Numerical algorithms for polynomial plus/minus factorization

M.Hromčík and M. Šebek

Abstract— **Two new algorithms are presented in the paper for the plus/minus factorization of a scalar discrete-time polynomial. The first method is based on the discrete Fourier transform theory (DFT) and its relationship to the Z-transform. Involving DFT computational techniques and the famous fast Fourier transform routine brings high computational efficiency and reliability. The method is applied in the case-study of H_2-optimal inverse dynamic filter to an audio equipment. The second numerical procedure originates in a symmetric spectral factorization routine, namely the Bauer's method of the 1950s. As a by product, a recursive LU factorization procedure for Toeplitz matrices is devised that is of more general impact and can be of use in other areas of applied mathematics as well. Performance of the method is demonstrated by an l_1 optimal controller design example.**

I. 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 [1], quadratically optimal filters for mobile phones [13], [14], and l_1 optimal regulators [2], to name just a few, all recall the +/- factorization as a crucial computational step.

II. EXISTING METHODS

In any case, the plus/minus factors for $n \geq 5$ cannot be achieved by a finite number of algebraic operations. This conclusion is due to the Galois's theorem stating that the roots of a polynomial of degree greater or equal to five cannot be expressed in a closed form. Therefore all numerical algorithms for plus/minus factorization are iterative in nature and give just an approximation to the genuine factors. Some existing approaches to this problem are mentioned in this section.

The most natural way is based on the computation of polynomial's roots. Having determined the roots $r_1, r_2, \ldots, r_n$ of $p(z)$ via any standard procedure for polynomial roots [10] and considering that $p(z) \neq 0$ for all $|z| = 1$ by assumption, one can divide the roots into two groups $R^+ = \{r_i : m^\star(r_i) = 0, |r_i| < 1\}, R^- = \{r_i : m^\star(r_i) = 0, |r_i| > 1\}$. Clearly, R^+ and R^- are the sets of roots of $p^+(z)$ and $p^-(z)$ respectively.

Performance of this procedure heavily hinges on the accuracy of the computed polynomial roots. If these roots are distinct and separated enough, standard numerical routines [10] can determine them with good precision. However, it is well known that the relative accuracy of a computed root decreases as its multiplicity grows [10], and so does the accuracy of the spectral factor thus obtained.

In addition, if the degree of the involved polynomial is high, say over 50, the very computation of the spectral factor coefficients is problematic due to rounding errors. It means that even if the desired *roots* of the spectral factor are evaluated with good accuracy, its particular *coefficients*, which are typically required in applications, are not accurate.

An 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 [6], [4]. 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 [4]. 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.

III. 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 $\boldsymbol{y} =$

The work of M. Hromčík was supported by the Ministry of Education of the Czech Republic (MSMT) under contract No. 1M0567. The work of M. Šebek was supported by the Grant Agency of the Czech Republic grant No. 102/08/0186 and by the Czech Ministry of Education, Grant 1P2008LA300.

M.Hromcik, Center for Applied Cybernetics, Czech Technical University in Prague, Faculty of Electrical Engineering, Karlovo namesti 13-G, Prague, Czech Republic `xhromcik@control.felk.cvut.cz`

M. Šebek, Department of Control Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic

$[y_0, y_1, \dots, y_N]$, 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 $\boldsymbol{y} = [y_0, y_1, \dots, y_N]$ is given, then its inverse *DFT* recovers the original vector $\boldsymbol{p} = [p_0, p_1, \dots, p_N]$, 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 [11]. The close relationship of DFT to interpolation is also well known and was used recently to solve some tasks of the polynomial control theory [5] and to treat robustness analysis problems of certain kind [12].

For numerical computation of DFT, the efficient recursive FFT algorithm was developed by Cooley and Tukey in 1965 [8]. If the length of the input is a power of two, a faster version of FFT (sometimes called *radix-2 FFT*) can be employed [8]. 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$^{\text{TM}}$, MATHEMATICA$^{\text{TM}}$ etc.). This is another good reason for employing the procedure proposed in this paper.

IV. PLUS/MINUS FACTORIZATION AND DFT

A. 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

$$x^+(z) = x_0^+ + x_1^+ z^{-1} + \cdots = \frac{n_0}{2} + n_{-1} z^{-1} + \cdots \ , \quad x^-(z) = x_0^- + x_1^- \tag{5}$$

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 [26], 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.

B. Numerical Algorithm

Numerical implementation follows the ideas considered above. A polynomial $p(z)$ is represented by its coefficients p_i, $i = 0 \dots r$ or, equivalently, by function values P_k in the Fourier interpolating points g^k, $k = -R \dots 0 \dots R$, where $R \geq d$, $\quad g = \mathrm{e}^{\mathrm{j}\frac{2\pi}{2R+1}}$. 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 [23] 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.

V. 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.

VI. 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 [8]. 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.

VII. UPGRADING LOUDSPEAKERS DYNAMICS

An original approach has been published by Sternad et al. in [15] 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 [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 [15]. 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 [15].

As for the spectral factor computation, the authors employ the Newton-Raphson iterative scheme [6] in the cited work [15]. According to their results and our experience, this method has been probably the best available procedure for scalar polynomial spectral factorization so far [15], [7]. 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 $B(z) = B_0 + B_1z^-1 + \cdots + B_{250}z^{-250}$ 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 section 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 [7]).

VIII. DISCUSSION

The success in modifying a selected numerical procedure, originally developed for polynomial spectral factorization, to handle the non-symmetric plus/minus decomposition suggests that other well known spectral factorization routines might work well in the non-symmetric context as well. Actually, we decided to go this way and succeeded in adapting a classical polynomial spectral factorization routine, the Bauer's algorithm [18], for the plus/minus case. Our results are presented in subsequent paragraphs.

IX. BAUER'S METHOD FOR SPECTRAL FACTORIZATION

F. I. Bauer published his method for spectral factorization of a discrete-time scalar polynomial as early as in 1955, see [18], [19]. The procedure is based on the relationship between polynomials and related infinite Toeplitz-type Sylvester matrices.

A. Algebra of Sylvester matrices

Given a two-sided polynomial $p(z) = p_{-m}z^{-m} + \cdots + p_0 + \cdots + p_n z^n$, we define its Sylvester companion matrix T_p^N of order N,

$$N \geq \max(n, m)$$

as an N by N square matrix constructed according to the following scheme:

$$T_p^N = \begin{pmatrix} p_0 & p_1 & \cdots & p_n & 0 & \cdots & 0 \\ p_{-1} & p_0 & p_1 & \cdots & p_n & \ddots & \vdots \\ \vdots & p_{-1} & \ddots & \ddots & & \ddots & 0 \\ p_{-m} & \vdots & \ddots & & & & p_n \\ 0 & p_{-m} & & & & \ddots & \vdots \\ \vdots & \ddots & \ddots & & \ddots & \ddots & p_1 \\ 0 & \cdots & 0 & p_{-m} & \cdots & p_{-1} & p_0 \end{pmatrix}$$

To show the relation between the polynomial algebra and the algebra of Sylvester matrices, let us consider two simple polynomials $p_1(z) = 3z^{-1} + 2 + z$ and $p_2(z) = z^{-1} + 3$. Their companion matrices of order four read respectively

$$T_{p1}^4 = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 0 & 3 & 2 & 1 \\ 0 & 0 & 3 & 2 \end{pmatrix}$$

$$T_{p2}^4 = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 0 & 1 & 3 & 0 \\ 0 & 0 & 1 & 3 \end{pmatrix}$$

Their sum $p_3(z) = p_1(z) + p_2(z)$ equals

$$p_3(z) = 4z^{-1} + 5 + z$$

and its companion matrix can be computed as direct sum of related companion matrices T_{p1}^4, T_{p2}^4:

$$T_{p3}^4 = \begin{pmatrix} 5 & 1 & 0 & 0 \\ 4 & 5 & 1 & 0 \\ 0 & 4 & 5 & 1 \\ 0 & 0 & 4 & 5 \end{pmatrix}$$

Similarly, their product $p_4 = p_1 p_2 = 3z^{-2} + 11z^{-1} + 7 + 3z$ has a companion matrix

$$T_{p4}^4 = T_{p1}^4 T_{p2}^4 = \begin{pmatrix} 7 & 3 & 0 & 0 \\ 11 & 7 & 3 & 0 \\ 3 & 11 & 7 & 3 \\ 0 & 3 & 11 & 6 \end{pmatrix}$$

B. Bauer's method for spectral factorization

As we have illustrated above, finite dimensional matrices are sufficient to accommodate "finite" algebraic problems. On the other hand, if we do not restrict to finite dimensionality of related matrices, transcendent problems, including spectral factorization, involving polynomials can be resolved by this approach as well.

We will illustrate the Bauer's spectral factorization method by means of a simple example. An interested reader can find detailed description in the original work [18] or, alternatively, in the survey paper [20].

Given $p(z) = 2z^{-1} + 5 + 2z$ its companion matrix of order five reads

$$T_p = \begin{pmatrix} 5 & 2 & 0 & 0 & 0 \\ 2 & 5 & 2 & 0 & 0 \\ 0 & 2 & 5 & 2 & 0 \\ 0 & 0 & 2 & 5 & 2 \\ 0 & 0 & 0 & 2 & 5 \end{pmatrix}$$

As p is symmetric and positive definite on the unit circle its spectral factor x exists such that

$$x^\star x = p$$

holds and x is stable. The spectral factor coefficients can be approximated using the Cholesky factorization of T_p:

$$T_x = \begin{pmatrix} 2.236 & 0.8944 & 0 & 0 & 0 \\ 0 & 2.049 & 0.9759 & 0 & 0 \\ 0 & 0 & 2.012 & 0.9941 & 0 \\ 0 & 0 & 0 & 2.003 & 0.9985 \\ 0 & 0 & 0 & 0 & 2.001 \end{pmatrix}$$

The diagonals of T_x obviously converge to the genuine spectral factor coefficients: $x(z) = 1 + 2z$.

An interesting feature of this routine is that particular columns of T_x can be computed iteratively, using only latest preceding column and the coefficients of $p(z)$, see [20] for details. As a result, the final algorithm is favorably memory efficient. Mainly for this reason the method is still quite popular in spite of the fact that some later approaches, see eg. [21], [22], provide a faster rate of convergence.

X. PLUS/MINUS FACTORIZATION VIA BAUER'S APPROACH

A modification of the Bauer's method for the non-symmetric polynomial plus/minus factorization is worked out in this section.

A. LU factorization

As we have shown in section 9.1, algebra of companion matrices is not limited to the symmetric case. Also the matrix theory provides useful factorization techniques for non-symmetric matrices along with stable and efficient procedures for their computation.

Bauer's method calls for the Cholesky factorization to get the desired spectral factor. This routine assumes the input matrix to be symmetric and positive definite which is the case of the spectral factorization problem. However, if we aim at modifying the method in order to capture the non-symmetric plus/minus factorization case, we need to leave this concept and employ another technique since the companion matrix is no longer symmetric.

The Cholesky factorization decomposes the input matrix into a product of two matrices basically that are upper and lower triangular respectively. Considering this observation, the most natural alternative for the non-symmetric plus/minus case seems to be the LU-factorization concept.

Definition 4.2 (general LU-factorization): LU factorization expresses any square matrix A as the product of a permutation of a lower triangular matrix and an upper triangular matrix,

$$A = LU$$

where L is a permutation of a lower triangular matrix with ones on its diagonal and U is an upper triangular matrix.

The permutations are necessary for theoretical reasons in the general case. For instance, the matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

cannot be expressed as the product of triangular matrices without interchanging its two rows. However, the special band structure of the companion matrices can be exploited to show that the permutations are not necessary and the factorization can be expressed simply as a product of a lower and an upper triangular matrix.

Lemma 4.1: Given a scalar discrete-time two-sided polynomial $p(z)$ with roots not lying on the unit circle, its companion matrix can be factored in the form $T_p = LU$ where L and U are lower and upper triangular matrices respectively.

Proof: If a (possibly two-sided) polynomial p is nonzero at the unit circle then the principal minors of its companion matrix are known to be nonzero, see the reasoning in [20]. Further, according to [9], Theorem 3.2.1, a matrix A has the desired lower-upper triangular factorization if its all principal minors are nonzero. Combining these two observations, we arrive at the statement of the lemma.

Following Lemma 4.1, a new algorithm for polynomial plus/minus factorization is suggested in the next subsection.

B. Plus/minus factorization algorithm

Given a (scalar, one-sided) 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 (4.1), 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

$$\tilde{p}(z) = \tilde{p}^+(z)\tilde{p}^-(z) \tag{6}$$

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

Having composed the companion matrix $T_{\tilde{p}}^N$ of sufficiently high order N, its LU factorization is performed. An approximation to the plus and minus factors of $\tilde{p}$ can then be read from the last column of the L and U factors respectively, similarly to the spectral factorization case.

The degree shift yielding the two-sided polynomial $\tilde{p}$ is necessary to assure correct decomposition of $\tilde{p}$ into stable and antistable parts. If the shift were not performed or were different from δ, the decomposition would still work in principle , however, the strict stability and antistability of particular factors would be lost.

Detailed description of the resulting algorithm follows.

Algorithm 4.2: Scalar discrete-time plus/minus factorization.

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

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

Step 1 - *Choice of the companion matrix size.*
Decide about the number N. N 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 [23] 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 - *Construction of $T_{\tilde{p}}^N$:*
Following the section 4.8.1, construct the Sylvester companion matrix related to $\tilde{p}$ of order N.

Step 4 - *LU decomposition of $T_{\tilde{p}}^N$:*
Perform the LU decomposition of $T_{\tilde{p}}^N$:

$$T_{\tilde{p}}^N = LU$$

L and U are lower and upper triangular matrices respectively.

Step 5 - *Construction of polynomial factors:*
Columns of the L and U matrices contains a nonzero vector l, u of length $\delta+1$ and $d-\delta+1$ lying under and above the main diagonal respectively. Take the last full column $l = [l_0, l_1, \ldots, l_\delta]$ to create the plus factor of $p(z)$ as

$$p^+(z) = l_0 + l_1 z + \cdots + l_\delta z^\delta$$

The minus factor is constructed in a similar way using the last vector u. $\diamond$

XI. EXAMPLE

To illutrate the usefulness of polynomial plus/minus factorization and to demonstrate the power of the proposed algorithm at the same time, we will discuss the l_1 optimal control problem.

l_1 optimization is a modern design technique, see [24] for a survey. The design goal lies in minimizing the l_1 norm of a closed loop transfer function. Such a way, the magnitude of measured output signal is minimized with respect to bounded, yet persistent input disturbances. l_1 optimal controllers have already found an application in some irrigation channel regulation problem, see [25] for instance.

Quite recently a new method has been suggested by Z. Hurak et al. for the computation of an l_1 optimal discrete-time SISO compensator, see [2]. Unlike their predcessors, the authors rely on the transfer function description purely and carefully exploit the algebraic structure of the problem. The resulting algorithm is given in [2] along with the following example.

Let us compute a feedback controller the minimizes ℓ_1 norm of the sensitivity function for a plant described by

$$G(z^{-1}) = \frac{b(z)}{a(z)} = \frac{-45 - 132z^{-1} + 9z^{-2}}{-20 - 48z^{-1} + 5z^{-2}}$$

The solution consists of the following computational steps

1) plus/minus factorization of $a(z^{-1}) = a^+(z^{-1})a^-(z^{-1})$ and $b(z^{-1}) = b^+(z^{-1})b^-(z^{-1})$
2) find the minimum degree solution to $a(z^{-1})x_0(z^{-1}) + b(z^{-1})y_0(z^{-1}) = 1$
3) find a solution to $a^-(z^{-1})b^-(z^{-1})x(z^{-1}) + y(z^{-1}) = a(z^{-1})x_0(z^{-1})$ of given degree of $y(z^{-1})$ and with minimum $\|.\|_1$ norm.
4) the optimal controller is given by

$$C(z^{-1}) = \frac{a^+(z^{-1})b^+(z^{-1})y_0(z^{-1}) + a(z^{-1})x(z^{-1})}{a^+(z^{-1})b^+(z^{-1})x_0(z^{-1}) - b(z^{-1})x(z^{-1})}$$

The first step can be efficiently and reliably performed using Algorithm 4.2. We take small-size Sylvester matrices first for illustrative purposes, say N equal to 4. T_a and T_b read respectively

$$Ta = \begin{bmatrix} -48 & 5 & 0 & 0 \\ -20 & -48 & 5 & 0 \\ 0 & -20 & -48 & 5 \\ 0 & 0 & -20 & -48 \end{bmatrix}$$

$$Tb = \begin{bmatrix} -132 & -45 & 0 & 0 \\ 9 & -132 & -45 & 0 \\ 0 & 9 & -132 & -45 \\ 0 & 0 & 9 & -132 \end{bmatrix}$$

and their LU factorization gives rise to

$$T_a^+ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.4167 & 1 & 0 & 0 \\ 0 & 0.3993 & 1 & 0 \\ 0 & 0 & 0.4003 & 1 \end{bmatrix}$$

$$T_a^- = \begin{bmatrix} -48 & 5 & 0 & 0 \\ 0 & -50.083 & 5 & 0 \\ 0 & 0 & -49.997 & 5 \\ 0 & 0 & 0 & -50 \end{bmatrix}$$

and

$$T_b^+ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -0.06818 & 1 & 0 & 0 \\ 0 & -0.06663 & 1 & 0 \\ 0 & 0 & -0.06667 & 1 \end{bmatrix}$$

$$T_b^- = \begin{bmatrix} -132 & -45 & 0 & 0 \\ 0 & -135.1 & -45 & 0 \\ 0 & 0 & -135 & -45 \\ 0 & 0 & 0 & -135 \end{bmatrix}$$

These matrix factors give a fair approximation to a^+, a^-, b^+, b^- polynomials:

$$a^+ = 0.40003z^{-1} + 1, a^- = -49.997z^{-1} + 5$$
$$b^+ = -0.067z^{-1} + 1, b^- = -135z^{-1} - 45$$

To get more accurate results, N is increased. Taking $N = 20$ yields perfectly accurate results,

$$a^+ = 2/5z^{-1} + 1, a^- = -50z^{-1} + 5$$
$$b^+ = -1/15z^{-1} + 1, b^- = -135z^{-1} - 45$$

XII. LU Factorization of Toeplitz Matrices

The LU decomposition can be performed via standard routines, see [9] for instance, implemented in standard packages such as LAPACK or commercial MATLAB. Nevertheless, thanks to the strong structurallity of involved Toeplitz matrices, dedicated efficient routines for their LU factorization can be developed.

We assume the L and U factors in special forms depicted in Figure 3. Analyzing the product LU, the procedure given in the figure in the form of a MATLAB pseudocode can be developed to receive subsequent iterations of l and u vectors.

XIII. 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.

Encouraged by the success in modifying a spectral factorization algorithm for the plus/minus factorization case, we decided to re-visit another classical spectral factorization routine, namely the Bauer's method of the 1950s. This idea has proved fruitful and our efforts resulted in another plus/minus factorization routine. As a by product, a recursive LU factorization procedure for Toeplitz matrices has been developed that is of more general impact and can be of use in other areas of applied mathematics as well. Performance of the method was demonstrated by an l_1 optimal control system design example.

XIV. Acknowledgments

The authors gratefully acknowledge the contribution of National Research Organization and reviewers' comments.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

References

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

[2] Hurák, Z., Böttcher, A., Šebek, M., *Minimum distance to the range of a lower triangular Toeplitz operator in l1 norm and application in l1 optimal control*, SIAM Journal on Control and Optimization, Vol. 45, No. 1, pp. 107-122, 2006.

[3] Kwakernaak H., Šebek M, *Polynomial J-Spectral Factorization*, IEEE Trans. Automatic Control, Vol. 39, No.2, pp. 315-328 (1994).

[4] 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.

[5] Hromčík M., Šebek M., *Numerical and Symbolic Computation of Polynomial Matrix Determinant*, Proceedings of the 38th Conference on Decision and Control CDC'99, Phoenix AZ, USA, December 7-10, 1999.

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

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

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

[9] Golub G. H., Van Loan C. F., "Matrix Computations", The Johns Hopkins University Press, Baltimore and London, 1990.

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

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

[12] 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.

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

[14] 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.

[15] 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.

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

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

[18] Bauer, F. L.: Ein direktes iterations verfahren zur Hurwitz-zerlegung eines polynoms (in German), Arch. Elektr. Uebertragung, vol. 9, pp. 285290, 1955.

[19] Bauer, F. L.: Beitrage zur entwicklung numerischer verfahren fur programmgesteuerte rechenanlagen, II. Direkte faktorisierung eines polynoms (in German), Sitz. Ber. Bayer. Akad. Wiss., pp. 163203, 1956.

[20] Wu, S.P., Boyd, S. and Vandenberghe, L.: FIR Filter Design via Spectral Factorization and Convex Optimization, Applied Computational Control, Signal and Communications, Biswa Datta editor, Birkhauser, 1997.

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

[22] Hromčík, M., Ježek, J. and Šebek, M.: *New Algorithm for Spectral Factorization and its Practical Application.* 6th European Control Conference ECC 2001, Porto, Portugal, September 4-7, 2001, pp. 3104-3109

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

[24] Dahleh, M.A. and Diaz-Bobillo, I.J.: Control of Uncertain Systems: A Linear Programming Approach, Prentice Hall, New Jersey, 1995.

[25] Malaterre, P.-O., Khammash, M.: l-1 Controller Design for a High-Order 5-pool Irrigation Canal System, IEEE-CDC conference, December, 2000, in Sydney, Australia.

[26] Needham, T.: Visual Complex Analysis, Oxford University Press, 1997.