Proceedings of the 8th IEEE Mediterranean Conference
on Control and Automation (MED 2000)
Rio, Patras, GREECE 17-19 July, 2000

SPECTRAL FACTORIZATION BY MEANS OF DISCRETE FOURIER TRANSFORM *

JAN JEŽEK†, MARTIN HROMČÍK†‡, MICHAEL ŠEBEK†‡

† Institute of Information Theory and Automation
182 08 Prague, Czech Republic
{hromcik,msebek}@utia.cas.cz

‡Trnka Laboratory for Automatic Control
Czech Technical University, Prague, Czech Republic

Abstract. A new algorithm is presented for the spectral factorization of a two-sided symmetric polynomial. This algorithm being combined with digonalization techniques for polynomial matrices can also be utilized in the multivariable case to calculate the spectral factor of a discrete-time para-Hermitian polynomial matrix. The proposed method is based on the discrete Fourier transform theory (DFT) and its relation to the Z-transform. Involving DFT into the computations brings high efficiency and reliability due to desirable numerical properties of the fast Fourier transform algorithm standing behind. The paper also includes an illustrative numerical example and discusses the numerical properties of the suggested routine with respect to other existing techniques.

Key Words: Spectral factorization, FFT, Z-transform, numerical methods.

1 Introduction

This paper describes a new method for the spectral factorization of a discrete-time symmetric two-sided polynomial $m(z)$. A two-sided polynomial

$$m(z) = \sum_{i=-d}^{d} m_i z^i,$$

where m_i are real numbers, is said to be discrete-time symmetric if it equals its adjoint $m^\sim(z)$ where $m^\sim(z)$ is defined as $m^\sim(z) = m(z^{-1})$. Clearly, $m(z)$ is symmetric if $m_i = m_{-i}$. It is also evident that the symmetry makes $m(z)$ real for $|z| = 1$. In addition, we require $m(z)$ being positive here. The spectral factor of such $m(z)$ is then the (one-sided) real polynomial $x(z) = x_0 + x_1 z^{-1} + \cdots + x_d z^{-d}$ such that $m(z) = x(z)x^\sim(z)$ and $x(z)$ is Schur, that is, has all its zeros z_i inside the unit circle in the complex plane. The problem has unique solution, up to the sign.

A matrix equivalent to the scalar case is the computation of the spectral factor $X(z) = X_0 + X_1 z^{-1} + \cdots + X_d z^{-d}$ to a discrete-time para-Hermitian two-sided polynomial matrix

$$M(z) = \sum_{i=0}^{d} (M_i z^i + M_i^T z^{-i}),$$

nonsingular for $|z| = 1$. Here $X(z)$ is required to be Schur-stable again and to fulfill the equation $M(z) = X(z)X^T(z^{-1})$.

The spectral factorization is one of the basic operations in the polynomial approach to the synthesis of linear control systems [4, 5]. It is a part of the procedure of quadratic functional minimization under a condition of causality. Particular forms of the functional to be minimized lead to some well known control design problems, such as Wiener filtering, linear-quadratic controller (LQ),

*This work has been supported by the Grant Agency of the Czech Republic under contract No. 102/99/1368 and by the Ministry of Education of the Czech Republic under contract No. VS97/034.

quadratic optimal observer (Kalman filter), linear-quadratic controller with Gaussian noise (LQG) and some others. All these tasks, being solved via polynomial methods, involve spectral factorization as the crucial computational step [4, 5].

In addition, a more general version of the standard spectral factorization, usually called the J-spectral factorization, is often considered in the H_∞ control theory [9]. Some numerical procedures for J-spectral factorization rely on successive scalar spectral factorizations [8] and hence the suggested algorithm can be of use in this context as well.

2 Existing methods

In any case, the spectral factor for $d \geq 2$ cannot be achieved by a finite number of algebraic operations. Therefore all numerical algorithms for its computation are iterative and give just an approximation to the genuine factor. Some existing approaches to this problem are mentioned in this section.

The most natural way is based on the computation of roots of a polynomial. Obviously, $m(z) = \sum_{i=-d}^{d} m_i z^i$ being a symmetric two-sided polynomial yields $m^\star(z) = z^{-d} m(z)$ to be a one-sided polynomial in z^{-1} with roots symmetric with respect to the unit circle in the complex plane. The roots of $m^\star(z)$ (all are nonzero) are the zeros of $m(z)$.

This idea can be directly used to evaluate the spectral factor. Having determined the roots $r_1, r_2, \ldots, r_d$ of $m^\star(z)$ via any standard procedure for polynomial roots [3] and considering that $m(z) \neq 0$ 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^- is the set of zeros of the spectral factor $x(z)$ and having this set at hand the factor itself can be easily constructed.

Employing also the symmetry of the roots, a more sophisticated procedure can be suggested to reduce the degree of the polynomial the roots of which have to be computed.

At any rate, performance of this procedure depends on the accuracy of the computed polynomial roots. If these roots are single and not close to each other, standard numerical routines [3] can determine them with good precision and the resulting factor can be given accurately as well. However, it is well known that the relative accuracy of a computed root decreases as its multiplicity grows [3], and so does the accuracy of the spectral factor thus obtained.

Since this case is rather frequent in control design tasks and since the explicit knowledge of roots is not necessary, attention has been paid to the development of other routines that avoid the direct roots evaluation. One such procedure, relying on successive Newton-Raphson iterations and solutions of symmetric polynomial equations, was published in [11] and is implemented in the Polynomial Toolbox 2.0 for Matlab [6]. Recently some new approaches to spectral factorization have appeared based on the analysis of quadratic forms [10]. These techniques can also be extended to the matrix case, unlike that based on simple roots evaluation.

In the next sections we will introduce a completely new approach to the problem. 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

For a vector of complex numbers, DFT is defined as follows:

Definition 1 (see [2, 1]) - *direct DFT*:
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

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

The vector $\boldsymbol{y}$ is called the image of vector $\boldsymbol{p}$. $\diamond$

Definition 3.2 (see [2, 1]) - *inverse DFT*:
If $\boldsymbol{y} = [y_0, y_1, \ldots, y_N]$ is a vector of complex numbers, then its inverse *DFT* is given by a vector $\boldsymbol{p} = [p_0, p_1, \ldots, p_N]$, where

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

$\diamond$

If $\boldsymbol{y}$ is an image of $\boldsymbol{p}$, then the formula (2) returns the original vector $\boldsymbol{p}$ [2, 1]. Hence the relation (2) is inverse to relation (1).

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 [2]. One of the experimental identification methods employs DFT as well [7]. 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, 13] and to treat robustness analysis problems of certain kind [14].

For numerical computation of DFT, the efficient recursive FFT algorithm was developed by Cooley and Tukey in 1965 [2], [1]. If the length of the input is a power of two, a faster version of

FFT (sometimes called *radix-2 FFT*) can be employed [2, 1]. In general, the FFT routine features a highly beneficial computational complexity and involves $O(N\log(N))$ multiplications and additions for a vector of length N.

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

4 Spectral Factorization and DFT

4.1 Theory

The spectral factorization problem consists in the following. Given a symmetric polynomial

$$m(z) = m_d z^d + \cdots + m_1 z + m_0 + m_1 z^{-1} + \cdots + m_d z^{-d} ,$$

positive for $|z| = 1$, we are looking for a Schur stable polynomial

$$x(z) = x_0 + x_1 z^{-1} + \cdots + x_d z^{-d}$$

satisfying

$$x(z)x(z^{-1}) = m(z) .$$

In order to solve the equation, we logarithmize it. As $m(z)$ is holomorphic and nonzero in $1-\varepsilon < |z| < 1+\varepsilon$ and $x(z)$ in $1-\varepsilon < |z|$ including $z = \infty$, the logarithms exist:

$$\ln x(z) = y(z), \quad \ln m(z) = n(z) .$$

Here $n(z)$, obtained from the given $m(z)$, is a symmetric (infinite) power series

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

It can be easily decomposed,

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

with power series

$$y(z^{-1}) = y_0 + y_1 z^{-1} + \cdots = \frac{n_0}{2} + n_1 z^{-1} + \cdots ,$$

holomorphic and nonzero for $1 - \varepsilon < |z|$. Finally, the spectral factor $x(z)$ is recovered as

$$x(z) = \mathrm{e}^{y(z^{-1})} = x_0 + x_1 z^{-}1 + \cdots . \qquad (3)$$

Since $y(z)$ is holomorphic in $1 - \varepsilon < |z|$, so is $x(z)$ and hence it can be expanded according to (3). Moreover, as a result of exponential function, $x(z)$ is nonzero in $1 - \varepsilon < |z|$. In other words, it has all its zeros inside the unit disk and is therefore Schur stable. Note also that $x(z)$ 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 $y(z)$ is an infinite power series.

4.2 Numerical Algorithm

Numerical implementation of this procedure follows the ideas considered above. Polynomials $x(z)$ is represented by its coefficients x_i, $i = 0 \ldots r$ or by function values X_k in the Fourier interpolating points g^k, $k = -R \ldots 0 \ldots R$, where $R \geq d$, $g = \mathrm{e}^{\mathrm{j}\frac{2\pi}{2R+1}}$. Power series are accordingly approximated by a finite set of their coefficients or by their values in a finite number of points on the complex unit circle. Some operations, namely the decomposition of $n(z)$ into $y(z)$ and $y(z^-1)$, are performed in the time domain x_i, other (evaluation of logarithmic and exponential functions) in the frequency domain X_k. For mutual conversion between domains, we use the shifted discrete Fourier transform defined as

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

approximating the Z-transform (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 taken into account during the computation. This value can be considered as a simple tuning knob of the computational process.

Resulting numerical routine looks then as follows:

Algorithm 1: Scalar discrete-time spectral factorization.

Input: Scalar symmetric polynomial $m(z) = m_d z^d + \cdots + m_1 z + m_0 + m_1 z^{-1} + \cdots + m_d z^{-d}$

Output: Polynomial $x(z) = x_0 + x_1 z^{-1} + \cdots + x_d z^{-d}$, the spectral factor of $m(z)$.

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

Step 2 - *Direct FFT (I):*
Using the FFT algorithm, perform direct DFT, defined by (1), on the vector

$$\boldsymbol{m} = [\underbrace{m_0, m_1, \ldots, m_n, 0, 0, \ldots, 0, m_n, \ldots, m_1}_{2R+1}]$$

In this way, the set $\boldsymbol{M} = [M_0, M_1, \ldots, M_{2R}]$ of the values of $m(z)$ at the Fourier points is obtained. Owing to the symmetry of $m(z)$ and assuming $m(z)$ with real coefficients only, M is symmetric and real as well.

Step 3 - *Logarithmization:*
Compute the logarithms $N_i = \ln(M_i)$ of all particular M_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 $n(z) = \ln(m(z))$ at related Fourier points on the unit complex circle.

Step 4 - *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 symmetric 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 $\ln(m(z))$ for the given R, perform inverse DFT, defined by (2), on the vector $\boldsymbol{N}$ using the FFT algorithm.

Step 5 - *Decomposition:*
Take the "causal part" $\boldsymbol{y}$ of $\boldsymbol{n}$: $\boldsymbol{y} = [n_0/2, n_1, \ldots, n_R]$.

Step 6 - *Direct FFT (II):*
Evaluate $y(z) = n_0/2 + n_1 z^-1 + \ldots + n_R z^{-R}$ at the Fourier points by applying direct FFT on the set

$$[\underbrace{\frac{n_0}{2}, n_1, \ldots, n_R, 0, 0, \ldots, 0}_{2R+1}]$$

and get $\boldsymbol{Y} = [Y_0, \ldots, Y_{2R+1}]$.

Step 7 - *Exponential function:*
To get the spectral factor, the exponential function $x(z) = e^{y(z)}$ remains to be evaluated. First we compute the values of $x(z)$ at the Fourier points: $\boldsymbol{X} = [e^{Y_1}, \ldots, e^{Y_{2R+1}}]$.

Step 8 - *Inverse FFT (II):*
Finally, the coefficients $\boldsymbol{x} = [x_0, \ldots, x_{2R}]$ of $x(z)$ are recovered by inverse FFT performed on the vector $\boldsymbol{X}$. The resulting approximation to the spectral factor $x(z)$ then equals $x(z) = x_0 + x_1 z^{-1} + \cdots + x_d z^{-d}$. $\diamond$

Note that one obtains $2R+1$ coefficients of $x(z) = x_0 + x_1 z^-1 + \cdots x_R z^{-2R}$ in the Step 8. However, $x(z)$ being the spectral factor of $m(z)$ is known to be of degree d only and only the first $d+1$ coefficients of $x(z)$ 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.

4.3 Example

Consider the two-sided polynomial $m(z) = 0.5z^{-1} + 1.25 + 0.5z$. Following the **Algorithm 1** and employing MATLAB, we receive the spectral factor in the next steps:

Step 1: Let us choose a small R first, say $R = 3$.

Step 2: Direct FFT of the vector $\boldsymbol{m} = [1.25, 0.5, 0, 0, 0.5]$ yields $\boldsymbol{M} = [2.2500, 1.5590, 0.4410, 0.4410, 1.5590]$. Note that M is real and symmetric.

Step 3: Natural logarithms of particular M_i's give rise to the vector $\boldsymbol{N} = [0.8109, 0.4441, -0.8187, -0.8187, 0.4441]$, which is real and symmetric as well.

Step 4: Inverse FFT of $\boldsymbol{N}$ results in the vector $\boldsymbol{n} = [0.0123, 0.4820, -0.0827, -0.0827, 0.4820]$ related to the symmetric two-sided polynomial $n(z) = -0.0827z^{-2} + 0.4820z^{-1} + 0.0123 + 0.4820z^1 - 0.0827z^2$.

Step 5: The causal part of $n(z)$ is the polynomial $y(z) = 0.0123/2 + 0.4820z^{-1} - 0.0827z^{-2}$

Step 6: Values of $y(z)$ at the Fourier points are obtained by direct FFT: $\boldsymbol{Y} = [0.4055, 0.2220 - 0.4098i, -0.4094 - 0.3620i, -0.4094 + 0.3620i, 0.2220 + 0.4098i]$.

Step 7: $\boldsymbol{X} = [e^{Y_i}] = [1.5000, 1.1452 - 0.4975i, 0.6210 - 0.2352i, 0.6210 + 0.2352i, 1.1452 + 0.4975i]$.

Step 8: Finally, inverse FFT of $\boldsymbol{X}$ returns the result:
$\boldsymbol{x} = [1.0065, 0.4851, 0.0337, -0.0213, -0.0040]$ and $x(z) = 1.0065 + 0.4851z^{-1} + 0.0337z^{-2} - 0.0213z^{-3} - 0.0040z^{-4}$. We know that the degree of the genuine spectral factor is 1 and therefore we take $x(z) = 1.0065 + 0.4851z^{-1}$ as the result. Indeed, the remaining coefficients are considerably smaller than the considered ones. $\diamond$

As R increases, the terms that should be zero according to the theory really become smaller while the valuable ones converge to the genuine spectral factor coefficients. For instance, taking $R = 50$ and repeating the whole procedure gives rise to a polynomial of degree 100 with the significant part $x(z) = (1+\epsilon) + (0.5+\epsilon)z^{-1}$ where $|\epsilon| < 10^{-15}$ and with all remaining coefficients in absolute value smaller than 10^{-14}. This output can be considered as a good result with respect to the exact spectral factor being $1 + 1/2z^{-1}$. Of course, the positive dependence of accuracy on R holds only up to a certain limit in practice - for R being extremely large, numerical problems usually arise that spoil the results, and therefore taking R many times greater than the suggested value cannot be recommended as a rule.

4.4 Modifications of the Algorithm

The basic version of the routine proposed above is based on the shifted dicrete Fourier transform. This modification of DFT appears beneficial 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). 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

$$\underbrace{[m_0, m_1, \ldots, m_n, 0, 0, \ldots, 0, m_n, \ldots, m_1]}_{2^R}$$

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 5, the decomposition reads $\boldsymbol{y} = [n_0/2, n_1, \ldots, n_R/2]$ instead of $\boldsymbol{y} = [n_0/2, n_1, \ldots, n_R]$. This minor modification of the proposed method further increases its efficiency since the powerful radix-2 FFT can be called.

The algorithm can also be extended to the situations when $m(z)$ has some coupled roots on the complex unit circle. In this case $|n(z)| = |\ln(m(z))|$ becomes infinite for some z on the unit circle and $N_i = n(q_i) = \ln(m(q^i))$ may appear very large for some Fourier interpolation points q^i. However, as the spectral factor is known to exist in this case [4, 5], the inverse Z-transform of $n(z)$ is guaranteed to exist as well. Hence we can ignore the contributions of such large N_i's to the approximating DFT sum in the Step 3, taking the singularity of related Fourier points into account and considering a large amount of interpolation points in total. Of course, this replacement by zeros further decreases the accuracy of the Z-transform approximation in this case. The bound above which the N_i's are supposed infinite is another "tuning parameter" of this embedded procedure.

Throughout the paper only m_i's real have been implicitly expected. Nevertheless, the algorithm can be directly applied to symmetric polynomials with complex coefficients as well. In this case, the definition of the adjoint slightly differs - having $m(z) = \sum_{i=0}^{d} x_i^* z^i$, its adjoint is given by $m^{\sim}(z) = \sum_{i=0}^{d} x_i^* z^{-i}$ where the star stands for complex conjugate. Taking this definition, all the ideas standing behind the algorithm remain valuable.

4.5 Implementation and Practical Experience

The algorithm has been implemented in MATLAB, using the standard routines for the fast Fourier transform (`fft` command). The practical experience is good. Though the complete routine seemingly issues a high number of manipulations, it is pretty fast thanks to the $O(R \log R)$ computational complexity of the FFT algorithm. It runs faster than the Newton-Raphson iterations routine as it is programmed in the Polynomial Toolbox 2.0 for Matlab [6], returning results of comparable accuracy. The execution times are similar with the method based on direct evaluation of roots, nevertheless, in the case of multiple roots the accuracy of the newly proposed method appears better provided the number of interpolation points is adequately chosen.

5 Further Research

This approach cannot be directly extended to the matrix case - the product of two matrices does not commute in general and since $X(z)X(z^{-1}) \neq X(z^{-1})X(z)$, one cannot write $\ln M(z) = \ln(X(z)) + \ln(X(z^{-1}))$ and perform the decomposition. Nevertheless, in combination with techniques for diagonalization of polynomial matrices and for factorization of a unimodular matrix, described for instance in [8], the proposed routine can be used to factor particular entries of an equivalent diagonal matrix.

A modification for continuous time polynomials is under research. However, it does not seem to be as beneficial as it is in the discrete time case since the relation between the Laplace transform, replacing the role of Z-transform in the continuous time domain, and the DFT is not so close.

6 Conclusion

A new method for the discrete-time spectral 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 illustrated by a simple numerical example and its numerical properties are discussed with respect to other existing algorithms.

Possible extensions for polynomial matrices and for the continuous time case have been discussed as well. They are under research now. The difficulties were described that occur and that prevent this approach from being adopted for these cases directly.

References

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

[2] Čížek V., *Discrete Fourier Transforms and Their Applications*, Adam Hilger Ltd, Bristol and Boston (1986).

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

[4] Kailath T., *Linear Systems*, Prentice Hall, New Jersey (1980).

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

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

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

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

[9] Green M., Glover K, Limebeer D. and Doyle J., *A J-spectral Factorization Approach to H_∞ control*, SIAM J. on Contr. Opt., vol. 28, pp. 1350-1371 (1990).

[10] Kaneko O. and Fujii T., *Discrete Time Behavioral Dissipativeness and Spectral Factorization via Quadratic Difference Forms*, Proceedings of the 5th European Control Conference ECC'99, Karlsruhe, Germany, October 31 - September 3 (Session BA9), 1999.

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

[12] Hromčík M., Šebek M., *New Algorithm for Polynomial Matrix Determinant Based on FFT*, Proceedings of the 5th European Control Conference ECC'99, Karlsruhe, Germany, October 31 - September 3 (Session DA1), 1999.

[13] 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.

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