Proceedings of the 4th IEEE Mediterranean Symposium
on New Directions in Control and Automation
Maleme, Krete, GREECE June 10-13, 1996

Numerical Methods for Zeros and Determinant of Polynomial Matrix

M. Šebek, S. Pejchová, D. Henrion
Institute of Information Theory and Automation
182 08 Prague, Czech Republic
(msebek@utia.cas.cz, pejchova@utia.cas.cz, henrion@utia.cas.cz)

H. Kwakernaak
Department of Applied Mathematics
University of Twente
Enschede, The Netherlands
(H.Kwakernaak@math.utwente.nl)

Abstract

As a next step in the project of systematic testing and evaluation of numerical algorithms for polynomial matrix operations[1], four algorithms for polynomial matrix determinant (modified Fadeev-Leverrier; triangularization; interpolation; and recovering from zeros) and four routines to compute polynomial matrix zeros (as roots of its determinant achieved by the first three methods above; and directly as generalized eigenvalues of the corresponding block companion matrix) are considered. The algorithms have been tested by running their MATLAB codes on numerical examples including both generic and ill-conditioned data. A typical example is reported here.

Introduction

One of two main approaches to linear control analysis and design (called factorization, algebraic, or polynomial approach) is based on manipulations with polynomial and rational matrices. The approach is relatively simple and easy to learn for students. However, although basic computational techniques have been proposed for almost every type of polynomial matrix operation and equation and for many of them prototype programs have been written, the algorithms are still far from being practically workable.

Therefore, a project[1] has been started recently to make one step forward in exploring polynomial matrix operations impact on control system design which consists of systematic investigation, modification and testing of numerical algorithms for each particular polynomial matrix operation used in control design. Due to the persistent lack of numerical mathematics, one is almost unable to compare the algorithms *theoretically*, thus we pursue the way of *experiment*. First results (concerning linear matrix polynomial equations) were reported recently [8]. In this paper, further numerical procedures are considered to compute determinant and zeros of polynomial matrices.

Determinant

The determinant of a square polynomial matrix

$$P(s) = P_0 + P_1 s + \ldots + P_m s^m,$$

denoted as

$$p(s) = \det P(s), \tag{1}$$

is defined as the standard determinant of a matrix over a ring. If $P(s)$ is $n \times n$, then

$$p(s) = p_0 + p_1 s + \ldots + p_k s^k \tag{2}$$

is a scalar polynomial of degree $k \leq mn$.

The following four algorithms have been considered to calculate the determinant: modified Fadeev-Leverrier's; triangularization; interpolation; and recovering from zeros.

[1] EC Copernicus Project CP:2424/93 entitled *Numerical algorithms for CAE based on modern polynomial control methods.*

Modified Fadeev-Leverrier's Method (MFL)

The well known procedure originally designed to compute the characteristic polynomial of a constant matrix [1, 9] (that is, $\det(Is - A)$ for a constant matrix A) was modified by Kwakernaak [3] to handle general polynomial matrices. In the series of n steps one computes scalar polynomials[2] d_i and polynomial matrices $Q_i(s)$ in the following manner:

$$\begin{array}{ll} & Q_0 = I_n \\ d_1 = -\mathrm{tr}\,(Q_0 P) & Q_1 = Q_0 P + d_1 I_n \\ d_2 = -\frac{1}{2}\mathrm{tr}\,(Q_1 P) & Q_2 = Q_1 P + d_2 I_n \\ \dots & \dots \\ d_i = -\frac{1}{i}\mathrm{tr}\,(Q_{i-1} P) & Q_i = Q_{i-1} P + d_i I_n \\ \dots & \dots \\ d_n = -\frac{1}{n}\mathrm{tr}\,(Q_{n-1} P) & 0 = Q_{n-1} P + d_n I_n \end{array}$$

As it is clear from the second relation of the last row, one finally gets the desired determinant as

$$p(s) = d_n(s).$$

Triangularization (TRI)

The given polynomial matrix $P(s)$ can by transformed to a triangular form $P_\flat(s)$ by a series of elementary column operations or, in other words, via postmultiplication by a unimodular matrix $U(s)$ with $\det U(s) = 1$

$$P_\flat(s) = P(s)U(s).$$

Then the desired determinant $p(s) = \det P_\flat(s)$ is found as the product of the diagonal elements of $P_\flat(s)$

$$p(s) = \prod [P_\flat(s)]_{ii}.$$

Interpolation (INT)

After substituting into $P(s)$ real (complex) numbers s_i, $i = 0, 1, \dots, k$, one gets $k+1$ scalar constants $p(s_i) = \det P(s_i)$ computed as determinants of constant matrices. The coefficients of the desired polynomial matrix determinant (2) can then be recovered from the solution of the (constant matrix) equation

$$[p_0 \dots p_k] \begin{bmatrix} 1 & \dots & 1 \\ s_0 & \dots & s_k \\ \dots & \dots & \dots \\ s_0^k & \dots & s_k^k \end{bmatrix} = [p(s_0), \dots, p(s_k)]$$

The square matrix on the left hand side is the Vandermonde matrix that is known to be ill-conditioned for larger k. This does not matter, however, since a special routine (*Vandermonde system solver*) is available [2] that is surprisingly numerically stable. Alternatively, one can take more than $k = mn$ points and look for a "least-squares" solution.

In fact, the leading and the closing coefficients of $p(s)$ can also be computed directly as $p_0 = \det P_0$ and $p_{mn} = \det P_m$, respectively, and so the problem size is 2 less. If, in addition, only purely imaginary interpolations points are chosen (that is, $s_i = j\omega_i$) the task further splits into two separate problems of half sizes as the odd and the even parts of $p(s)$ can be computed independently via

$$[p_0, p_2 \dots] \begin{bmatrix} 1 & \dots & 1 \\ -\omega_1^2 & \dots & -\omega_k^2 \\ \omega_1^4 & \dots & \omega_k^4 \\ \dots & \dots & \dots \end{bmatrix} =$$

$$= \mathrm{Re}\,[p(j\omega_1), \dots, p(j\omega_k)]$$

and

$$[p_1, p_3 \dots] \begin{bmatrix} 1 & \dots & 1 \\ -\omega_1^2 & \dots & -\omega_k^2 \\ \omega_1^4 & \dots & \omega_k^4 \\ \dots & \dots & \dots \end{bmatrix} =$$

$$= \mathrm{Im}\,[p(j\omega_1)/\omega_1, \dots, p(j\omega_k)/\omega_k].$$

Recovering Determinant from Zeros (RDZ)

The last but not least method of computing the determinant consists of the following two steps: At first, k zeros $s_1, \dots, s_k$ of the polynomial matrix $P(s)$ are calculated by a direct numerically stable procedure (see below) and, at second, the desired determinant is derived by expanding the product

$$p(s) = c(s - s_1)(s - s_2) \dots (s - s_k).$$

The constant $c = \det P(z) / \prod (z - s_i)$ $(z \neq s_i)$ is to accomplish proper scaling. This approach is know to be very efficient for calculating of the characteristic polynomial[3] of a constant matrix [6]. The classical approach which characterizes zeros as roots of the determinant is here actually reversed.

2-D Polynomial Matrix

In various control problems (robust control of MIMO, 2-D systems control), one faces the challenge of computing the determinant of a matrix that contains polynomials in two (or even more)

[2] In the original version, d_i's and $Q_i(s)$'s are constant scalars and matrices, respectively.

[3] That is $p(s) = \det(sI - A)$.

indeterminates. In principle, this job can also be done by (MFL) or modified (TRI), but the recursive application of (INT) or the combination of (INT) and (RDZ) has been found especially efficient.

Zeros

Zeros of an $n \times n$ polynomial matrix $P(s)$ of degree m are usually defined as complex values s_i, $i = 1, \ldots, k \leq mn$ for which

$$\text{rank } P(s_i) < \text{rank } P(s) = n, \qquad (3)$$

or, equivalently, as the roots of the scalar polynomial $p(s) = \det P(s)$.

Zeros as Roots of Determinant (ZRD)

Naturally, three methods arise that consist in computing the determinant $p(s) = \det P(s)$ via each of the first three algorithms above and then finding the desired zeros as the roots of the scalar polynomial $p(s)$. All the first three methods above were considered and the MATLAB function `roots` was applied to complete the job [6].

Zeros as Generalized Eigenvalues (RDZ)

Another and probably the best way to compute the zeros of a polynomial matrix $P(s)$ is to consider a generalized eigenvalue problem

$$Cv = \lambda Dv \qquad (4)$$

where v is the eigenvector corresponding to an eigenvalue λ, C is a modified block companion matrix corresponding to $P(s)$

$$C = \begin{bmatrix} 0 & I & 0 & \ldots & 0 \\ 0 & 0 & I & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & I \\ -P_0 & -P_1 & \ldots & \ldots & -P_{m-1} \end{bmatrix}$$

and

$$D = \begin{bmatrix} I & 0 & \ldots & 0 \\ 0 & I & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & I & 0 \\ 0 & \ldots & 0 & P_m \end{bmatrix}$$

Now if the leading coefficient matrix P_m is nonsingular, the problem reduces to usual eigenvalue computation as the desired zeros of $P(s)$ simply equal to the eigenvalues of $D^{-1}C$ (which is the block companion matrix[4] corresponding to $P(s)$), that is, $s_i = \lambda_i$ [5]. If this is not the case, two variations are available:

One can always easily find a polynomial matrix $N(s)$ such that $\det N(s) = cs^l$ (where c is a real constant) and that makes the leading coefficient of the product $\bar{P}(s) = P(s)N(s)$ nonsingular. Then the zeros of $\bar{P}(s)$ are found by solving a standard eigenvalue problem and, having removed the l spurious zeros $s_i = 0$ (introduced by $N(s)$), genuine zeros of $P(s)$ result [5].

Alternatively, one can directly compute the generalized eigenvalues (3) using the MATLAB [6] function `eig(C,D)` and then the finite ones among them are exactly the desired zeros of $P(s)$ (that is, the infinites due to the singularity of P_m must be removed).

Rectangular Matrix

If $P(s)$ is rectangular, say wide with full row rank, its zeros can be defined by (3) as well.

The following modification of RDZ was proposed recently by Kwakernaak [4] to compute zeros in such a case[5]: At first, the matrix is split to $P(s) = [D(s)\ N(s)]$, $D(s)$ square, and an observable state-space realization is found to be the "system"

$$D^{-1}(s)N(s) = C(sI - A)^{-1}B.$$

Then it is transformed to controllability staircase form (via `ctrbf` from [7])

$$A = \begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ B_2 \end{bmatrix}$$

with the pair (A_{22}, B_2) completely controllable. The desired zeros of $P(s)$ are finally computed as the eigenvalues of the matrix A_{11} (corresponding to the uncontrollable modes of the "system"). If $P(s)$ is square, $N(s)$ makes no sense, the whole system (now $D(s)y(s) = 0$) is uncontrollable and the method reduces to RDZ above (with $C = A_{11}$).

Example

As an implementation of the above mentionned techniques, the MATLAB function `pdet` from the *Polynomial Control Toolbox* allows the user to specify the algorithm to compute the determinant

[4] Yet another way of computing $p(s) = \det P(s)$ may consist in reducing the block companion matrix above to the standard companion matrix form in which the coefficients of $p(s)$ are directly readable from its last row.

[5] Here $P(s)$ is assumed to be row reduced with a row leading coefficient matrix $P_L = [D_L\ 0]$ where D_L square and nonsingular. So a general matrix must be first transformed into such a form. Alternatively, a singular system realization could be constructed directly and the desired zeros could be computed as (the finite) generalized eigenvalues of its uncontrollable part.

$$P(s) = \begin{bmatrix} -24-3s-4s^2 & 0 & 3-6s-s^2 & 16+8s+2s^2 & -10-s+s^2-6s^3 \\ -4+4s & 15-25s-11s^2 & 11 & 5-3s-4s^2 & -10-11s+3s^2 \\ 15+7s-6s^2+8s^3 & -3-8s & 0 & -5-7s-3s^2+5s^3 & -11+3s+11s^2 \\ 1+7s-s^2 & 20+9s-19s^2 & -19s+10s^2 & 3 & 6+8s+17s^2 \\ -4-2s & 6+9s-12s^2-6s^3 & 0 & -25 & 0 \end{bmatrix}$$

of a polynomial matrix. For each of these methods, zeroing (suppression of the "nearly" zero components of the resulting polynomial determinant) is the critical step, as it is illustrated in the following example. Let us consider a 5×5 polynomial matrix $P(s)$ of degree 3 given above.

Using the MFL method, the computed determinant $p_{\mathrm{MFL}}(s) = \det P(s)$ reads

$$\begin{aligned} &p_{\mathrm{MFL}}(s) = \\ &2050825 + 5308730s - 8364573s^2 - 4460254s^3 + \\ &6605708s^4 - 3830859s^5 - 1077006s^6 + 2142092s^7 - \\ &1623284s^8 - 561001s^9 + 209883s^{10} - 68838s^{11} - \\ &21942s^{12} + 11520s^{13} = \\ &p_0 + p_1 s + p_2 s^2 + \ldots + p_{12}s^{12} + p_{13}s^{13}. \end{aligned}$$

Its coefficients p_i will serve us for further reference.

If now the determinant is built from the generalized eigenvalues of a block companion matrix (RDZ), the program returns $p_{\mathrm{RDZ}}(s) = \det P(s)$ as

$$\begin{aligned} &p_{\mathrm{RDZ}}(s) = \\ &p_0 + (p_1 - 1.86 \cdot 10^{-9})s + (p_2 + 4.66 \cdot 10^{-9})s^2 + \\ &\ldots + (p_{12} - 1.82 \cdot 10^{-10})s^{12} + \\ &(p_{13} + 4.73 \cdot 10^{-11})s^{13} - 2.09 \cdot 10^{-13}s^{14} \end{aligned}$$

which leading coefficient can be cancelled by first cutting the almost infinite eigenvalues, that is the eigenvalues which norm exceed a fixed upper bound (here $\|P(s)\| \cdot 10^4$). Sometimes it is also necessary to suppress afterwards the almost zero components of the resulting $\det P(s)$, where the lower bound is a small enough constant.

Such a zeroing is even more necessary in the case of interpolation (INT), where the computed $p_{\mathrm{INT}}(s) = \det P(s)$ is a polynomial which degree is higher than the expected one :

$$\begin{aligned} &p_{\mathrm{INT}}(s) = \\ &(p_0 + 9.31 \cdot 10^{-10}) + (p_1 + 4.67 \cdot 10^{-9})s + \ldots + \\ &(p_{12} + 1.02 \cdot 10^{-3})s^{12} + (p_13 + 5.26 \cdot 10^{-4})s^{13} + \\ &5.14 \cdot 10^{-4}s^{14} + 2.26 \cdot 10^{-4}s^{15}. \end{aligned}$$

Leaving out the almost zero components of $\det P(s)$, according to a fixed lower bound of $\|P(s)\| \cdot 10^{-6}$ (value which gave satisfying results in some other cases, here $4.42 \cdot 10^{-5}$) doesn't modify the result. In such a case, the user must find what could be a correct value for zeroing (here a lower bound of 10^{-4} leads to the correct degree of 13 for the determinant). This value is then given to the function **pdet** as a supplementary optional parameter.

Finally, the triangularization procedure (TRI), with considerable more efforts (polynomial elementary operations are time consuming (in MATLAB) and numerically unstable operations), returns with or without zeroing

$$\begin{aligned} &p_{\mathrm{TRI}}(s) = \\ &(p_0 - 2.33 \cdot 10^{-10}) + (p_1 - 3.73 \cdot 10^{-9})s + \ldots + \\ &(p_{12} + 6.18 \cdot 10^{-11})s^{12} + (p_{13} - 3.64 \cdot 10^{-11})s^{13}. \end{aligned}$$

References

[1] F.R. Gantmacher. *The Theory of Matrices.* Chelsea Publishing Company, New York, 1960.

[2] G.M. Golub and C. van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore, Maryland, 1983.

[3] Huibert Kwakernaak. MATLAB macros for polynomial $\mathcal{H}_\infty$-optimal regulation. Memorandum No. 881, Department of Applied Mathematics, University of Twente, The Netherlands, September 1990.

[4] Huibert Kwakernaak. State space algorithms for polynomial matrix computations. Memorandum No. 1168, Department of Applied Mathematics, University of Twente, The Netherlands, October 1993.

[5] Huibert Kwakernaak and Michael Šebek. Polynomial J-spectral factorization. *IEEE Transactions on Automatic Control*, AC-39(2):315–328, February 1994.

[6] The MathWorks, Inc. *PRO-MATLAB™ User's Guide*, 1989.

[7] The MathWorks, Inc. *Control Systems TOOLBOX For Use with MATLAB™: User's Guide*, 1992.

[8] Michael Šebek, Soňa Pejchová, and Ferdinand Kraffer. Testing algorithms for linear polynomial equations. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 2518–2519, Orlando,Fl, December 1994. IEEE Control Systems Society.

[9] Lofti A. Zadeh and Charles A. Desoer. *Linear Systems Theory: The State Space Approach.* McGraw-Hill Book Company, Inc., New York, 1963.