FFT BASED ALGORITHM FOR POLYNOMIAL PLUS-MINUS FACTORIZATION

Martin Hromčík, Michael Šebek

Centre for Applied Cybernetics
Czech Technical University, Prague, Czech Republic
e-mail: m.hromcik@c-a-k.cz, m.sebek@c-a-k.cz

Keywords: Polynomial design methods, numerical algorithms.

Abstract

In this report a new algorithm is presented for the plus/minus factorization of a scalar discrete-time polynomial. The method is based on the discrete Fourier transform theory (DFT) and its relationship to the *Z*-transform. Involving DFT computational techniques, namely the famous fast Fourier transform routine (FFT), brings high computational efficiency and reliability. The effectiveness of the proposed algorithm is demonstrated by a particular practical application. Namely the problem of computing an H_2-optimal inverse dynamic filter to an audio equipment is considered as it was proposed by M. Sternad and colleagues in [16] to improve behavior of moderate quality loudspeakers. Involved spectral factorization can be converted into plus-minus factorization in a special case which in turn is resolved by our new method.

1 Introduction

This paper describes a new method for the plus-minus factorization of a discrete-time polynomial. Given a polynomial in the z variable,

$$p(z) = p_0 + p_1 z + p_2 z^2 + \cdots + p_n z^n,$$

without any roots on the unit circle, its plus/minus factorization is defined as

$$p(z) = p^+(z)p^-(z) \tag{1}$$

where $p^+(z)$ has all roots inside and $p^-(z)$ outside the unit disc. Clearly, the scalar plus/minus factorization is unique up to a scaling factor.

Polynomial plus/minus factorization has many applications in control and signal processing problems. For instance, efficient algebraic design methods for time-optimal controllers [7], quadratically optimal filters for mobile phones [14, 15], and l_1 optimal regulators [3], to name just a few, all recall the +/- factorization as a crucial computational step.

2 Existing Methods

From the computational point of view, nevertheless, the task is not well treated. There are two quite natural methods.

One of them is based on direct computation of roots. Using standard methods for polynomial roots evaluation, see [5, 18] for instance, one can separate the stable and unstable roots of $p(s)$ directly and construct the plus and minus parts from related first order factors or, alternatively, emply a more efficient recursive procedure based on the matrix eigenvalue theory [18].

Alternative algorithm relies on polynomial spectral factorization and gratest polynomial divisor computation. If $q(z)$ is the spectral factor of the symmetric product $p(z)p(z^{-1})$ then the greatest common divisor of $p(z)$ and $q(z)$ is obviously the plus factor of $p(z)$. The minus factor can be derived similarly from $p(z^{-1})$ and $q(z^{-1})$. As opposed to the previous approach based on direct roots computation which typically makes problems for higher degrees and/or roots multiplicities, this procedure relies on numerically reliable algorithms for polynomial spectral factorization [11, 2]. Unfortunately, the polynomial greatest common divisor computation is much more sensitive. As a result, both these techniques do not work properly for high degrees (say over 50).

In this report we will introduce a completely new approach to the problem, inspired by our work on efficient algorithms for polynomial spectral factorization, see [2]. It is based on the DFT theory and provides both a fruitful view on the relation between DFT and the Z-transform theory, and a powerful computational tool in the form of the fast Fourier transform algorithm.

3 Discrete Fourier Transform

If $\boldsymbol{p} = [p_0, p_1, \ldots, p_N]$ is a vector of complex numbers, then its *direct DFT* is given by the vector $\boldsymbol{y} = [y_0, y_1, \ldots, y_N]$, where

$$y_k = \sum_{i=0}^{N} p_i e^{-j\frac{2\pi k}{N+1}i} \tag{2}$$

The vector $\boldsymbol{y}$ is called the image of vector $\boldsymbol{p}$. Conversely, if $\boldsymbol{y} = [y_0, y_1, \ldots, y_N]$ is given, then its inverse *DFT* recovers the original vector $\boldsymbol{p} = [p_0, p_1, \ldots, p_N]$, where

$$p_i = \frac{1}{N+1} \sum_{k=0}^{N} y_k e^{j\frac{2\pi i}{N+1}k} \tag{3}$$

DFT is of great interest in various engineering fields. For its relationship to Fourier series of sampled signals, DFT is frequently used in signal processing. One of the experimental

identification methods employs DFT as well [9]. The close relationship of DFT to interpolation is also well known and was used recently to solve some tasks of the polynomial control theory [12] and to treat robustness analysis problems of certain kind [13].

For numerical computation of DFT, the efficient recursive FFT algorithm was developed by Cooley and Tukey in 1965 [4]. If the length of the input is a power of two, a faster version of FFT (sometimes called *radix-2 FFT*) can be employed [4]. In general, the FFT routine features a highly beneficial computational complexity and involves $\mathcal{O}(N\log(N))$ multiplications and additions for a vector of length N.

Thanks to the importance of DFT mentioned above, the FFT algorithms are naturally available as built-in functions of many computing packages (MATLAB$^{\mathrm{TM}}$, MATHEMATICA$^{\mathrm{TM}}$ etc.). This is another good reason for employing the procedure proposed in this paper.

4 Plus-minus Factorization and DFT

4.1 Theory

Given a polynomial

$$p(z) = p_0 + p_1 z + \cdots + p_d z^d \,,$$

nonzero for $|z| = 1$, we first apply a direct degree shift to arrive at a two-sided polynomial

$$\tilde{p}(z) = p_0 z^{-\delta} + \cdots + p_d z^{d-\delta},$$

where δ is the number of roots of $p(z)$ lying inside the unit circle. Now, instead of solving equation (1), we look for $\tilde{p}^+(z) = \tilde{p}_0^+ + \tilde{p}_1^+ z^{-1} + \cdots + \tilde{p}_\delta^+ z^{-\delta}$ and $\tilde{p}^-(z) = \tilde{p}_0^- + \tilde{p}_1^- z + \cdots + \tilde{p}_{d-\delta}^- z^{d-\delta}$ such that

$$\tilde{p}(z) = \tilde{p}^+(z)\tilde{p}^-(z) \tag{4}$$

Relation between the pairs $\tilde{p}^+, \tilde{p}^-$ and p^+, p^- are obvious.

In order to solve the equation (4), logarithm is applied. As $\tilde{p}(z), \tilde{p}^+(z)$ and $\tilde{p}^-(z)$ are all analytic and nonzero in $1-\varepsilon < |z| < 1+\varepsilon$ the logarithms exist. Let us denote them as $\ln\tilde{p}(z) = n(z),\ \ln\tilde{p}^+(z) = x^+(z),\ \ln\tilde{p}^-(z) = x^-(z)$. Here $n(z)$, obtained from the given $\tilde{p}(z)$, is a Laurent infinite power series

$$n(z) = \cdots + n_1 z + n_0 + n_{-1} z^{-1} + \cdots \,.$$

It can be directly decomposed,

$$n(z) = x^+(z) + x^-(z^{-1})$$

with power series

$$x^+(z) = x_0^+ + x_1^+ z^{-1} + \cdots = \frac{n_0}{2} + n_{-1} z^{-1} + \cdots,$$

$$x^-(z) = x_0^- + x_1^- z + \cdots = \frac{n_0}{2} + n_1 z + \cdots \tag{5}$$

analytic for $1-\varepsilon < |z|$ and $1+\varepsilon > |z|$ respectively.

At this time the necessity of the degree shift yielding the two-sided polynomial $\tilde{p}$ can be explained. According to the Cauchy's theorem of argument [6], the curve $p(z)$ for $|z| = 1$ encircles the origin in the complex plane as many times as is the number of roots of $p(z)$ lying in the complex unit disc. Hence the logarithms cannot be applied directly as its imaginary part, reading the phase of $p(z)$, would not be continuous. An easy solution to avoid this situation is to move the desired number of roots of $p(z)$ from infinity to zero by performing proper degree shift.

Once $x^+(z)$ and $x^-(z)$ are computed, the plus/minus factors $\tilde{p}^+,\ \tilde{p}^-$ are recovered as

$$\tilde{p}^+ = \mathrm{e}^{x^+(z)} = \tilde{p}_0^+ + \tilde{p}_1^+ z^{-1} + \cdots,\ \ \tilde{p}^- = \mathrm{e}^{x^-(z)} = \tilde{p}_0^- + \tilde{p}_1^- z + \cdots.$$

Since $x^+(z)$ is analytic in $1-\varepsilon < |z|$, so is $\tilde{p}^+(z)$ and hence it can be expanded according to (3). Moreover, as a result of exponential function, $\tilde{p}^+(z)$ is nonzero in $1-\varepsilon < |z|$. In other words, it has all its zeros inside the unit disc and is therefore Schur stable. Note also that $\tilde{p}^+(z)$ has to be a (finite) polynomial of degree d (due to the uniqueness of the solution to the problem which is known to be a polynomial) though $n(z)$ is an infinite power series. Similar reasoning proves the $\tilde{p}^-$ factor desired properties.

4.2 Numerical Algorithm

Numerical implementation follows the ideas considered above. A polynomial $p(z)$ is represented by its coefficients $p_i,\ i = 0 \ldots r$ or, equivalently, by function values P_k in the Fourier interpolating points $g^k,\ k = -R \ldots 0 \ldots R$, where $R \geq d,\ \ g = \mathrm{e}^{\mathrm{j}\frac{2\pi}{2R+1}}$. Accordingly, a power series can be approximated by a finite set of its coefficients or by its values in a finite number of interpolation points on the unit circle. Some operations of the procedure, namely the decomposition of $n(z)$ into $x^+(z)$ and $x^-(z)$, are performed in the time domain (operations on coefficients), while the others (evaluation of logarithmic and exponential functions) are executed in the frequency domain (operations with values over $|z| = 1$). Mutual conversion between the two domains is mediated by the shifted discrete Fourier transform operator defined as

$$X_k = \sum_{i=-R}^{R} x_i g^{-ki}, \quad x_i = \frac{1}{2R+1} \sum_{k=-R}^{R} X_k g^{ki} \,,$$

which approximates the Z-transform by dealing with $-R \leq i \leq +R$ instead of infinite $-\infty < i < +\infty$, and with $z = g^k,\ -R \leq k \leq +R$ instead of continuum $z = \mathrm{e}^{\mathrm{j}\phi},\ -\pi \leq \phi \leq +\pi$.

The accuracy of results depends on the number of interpolation points $2R+1$ involved in the computation. This number can be considered as a simple tuning knob of the computational process.

Resulting numerical routine looks then as follows:

Algorithm 1: Scalar discrete-time plus-minus factorization.

Input: Scalar polynomial
$p(z) = p_0 + p_1 z + \cdots + p_d z^d$, nonzero for $|z| = 1$.

Output: Polynomials $p^+(z)$ and $p^-(z)$, the plus and minus factors of $p(z)$.

Step 1 - *Choice of the number of interpolation points.*
Decide about the number R. R approximately 10 to 50 times larger than d is recommended up to our practical experience.

Step 2 - *Degree shift.*
Find out the number δ of zeros of $p(z)$ inside the unit disc. A modification of well known Schur stability criterion can be employed, see [10] for instance.

Having δ at hand, construct a two-sided polynomial $\tilde{p}(z)$ as

$$\tilde{p}(z) = p(z)z^{-\delta} = p_0 z^{-\delta} + \cdots + p_d z^{d-\delta} =$$
$$\tilde{p}_{-\delta} z^{-\delta} + \cdots + \tilde{p}_0 + \cdots + \tilde{p}_{d-\delta} z^{d-\delta}$$

Step 3 - *Direct FFT (I):*
Using the FFT algorithm, perform direct DFT, defined by (2), on the vector

$$\boldsymbol{p} = [\underbrace{\tilde{p}_0, \tilde{p}_1, \ldots, \tilde{p}_{d-\delta}, 0, 0, \ldots, 0, \tilde{p}_{-\delta}, \ldots, \tilde{p}_{-1}}_{2R+1}]$$

In this way, the set $\boldsymbol{P} = [P_0, P_1, \ldots, P_{2R}]$ of the values of $\tilde{p}(z)$ at the Fourier points is obtained.

Step 4 - *Logarithmization:*
Compute the logarithms $N_i = \ln(P_i)$ of all particular P_i's and form the vector $\boldsymbol{N} = [N_0, N_1, \ldots, N_{2R}]$ of them. N_i's thus obtained are the values of the function $n(z) = \ln(\tilde{p}(z))$ at related Fourier points on the unit complex circle.

Step 5 - *Inverse FFT (I):*
To get the vector $\boldsymbol{n} = [n_0, n_1, \ldots, n_R, n_{-R}, \ldots, n_{-1}]$, containing the coefficients of the two-sided polynomial $n(z) = n_{-R}z^{-R} + \cdots + n_{-1}z^{-1} + n_0 + n_1 z + \cdots + n_R z^R$ approximating the power series of $\ln(m(z))$ for the given R, perform inverse DFT, defined by (3), on the vector $\boldsymbol{N}$ using the FFT algorithm.

Step 6 - *Decomposition:*
Take the "causal part" $\boldsymbol{x}^+$ of $\boldsymbol{n}$:
$\boldsymbol{x}^+ = [n_0/2, n_1, \ldots, n_R]$. Similarly, construct $\boldsymbol{x}^-$ as $\boldsymbol{x}^- = [n_0/2, n_{-1}, \ldots, n_{-R}]$.

Step 7 - *Direct FFT (II):*
Evaluate $x^+(z) = n_0/2 + n_1 z^{-1} + \ldots + n_R z^{-R}$ at the Fourier points by applying direct FFT on the set $\boldsymbol{x}^+$ and get $\boldsymbol{X}^+ = [X_0^+, \ldots, X_R^+]$. Proceed with $x^-(z)$ in obvious way.

Step 8 - *Exponential function:*
To get the plus/minus factors, the exponential functions $\tilde{p}^+(z) = e^{x^+(z)}$ and $\tilde{p}^-(z) = e^{x^-(z)}$ remain to be evaluated. First we compute the values of $\tilde{p}^+(z)$ and $\tilde{p}^-(z)$ at the Fourier points: $\tilde{\boldsymbol{P}}^+ = [e^{X_0^+}, \ldots, e^{X_R^+}]$. Similar steps apply for the minus part.

Step 9 - *Inverse FFT (II):*
Finally, the coefficients $\tilde{\boldsymbol{p}}^+ = [\tilde{p}_0^+, \ldots, \tilde{p}_R^+]$ of $\tilde{p}^+(z)$ are recovered by inverse FFT performed on the vector $\tilde{\boldsymbol{P}}^+$. The resulting approximation to the plus factor $\tilde{p}^+(z)$ then equals $\tilde{p}^+(z) = \tilde{p}_0^+ + \tilde{p}_{-1}^+ z^{-1} + \cdots + \tilde{p}_{-\delta}^+ z^{-\delta}$. Proceed with the minus part accordingly.

Step 10 - *Finalization:*
Convert the plus-minus factors $\tilde{p}^+(z)$ and $\tilde{p}^-(z)$ of $\tilde{p}(z)$ into the desired factors of $p(z)$ using the following formulas

$$p^- = \tilde{p}^-, \quad p^+ = \tilde{p}^{+\star},$$

where the star stands for discrete-time conjugate, $z \to z^{-1}$. ◇

Note that one obtains R coefficients of $\tilde{p}^+$ and $\tilde{p}^-$ in the step 9. However, $p^+(z)$ being the plus factor of $p(z)$ is known to be of degree δ only and only the first $\delta + 1$ coefficients of $\tilde{p}^+(z)$ should be significant as a result while the remaining ones should be negligible. As the number R increases, these values theoretically converge to zero indeed since the formulas of DFT become better approximations to the Z-transform definitions.

5 Radix-2 Modification of the Algorithm

The basic version of the routine proposed above is based on the shifted dicrete Fourier transform. This modification of DFT appears useful during the derivation of the Algorithm 1 due to its more transparent relationship to the spectral theory. It can be easily transformed to the standard DFT as it is defined in the section 3, simply by reordering related vector entries (see the steps 2 and 4 of Algorithm 1). However, $2R + 1$ interpolation points are used for the FFT algorithm and unfortunately this number is always odd and cannot equal any power of two. Therefore the radix-2 fast version of the FFT routine cannot be addressed. Nevertheless, this slight drawback can be easily avoided if the periodicity of direct and inverse DFT formulas is taken into account. Basically, one can construct the initial set as

$$[\underbrace{\tilde{p}_0, \tilde{p}_1, \ldots, \tilde{p}_{d-\delta}, 0, 0, \ldots, 0, \tilde{p}_{-\delta}, \ldots, \tilde{p}_{-1}}_{2^R}]$$

which has a power-of-two entries in total. The Algorithm 1 remains valid also in this case with $2R + 1$ replaced by 2^R and $R + 1$ by 2^{R-1} respectively, up to one point: in the Step 6, the decomposition reads $\boldsymbol{x}^+ = [n_0/2, n_1, \ldots, n_R/2]$ instead of $\boldsymbol{x}^+ = [n_0/2, n_1, \ldots, n_R]$. This minor modification of the proposed method further increases its efficiency since the powerful radix-2 FFT can be called.

6 Computational Complexity

Thanks to the fact that the fast Fourier transform algorithm is extensively used during the computation, the overall routine features an expedient computational complexity.

Provided that the above modifications of the computational procedure are considered, namely if the resulting number of interpolation points is taken as a power of two, the fast radix-2 FFT can be employed. In this case, $(R \log_2 R)/2$ multiplications and $R \log_2 R$ additions are needed to evaluate either direct or inverse DFT of a vector of length R [4]. Let us suppose in addition that computing the logarithm or exponential of a scalar constant takes at most k multiplications and l additions. Then the particular steps of the modified Algorithm 1 involve $(R \log_2 R)/2$ multiplications and $R \log_2 R$ additions (Steps 3, 5, 7, 9), and kR multiplications and lR additions (Steps 4, 8) respectively. Hence the overall procedure consumes

$$4\frac{R \log R}{2} + 2kR = 2R \log R + 2lR$$

complex multiplications, and

$$4R \log R + 2lR$$

complex additions. By inspecting the above formulas one can see that asymptotically the proposed method features $\mathcal{O}(R \log R)$ complex multiplications and additions.

7 Upgrading Loudspeakers Dynamics

An original approach has been published by Sternad et al. in [16] how to improve performance of an audio equipment at low additional costs. The authors use the LQG optimal feedforward compensator technique to receive an inverse dynamic filter for a moderate quality loudspeaker. By attaching a signal processor implementing this filter prior to the loudspeaker, the dynamical imperfections of the original device are eliminated and the overall equipment behaves as an aparatus of a much higher class. To learn more about this research and to get some working examples, visit [17].

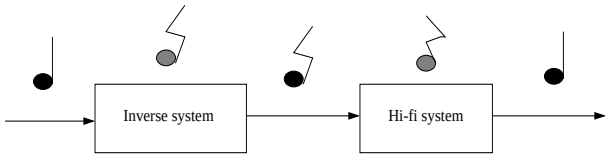

Figure 1: Pre-filter compensation scheme (adopted from [16]).

Unlike their predecessors, the authors try to modify the sound over the whole range of frequencies. Such a complex compensation fully employs the increasing performance of signal hardware dedicated to CD-quality audio signals, and at the same time calls for fast and reliable factorization solvers [16]. We believe our new algorithm will significantly contribute to this goal.

The loudspeaker dynamics is considered in the form of an ARX model

$$y(t) = z^{-k}\frac{B(z)}{A(z)}u(t).$$

Since the impulse response is rather long for a high sampling frequency (CD-quality standard of 44 kHz was used), both the numerator and denominator of the model are of high orders, say one to five hundred.

The model has an unstable inverse in general since some of its zeros may lie outside the unit disc. Hence a stable approximation has to be calculated to be used in the feedforward structure. The authors recall the LQG theory and seek for a compensating filter

$$u(t) = \frac{Q(z)}{P(z)}w(t)$$

such that the criterion $J = E|y(t) - w(t-d)|^2 + \rho|u(t)|^2$ is minimized.

For broadband audio signals, the optimal filter is given in the form

$$u(t) = \frac{Q_1(z)A(z)}{\beta(z)}w(t)$$

where β results from the spectral factorization

$$\beta\beta^* = BB^* + \rho AA^*$$

and Q_1 is the solution of a subsequent Diophantine equation

$$z^{k-d}B^*(z) = r\beta^*(z)Q_1(z^{-1}) + zL^*(z),$$

see [16].

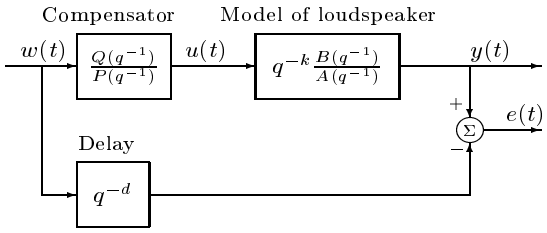

Figure 2: Optimal filtering problem setup (adopted from [16]).

As for the spectral factor computation, the authors employ the Newton-Raphson iterative scheme [11] in the cited work [16]. According to their results and our experience, this method has been probably the best available procedure for scalar polynomial spectral factorization so far [16, 8]. This method works quite well also for high degrees of involved polynomials in contrast to the straightforward way of computing and distributing the roots of $BB^* + \rho AA^*$.

Let us perform a benchmark experiment to compare the existing approach and our newly proposed algorithm for particular numerical data kindly provided by Mikael Sternad and colleagues from the University of Uppsala. Up to now, two models of the loudspeakers dynamics have been sent to us for testing purposes and the results related to the more complex one are presented in the following.

The data in concern are given as follows. The numerator $B(z) = B_0 + B_1 z^{-}1 + \cdots + B_{250} z^{-250}$ is an unstable polynomial of degree 250, $A(z)$ is stable of degree 90, and $k = 160$. Taking $\rho = 0$, the spectral factorization of $m(z) = B(z)B^*(z) = m_{250} z^{-250} + \ldots + m_0 + \ldots + m_{250} z^{250}$ is to be performed. In this special case, the spectral factor $x(z)$ of $m(z)$ can be effectively constructed as

$$x(z) = B^+(z) \left(B^-(z)\right)^* z^{-k}$$

where B^+, B^- are the plus and minus factors of B respectively and k is the degree of B^-.

All presented experiments were realized on a PC computer with Pentium III/1.2GHz processor and 512 MB RAM, under MS Windows 2000 in MATLAB version 6.1.

Results of this experiment for various values of the parameters N are summarized and related in the following table. Namely, the computational time and accuracy of results are of interest. To obtain the former characteristic, the MATLAB abilities were employed (the built-in functions `tic/toc`). The computational error is defined here as the largest coefficient of the expression $B^+B^- - B$, evaluated in the MATLAB workspace, divided by the largest coefficient of B (all in absolute value).

	Time [s]	Accuracy
FFT(14)	0.23 sec	$6.28 \cdot 10^{-3}$
FFT(15)	0.45 sec	$2.52 \cdot 10^{-8}$
FFT(16)	0.89 sec	$4.65 \cdot 10^{-11}$
FFT(17)	1.75 sec	$2.40 \cdot 10^{-12}$

TABLE 1: Accuracy and efficiency of compared algorithms.

These tests prove the power of the new algorithm in such tough examples. Neither of the two procedures described in paragraph 2 can factor this large polynomial. Direct roots evaluation method, based on the standard MATLAB function `roots`, gives totally meaningless results (accuracy of 10^{43}) while the routine based on spectral factorization fails due to numerical problems with greatest common polynomial divisor evaluation (Polynomial Toolbox function `rdiv` was used [8]).

8 Further Research

The success in modifying a selected numerical procedure, originally developed for polynomial spectral factorization, to handle the unsymmetric plus-minus decomposition suggests that other well known spectral factorization routines might work well in the non-symmetric context as well. Actually, we achieved some results of this kind recently and plan to incorporate them in the final camera-ready version of our contribution.

Unfortunately, the described approach cannot be directly extended to the matrix case - the product of two matrices does not commute in general and since $\tilde{P}^+(z)\tilde{P}^-(z) \neq \tilde{P}^-(z)\tilde{P}^+(z)$, one cannot write $\ln \tilde{P}(z) = \ln(\tilde{P}^+(z)) + \ln(\tilde{P}^-(z))$ and perform the decomposition. Clearly, in combination with techniques for the Smith form of a polynomial matrix [7, 8], the proposed routine can be used to factor particular entries of an equivalent diagonal matrix. Unfortunately, from the computational point of view it is not particularly useful as the Smith form transformation is known to be numerically fragile [8].

A modification for continuous time polynomials is under research too. However, the suggested approach does not seem to be as fruitful for continuous time systems as it is in the discrete time case. The reason is that the relation between the Laplace transform, replacing the role of Z-transform in the continuous time domain, and the DFT is not so close. The idea of a direct bilinear transform converting the s variable to z cannot be recommended for numerical reasons.

9 Conclusion

A new method for the discrete-time plus-minus factorization problem in the scalar case has been proposed. The new method relies on numerically stable and efficient FFT algorithm. Besides its good numerical properties, the derivation of the routine also provides an interesting look into the related mathematics, combining the results of the theory of functions of complex variable, the theory of sampled signals, and the discrete Fourier transform techniques. The suggested method is employed in a practical application of improving the quality of a hi-fi system.

Acknowledgements

The work of M. Hromčík and M. Šebek has been supported by the Ministry of Education of the Czech Republic under contract No. LN00B096.

References

[1] Kučera V., *Analysis and Design of Discrete Linear Control Systems*, Academia Prague (1991).

[2] M. Hromcik, J. Jezek, M. Sebek, *New Algorithm for Spectral Factorization and its Practical Application*, Proceedings of the European Control Conference ECC'2001, Porto, Portugal, September 1-5, 2001.

[3] Z. Hurak, M. Sebek, *Algebraic Approach to l-1 Optimal Control*, to appear.

[4] Bini D., Pan V., *Polynomial and Matrix Computations, Volume 1: Fundamental algorithms*, Birkhäuser, Boston (1994).

[5] Higham N. J., *Accuracy and Stability of Numerical Algorithms*, S.I.A.M., Philadelphia (1996).

[6] Needham T., *Visual Complex Analysis*, Oxford University Press, 1997.

[7] Kučera V., *Analysis and Design of Discrete Linear Control Systems*, Academia Prague (1991).

[8] Kwakernaak H., Šebek M., *PolyX Home Page*, `http://www.polyx.cz/`, `http://www.polyx.com/`.

[9] Ljung L., *System Identification: Theory for the User*, Prentice-Hall Information and Systems Sciences Series. Englewood Cliffs, Prentice-Hall (1987).

[10] Barnett S., *Polynomials and Linear Control*. Marcel Dekker, New York and Basel (1983).

[11] Ježek J. and Kučera V., *Efficient Algorithm for Matrix Spectral Factorization*,Automatica, vol. 29, pp. 663-669, 1985.

[12] Hromčík M., Šebek M., *Numerical and Symbolic Computation of Polynomial Matrix Determinant*,

[13] Hromčík M., Šebek M, *Fast Fourier Transform and Robustness Analysis with Respect to Parametric Uncertainties*, Proceedings of the 3rd IFAC Symposium on Robust Control Design ROCOND 2000, Prague, CZ, June 21-23, 2000.

[14] M. Sternad and A. Ahleén, *Robust Filtering and Feedforward Control Based on Probabilistic Descriptions of Model Errors*, Automatica, 29, pp. 661-679.

[15] K. Ohrn, A. Ahleén and M. Sternad, *A Probabilistic Approach to Multivariable Robust Filtering and Open-loop Control*, IEEE Transactions on Automatic Control, 40, pp. 405-417.

[16] M. Sternad, M. Johansson, J. Rutstrom, *Inversion of Loudspeaker Dynamics by Polynomial LQ Feedforward Control*, Proceedings of the 3rd IFAC Symposium on Robust Control Design ROCOND 2000, Prague, CZ, June 21-23, 2000.

[17] University of Uppsala, Signals and Systems Department, *Adaptive Signal Processing, Course Homepage 2000*, http://www.signal.uu.se/Courses/CourseDirs/AdaptSignTF/Adapt00.html

[18] The Mathworks, *Using MATLAB 5.3*, The Matrhworks, 1999.