(p)=[];
      else
         xI = xnew;
         alpha = 1;
      end

   end

   % compute Lagrange multipliers
   if ~isempty(I),
      l = H(:,I)*xI + f; l(I) = 0;
   else
      l = f;
   end
   [minl,q] = min(l);
   if minl < 0,             % new constraint becomes inactive
      I = [I;q];
      xI = [xI;0];
   else,                    % optimum found!
      x = zeros(n,1); x(I) = xI;
      break;
    end

  end
```

Appendix B

Levinson's Algorithm

Levinson's algorithm solves systems of linear equations with Toeplitz system matrices. The original work by Levinson [70] considers the problem of linear prediction. The structure of this problem is identical to the one of fitting an autoregressive model to given data. This problem has been considered by Durbin who rediscovered Levinson's algorithm [31]. For this reason the algorithm is sometimes called Levinson-Durbin algorithm.

Levinson's algorithm can be used with general Toeplitz matrices. However, we will only discuss it for the special case of Hermitian Toeplitz matrices which are of more practical relevance and which exclusively occur in the algorithms discussed in this thesis.

A Hermitian Toeplitz matrix $\boldsymbol{T}_n = \boldsymbol{T}_n^H$ of size $n \times n$ has the following structure:

$$\boldsymbol{T}_n = \begin{pmatrix} t_0 & t_1^* & & \cdots & t_{n-1}^* \\ t_1 & t_0 & t_1^* & \cdots & \vdots \\ \vdots & t_1 & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & t_1^* \\ t_{n-1} & & \cdots & t_1 & t_0 \end{pmatrix}. \tag{B.1}$$

It is completely determined by its first column or row because all the elements along any diagonal are identical. The basic problem solved by Levinson's algorithm can be formulated as follows: Solve

$$\boldsymbol{T}_n \boldsymbol{x}_n = \begin{pmatrix} \alpha_n \\ \mathbf{0} \end{pmatrix} \tag{B.2}$$

for $\boldsymbol{x}_n$ and α_n, where $\boldsymbol{x}_n = [1, x_n(2), \ldots, x_n(n)]^T$. In the context of autoregressive parameter estimation, the equations (B.2) are called Yule-Walker equations and the first column of $\boldsymbol{T}_n$ is an autocorrelation sequence.

Define the square matrix $\boldsymbol{J}$ by

$$\boldsymbol{J} = \begin{pmatrix} \mathbf{0} & & & 1 \\ & & 1 & \\ & \cdots & & \\ 1 & & & \mathbf{0} \end{pmatrix}. \tag{B.3}$$

The matrix $\boldsymbol{J}$ is usually called *exchange* or *reflection* matrix. Denote by

$$\hat{\boldsymbol{x}} = \boldsymbol{J}\boldsymbol{x}^* \tag{B.4}$$

the vector that has the same elements as $\boldsymbol{x}^*$ but in reversed order. Note that for a Hermitian Toeplitz matrix $\boldsymbol{T}$ the equality

$$\boldsymbol{T}\boldsymbol{J} = \boldsymbol{J}\boldsymbol{T}^* \tag{B.5}$$

holds. Assume $\boldsymbol{T}\boldsymbol{x} = \boldsymbol{b}$ is satisfied. From (B.4) and (B.5)

$$\boldsymbol{T}\hat{\boldsymbol{x}} = \boldsymbol{T}\boldsymbol{J}\boldsymbol{x}^* = \boldsymbol{J}\boldsymbol{T}^*\boldsymbol{x}^* = \boldsymbol{J}\boldsymbol{b}^* = \hat{\boldsymbol{b}}. \tag{B.6}$$

follows. This property implied by the Hermitian Toeplitz structure of $\boldsymbol{T}$ is exploited by Levinson's algorithm. Denote by $\boldsymbol{T}_m$, $m \leq n$, the upper left $m \times m$ partition of the matrix $\boldsymbol{T}_n$. Note that $\boldsymbol{T}_m$ is also Hermitian Toeplitz. From a known solution to the problem

$$\boldsymbol{T}_m \boldsymbol{x}_m = \begin{pmatrix} \alpha_m \\ \mathbf{0} \end{pmatrix}, \quad x_m(1) = 1, \tag{B.7}$$

Levinson's recursion computes the solution $\boldsymbol{x}_{m+1}$ to the problem of dimension $m+1$. From (B.7)

$$\boldsymbol{T}_{m+1} \begin{pmatrix} \boldsymbol{x}_m \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha_m \\ \mathbf{0} \\ \beta_m \end{pmatrix} \tag{B.8}$$

follows, where $\beta_m = \hat{\boldsymbol{t}}_m^H \boldsymbol{x}_m$ with $\boldsymbol{t}_m = [t_1, \ldots, t_m]^T$. Using (B.6) we get

$$\boldsymbol{T}_{m+1} \left[\begin{pmatrix} \boldsymbol{x}_m \\ 0 \end{pmatrix} + k_m \begin{pmatrix} 0 \\ \hat{\boldsymbol{x}}_m \end{pmatrix} \right] = \begin{pmatrix} \alpha_m \\ \mathbf{0} \\ \beta_m \end{pmatrix} + k_m \begin{pmatrix} \beta_m^* \\ \mathbf{0} \\ \alpha_m^* \end{pmatrix} \tag{B.9}$$

with a scalar constant k_m yet to be determined. Comparing (B.9) with

$$\boldsymbol{T}_{m+1} \boldsymbol{x}_{m+1} = \begin{pmatrix} \alpha_{m+1} \\ \mathbf{0} \end{pmatrix}, \tag{B.10}$$

we see that $\boldsymbol{x}_{m+1}$ and α_{m+1} can be computed from (B.9) by appropriate choice of k_m:

$$\beta_m + k_m \alpha_m^* = 0 \quad \Rightarrow \quad k_m = -\frac{\beta_m}{\alpha_m^*} = -\frac{\beta_m}{\alpha_m}, \tag{B.11}$$

where the last equality follows from the fact that α_m is real-valued not equal to zero. The first is a consequence of the matrices $\boldsymbol{T}_m$ being Hermitian. If $\boldsymbol{T}_n$ is positive definite, then $\alpha_m > 0$, $m = 1, 2, \ldots, n$, holds. In the context of autoregressive parameter estimation, α_m is the power of the assumed driving noise process for a model of order $m-1$. In the context of linear prediction, α_m is the power of the prediction error for an order $m-1$ prediction error filter. The scalars k_m computed in each iteration step are called reflection coefficients. Equations (B.9) – (B.11) provide the recursions for $\boldsymbol{x}_n$ and α_n:

$$\begin{aligned} \boldsymbol{x}_{m+1} &= \begin{pmatrix} \boldsymbol{x}_m \\ 0 \end{pmatrix} - \frac{\beta_m}{\alpha_m} \begin{pmatrix} 0 \\ \hat{\boldsymbol{x}}_m \end{pmatrix} \quad \text{and} \\ \alpha_{m+1} &= \alpha_m - \frac{|\beta|_m^2}{\alpha_m} = \alpha_m \left(1 - |k_m|^2\right), \quad m = 1(1)n-1, \end{aligned} \tag{B.12}$$

with initializations $\boldsymbol{x}_1 = 1$ and $\alpha_1 = t_0$. The fact that α_m is real-valued is most easily seen by looking at its recursion formula (B.12) and noting that its initial value $\alpha_1 = t_0$ is real-valued.

The derivation of Levinson's recursion makes clear that it cannot be applied to all Toeplitz matrices. We implicitly assumed the non-singularity of the partitions $\boldsymbol{T}_m$, $m < n$, although this is not necessary for the non-singularity of $\boldsymbol{T}_n$. However, the definiteness of $\boldsymbol{T}_n$ is sufficient for Levinson's algorithm to be applicable. The Toeplitz matrices used in this thesis have been shown to be positive definite. The Matlab program `levdurb` implements Levinson's algorithm as described above.

```
function [x,alpha] = levdurb(t)
% function [x,alpha] = levdurb(t)
% Levinson-Durbin Algorithm
% t:     autocorrelation sequence
% x:     AR-model/prediction error filter coefficients
% alpha: driving noise/error power

t = t(:); n = length(t);
x = 1; alpha = t(1);
for m = 1:n-1,
   k = -(t(m+1:-1:2)'*x)/alpha;
   x = [x;0] + k*flipud([conj(x);0]);
   alpha = alpha*(1 - abs(k)^2);
end
```

It is easily verified that the computational complexity of Levinson's algorithm is $O(n^2)$ which is to be compared to $O(n^3)$ for solving arbitrary systems of linear equations.

Note that the Matlab Signal Processing Toolbox provides a program called `levinson` which also implements the Levinson-Durbin recursion. The commands

```
    x = levdurb(t)
```

and

```
    x = levinson(t)
```

will give the same results if `t` is a vector. Note that `levinson` can also be used if `t` is a matrix. Unlike `levdurb`, the program `levinson` does not return the value α_n. The algorithms presented in the following two sections are based on the Levinson-Durbin recursion. However, the problems considered there cannot be solved by the program `levinson` provided in the Matlab Signal Processing Toolbox.

B.1 Hermitian Toeplitz Systems of Linear Equations

Levinson's algorithms as described so far can only be used to solve the special linear system (B.2). However, a slightly modified version of Levinson's algorithm can solve linear Toeplitz systems with arbitrary right-hand side vectors (see e. g. [78, 104]). We will describe the procedure for Hermitian Toeplitz systems.

Consider the problem

$$\boldsymbol{T}_n \boldsymbol{y}_n = \boldsymbol{b}_n, \tag{B.13}$$

with specified $\boldsymbol{T}_n$ and $\boldsymbol{b}_n$ and unknown $\boldsymbol{y}_n$. The matrix $\boldsymbol{T}_n$ is an $n \times n$ Hermitian Toeplitz matrix as defined by (B.1), and the length n column vector $\boldsymbol{b}_n = [b_1, b_2, \dots, b_n]^T$ is arbitrary. The solution vector $\boldsymbol{y}_n$ is computed recursively from solutions to the lower order problems

$$\boldsymbol{T}_m \boldsymbol{y}_m = \boldsymbol{b}_m, \quad m < n, \tag{B.14}$$

where $\boldsymbol{T}_m$ is an upper left $m \times m$ partition of the matrix $\boldsymbol{T}_n$ containing the elements $t_0, t_1, \dots, t_{m-1}$, and $\boldsymbol{b}_m = [b_1, b_2, \dots, b_m]^T$ contains the first m elements of $\boldsymbol{b}_n$. In addition, we assume that the solution to the problem

$$\boldsymbol{T}_{m+1} \boldsymbol{x}_{m+1} = \begin{pmatrix} \alpha_{m+1} \\ \boldsymbol{0} \end{pmatrix}, \quad m < n, \quad x_{m+1}(1) = 1, \tag{B.15}$$

is known. It can be computed by Levinson's algorithm. With these assumptions the quantities $\boldsymbol{y}_m$, $\boldsymbol{x}_{m+1}$, and α_{m+1} are known for some $m < n$. From (B.6) and (B.15)

$$\boldsymbol{T}_{m+1} \left[\begin{pmatrix} \boldsymbol{y}_m \\ 0 \end{pmatrix} + k_m \, \hat{\boldsymbol{x}}_{m+1} \right] = \begin{pmatrix} \boldsymbol{b}_m \\ \gamma_m \end{pmatrix} + k_m \begin{pmatrix} \boldsymbol{0} \\ \alpha_{m+1}^* \end{pmatrix} \tag{B.16}$$

follows, where k_m is an arbitrary scalar constant and γ_m is given by

$$\gamma_m = \hat{\boldsymbol{t}}_m^H \boldsymbol{y}_m \tag{B.17}$$

with $\boldsymbol{t}_m = [t_1, \ldots, t_m]^T$. Choosing the last element of the right-hand side vector in (B.16) such that

$$\gamma_m + k_m \alpha_{m+1}^* = b_{m+1}$$

is satisfied, the recursion

$$\boldsymbol{y}_{m+1} = \begin{pmatrix} \boldsymbol{y}_m \\ 0 \end{pmatrix} + k_m \hat{\boldsymbol{x}}_{m+1}, \quad m = 1(1)n-1, \tag{B.18}$$

follows with

$$k_m = \frac{b_{m+1} - \gamma_m}{\alpha_{m+1}},$$

where the fact that α_{m+1} is real-valued and not equal to zero has been taken into account. The initial values are $\boldsymbol{x}_1 = 1$, $\alpha_1 = t_0$, and $\boldsymbol{y}_1 = b_1/t_0$.

Hence, the linear system (B.13) can be solved using two recursion formulas. One for computing $\boldsymbol{x}_m$, $m = 1(1)n$, as in the standard version of Levinson's algorithm, and the other for computing the solutions $\boldsymbol{y}_m$, $m = 1(1)n$, of the general problems. The computational effort is approximately twice the effort of Levinson's algorithm in its standard form, but the complexity of this more general algorithm is still $O(n^2)$.

The Matlab program `levin` implements this method.

```
function x = levin(a,b)
% function x = levin(a,b)
% solves system of complex linear equations toeplitz(a)*x=b
% using Levinson's algorithm
% a ... first row of positive definite Hermitian Toeplitz matrix
% b ... right hand side vector
%
% Author: Mathias C. Lang, Vienna University of Technology, AUSTRIA
% 9-97

a = a(:); b = b(:); n = length(a);
t = 1; alpha = a(1); x = b(1)/a(1);
for i = 1:n-1,
   k = -(a(i+1:-1:2)'*t)/alpha;
   t = [t;0] + k*flipud([conj(t);0]);
   alpha = alpha*(1 - abs(k)^2);
   k = (b(i+1) - a(i+1:-1:2)'*x)/alpha;
   x = [x;0] + k*flipud(conj(t));
end
```

Again it should be noted that this algorithm only works if all partitions $\boldsymbol{T}_m$, $m < n$ of $\boldsymbol{T}_n$ are non-singular. This is satisfied for positive definite matrices $\boldsymbol{T}_n$.

B.2 Inversion of a Hermitian Toeplitz Matrix

Sometimes it is necessary to explicitly compute the inverse of a matrix. This section shows that this is accomplished by Levinson's algorithm if the matrix is a positive definite Toeplitz matrix. Again we will concentrate on Hermitian Toeplitz matrices. Levinson's algorithm actually provides a factorization of the inverse matrix of the form

$$\boldsymbol{T}_n^{-1} = \boldsymbol{L}_n \boldsymbol{D}_n \boldsymbol{L}_n^H \tag{B.19}$$

where the $n \times n$ matrix $\boldsymbol{L}_n$ is a lower triangular matrix and the $n \times n$ matrix $\boldsymbol{D}_n$ is a diagonal matrix. The factorization (B.19) is known as *Cholesky* factorization.

Let the columns of the lower triangular matrix $\boldsymbol{L}_n$ be the solutions $\boldsymbol{x}_m$ of the problems (B.7):

$$\boldsymbol{L}_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ x_n(2) & 1 & & 0 \\ x_n(3) & x_{n-1}(2) & & \vdots \\ \vdots & \vdots & \ddots & 0 \\ x_n(n) & x_{n-1}(n-1) & \cdots & 1 \end{pmatrix}. \tag{B.20}$$

Note that all diagonal elements of the matrix $\boldsymbol{L}_n$ are unity. From (B.7), the matrix $\boldsymbol{T}_n\boldsymbol{L}_n$ is an upper triangular matrix with the elements $\alpha_n, \alpha_{n-1}, \ldots, \alpha_1$ in its main diagonal. Since $\boldsymbol{L}_n^H$ is also upper triangular, the product $\boldsymbol{L}_n^H\boldsymbol{T}_n\boldsymbol{L}_n$ must be upper triangular as well. Its diagonal elements are $\alpha_n, \alpha_{n-1}, \ldots, \alpha_1$. Moreover, since the matrix $\boldsymbol{L}_n^H\boldsymbol{T}_n\boldsymbol{L}_n$ is Hermitian, it must be a diagonal matrix:

$$\boldsymbol{L}_n^H\boldsymbol{T}_n\boldsymbol{L}_n = \begin{pmatrix} \alpha_n & & & \mathbf{0} \\ & \alpha_{n-1} & & \\ & & \ddots & \\ \mathbf{0} & & & \alpha_1 \end{pmatrix} = \boldsymbol{D}_n^{-1}. \tag{B.21}$$

From (B.21)

$$\boldsymbol{T}_n = \boldsymbol{L}_n^{-H}\boldsymbol{D}_n^{-1}\boldsymbol{L}_n^{-1}$$

and

$$\boldsymbol{T}_n^{-1} = \boldsymbol{L}_n \boldsymbol{D}_n \boldsymbol{L}_n^H \tag{B.22}$$

follow. Equation (B.22) is exactly the required factorization (B.19). Levinson's algorithm provides all elements of the matrices $\boldsymbol{L}_n$ and $\boldsymbol{D}_n$. It is only necessary to store the intermediate results $\boldsymbol{x}_m$ and α_m for $1 \leq m \leq n$. This is accomplished by the Matlab program `invtoep`.

```
function [L,d] = invtoep(t)
% function [L,d] = invtoep(t)
% Cholesky factorization of the inverse of the positive
% definite Hermitian Toeplitz matrix T = toeplitz(t) with
% its first row given by t using Levinson's algorithm:
% inv(T) = L*diag(d)*L'

t = t(:); n = length(t);
x = 1; alpha = t(1);
L = eye(n,n); d = zeros(n,1); d(n) = 1/alpha;
for m = 1:n-1,
   k = -(t(m+1:-1:2)'*x)/alpha;
   x = [x;0] + k*flipud([conj(x);0]);
   alpha = alpha*(1 - abs(k)^2);
   L(n-m:n,n-m) = x;
   d(n-m) = 1/alpha;
end
```

Appendix C

Rouché's Theorem and a Stability Condition

In this appendix, we give a proof of Rouché's theorem. Furthermore, we formulate a condition on the coefficients of a polynomial such that the moduli of its zeros are bounded by an arbitrary positive constant.

C.1 Preliminaries

In order to prove Rouché's theorem, we need the following result:

If f(z) is analytic inside and on a closed contour C and is not zero on the contour, then

$$N = \frac{1}{2\pi j} \oint_C \frac{f'(z)}{f(z)} dz, \tag{C.1}$$

where N is the number of zeros inside the contour (a zero of order m being counted m times).

A proof of this basic result can be found in [131].

We now formulate a different representation of the expression for N given in (C.1) which will be used in the proof of Rouché's theorem. Since

$$\frac{f'(z)}{f(z)} = \frac{d}{dz} \ln f(z),$$

we get

$$\oint_C \frac{f'(z)}{f(z)} dz = \Delta_C \left\{ \ln f(z) \right\}, \tag{C.2}$$

where $\Delta_C \left\{ \ln f(z) \right\}$ denotes the increase of the function $\ln f(z)$ when z moves round the contour C. Since

$$\ln f(z) = \ln |f(z)| + j \arg[f(z)],$$

and since $\ln |f(z)|$ returns to its original value when z moves round the closed contour C,

$$\Delta_C \{\ln f(z)\} = j\Delta_C \{\arg[f(z)]\} \tag{C.3}$$

holds. From (C.1), (C.2), and (C.3) it follows that the number of zeros of $f(z)$ inside C can be written as

$$N = \frac{1}{2\pi}\Delta_C \{\arg[f(z)]\}. \tag{C.4}$$

We will use (C.4) in the following proof of Rouché's theorem.

C.2 Rouché's Theorem

The following proof of Rouché's theorem is taken from [131]. We sketch this proof to provide the reader with insight into this important theorem. Furthermore, from the proof it becomes clear that the condition implied by Rouché's theorem is sufficient but not necessary. However, the examples given in Section 5.5 show that the proposed method which is based on this condition yields results that are usually considerably better than solutions obtained by a broad range of other methods which use different stability conditions.

Rouché's theorem. *If $f(z)$ and $g(z)$ are analytic inside and on a closed contour C, and $|g(z)| < |f(z)|$ on C, then $f(z)$ and $f(z) + g(z)$ have the same number of zeros inside C.*

Note that due to the requirement $|g(z)| < |f(z)|$, neither $f(z)$ nor $f(z) + g(z)$ can have zeros on C. Let N denote the number of zeros of $f(z)$ inside C, and let N' denote the number of zeros of $f(z) + g(z)$ inside C. Then, according to (C.4),

$$\begin{aligned} 2\pi N' &= \Delta_C \{\arg[f(z)+g(z)]\} \\ &= \Delta_C \{\arg[f(z)]\} + \Delta_C \left\{\arg\left[1 + \frac{g(z)}{f(z)}\right]\right\} \\ &= 2\pi N + \Delta_C \{\arg[F(z)]\}, \end{aligned}$$

where $F(z) = 1 + \frac{g(z)}{f(z)}$. To prove Rouché's theorem, we have to prove that

$$\Delta_C \{\arg[F(z)]\} = 0. \tag{C.5}$$

Since $|g(z)| < |f(z)|$ on C, the value of the complex function $F(z)$ is always inside a circle in the complex plane with center 1 and radius 1 when z describes C. Consequently,

$$-\frac{\pi}{2} < \arg[F(z)] < \frac{\pi}{2}$$

is satisfied on C. Since $\arg[F(z)]$ cannot increase or decrease by an integer multiple of 2π, it must return to its original value when z moves round the closed contour C and therefore, (C.5) is true. This proves Rouché's theorem.

Note that the condition $|g(z)| < |f(z)|$ on C is sufficient but not necessary for (C.5) to be true. As a simple example, take $g(z) = cf(z)$ with an arbitrary complex constant c. Then, $f(z) + g(z) = (1 + c)f(z)$ will always have the same zeros as $f(z)$, regardless of the magnitude of c.

C.3 A Sufficient Stability Condition

Let $A(z) = 1 + a_1 z^{-1} + \ldots + a_N z^{-N}$ be a polynomial in z^{-1} of degree N with all its roots inside the circle $|z| = \rho$. Rouché's theorem provides a condition an update polynomial $B(z) = b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}$ of degree $M \leq N$ must satisfy, such that $A(z) + B(z)$ also has all its roots inside the circle $|z| = \rho$. However, it does not provide an explicit condition on the coefficients of a polynomial such that a constraint on the moduli of its roots is satisfied. Such an explicit condition is given in this section. Before stating this condition for arbitrary degrees N, we provide conditions on the coefficients of a polynomial of degree $N \leq 2$ which are necessary and sufficient for all the zeros of the polynomial to lie inside or on the circle $|z| = \rho$.

For $N \leq 2$ it is straightforward to give such explicit conditions. Assume that the coefficients of $A(z)$ are real-valued. It is well known that the zeros of $A(z) = 1 + a_1 z^{-1} + a_2 z^{-2}$ are inside the unit circle if and only if the coefficients a_1 and a_2 are inside the stability triangle [47, 104, 113], i. e. if they satisfy

$$|a_1| < 1 + a_2, \qquad |a_2| < 1. \tag{C.6}$$

The mapping $\xi = \dfrac{z}{\rho}$ maps the circle $|z| = \rho$ to the unit circle in the ξ-plane. $A(z)$ can be written as

$$\begin{aligned} A(z) &= 1 + a_1 z^{-1} + a_2 z^{-2} \\ &= 1 + a_1 \rho^{-1} \xi^{-1} + a_2 \rho^{-2} \xi^{-2} \\ &= 1 + \tilde{a}_1 \xi^{-1} + \tilde{a}_2 \xi^{-2}. \end{aligned}$$

If the coefficients $\tilde{a}_1$ and $\tilde{a}_2$ are inside the stability triangle, all roots of $A(z)$ are inside the circle $|z| = \rho$. The corresponding conditions on the coefficients a_1 and a_2 read

$$|a_1| \leq \rho + \frac{a_2}{\rho}, \qquad |a_2| \leq \rho^2. \tag{C.7}$$

If at least one of the inequalities (C.7) is satisfied with equality, the zeros of $A(z)$ are on the circle $|z| = \rho$.

For arbitrary N, the following explicit condition guarantees that all roots of $A(z)$ are inside the unit circle (see e. g. [23]):

$$\mathrm{Re}\,\{A(z)\} > 0, \quad |z| = 1. \tag{C.8}$$

This condition is sufficient but not necessary. In [23, 74, 76], condition (C.8) has been replaced by

$$\mathrm{Re}\{A(z)\} \geq \delta, \quad |z| = 1, \tag{C.9}$$

where δ is a small positive constant which is used to ensure some stability margin. There is, however, no direct relationship between this constant δ and the maximum modulus of the roots of $A(z)$. If an explicit condition is desired which guarantees that the maximum modulus of the roots of $A(z)$ is bounded by the constant ρ, then the following condition is more appropriate:

$$\mathrm{Re}\{A(z)\} \geq 0, \quad |z| = \rho. \tag{C.10}$$

The following proof of (C.10) is a generalization of the proof of (C.8) given in [23].

Let $A(z)$ be a polynomial in z^{-1} of degree N. For $z = re^{j\varphi}$, $A(z)$ is given by

$$A(re^{j\varphi}) = \sum_{n=0}^{N} a_n r^{-n} e^{-jn\varphi}. \tag{C.11}$$

The coefficients a_n can be calculated from

$$a_n = \frac{\rho^n}{2\pi} \int_0^{2\pi} A(\rho e^{j\alpha}) e^{jn\alpha} d\alpha, \quad \rho > 0, \tag{C.12}$$

with arbitrary positive ρ. Note that the integral on the right-hand side of (C.12) vanishes for $n < 0$ and for $n > N$. Replacing the coefficients a_n in (C.11) by the expression in (C.12), $A(re^{j\varphi})$ can be written as

$$A(re^{j\varphi}) = \sum_{n=0}^{N} \left(\frac{\rho}{r}\right)^n \frac{1}{2\pi} \int_0^{2\pi} A(\rho e^{j\alpha}) e^{jn(\alpha-\varphi)} d\alpha. \tag{C.13}$$

Using the fact that the integral in (C.13) vanishes for $n < 0$ and for $n > N$, (C.13) can be written as

$$\begin{aligned} A(re^{j\varphi}) &= \frac{1}{2\pi} \int_0^{2\pi} A(\rho e^{j\alpha}) d\alpha + \sum_{n=1}^{\infty} \left(\frac{\rho}{r}\right)^n \frac{1}{2\pi} \int_0^{2\pi} A(\rho e^{j\alpha}) \left[e^{jn(\alpha-\varphi)} + e^{-jn(\alpha-\varphi)}\right] d\alpha \\ &= \frac{1}{2\pi} \int_0^{2\pi} A(\rho e^{j\alpha}) \left[1 + 2\sum_{n=1}^{\infty} \left(\frac{\rho}{r}\right)^n \cos n(\alpha - \varphi)\right] d\alpha, \quad r > \rho, \end{aligned} \tag{C.14}$$

where the integral and the sum have been interchanged in the second expression assuming that $r > \rho$ holds. For $r > \rho$, the term in brackets in the second line of (C.14) sums up to

$$1 + 2\sum_{n=1}^{\infty} \left(\frac{\rho}{r}\right)^n \cos n(\alpha - \varphi) = \frac{r^2 - \rho^2}{r^2 - 2r\rho \cos n(\alpha - \varphi) + \rho^2}, \quad r > \rho. \tag{C.15}$$

This term is easily shown to be strictly positive for $r > \rho$. Using (C.15), $A(re^{j\varphi})$ can finally be written as

$$A(re^{j\varphi}) = \frac{1}{2\pi}\int_0^{2\pi} A(\rho e^{j\alpha})\frac{r^2-\rho^2}{r^2-2r\rho\cos n(\alpha-\varphi)+\rho^2}d\alpha, \quad r > \rho. \tag{C.16}$$

From equation (C.16), any value of $A(z)$ outside the circle $|z| = \rho$ can be computed from the values of $A(z)$ on the circle $|z| = \rho$. Since the term in (C.15) is positive for $r > \rho$, the integral in (C.16) will not vanish if either $\text{Re}\{A(z)\}$ or $\text{Im}\{A(z)\}$ do not change their signs on the circle $|z| = \rho$. Therefore, $A(z)$ will not have any zeros outside the circle $|z| = \rho$ if one of the following conditions is met:

$$\begin{aligned} \text{Re}\{A(z)\} &\geq (\leq)\; 0, \qquad |z| = \rho, \\ \text{Im}\{A(z)\} &\geq (\leq)\; 0, \qquad |z| = \rho. \end{aligned} \tag{C.17}$$

The conditions on the imaginary part $\text{Im}\{A(z)\}$ cannot be satisfied by a polynomial with real-valued coefficients. Consequently, in the case of real-valued coefficients a_n, only one of the conditions on the real part $\text{Re}\{A(z)\}$ can be used. Note that both conditions on $\text{Re}\{A(z)\}$ are equivalent if used as constraints in an optimization process. The resulting transfer functions will be identical, just the signs of numerator and denominator polynomials will be different. Obviously, the conditions (C.17) are sufficient and not necessary for $A(z)$ to have no zeros outside the circle $|z| = \rho$. Note, however, that the conditions (C.17) are linear with respect to the polynomial coefficients a_n. This makes them especially attractive for the use in linear or quadratic programming methods. If a stability constraint is desired which explicitly involves the polynomial coefficients, the constraints (C.17) should be preferred over (C.9) because they allow for a direct specification of a maximum pole radius ρ.

Bibliography

[1] J. W. Adams, J. E. Nelson, J. J. Moncada, and R. W. Bayma, "FIR Digital Filter Design with Multiple Optimality Criteria and Constraints," in *Proc. IEEE Int. Symp. Circuits and Systems*, Portland, Oregon, May 1989, vol. 1, pp. 343-346.

[2] J. W. Adams, "FIR Digital Filters with Least-Squares Stopbands Subject to Peak-Gain Constraints," *IEEE Transactions on Circuits and Systems*, vol. CAS-39, pp. 376-388, April 1991.

[3] J. W. Adams, "A New Optimal Window," *IEEE Transactions on Signal Processing*, vol. SP-39, pp. 1753-1769, Aug. 1991.

[4] J. W. Adams, "Constrained Least-Squares Digital Filters," in *Proc. IEEE Int. Symp. Circuits and Systems*, San Diego, CA, May 1992, vol. 2, pp. 565-568.

[5] J. W. Adams, J. L. Sullivan, R. Hashemi, C. Ghadimi, J. Franklin, and B. Tucker, "New Approaches to Constrained Optimization of Digital Filters," in *Proc. IEEE Int. Symp. Circuits and Systems*, Chicago, May 1993, vol. 1, pp. 80-83.

[6] J. W. Adams and J. L. Sullivan, "Peak-Constrained Least-Squares Optimization," *IEEE Transactions on Signal Processing*, vol. SP-46, pp. 306-321, Feb. 1998.

[7] V. R. Algazi, M. Suk, and C.-S. Rim, "Design of Almost Minimax FIR Filters in One and Two Dimensions by WLS Techniques," *IEEE Transactions on Circuits and Systems*, vol. CAS-33, pp. 590-596, June 1986.

[8] A. S. Alkhairy, K. G. Christian, and J. S. Lim, "Design and Characterization of Optimal FIR Filters with Arbitrary Phase," *IEEE Transactions on Signal Processing*, vol. SP-41, pp. 559-572, Feb. 1993.

[9] A. Alkhairy, "A Complex Chebyshev Approximation Algorithm for FIR Filter Design," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Adelaide, Australia, April 1994, vol. 3, pp. 545-548.

[10] A. Alkhairy, "Synthesis of Optimal Nonrecursive Filters," in *Proc. IEEE Int. Symp. Circuits and Systems*, London, May/June 1994, vol. 2, pp. 557-559.

[11] J. A. Barreto and C. S. Burrus, "L_p-Complex Approximation using Iteratively Reweighted Least Squares for FIR Digital Filters," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Adelaide, Australia, April 1994, vol. 3, pp. 545-548.

[12] I. Barrodale, L. M. Delves, and J. C. Mason, "Linear Chebyshev Approximation of Complex-Valued Functions," *Mathematics of Computation*, vol. 32, pp. 853-863, July 1978.

[13] D. Burnside and T. W. Parks, "Accelerated Design of FIR Filters in the Complex Domain," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Minneapolis, Minnesota, USA, April 1993, vol. 3, pp. 81-84.

[14] D. Burnside and T. W. Parks, "Optimal Design of FIR Filters with the Complex Chebyshev Error Criteria," *IEEE Transactions on Signal Processing*, vol. SP-43, pp. 605-616, March 1995.

[15] C. S. Burrus, J. A. Barreto, and I. W. Selesnick, "Iterative Reweighted Least-Squares Design of FIR Filters," *IEEE Transactions on Signal Processing*, vol. SP-42, pp. 2926-2936, Nov. 1994.

[16] G. Calvagno, G. M. Cortelazzo, and G. A. Mian, "A Technique for Multiple Criterion Approximation of FIR Filters in Magnitude and Group Delay," *IEEE Transactions on Signal Processing*, vol. SP-43, pp. 393-400, Feb. 1995.

[17] C-K. Chen and J.-H. Lee, "Design of Digital All-Pass Filters Using a Weighted Least Squares Approach," *IEEE Transactions on Circuits and Systems*-II, vol. 41, pp. 346-351, May 1994.

[18] X. Chen and T. W. Parks, "Design of FIR Filters in the Complex Domain," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, pp. 144-153, Feb. 1987.

[19] X. Chen and T. W. Parks, "Design of IIR Filters in the Complex Domain," *IEEE Trans. Acoust., Speech, and Signal Processing*, vol. ASSP-38, pp. 910-920, June 1990.

[20] E. W. Cheney, *Introduction to Approximation Theory*, McGraw-Hill, New York, 1966.

[21] C.-Y. Chi and S.-L. Chiou, "A New WLS Chebyshev Approximation Method for the Design of FIR Digital Filters with Arbitrary Complex Frequency Response," *Signal Processing*, vol. 29, pp. 335-347, Dec. 1992.

[22] N. N. Chit and J. S. Mason, "Complex Chebyshev Approximation for FIR Digital Filter Design," *IEEE Transactions on Signal Processing*, vol. SP-39, pp. 49-54, Jan. 1991.

[23] A. T. Chottera and G. A. Jullien, "A Linear Programming Approach to Recursive Digital Filter Design with Linear Phase,", *IEEE Trans. Circuits and Systems*, vol. CAS-29, March 1982, pp. 139-149.

[24] A. K. Cline, "Rate of Convergence of Lawson's algorithm," *Mathematics of Computation*, vol. 26, pp. 167-176, 1972.

[25] L. Collatz and W. Wetterling, *Optimierungsaufgaben*, Springer-Verlag, Berlin, 2nd Edition, 1971.

[26] G. Cortelazzo and M. R. Lightner, "Simultaneous Design in Both Magnitude and Group-Delay of IIR and FIR Filters Based on Multiple Criterion Optimization," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-32, pp. 949-967, Oct. 1984.

[27] G. Cortelazzo, "On the Role of Input Signals in Deterministic Linear Filtering," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, pp. 203-205, Feb. 1986.

[28] G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, N. J. , 1963.

[29] A. G. Deczky, "Synthesis of Recursive Digital Filters Using the Minimum p-Error Criterion," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-20, pp. 257-263, 1972.

[30] A. G. Deczky, *Computer Aided Synthesis of Digital Filters in the Frequency Domain*, Ph.D. thesis, Swiss Federal Institute of Technology, Zurich, 1973.

[31] J. Durbin, "The Fitting of Time Series Models," *Rev. Instrum. Int. Statistics*, vol. 28, pp 233-244, 1960.

[32] S. Ellacott and J. Williams, "Linear Chebyshev Approximation in the Complex Plane Using Lawson's Algorithm," *Mathematics of Computation*, vol. 30, pp. 35-44, Jan. 1976.

[33] S. Ellacott and J. Williams, "Rational Chebyshev Approximation in the Complex Plane," *SIAM J. Numer. Anal.*, vol. 13, no. 3, June 1976, pp. 310-323.

[34] R. Fletcher, *Practical Methods of Optimization*, Wiley, Chichester, 2nd Edition, 1987.

[35] M. Gerken, H. W. Schüßler, and P. Steffen, "On the Design of Recursive Digital Filters Consisting of a Parallel Connection of Allpass Sections and Delay Elements," *AEÜ*, vol. 49, pp. 1-11, 1995.

[36] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, San Diego, 1981.

[37] K. Glashoff and K. Roleff, "A New Method for Chebyshev Approximation of Complex-Valued Functions," *Mathematics of Computation*, vol. 39, pp. 233-239, Jan. 1981.

[38] G. Haberstumpf, P. Möhringer, H. W. Schüßler, and P. Steffen, "On Anti-Aliasing Filters with Chebyshev Behaviour in the Passband," *AEÜ*, vol. 35, pp. 489-492, 1981.

[39] O. Herrmann, "Design of Nonrecursive Digital Filters with Linear Phase," *Electronics Letters*, vol. 6, pp. 328-329, 1970.

[40] O. Herrmann and H. W. Schüßler "Design of Nonrecursive Digital Filters with Minimum Phase," *Electronics Letters*, vol. 6, pp. 329-330, 1970.

[41] O. Herrmann and H. W. Schüßler "On the Design of Nonrecursive Digital Filters," *IEEE Arden House Workshop*, Jan. 1970.

[42] R. Hettich and P. Zencke, *Numerische Methoden der Approximation und semi-infiniten Optimierung*, Teubner, Stuttgart, 1982.

[43] E. Hofstetter, A. Oppenheim and J. Siegel, "A New Technique for the Design of Nonrecursive Digital Filters," in *Proc. of the Fifth Annual Princeton Conference on Information Sciences and Systems*, pp. 64-72, 1971.

[44] R. Horst, P. M. Pardalos, and N. V. Thoai, *Introduction to Global Optimization*, Kluwer, Dordrecht, 1995.

[45] T. Inukai, "A Unified Approach to Optimal Recursive Digital Filter Design," *IEEE Transactions on Circuits and Systems*, vol. CAS-27, pp. 646-649, July 1980.

[46] L. B. Jackson, "An Improved Martinez/Parks Algorithm for IIR Design with Unequal Numbers of Poles and Zeros," *IEEE Trans. Signal Processing*, vol. 42, pp. 1234-1238, May 1994.

[47] L. B. Jackson, *Digital Filters and Signal Processing*, Kluwer Academic Publishers, 3rd Edition, 1996.

[48] L. J. Karam and J. H. McClellan, "A Multiple Exchange Remez Algorithm for Complex FIR Filter Design in the Chebyshev Sense," in *Proc. IEEE Int. Symp. Circuits and Systems*, London, May/June 1994, vol. 2, pp. 517-520.

[49] L. J. Karam and J. H. McClellan, "Complex Chebyshev Approximation for FIR Filter Design," *IEEE Transactions on Circuits and Systems*-II, vol. 42, pp. 207-216, March 1995.

[50] L. J. Karam and J. H. McClellan, "Design of Optimal Digital FIR Filters with Arbitrary Magnitude and Phase Responses," in *Proc. IEEE Int. Symp. Circuits and Systems*, Atlanta, May 1996, vol. 2, pp. 385-388.

[51] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorics*, **4**, pp. 373-395.

[52] M. Z. Komodromos, S. F. Russell, and P. T. P. Tang, "Design of FIR Filters with Complex Desired Frequency Response Using a Generalized Remez Algorithm," *IEEE Transactions on Circuits and Systems*-II, vol. 42, pp. 274-278, April 1995.

[53] M. Z. Komodromos, S. F. Russell, and P. T. P. Tang, "Design of FIR Hilbert Transformers and Differentiators in the Complex Domain," *IEEE Transactions on Circuits and Systems*-I, vol. 45, pp. 64-67, Jan. 1998.

[54] M. Lang, *Ein Beitrag zur Phasenapproximation mit Allpässen*, Ph.D. Thesis, Universität Erlangen-Nürnberg, Erlangen, Germany, 1993.

[55] M. Lang and J. Bamberger, "Nonlinear Phase FIR Filter Design According to the L_2 Norm with Constraints for the Complex Error," *Signal Processing*, vol. 36, pp. 31-40, March 1994.

[56] M. Lang and T. I. Laakso, "Simple and Robust Method for the Design of Allpass Filters Using Least-Squares Phase Error Criterion," *IEEE Transactions on Circuits and Systems*-II, vol. 41, pp. 40-48, Jan. 1994.

[57] M. Lang, "Allpass Filter Design and Applications," *IEEE Transactions on Signal Processing*, vol. SP-46, pp. 2505-2514, Sept. 1998.

[58] M. C. Lang, "Chebyshev Design of FIR Filters with Arbitrary Magnitude and Phase Responses," *Signal Processing VIII (EUSIPCO'96)*, Trieste, Italy, Sept. 1996, vol. 3, pp. 1757-1760.

[59] M. C. Lang, "Design of Nonlinear Phase FIR Digital Filters Using Quadratic Programming," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Munich, April 1997, vol. 3, pp. 2169-2172.

[60] M. C. Lang, "Reweighted Least Squares Design of FIR Filters in the Complex Domain," in *Proc. 4th International Workshop on Systems, Signals and Image Processing*, Poznan, May 1997, pp. 17-20.

[61] M. C. Lang, "Constrained Least Square Design of FIR Filters with Arbitrary Magnitude and Phase Responses," in *Proc. IEEE Int. Symp. Circuits and Systems*, Hong Kong, June 1997, vol. 4, pp. 2264-2267.

[62] M. C. Lang, "Weighted Least Squares Design of FIR Filters Subject to Magnitude and Phase Constraints," in *Proc. 13th Int. Conf. Digital Signal Processing*, Santorini, Greece, July 1997, pp. 451-454.

[63] M. C. Lang, "An Iterative Reweighted Least Squares Algorithm for Constrained Design of Nonlinear Phase FIR Filters," in *Proc. IEEE Int. Symp. Circuits and Systems*, Monterey (CA), May/June 1998.

[64] M. C. Lang, "Weighted Least Squares IIR Filter Design with Arbitrary Magnitude and Phase Responses and Specified Stability Margin," in *Proc. IEEE Symp. on Digital Filtering and Signal Processing*, Victoria, BC, Canada, June 1998, pp. 82-86.

[65] M. C. Lang, "Least Squares Design of IIR Filters with Arbitrary Magnitude and Phase Responses and Specified Stability Margin," *Signal Processing IX (EUSIPCO'98)*, Rhodes, Greece, Sept. 1998, vol. III, pp. 1909-1912.

[66] M. C. Lang, "A Multiple Exchange Algorithm for Constrained Design of FIR Filters in the Complex Domain," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Phoenix, Arizona, March 1999, vol. 3, pp. 1149-1152.

[67] C. L. Lawson, *Contributions to the Theory of Linear Least Maximum Approximations*, Ph.D. Thesis, University of California, Los Angeles, 1961.

[68] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice Hall, Englewood Cliffs, New Jersey, 1974.

[69] S. S. Lawson, "On Design Techniques for Approximately Linear Phase Recursive Digital Filters," in *Proc. IEEE Int. Symp. Circuits and Systems*, Hong Kong, June 1997, vol. 4, pp. 2212-2215.

[70] N. Levinson, "The Wiener RMS Criterion in Filter Design and Prediction," *J. Math. Phys.*, vol. 25, pp. 261-278, Jan. 1947.

[71] J. T. Lewis, R. Murphy, and D. W. Tufts, "Design of Minimum Noise Digital Filters Subject to Inequality Constraints Using Quadratic Programming," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, pp. 434-436, Oct. 1976.

[72] Y. C. Lim, J.-H. Lee, C. K. Chen, and R.-H. Yang, "A Weighted Least Squares Algorithm for Quasi-Equiripple FIR and IIR Digital Filter Design," *IEEE Trans. Signal Processing*, vol. SP-40, pp. 551-558, March 1992.

[73] G. G. Lorentz, *Approximation of Functions*, Holt, Rinehart and Winston, New York, 1966.

[74] W.-S. Lu, "Design of Stable IIR Digital Filters with Equiripple Passbands and Peak-Constrained Least Squares Stopbands," in *Proc. IEEE Int. Symp. Circuits and Systems*, Hong Kong, June 1997, vol. 4, pp. 2192-2195.

[75] W.-S. Lu, "A Parameterization Method for the Design of IIR Digital Filters with Prescribed Stability Margin," in *Proc. IEEE Int. Symp. Circuits and Systems*, Hong Kong, June 1997, vol. 4, pp. 2188-2191.

[76] W.-S. Lu, "A Weighted Least-Squares Method for the Design of Stable 1-D and 2-D IIR Digital Filters," *IEEE Transactions on Signal Processing*, vol. SP-46, pp. 1-10, Jan. 1998.

[77] W.-S. Lu, "Design of Recursive Digital Filters with Prescribed Stability Margin: A Parameterization Approach," *IEEE Transactions on Circuits and Systems*-II, vol. 45, pp. 1289-1298, Sept. 1998.

[78] S. L. Marple Jr., *Digital Spectral Analysis*, Prentice-Hall, Englewood Cliffs (NJ), 1987.

[79] H. G. Martinez and T. W. Parks, "Design of Recursive Digital Filters with Optimum Magnitude and Attenuation Poles on the Unit Circle," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-26, pp. 150-156, April 1978.

[80] M. T. McCallig and B. J. Leon, "Constrained Ripple Design of FIR Digital Filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-25, pp. 893-902, Nov. 1978.

[81] M. T. McCallig, "Design of Digital FIR Filters with Complex Conjugate Pulse Responses," *IEEE Transactions on Circuits and Systems*, vol. CAS-25, pp. 1103-1105, Dec. 1978.

[82] J. H. McClellan and T. W. Parks, "A Unified Approach to the Design of Optimum FIR Linear-Phase Digital Filters," *IEEE Trans. Circuit Theory*, vol. CT-20, pp. 697-701, November 1973.

[83] J. H. McClellan, T. W. Parks, and L. R. Rabiner, "A Computer Program for Designing Optimum FIR Linear Phase Digital Filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 506-526, Dec. 1973.

[84] G. A. Merchant and T. W. Parks, "Efficient Solution of a Toeplitz-Plus-Hankel coefficient matrix system of equations," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-30, pp. 40-44, Feb. 1982.

[85] S. K. Mitra and K. Hirano, "Digital All-Pass Networks," *IEEE Transactions on Circuits and Systems*, vol. CAS-21, pp. 688-700, 1974.

[86] M. Okuda, M. Ikehara, and S. Takahashi, "Fast and Stable Least-Squares Approach for the Design of Linear Phase FIR Filters," *IEEE Transactions on Signal Processing*, vol. SP-46, pp. 1485-1493, June 1998.

[87] T. W. Parks, C. S. Burrus, *Digital Filter Design*, John Wiley & Sons, New York, 1987.

[88] T. W. Parks and J. H. McClellan, "Chebyhsev Approximation for Nonrecursive Digital Filters with Linear Phase," *IEEE Transactions on Circuit Theory*, vol. CT-19, pp. 189-194, March 1972.

[89] A. W. Potchinkov, *Der Entwurf digitaler FIR-Filter mit Methoden der semiinfiniten konvexen Optimierung*, Ph.D. Thesis, Technische Universität Berlin, Berlin, Germany, 1994.

[90] A. W. Potchinkov and R. M. Reemtsen, "FIR Filter Design in the Complex Domain by a Semi-Infinite Programming Technique," Parts I&II, *AEÜ*, vol. 48, pp. 135-144 and 200-209, 1994.

[91] A. W. Potchinkov and R. M. Reemtsen, "The Design of FIR Filters in the Complex Plane by Convex Optimization," *Signal Processing*, vol. 46, pp. 127-146, 1995.

[92] A. W. Potchinkov and R. M. Reemtsen, "The Simultaneous Approximation of Magnitude and Phase by FIR Digital Filters," Parts I&II, *Journal of Circuit Theory and Applications*, vol. 25, pp. 167-197, 1997.

[93] A. W. Potchinkov, "Design of optimal linear phase FIR filters by a semi-infinite programming technique," *Signal Processing*, vol. 58, pp. 165-180, 1997.

[94] K. Preuss, "A Novel Approach for Complex Chebyshev-Approximation with FIR-Filters Using the Remez Exchange Algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, , April 1987, vol. 2, pp. 872-875.

[95] K. Preuss, "On the Design of FIR Filters by Complex Chebyshev Approximation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-37, pp. 702-712, May 1989.

[96] F. Qin, "A Practical Method for Designing FIR Digital Filters in the Complex Domain," *IEEE Transactions on Signal Processing*, vol. SP-45, pp. 2092-2096, Aug. 1997.

[97] L. R. Rabiner, "Linear Program Design of Finite Impulse Response (FIR) Digital Filters," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-20, pp. 280-288, Oct. 1972.

[98] L. R. Rabiner, J. F. Kaiser, O. Herrmann, and M. T. Dolan, "Some Comparisons Between FIR and IIR Digital Filters," *The Bell System Technical Journal*, vol. 53, pp. 305-331, Feb. 1974.

[99] R. Reemtsen, "Some outer approximation methods for semi-infinite optimization problems," *Journal of Computational and Applied Mathematics*, 53, pp. 87-108, 1994.

[100] R. Reemtsen and St. Görner, "Numerical Methods for Semi-Infinite Programming: A Survey", in *Semi-Infinite Programming*, R. Reemtsen and J.-J. Rückmann, ed., pp. 195-275, Kluwer Academic Publishers, Boston-London-Dordrecht, 1998.

[101] P. A. Regalia, S. K. Mitra, and P. P. Vaidyanathan, "The Digital All-Pass Filter: A Versatile Signal Processing Building Block," Proceedings of the IEEE, vol. 76, pp. 19-37, Jan. 1988.

[102] J. R. Rice, *The Approximation of Functions*, Volume 2, Addison-Wesley, Reading (MA), 1969.

[103] T. J. Rivlin, *An Introduction to the Approximation of Functions*, Dover, New York, 1981.

[104] R. A. Roberts and C. T. Mullis, *Digital Signal Processing*, Addison-Wesley, Reading (MA), 1987.

[105] K. Roleff, "A stable multiple exchange algorithm for linear SIP", in *Semi-infinite programming*, Proc. of a Workshop, Bad Honnef, Aug./Sept. 1978, R. Hettich, ed., pp. 83-96, Springer, Berlin-Heidelberg-New York, 1979.

[106] T. Saramäki, "Design of Optimum Recursive Digital Filters with Zeros on the Unit Circle," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-31, pp. 450-458, April 1983.

[107] T. Saramäki, "On the Design of Digital Filters as a Sum of Two All-Pass Filters," *IEEE Transactions on Circuits and Systems*, vol. CAS-32, pp. 1191-1193, Nov. 1985.

[108] M. Schulist, "Improvements of a Complex FIR Filter Design Algorithm," *IEEE Transactions on Signal Processing*, vol. SP-20, pp. 81-90, May 1990.

[109] M. Schulist, *Ein Beitrag zum Entwurf nichtrekursiver Filter*, Ph.D. Thesis, Universität Erlangen-Nürnberg, Erlangen, Germany, 1992.

[110] M. Schulist, "Complex Approximation with Additional Constraints," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, San Francisco, March 1992, vol. 5, pp. 101-104.

[111] M. Schulist, "FIR Filter Design with Additional Constraints Using Complex Chebyshev Approximation," *Signal Processing*, vol. 33, pp. 111-119, 1993.

[112] H. W. Schüßler, *Digitale Systeme zur Signalverarbeitung* (in German), Springer, Berlin, 1973.

[113] H. W. Schüßler, *Digitale Signalverarbeitung 1* (in German), Fourth Edition, Springer, Berlin, 1994.

[114] H. W. Schüßler, *Digitale Signalverarbeitung 2* (in German), Springer, in preparation.

[115] H. W. Schüßler, P. Möhringer, and P. Steffen, "On Partly Digital Anti-Aliasing Filters," *AEÜ*, vol. 36, pp. 349-355, 1982.

[116] H. W. Schüßler and P. Steffen, "Some Advanced Topics in Filter Design," in *Advanced Topics in Signal Processing*, J. S. Lim and A. V. Oppenheim, ed., pp. 416-491, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[117] I. W. Selesnick, "Magnitude Squared Design of Recursive Filters with the Chebyshev Norm Using a Constrained Rational Remez Algorithm," *IEEE DSP Workshop*, Oct. 1994.

[118] I. W. Selesnick, *New Techniques for Digital Filter Design*, Ph.D. Thesis, Rice University, Houston, Texas, 1996.

[119] I. W. Selesnick, M. Lang, and C. S. Burrus, "Constrained Least Square Design of FIR Filters without Specified Transition Bands," *IEEE Transactions on Signal Processing*, vol. SP-44, pp. 1879-1892, August 1996.

[120] I. W. Selesnick and C. S. Burrus, "Exchange Algorithms that Complement the Parks-McClellan Algorithm for Linear-Phase FIR Filter Design," *IEEE Transactions on Circuits and Systems*-II, vol. 44, pp. 137-143, Feb. 1997.

[121] I. W. Selesnick and C. S. Burrus, "Maximally Flat Low-pass FIR Filters with Reduced Delay," *IEEE Transactions on Circuits and Systems*-II, vol. 45, pp. 53-68, Jan. 1998.

[122] M. A. Sid-Ahmed, A. Chottera, and G. A. Jullien, "Computational Techniques for Least-Square Design of Recursive Digital Filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-26, pp. 477-480, Oct. 1978.

[123] P. Steffen, H. W. Schüßler, and P. Möhringer, "On Optimum Anti-Aliasing Filters," *AEÜ*, vol. 35, pp. 185-191, 1981.

[124] K. Steiglitz, "Design of FIR Digital Phase Networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-29, pp. 171-176, April 1981.

[125] K. Steiglitz, T. W. Parks, and J. F. Kaiser, "METEOR: A Constraint-Based FIR Filter Design Program," *IEEE Transactions on Signal Processing*, vol. SP-40, pp. 1901-1909, Aug. 1992.

[126] R. L. Streit and A. H. Nuttall, "A Note on the Semi-Infinite Programming Approach to Complex Approximation," *Mathematics of Computation*, vol. 40, pp. 599-605, April 1983.

[127] R. L. Streit and A. H. Nuttall, "A General Chebyshev Complex Function Approximation Procedure and an Application fo Beamforming," *Journal of the Acoustical Society of America*, vol. 72, pp. 181-190, July 1982.

[128] J. L. Sullivan and J. W. Adams, "A New Nonlinear Optimization Algorithm for Asymmetric FIR Digital Filters," *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 2, pp. 541-544, May-June 1994.

[129] J. L. Sullivan and J. W. Adams, "A Nonlinear Optimization Algorithm for Asymmetric FIR Digital Filters," *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 2, pp. 869-872, April-May 1995.

[130] P. T. P. Tang, "A Fast Algorithm for Linear Complex Chebyshev Approximations," *Mathematics of Computation*, vol. 51, pp. 721-739, Oct. 1988.

[131] E. C. Titchmarsh, *The Theory of Functions*, Second Edition, Oxford University Press, Oxford, 1979.

[132] C.-Y. Tseng, "A Numerical Algorithm for Complex Chebyshev FIR Filter Design," in *Proc. IEEE Int. Symp. Circuits and Systems*, San Diego, CA, May 1992, vol. 2, pp. 549-552.

[133] C.-Y. Tseng, "A Generalized Remez Multiple Exchange Algorithm for Complex FIR Filter Design," in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Adelaide, Australia, April 1994, vol. 3, pp. 553-556.

[134] C.-Y. Tseng, "An Efficient Implementation of Lawson's Algorithm with Application to Complex Chebyshev FIR Filter Design," *IEEE Transactions on Circuits and Systems*-II, vol. 42, pp. 245-260, April 1995.

[135] C.-Y. Tseng, "A Multiple-Exchange Algorithm for Complex Chebyshev Approximation by Polynomials on the Unit Circle," *SIAM Journal on Numerical Analysis*, vol. 33, pp. 2017-2049, Oct. 1996.

[136] A. W. M. Van den Enden and G. A. L. Leenknegt, "Design of Optimal IIR Filters with Arbitrary Amplitude and Phase Requirements," *Signal Processing III: Theories and Applications, EUSIPCO-86*, 1986, pp. 183-186.

[137] P. P. Vaidyanathan, S. K. Mitra, and Y. Neuvo, "A New Approach to the Realization of Low-Sensitivity IIR Digital Filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, pp. 350-361, April 1986.

[138] R. Vuerinckx, Y. Rolain, J. Schoukens, and R. Pintelon, "Design of Stable IIR Filters in the Complex Domain by Automatic Delay Selection," *IEEE Transactions on Signal Processing*, vol. SP-44, pp. 2339-2344, Sept. 1996.

[139] R. Vuerinckx, "Design of High-Order Chebyshev FIR Filters in the Complex Domain Under Magnitude Constraints," *IEEE Transactions on Signal Processing*, vol. SP-46, pp. 1676-1681, June 1998.

[140] G. A. Watson, "A Multiple Exchange Algorithm for Multivariate Chebyshev Approximation," *SIAM Journal on Numerical Analysis*, vol. 12, pp. 46-52, March 1975.

[141] B. A. Weisburn, T. W. Parks, and R. G. Shenoy, "Error Criteria for Filter Design," in in *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Adelaide, Australia, April 1994, vol. 3, pp. 565-568.

[142] R. H. Yang and Y. C. Lim, "Efficient Computational Procedure for the Design of FIR Digital Filters Using WLS Technique," *IEE Proc. G*, vol. 140, no. 5, pp. 355-359, Oct. 1993.