Research Article

An Efficient Constrained Learning Algorithm for Stable 2D IIR Filter Factorization

Nicholas Ampazis¹ and Stavros J. Perantonis²

¹ Department of Financial and Management Engineering, University of the Aegean, 82100 Chios, Greece

² Institute of Informatics and Telecommunications, NCSR “Demokritos”, 15310 Athens, Greece

Correspondence should be addressed to Nicholas Ampazis; n.ampazis@fme.aegean.gr

Received 9 November 2012; Accepted 12 January 2013

Academic Editor: Tingwen Huang

Copyright © 2013 N. Ampazis and S. J. Perantonis. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A constrained neural network optimization algorithm is presented for factorizing simultaneously the numerator and denominator polynomials of the transfer functions of 2-D IIR filters. The method minimizes a cost function based on the frequency response of the filters, along with simultaneous satisfaction of appropriate constraints, so that factorization is facilitated and the stability of the resulting filter is respected.

1. Introduction

Factorization of 2D polynomials is an important problem in the design of IIR filters in two-dimensional signal processing because a factorized filter transfer function can be efficiently implemented in cascade form. However, the fundamental theorem of Algebra, concerning polynomial factorization, cannot be extended to the two-dimensional case; that is, a bivariate polynomial $F(z_1, z_2)$ cannot, in general, be factored as a product of lower-order polynomials. The 2D IIR cascade structure may be considered as an attractive circuit design alternative, because of its relative insensitivity to coefficient quantization, the requirement of smaller number of arithmetic operations for a given filter size, and finally, because issues related to the stability of the filters are easier to deal with for filters with a smaller number of denominator coefficients [1]. Hence, an efficient method for the factorization of 2D IIR filters would be most beneficial.

Many methods have been proposed for dealing with the two-dimensional polynomial factorization problem [2–6]. Most of those methods, however, adopt a conventional numerical method of roots finding (e.g., Laguerre’s, Newton-Raphson’s, Jenkins-Traub’s methods, etc.), in which successive approximations to the roots are obtained. In addition, almost all numerical methods can only find the roots one by one, that

is, by deflation method, where the next root is obtained by the deflated polynomial after the former root is found. This means that numerical root-finding methods are inherently sequential.

In an earlier work we have proposed a neural network-based Constrained Learning Algorithm (CLA) for factorizing 2D polynomials using constrained optimization techniques [7]. Using that approach we were able to obtain exact solutions for factorable polynomials and excellent approximate solutions for nonfactorable polynomials. The technique offers the advantage of incorporating a priori information about the relations between the coefficients of the original polynomial and the coefficients of the desired factor polynomials. By incorporating additional stability constraints into the formalism, the stability (in the sense of Huang [8]) of a transfer function with the resulting factorized polynomial in the denominator (of up to second order) is also made possible. Our CLA method has been explored further in [9] by utilizing root moments, in [10] with an adaptive neural network architecture, and in [11] for handling complex root cases.

Despite its success in 2D polynomial factorization, CLA suffers from disadvantages which can limit its effectiveness for its application to IIR filter factorization. CLA’s cost function is not directly related to a proximity measure

between the frequency response characteristics of the filters, but is based on the Mean Square Error (MSE) criterion between the output of the original polynomial and the output of the resulting factorable polynomial for various real values of $|z_1| < 1$ and $|z_2| < 1$. Thus, CLA requires separate factorizations for the numerator and denominator polynomials of the IIR filter, while better solutions, in terms of the proximity of filter characteristics, could be achieved by simultaneous variation of all available parameters (numerator and denominator coefficients). In this paper we apply constrained optimization techniques to the more general problem of implementing a stable cascade IIR filter, given the specifications of an original 2D filter. This is achieved by suitably modifying our previous formalism as it is explained in the following section.

2. Derivation of the Algorithm

Consider the polynomial

$$f(z_1, z_2) = \sum_{i=0}^{N_A} \sum_{j=0}^{N_A} a_{ij} z_1^i z_2^j \quad (1)$$

with N_A even and $a_{00} = 1$. For the above polynomial, we seek to achieve an exact or approximate factorization of the form

$$f(z_1, z_2) \approx \prod_{i=1,2} f^{(i)}(z_1, z_2), \quad (2)$$

where

$$f^{(i)}(z_1, z_2) = \sum_{j=0}^{M_A} \sum_{k=0}^{M_A} w_{jk}^{(i)} z_1^j z_2^k \quad (3)$$

with $M_A = N_A/2$.

To this end, we can utilize a feedforward neural network architecture in order to learn the appropriate weights $w_{jk}^{(i)}$ as we proposed in [7]. For example, Figure 1 shows the architecture of such a network for the case where $N_A = 2$. The units in the hidden layer have a logarithmic activation function $\ln$, and the output unit is linear. This architecture has been preferred to the $\sum$ - $\prod$ architecture employed in [12], because it produces smoother cost function landscapes avoiding deep valleys and thus facilitates learning. As we showed in [7], this neural network can be trained using a Constrained Learning Algorithm (CLA) that achieves minimization of the MSE criterion along with simultaneous satisfaction of multiple equality and inequality constraints between the polynomial coefficients. Using that method, we are able to obtain very good approximate solutions for nonfactorable polynomials.

Now consider the 2D IIR filter transfer function

$$H(z_1, z_2) = \frac{B(z_1, z_2)}{A(z_1, z_2)}, \quad (4)$$

FIGURE 1: Neural network architecture used to solve the factorization problem.

where

$$B(z_1, z_2) = \sum_{i=0}^{N_B} \sum_{j=0}^{N_B} b_{ij} z_1^i z_2^j, \quad (5)$$

$$A(z_1, z_2) = \sum_{i=0}^{N_A} \sum_{j=0}^{N_A} a_{ij} z_1^i z_2^j$$

with N_A, N_B even and $b_{00} = a_{00} = 1$.

For the above polynomials, we seek to achieve an exact or approximate factorization of the form

$$B(z_1, z_2) \approx \prod_{i=1,2} B^{(i)}(z_1, z_2), \quad (6)$$

where

$$B^{(i)}(z_1, z_2) = \sum_{j=0}^{M_B} \sum_{k=0}^{M_B} u_{jk}^{(i)} z_1^j z_2^k, \quad (7)$$

$$A(z_1, z_2) \approx \prod_{i=1,2} A^{(i)}(z_1, z_2), \quad (8)$$

where

$$A^{(i)}(z_1, z_2) = \sum_{j=0}^{M_A} \sum_{k=0}^{M_A} v_{jk}^{(i)} z_1^j z_2^k \quad (9)$$

with $M_A = N_A/2$, and $M_B = N_B/2$ so that the transfer function of (4) can be approximated by

$$\widehat{H}(z_1, z_2) = \frac{\prod_{i=1,2} B^{(i)}(z_1, z_2)}{\prod_{i=1,2} A^{(i)}(z_1, z_2)}. \quad (10)$$

Unfortunately, if it were to apply CLA in this problem we would have to derive two separate factorizations for the numerator and denominator polynomials. Such an optimization scheme, however, is not directly related to a proximity

measure between the original and the factorized filter frequency response characteristics. In terms of the proximity of filter characteristics, obviously a better approximation can be achieved by a simultaneous variation of both the numerator and denominator coefficients.

In this case, we should ask: in what sense are $H(z_1, z_2)$ and $\widehat{H}(z_1, z_2)$ similar? The answer to this question is that we would at least expect the amplitude spectra of $H(z_1, z_2)$ and $\widehat{H}(z_1, z_2)$ to be roughly equal so that the specifications of the original and of the factorized filter match as close as possible. Hence, the main purpose of the factorization algorithm is to minimize an MSE cost function based on the magnitude of the frequency responses of the original and the resulting filter:

$$\begin{aligned} E &= \frac{1}{2} \sum_{\theta_1=-\pi}^{\pi} \sum_{\theta_2=-\pi}^{\pi} (T - O)^2, \\ T &= \ln \left(\left| H(e^{j\theta_1}, e^{j\theta_2}) \right| \right), \\ O &= \ln \left(\left| \widehat{H}(e^{j\theta_1}, e^{j\theta_2}) \right| \right). \end{aligned} \quad (11)$$

The double summation is carried over a regular grid of points. Logarithmic functions are used also in this case in order to suppress deep valleys and thus facilitate convergence. However, E is still a highly nonlinear function of the polynomial coefficients, and therefore gradient-based techniques can run into problems in the presence of valleys or ridges in the cost function landscape. For such cases it has proved beneficial to incorporate into the adaptation rule additional a priori problem-specific information. This often facilitates the convergence of the algorithm and leads to better solutions.

For the problem at hand, additional information is available in the form of constraints between the coefficients of the given numerator and denominator polynomials and the coefficients of the corresponding desired factor polynomials. More explicitly, if we assume that $B(z_1, z_2)$ and $A(z_1, z_2)$ are factorable, then these constraints can be expressed as follows:

$$\begin{aligned} \Phi_{j+(N_B+1)i}^u &= b_{ij} - \sum_{l=1}^i \sum_{m=1}^j u_{lm}^{(1)} u_{i-l, j-m}^{(2)} = 0, \quad 0 \leq i \leq N_B, \quad 0 \leq j \leq N_B, \\ \Phi_{j+(N_A+1)i}^v &= a_{ij} - \sum_{l=1}^i \sum_{m=1}^j v_{lm}^{(1)} v_{i-l, j-m}^{(2)} = 0, \quad 0 \leq i \leq N_A, \quad 0 \leq j \leq N_A. \end{aligned} \quad (12)$$

Thus, for the general case in which $B(z_1, z_2)$ and $A(z_1, z_2)$ are nonfactorable, the objective of the adaptation process is to reach a minimum of the cost function of (11) with respect

to the variables $u_{jk}^{(i)}$ and $v_{jk}^{(i)}$, which satisfies as best as possible the constraint $\Phi = \mathbf{0}$, where $\Phi^T = (\Phi^u, \Phi^v)$ with

$$\begin{aligned} \Phi^u &= \left(\Phi_{j+(N_B+1)i}^u, 0 \leq i \leq N_B, 0 \leq j \leq N_B \right), \\ \Phi^v &= \left(\Phi_{j+(N_A+1)i}^v, 0 \leq i \leq N_A, 0 \leq j \leq N_A \right). \end{aligned} \quad (13)$$

The strategy which will be followed in order to solve this problem is similar to the method we proposed in [7], but with appropriate modifications in order to take care of the above constraints, as well as the new cost function that we introduced in (11).

At each iteration of our algorithm, the vector of unknown coefficients $\mathbf{w} = (\mathbf{u}, \mathbf{v})^T$ is changed by $d\mathbf{w}$, so that

$$d\mathbf{w}^T d\mathbf{w} = (\delta P)^2, \quad (14)$$

where δP is a small constant. Thus, at each iteration, the search for an optimum new point in the $\mathbf{w}$ space is restricted to a small hypersphere centered at the point defined by the current vector $\mathbf{w}$. If δP is small enough, the changes to E and to Φ induced by changes in the coefficients can be approximated by the first differentials dE and $d\Phi$. Also, at each iteration, we seek to achieve the maximum possible change in $|dE|$, so that (14) is respected and the change $d\Phi$ in Φ is equal to a predetermined vector quantity $\delta\mathbf{Q}$, designed to bring Φ closer to its target (zero).

Maximization of dE can be carried out analytically by introducing suitable Lagrange multipliers. Thus, a vector $\mathbf{v}$ of Lagrange multipliers is introduced for the constraints in (12) and a multiplier μ is introduced for (14).

Consider a function ε , whose differential is defined as follows:

$$d\varepsilon = dE + (\delta\mathbf{Q}^T - d\Phi^T) \mathbf{v} + \mu [(\delta P)^2 - d\mathbf{w}^T d\mathbf{w}]. \quad (15)$$

On evaluating the differentials involved in the right-hand side, we readily obtain

$$d\varepsilon = d\mathbf{w}^T \mathbf{J} + (\delta\mathbf{Q}^T - d\mathbf{w}^T \mathbf{F}^T) \mathbf{v} + \mu [(\delta P)^2 - d\mathbf{w}^T d\mathbf{w}], \quad (16)$$

where

- (i) $\mathbf{J}$ is a vector with elements $J_i = \partial E / \partial w_i$;
- (ii) $\mathbf{F}$ is a matrix given by $\mathbf{F} = \begin{bmatrix} \mathbf{F}^u & \mathbf{0} \\ \mathbf{0} & \mathbf{F}^v \end{bmatrix}$;
- (iii) $\mathbf{F}^u$ and $\mathbf{F}^v$ are matrices with elements $F_{mi}^u = \partial \Phi_m^u / \partial u_i$ and $F_{mi}^v = \partial \Phi_m^v / \partial v_i$, respectively.

To minimize $d\varepsilon$ (maximize its magnitude) at each iteration, we demand that

$$d^2 \varepsilon = \mathbf{J} - \mathbf{F}^T \mathbf{v} - 2\mu d\mathbf{w} = 0, \quad d^3 \varepsilon = -2\mu > 0. \quad (17)$$

From (17) it immediately follows that

$$d\mathbf{w} = \frac{\mathbf{J}}{2\mu} - \frac{\mathbf{F}^T \mathbf{v}}{2\mu}. \quad (18)$$

Equation (18) is the update rule for the coefficients of the factor polynomials, but μ and $\mathbf{v}$ must still be evaluated in terms of known quantities.

Noting that $d\Phi = \delta\mathbf{Q} = \mathbf{F}d\mathbf{w}$, we can multiply both sides of (18) by $\mathbf{F}$ and solve for $\mathbf{v}$ to obtain

$$\mathbf{v} = -2\mu\mathbf{I}_{FF}^{-1}\delta\mathbf{Q} + \mathbf{I}_{FF}^{-1}\mathbf{I}_{JF}, \quad \text{where } \mathbf{I}_{FF} = \mathbf{F}\mathbf{F}^T, \mathbf{I}_{JF} = \mathbf{F}\mathbf{J}. \quad (19)$$

To evaluate μ , we substitute (18) and (19) into (14) and obtain

$$\mu = -\frac{1}{2} \left[\frac{I_{JJ} - \mathbf{I}_{JF}^T \mathbf{I}_{FF}^{-1} \mathbf{I}_{JF}}{(\delta P)^2 - \delta\mathbf{Q}^T \mathbf{I}_{FF}^{-1} \delta\mathbf{Q}} \right]^{1/2}, \quad \text{where } I_{JJ} = \mathbf{J}^T \mathbf{J}. \quad (20)$$

Note that due to the block-diagonal form of matrix $\mathbf{F}$, it is possible to avoid the tedious evaluation of $\mathbf{I}_{FF}^{-1}$ by rewriting the quantities I_{JJ} , $\mathbf{I}_{FF}$, and $\mathbf{I}_{JF}$ as follows:

- (i) $I_{JJ} = (\mathbf{J}^u)^T \mathbf{J}^u + (\mathbf{J}^v)^T \mathbf{J}^v$, where $J_i^u = \partial E / \partial u_i$ and $J_i^v = \partial E / \partial v_i$,
- (ii) $\mathbf{I}_{FF} = \mathbf{I}_{FF}^u + \mathbf{I}_{FF}^v = \mathbf{F}^u (\mathbf{F}^u)^T + \mathbf{F}^v (\mathbf{F}^v)^T$,
- (iii) $\mathbf{I}_{JF} = \mathbf{I}_{JF}^u + \mathbf{I}_{JF}^v = \mathbf{F}^u \mathbf{J}^u + \mathbf{F}^v \mathbf{J}^v$.

Hence, (19) and (20) can be rewritten as

$$\begin{aligned} \mathbf{v} = & -2\mu(\mathbf{I}_{FF}^u)^{-1}\delta\mathbf{Q}^u - 2\mu(\mathbf{I}_{FF}^v)^{-1}\delta\mathbf{Q}^v \\ & + (\mathbf{I}_{FF}^u)^{-1}\mathbf{I}_{JF}^u + (\mathbf{I}_{FF}^v)^{-1}\mathbf{I}_{JF}^v, \\ \mu = & \frac{1}{2} \left[\frac{I_{JJ} - (\mathbf{I}_{JF}^u)^T (\mathbf{I}_{FF}^u)^{-1} \mathbf{I}_{JF}^u - (\mathbf{I}_{JF}^v)^T (\mathbf{I}_{FF}^v)^{-1} \mathbf{I}_{JF}^v}{(\delta P)^2 - (\delta\mathbf{Q}^u)^T (\mathbf{I}_{FF}^u)^{-1} \delta\mathbf{Q}^u - (\delta\mathbf{Q}^v)^T (\mathbf{I}_{FF}^v)^{-1} \delta\mathbf{Q}^v} \right]^{1/2}, \end{aligned} \quad (21)$$

since the matrices $(\mathbf{I}_{FF}^u)^{-1}$ and $(\mathbf{I}_{FF}^v)^{-1}$ can be computed more easily than the composite matrix $\mathbf{I}_{FF}^{-1}$.

We next discuss the choice of $\delta\mathbf{Q}$. It is natural to choose the elements δQ_m of $\delta\mathbf{Q}$ proportional to Φ_m with a negative constant of proportionality $-k$, $k > 0$, so that the constraints move towards zero at an exponential rate. Then, the bound $k \leq \delta P (\Phi^T \mathbf{I}_{FF}^{-1} \Phi)^{-1/2}$ set on the value of k by (20) forces us to choose k adaptively. The simplest choice for k will be used; namely, $k = [\xi(\delta P)^2 / ((\Phi^u)^T (\mathbf{I}_{FF}^u)^{-1} \Phi^u + (\Phi^v)^T (\mathbf{I}_{FF}^v)^{-1} \Phi^v)]^{1/2}$, where $0 < \xi < 1$. Thus, the final update rule has only two free parameters, namely, δP and ξ .

3. Stability Constraints

For the filter $H(z_1, z_2)$ defined in (4), it is reasonable to assume that for most practical purposes the numerator polynomial $B(z_1, z_2)$ does not affect the system stability [1]. In the important special case of a second-order denominator polynomial $A(z_1, z_2)$ ($N_B = 2$), the stability conditions according to Huang [8] for the factor polynomials can be written as $X_k < 1$, $k = 1, \dots, 6$, as shown in Table 1. These conditions can be incorporated in the formalism as follows:

TABLE 1: Stability constraints for the denominator polynomial.

$X_1 = (v_{10}^{(1)})^2$
$X_2 = (v_{10}^{(2)})^2$
$X_3 = (v_{01}^{(1)})^2 + (v_{11}^{(1)})^2 + 2v_{01}^{(1)}v_{11}^{(1)} - (v_{10}^{(1)})^2 - 2v_{10}^{(1)}$
$X_4 = (v_{01}^{(1)})^2 + (v_{11}^{(1)})^2 - 2v_{01}^{(1)}v_{11}^{(1)} - (v_{10}^{(1)})^2 + 2v_{10}^{(1)}$
$X_5 = (v_{01}^{(2)})^2 + (v_{11}^{(2)})^2 + 2v_{01}^{(2)}v_{11}^{(2)} - (v_{10}^{(2)})^2 - 2v_{10}^{(2)}$
$X_6 = (v_{01}^{(2)})^2 + (v_{11}^{(2)})^2 - 2v_{01}^{(2)}v_{11}^{(2)} - (v_{10}^{(2)})^2 + 2v_{10}^{(2)}$

at a certain iteration of the algorithm, let a number M of the stability constraints be violated, so that $X_{k_r} \geq 1$, $r = 1, \dots, M$. We can augment the vector Φ^v of the previous section by adding extra components $\Phi_{[j+(N_A+1)i]+r}^v = X_{k_r}$, $r = 1, \dots, M$. By following the formalism of Section 2 using the augmented vector Φ^v , the quantities X_{k_r} are driven towards the stability region. Note that the dimensionality of the vector Φ^v is adaptive, as only the violated constraints are included in each iteration. Iterative application of this scheme can lead to a final stable solution.

4. Experimental Results

We seek an approximate factorization of the second-order 2D IIR filter whose numerator and denominator polynomial coefficients are shown in Tables 2 and 3, respectively. Note that both polynomials are nonfactorable.

Figure 2(a) shows the amplitude spectrum of the original low-pass IIR filter, and Figure 3 shows the error curve of the *unconstrained* gradient descent method applied to (11) using a regular grid of 32^2 points with an adaptation gain of 10^{-6} . From the latest figure we can see that after an initial error reduction, an undesirable flat minimum is reached followed by unbounded oscillations caused by a transition into a region of the parameter space in which the denominator polynomial is unstable. The amplitude spectrum of the filter corresponding to the factorization result achieved at the flat minimum is shown in Figure 2(b) and is clearly unacceptable.

Using the constraints $\Phi_1^u, \Phi_3^u, \Phi_4^u, \Phi_1^v, \Phi_3^v$, and Φ_4^v and all stability constraints of Section 3, we have applied both CLA, as described in [7], and the method proposed in the present paper using a regular grid of 32^2 training points. Figure 2(c) shows the amplitude spectrum of the filter obtained from the original CLA approach (with $\delta P = 0.01$ and $\xi = 0.99$) applied sequentially to the numerator and denominator polynomials. Obviously CLA outperforms gradient descent in all aspects. However, there are still small (but not negligible) differences between the amplitude spectra shown in Figures 2(a) and 2(c).

Figure 2(d) shows the amplitude spectrum of the filter obtained by the proposed method, with $\delta P = 0.07$ and $\xi = 0.99$. Clearly, there is very good agreement between the amplitude spectra of Figures 2(a) and 2(d). In order to visualize this better, in Figure 4, we present the differences in the amplitude spectra between the original filter and each of the filters obtained by gradient descent (Figure 4(a)), CLA (Figure 4(b)), and the proposed method

FIGURE 2: Amplitude spectrum of (a) original filter. (b) Filter obtained with gradient descent. (c) Filter obtained with CLA. (d) Filter obtained with the proposed method.

FIGURE 3: Error plot of gradient descent method.

(Figure 4(c)). From this figure it is clear that, of all the methods, the one proposed in the present paper achieves

the maximum proximity to the original amplitude spectrum, leading to the design of a cascade filter that meets as closely as possible the specifications of the original nonfactorable filter.

The factorization results are summarized in Tables 2 and 3. Note that the separate factorization of the CLA approach produces coefficients that are closer to those of both the numerator and denominator polynomials of the original filter, while the proposed method achieves maximum similarity between the frequency responses in expense of a smaller coefficient similarity, distributed along both the numerator and denominator coefficients.

5. Conclusions

In this paper, an efficient algorithm was proposed for factorizing simultaneously the numerator and denominator polynomials of the transfer functions of 2D IIR filters. The method is useful for designing IIR filters in cascade form for more efficient hardware implementation. By introducing a cost function based on the difference of frequency responses

TABLE 2: Factorization results for the numerator polynomial.

	Original non-factorable	Proposed method	CLA
Numerator polynomial coefficient matrix B	$\begin{bmatrix} 1 & 0.4000 & 0.0300 \\ 0.2500 & 0.4500 & 0.0700 \\ 0.0125 & 0.0350 & 0.0100 \end{bmatrix}$	$\begin{bmatrix} 1 & 0.4001 & 0.0384 \\ 0.2501 & 0.4649 & 0.0809 \\ 0.0078 & 0.0451 & 0.0427 \end{bmatrix}$	$\begin{bmatrix} 1 & 0.4000 & 0.0276 \\ 0.2500 & 0.4418 & 0.0602 \\ 0.0156 & 0.0484 & 0.0318 \end{bmatrix}$
1st factor polynomial coefficient matrix	—	$\begin{bmatrix} 1 & 0.1598 \\ 0.0368 & 0.1678 \end{bmatrix}$	$\begin{bmatrix} 1 & 0.0888 \\ 0.1212 & 0.1141 \end{bmatrix}$
2nd factor polynomial coefficient matrix	—	$\begin{bmatrix} 1 & 0.2403 \\ 0.2133 & 0.2542 \end{bmatrix}$	$\begin{bmatrix} 1 & 0.3111 \\ 0.1287 & 0.2785 \end{bmatrix}$

TABLE 3: Factorization results for the denominator polynomial.

	Original non-factorable	Proposed method	CLA
Denominator polynomial coefficient matrix A	$\begin{bmatrix} 1 & -0.9636 & 0.2321 \\ -0.9636 & 0.8360 & -0.1791 \\ 0.2321 & -0.1791 & 0.0445 \end{bmatrix}$	$\begin{bmatrix} 1 & -0.9638 & 0.2315 \\ -0.9638 & 0.8910 & -0.2030 \\ 0.2320 & -0.2065 & 0.0438 \end{bmatrix}$	$\begin{bmatrix} 1 & -0.9635 & 0.2320 \\ -0.9635 & 0.8332 & -0.1778 \\ 0.2317 & -0.1767 & 0.0335 \end{bmatrix}$
1st factor polynomial coefficient matrix	—	$\begin{bmatrix} 1 & -0.5087 \\ -0.4649 & 0.2518 \end{bmatrix}$	$\begin{bmatrix} 1 & -0.4902 \\ -0.4616 & 0.1625 \end{bmatrix}$
2nd factor polynomial coefficient matrix	—	$\begin{bmatrix} 1 & -0.4551 \\ -0.4989 & 0.1739 \end{bmatrix}$	$\begin{bmatrix} 1 & -0.4733 \\ -0.5019 & 0.2062 \end{bmatrix}$

FIGURE 4: Differences in the amplitude spectra between the original filter and the resulting factorable filters: (a) Gradient descent. (b) CLA. (c) Proposed method.

between the original and cascade filters and using a flexible framework for incorporating additional information in the form of constraints, we obtained optimal stable solutions for nonfactorable filters.

References

- [1] J. S. Lim, *Two-Dimensional Signal and Image Processing*, Prentice-Hall International, 1990.
- [2] Z. Mou-Yan and R. Unbehauen, "On the approximate factorization of 2-D polynomials," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 4, pp. 577–579, 1987.
- [3] N. E. Mastorakis, N. J. Theodorou, and S. G. Tzafestas, "A general factorization method for multivariable polynomials," *Multidimensional Systems and Signal Processing*, vol. 5, no. 2, pp. 151–178, 1994.
- [4] P. Misra and R. V. Patel, "Simple factorizability of 2-D polynomials," in *Proceedings of the International Symposium on Circuits and Systems*, pp. 1207–1210, New Orleans, La, USA, 1990.
- [5] B. C. Langand Frenzel, "Polynomial root finding," *IEEE Signal Processing Letters*, vol. 1, no. 10, pp. 141–143, 1994.
- [6] L. Hoteit, "FFT-based fast polynomial rooting," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP'00)*, vol. 6, pp. 3315–3318, Istanbul, Turkey, June 2000.
- [7] S. Perantonis, N. Ampazis, S. Varoufakis, and G. Antoniou, "Constrained learning in Nneural networks: application to stable factorization of 2-D polynomials," *Neural Processing Letters*, vol. 7, no. 1, pp. 5–14, 1998.
- [8] T. S. Huang, "Stability of two-dimensional recursive filters," *IEEE Transactions on Audio and Electroacoustics*, vol. 20, no. 2, pp. 158–163, 1972.
- [9] D. S. Huang, H. H. S. Ip, and Z. Chi, "A neural root finder of polynomials based on root moments," *Neural Computation*, vol. 16, no. 8, pp. 1721–1762, 2004.
- [10] D. S. Huang, H. S. I. Horace, C. K. L. Ken, Z. Chi, and H. S. Wong, "A new partitioning neural network model for recursively finding arbitrary roots of higher order arbitrary polynomials," *Applied Mathematics and Computation*, vol. 162, no. 3, pp. 1183–1200, 2005.
- [11] D. S. Huang, "constructive approach for finding arbitrary roots of polynomials by neural networks," *AIEEE Transactions On Neural Networks*, vol. 15, no. 2, pp. 477–491, 2004.
- [12] R. Hormis, G. Antoniou, and S. Mentzelopoulou, "Separation of two- dimensional polynomials via a sigma-pi neural net," in *Proceedings of the International Conference on Modelling and Simulation*, pp. 304–306, Pittsburg, Pa, USA, 1995.