ALGEBRA FOR 2-D POLYNOMIAL MATRICES

P. Zezula[1], J. Ježek[2]

[1]xzezulap@fel.cvut.cz
ČVUT FEL
Katedra řídicí techniky (K335)
Karlovo nám. 13
121 35 Praha 2

[2]jezek@utia.cas.cz
ÚTIA AV ČR
Pod vodárenskou věží 4
182 08 Praha 8

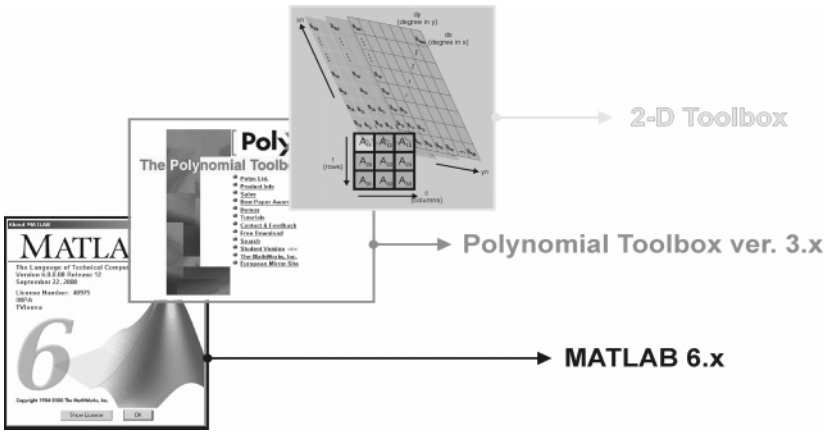
Keywords: 2-D polynomial algebra, spectra methods, polynomial algebra via spectra,

Since the 80', some 2-D problems have been described in connection with the polynomial approach. The 2-D area stands outside the mainstream investigation, whereas the area of 1-D problems has been worked up as far as tools like Polynomial-Toolbox for Matlab, the first serviceable set of functions for practical work with polynomials.

Now the first small step resulted in a package of elementary algebraical operations with 2-D polynomial matrices called shortly 2-D Toolbox. The toolbox allow to work with 2-D polynomial matrices in the form of classical polynomial matrices, spectra of polynomial matrices and two sided polynomial matrices. Obviously each form is supported with the whole set of algebraical operations, constructors and converters. Especially the spectra approach offers often result many times faster as via classical way.

THE EMBEDDING AND RESOURCES

The 2-D toolbox use one of the most enlarged CAD applications for engineers - Matlab as basic platform and interface. Further is used the Polynomial Toolbox 3.0 .



The advantages of both platforms used for the 2-D Toolbox are:

Matlab 6.x
- object management
- scalar matrix operations
- built-in FFT

Polynomial Toolbox for Matlab
- data structures
- handling of numerical accuracy

2-D Toolbox
- own objects
- 2-D polynomial algebra
- 2-D spectra algebra

IMPLEMENTATION OF 2-D TOOLBOX

The data structures have been chosen as close as possible to the 1-D polynomial data structures. Instead of variables like z_1 and z_2 we have choosen variables x and y – see (1) .

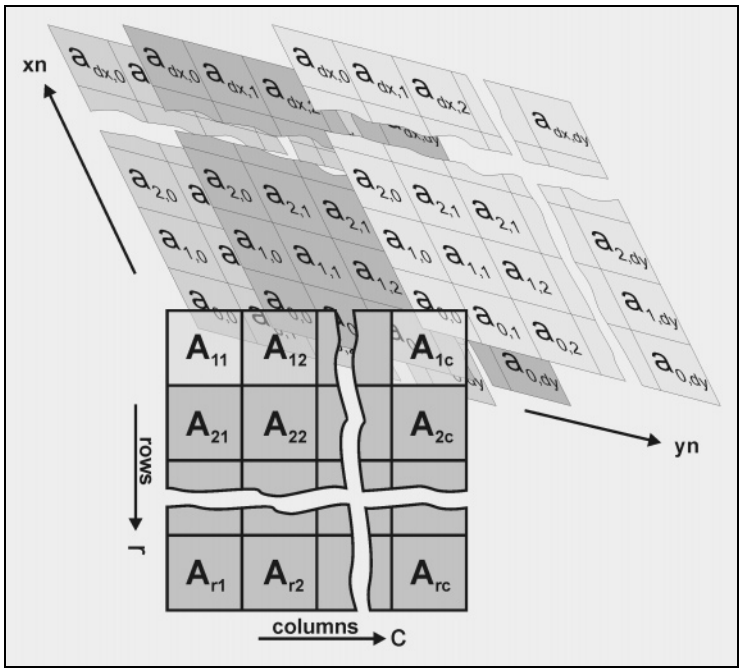


Figure 1: 2-D polynomial matrix

As introduced above, 2-D Toolbox contain two separate sets of algebraical functions (methods). One for the 2-D algebra via polynomial coefficients and the second one for the same algebra but via spectra. These require two different Matlab-objects. For comparing the data structures see Tab 1. There are obviously converters between each other.

Tab 1: Data structure table – Dual description of 2-D polynomials.

"TIME" Structure of class **POL2**	⇔	"SPECTRA" Structure of class **POL2SP**
X.d	Polynomial degrees in "TIME" [dx,dy]	X.d
X.s	Matrix size [r,c]	X.s
X.c	Matrix of polynomial coefficients [r, c, xn, yn]	X.c
X.v	Vector of used variables default ['x','y']	X.v
X.u	Users info	X.u
X.version	Version number	X.version
-	Number of used spectra points	X.n
-	0/1 – imag/real indicator	X.k

Example: matrix in polynomial coefficients 4x1 with degrees [2,1] - appearance of the user interface

```
>> C=pol2(A)

2-D polynomial matrix: 4-by-1
deg(x)=2, deg(y)=1

C =

 (31-11i) + (-8+32i)x + (37+16i)y + (34-1i)x^2 + (45+21i)x*y + (-31+19i)x^2*y
 (11+43i) + (-12-12i)x + (27+10i)y + (-5+4i)x^2 + (14+11i)x*y + (-1+39i)x^2*y
 (20+19i) + (-33+25i)x + (-6+19i)y + (46-4i)x^2 - 25x*y + (-9+6i)x^2*y
 (-41-11i) + (33-9i)x + (12+9i)y + (-35-44i)x^2 + (-15-22i)x*y + (-4+3i)x^2*y
```

DESCRIPTION OF 2-D POLYNOMIALS

Note: For 2-D polynomials, roots are not scalars but still 1-D polynomials (one parameter curves). Properties of such pole-curves are matter of ongoing investigations. Therefore, this description way, often used for 1-D polynomials, can not been used in this line.

There are two classical ways to describe a ordinary 2-D polynomial

$$\begin{aligned} a(x,y) = \quad & a_{0,0} + a_{1,0}x + a_{2,0}x^2 + \ldots + a_{N_x-1,0}x^{N_x-1} + \\ & + a_{0,1}y + a_{1,1}xy + a_{2,1}x^2y + \ldots + a_{N_x-1,1}x^{N_x-1}y + \\ & + a_{0,2}y^2 + a_{1,2}xy^2 + a_{2,2}x^2y^2 + \ldots + a_{N_x-1,2}x^{N_x-1}y^2 + \\ & \vdots \\ & + a_{0,N_y-1}y^{N_x-1} + a_{1,N_y-1}xy^{N_x-1} + \ldots + a_{N_x-1,N_y-1}x^{N_x-1}y^{N_y-1} \end{aligned} \tag{1}$$

via matrix of coefficients

$$\mathbf{A} = \begin{bmatrix} a_{00} & a_{10} & a_{20} & \cdots & a_{N_x-1,0} \\ a_{01} & a_{11} & a_{21} & \cdots & a_{N_x-1,1} \\ a_{02} & a_{12} & a_{22} & \cdots & a_{N_x-1,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{0,N_y-1} & a_{1,N_y-1} & a_{2,N_y-1} & \cdots & a_{N_x-1,N_y-1} \end{bmatrix}, \tag{2}$$

or via matrix of interpolated values

$$\mathbf{P} = \begin{bmatrix} p_{00} & p_{10} & p_{20} & \cdots & p_{N_x-1,0} \\ p_{01} & p_{11} & p_{21} & \cdots & p_{N_x-1,1} \\ p_{02} & p_{12} & p_{22} & \cdots & p_{N_x-1,2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{0,N_y-1} & p_{1,N_y-1} & p_{2,N_y-1} & \cdots & p_{N_x-1,N_y-1} \end{bmatrix}, \tag{3}$$

evaluated for points $\mathbf{s}_x = [s_0, s_1, s_2, s_3 \ldots s_{Nx-1}]$ substituted for variable x and
$\mathbf{s}_y = [s_0, s_1, s_2, s_3 \ldots s_{Ny-1}]$ substituted for variable y.

Special properties of such points are adduced below.

It is depend by the situation with one description form is proper to be used. User-readable is just the first description mode via matrix **A** supported by a complex method for user-friendly displaying of variables (see example below). A similar method is available for data entry. Interpretation of spectra points, especially for 2-D spectra, is not a common ability. Nevertheless, it is also possible to

show and edit the spectra of 2-D polynomials. the problem is the convertibility between this descriptions. Numerical algorithms are the core of all toolboxes promising a faster computation. Our way to get interpolated values is to choose points $\mathbf{s}_x$ and $\mathbf{s}_y$ as so-called Fourier-points and use the very fast build-in FFT dimension by dimension (for more details see [1] . Results of such interpolations are called spectra. Although without physically interpretation in this case it is now possible to apply the whole knowledge about operations with the origin and image in this transformation.

MULTIPLICATION OF 2-D POLYNOMIALS

The elementary operation which has to be inherent to all algebraical toolboxes is multiplication. Functions like determinant and power use internal this function. So that multiplication is definitely the most important function. There are some numerical algorithms to compute the multiplication.

VIA CONVOLUTION

A way to compute the multiplication is to follow the definition of multiplication and realize this one 1-by-1. It results in a convolution.

Consider the multiplication: $Z_{(x,y)} = P_{(x,y)} R_{(x,y)}$

$$z_{i,k} = p_{i,k} \otimes r_{i,k} = \sum_{w=0}^{N_k-1} \sum^{N_i-1} p_{v,w} r_{i-v,k-w} \ , \qquad (4)$$

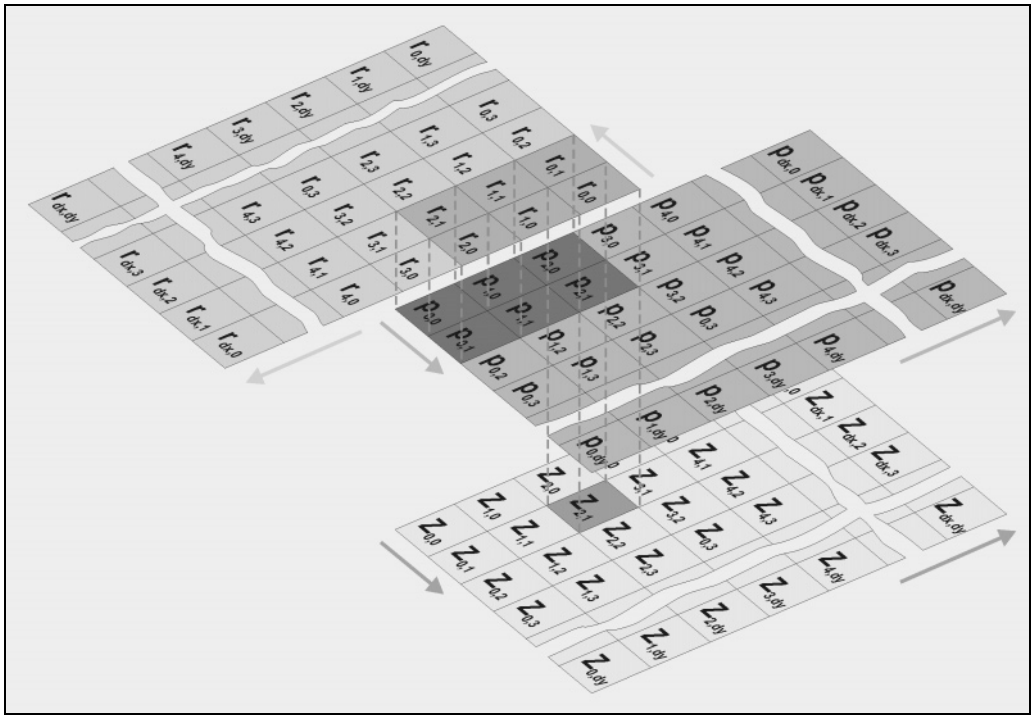


Figure 2: Computing the element $z_{2,1}$ of one 2-D polynomial via 2-D convolution

It is obviously that such a computing element-by-element and polynomial-by-polynomial realized via program cycles can not provide fast, satisfying results.

A much faster way is to avoid slow cycles and prepare the sub-matrices with coefficients into so-called Sylvester matrix. That is the usual way for polynomial matrix multiplication today.

VIA SYLVESTER MATRIX

This algorithm of the 2nd generation prepare the data in strips column-by-column $\mathbf{P}$, $\mathbf{R}$ and need just one multiplication to compute the multiplication $\mathbf{Z}$. Such pseudo-result has to bee reshaped following explicit knowledge about result degree etc. into final result.

$$
\begin{bmatrix}
\overbrace{\begin{bmatrix} P_{0,0} & 0 & \cdots & 0 \\ P_{1,0} & P_{0,0} & & 0 \\ \vdots & P_{1,0} & \ddots & 0 \\ P_{dx,0} & \vdots & \ddots & P_{0,0} \\ 0 & P_{dx,0} & \ddots & P_{1,0} \\ 0 & 0 & & \vdots \\ 0 & 0 & \cdots & P_{dx,0} \end{bmatrix}}^{dx_R+1} & & & \overbrace{\phantom{\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}}}^{dy_R+1} \\
\begin{bmatrix} P_{0,1} & 0 & \cdots & 0 \\ P_{1,1} & P_{0,1} & & 0 \\ \vdots & P_{1,1} & \ddots & 0 \\ P_{dx,1} & \vdots & \ddots & P_{0,1} \\ 0 & P_{dx,1} & \ddots & P_{1,1} \\ 0 & 0 & & \vdots \\ 0 & 0 & \cdots & P_{dx,1} \end{bmatrix} & \begin{bmatrix} P_{0,0} & 0 & \cdots & 0 \\ P_{1,0} & P_{0,0} & & 0 \\ \vdots & P_{1,0} & \ddots & 0 \\ P_{dx,0} & \vdots & \ddots & P_{0,0} \\ 0 & P_{dx,0} & \ddots & P_{1,0} \\ 0 & 0 & & \vdots \\ 0 & 0 & \cdots & P_{dx,0} \end{bmatrix} & & 0 \\
 & \begin{bmatrix} P_{0,1} & 0 & \cdots & 0 \\ P_{1,1} & P_{0,1} & & 0 \\ \vdots & P_{1,1} & \ddots & 0 \\ P_{dx,1} & \vdots & \ddots & P_{0,1} \\ 0 & P_{dx,1} & \ddots & P_{1,1} \\ 0 & 0 & & \vdots \\ 0 & 0 & \cdots & P_{dx,1} \end{bmatrix} & \ddots & 0 \\
\begin{bmatrix} P_{0,dy} & 0 & \cdots & 0 \\ P_{1,dy} & P_{0,dy} & & 0 \\ \vdots & P_{1,dy} & \ddots & 0 \\ P_{dx,dy} & \vdots & \ddots & P_{0,dy} \\ 0 & P_{dx,dy} & \ddots & P_{1,dy} \\ 0 & 0 & & \vdots \\ 0 & 0 & \cdots & P_{dx,dy} \end{bmatrix} & & \ddots & \begin{bmatrix} P_{0,0} & 0 & \cdots & 0 \\ P_{1,0} & P_{0,0} & & 0 \\ \vdots & P_{1,0} & \ddots & 0 \\ P_{dx,0} & \vdots & \ddots & P_{0,0} \\ 0 & P_{dx,0} & \ddots & P_{1,0} \\ 0 & 0 & & \vdots \\ 0 & 0 & \cdots & P_{dx,0} \end{bmatrix} \\
0 & \begin{bmatrix} P_{0,dy} & 0 & \cdots & 0 \\ P_{1,dy} & P_{0,dy} & & 0 \\ \vdots & P_{1,dy} & \ddots & 0 \\ P_{dx,dy} & \vdots & \ddots & P_{0,dy} \\ 0 & P_{dx,dy} & \ddots & P_{1,dy} \\ 0 & 0 & & \vdots \\ 0 & 0 & \cdots & P_{dx,dy} \end{bmatrix} & \ddots & \begin{bmatrix} P_{0,1} & 0 & \cdots & 0 \\ P_{1,1} & P_{0,1} & & 0 \\ \vdots & P_{1,1} & \ddots & 0 \\ P_{dx,1} & \vdots & \ddots & P_{0,1} \\ 0 & P_{dx,1} & \ddots & P_{1,1} \\ 0 & 0 & & \vdots \\ 0 & 0 & \cdots & P_{dx,1} \end{bmatrix} \\
0 & 0 & \cdots & \vdots \\
0 & 0 & \cdots & \begin{bmatrix} P_{0,dy} & 0 & \cdots & 0 \\ P_{1,dy} & P_{0,dy} & & 0 \\ \vdots & P_{1,dy} & \ddots & 0 \\ P_{dx,dy} & \vdots & \ddots & P_{0,dy} \\ 0 & P_{dx,dy} & \ddots & P_{1,dy} \\ 0 & 0 & & \vdots \\ 0 & 0 & \cdots & P_{dx,dy} \end{bmatrix}
\end{bmatrix}
\times
\begin{bmatrix} \begin{bmatrix} R_{0,0} \\ R_{1,0} \\ \vdots \\ R_{dx_R,0} \end{bmatrix} \\ \begin{bmatrix} R_{0,1} \\ R_{1,1} \\ \vdots \\ R_{dx_R,1} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} R_{0,dy_R} \\ R_{1,dy_R} \\ \vdots \\ R_{dx_R,dy_R} \end{bmatrix} \end{bmatrix}
=
\begin{bmatrix} \begin{bmatrix} Z_{0,0} \\ Z_{1,0} \\ \vdots \\ Z_{dx_Z,0} \end{bmatrix} \\ \begin{bmatrix} Z_{0,1} \\ Z_{1,1} \\ \vdots \\ Z_{dx_Z,1} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} Z_{0,dy_Z} \\ Z_{1,dy_Z} \\ \vdots \\ Z_{dx_Z,dy_Z} \end{bmatrix} \end{bmatrix}
\tag{5}
$$

The disadvantage of this way is the redundancy of data which cause useless operation burden.

Via spectra

Today, the most elegant way to compute a polynomial matrix product. The computing bulk is hidden in the well-implemented FFT. The operation outside the FFT is than just the element-by-element multiplication realized via so-called *dot* syntax **Z = P. * R** with spectra matrices. That is the corresponding operations to the convolution with polynomial coefficient matrices.

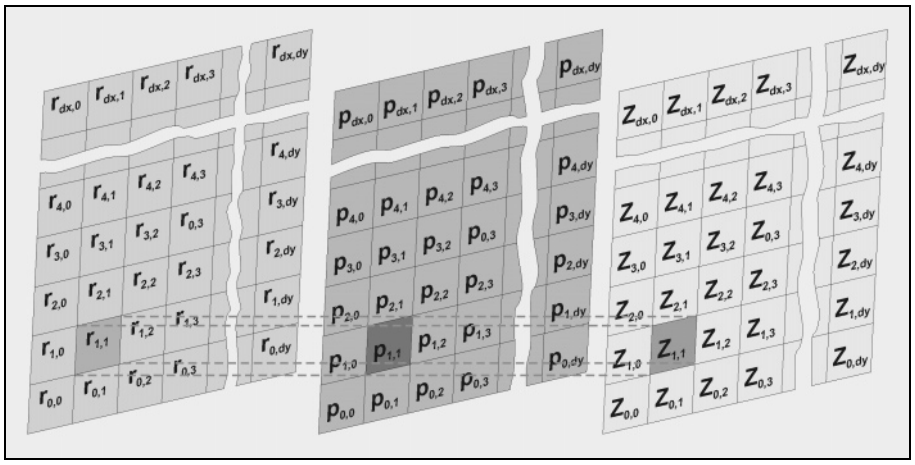


Figure 3: Computing the element $z_{1,1}$ via spectra points

Just the both degrees of the result will be computed at first to reserve enough spectra points for correct image-origin correspondence.

OPERATION BURDEN

Operation burden by computing of 2nd power of a 2-D polynomial matrix with size r × r and degrees d=dx=dy.

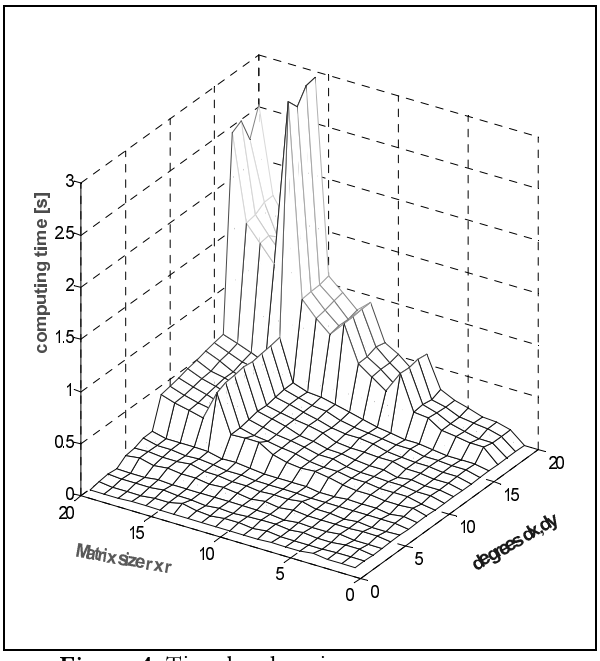


Figure 4: Time burden via spectra - measuretr

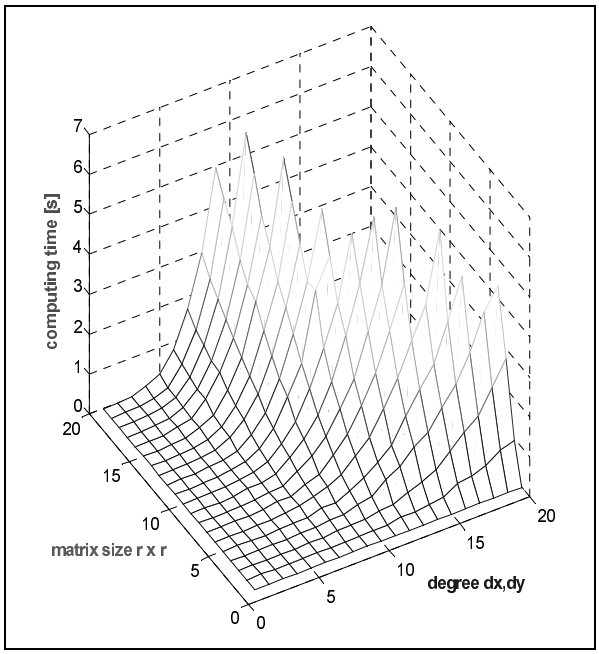


Figure 5: Time burden via polynomial coefficients

- via polynomial coefficient we need $Q = r^3\left(2d^3 - d\right)^2$ multiplications
- via spectra we need just $Q = r^3\left(4d\right)^2$ multiplications

Depictured results computed on PIV / 2,6GHz / 512MB RAM / Win2k / Matlab 6.5

Tab 2: Rate of computing time via polynomial coefficients and spectra ($t_{coefficients} / t_{spectra}$)

Degrees dx,dy \ Matrix size r×r	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2							0,5	1,9	1,0	3,5	2,5	4,5	6,0	12,6	3,5	5,2	7,4	8,5
3							1,9	1,3	4,5	5,5	10,6	15,1	20,0	30,3	10,4	14,3	15,4	19,4
4					1,9	5,2	3,0	4,0	5,7	12,0	16,3	23,3	28,3	56,7	14,4	20,8	24,3	27,6
5					2,9	8,8	3,0	6,3	10,6	24,2	22,3	34,6	46,2	56,9	21,1	26,6		
6				3,1	7,3	6,1	6,5	7,4	12,9	18,4	21,8	37,4	40,5	63,0	22,6			
7			2,1	5,9	5,8	16,6	6,0	11,4	17,6	26,8	30,4	42,1	55,9					
8			1,0	2,0	6,9	22,9	6,3	10,1	19,2	27,2	34,1	38,6	64,5					
9		0,9	4,9	2,9	6,0	14,6	7,1	13,3	17,8	31,9	44,5	45,3						
10		1,0	1,9	3,9	8,1	17,6	8,9	15,2	23,5	32,4	46,7							
11		1,9	4,9	4,5	10,6	23,2	11,0	14,7	26,7	31,8								
12		2,9	2,5	3,7	13,7	27,7	10,1	16,2	21,8	34,4								
13		2,1	2,0	6,0	12,6	23,3	8,1	16,0	23,7									
14	1,0	3,1	2,3	4,7	15,0	20,6	9,8	16,7	24,1									
15	1,1	3,1	2,7	5,5	13,6	24,3	11,9	17,2	24,4									
16	0,5	6,3	2,5	5,5	13,3	18,9	5,0	7,4										
17	1,9	5,0	2,6	6,5	14,2	21,3	10,7	16,4										
18	2,1	5,3	3,5	7,5	13,4	24,1	11,4	17,2										
19	1,9	6,3	2,5	7,1	14,8	27,1	11,8											
20	1,0	9,8	4,2	8,7	13,5	23,6	11,7											

CONCLUSION

As presented, spectra methods offer a very fast alternative to the common way of polynomial matrix handling. Especially 2-D polynomials represent a hopeful area for this approach. Further topics like non-explicit equation, well prepared under 1-D, are the mater of ongoing investigation.

REFERENCES

[1] P. Zezula, *Implementace polynomiální algebry ve spektrech*. Diplomová práce, Praha, FEL ČVUT, 2001

[2] P. Hušek and R. Pytelková, *Product of Multivariable Polynomials Using Fourier Transform*. Proc. Of 12th International Conf. on Process Control (ŘÍP 2000), Kouty nad Desnou

[3] M. Hromčík, *Numerical Algorithms for Polynomial Matrices*, Diploma Thesis, Prague, Dept. of Control Engineering FEE CTU, 2000.

[4] M. Šebek, *Multi-Dimensional Systems: Control Via Polynomial Techniques*. Doctoral Thesis, Prague, ÚTIA AV ČR, 1996

[5] G. H. Golub and C. F. van Loan, *Matrix Computation. Baltimore*, Maryland: The Johns Hopkins University Press, 1996

[6] V. Čížek, *Discrete Fourier Transforms and Their Applications*, Bristol, Adam Hilger, 1986.

[7] D. Henrion, *Reliable Algorithms for Polynomial Matrices*, CSc Thesis, Prague, ÚTIA AV ČR, 1998.

ACKNOWLEDGEMENT

The investigation of the first author has been supported by the GACR grant number 102/02/0709. The second author has been supported by the Ministry of Education of CR under contract ME 496.