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 Faddeev-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
\[ p(s) = \text{det} P(s), \tag{1} \]
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 Faddeev-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 \( [I - A] \) (that is, \( \text{det}(I - A) \)) 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 \( d_i \) and polynomial matrices \( Q_i(s) \) in the following manner:

\[
\begin{align*}
Q_0 &= I_n, \\
d_0 &= -\text{tr} (Q_0 P) \\
Q_1 &= Q_0 P + d_1 I_n \\
d_1 &= -\frac{1}{2} \text{tr} (Q_1 P) \\
Q_2 &= Q_1 P + d_2 I_n \\
d_2 &= -\frac{1}{4} \text{tr} (Q_2 P) \\
&\vdots
\end{align*}
\]

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 be transformed to a triangular form \( P_n(s) \) by a series of elementary column operations or, in other words, via postmultiplication by a unimodular matrix \( U(s) \) with \( \text{det} U(s) = 1 \)

\[
P_n(s) = P(s)U(s).
\]

Then the desired determinant \( p(s) = \text{det} P_n(s) \) is found as the product of the diagonal elements of \( P_n(s) \)

\[
p(s) = \prod_{i=1}^{n} [P_n(s)]_{ii}.
\]

Interpolation (INT)

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

\[
\begin{bmatrix}
1 & \cdots & 1 \\
\vdots & \ddots & \vdots \\
S_0 & \cdots & S_k
\end{bmatrix}
\begin{bmatrix}
p_0 \\
p_k
\end{bmatrix}
= \begin{bmatrix}
p(s_0) \\
p(s_k)
\end{bmatrix}
\]

and

\[
\begin{bmatrix}
\bar{p}_1 \\
\bar{p}_3 \\
\vdots
\end{bmatrix}
= \text{Re} \begin{bmatrix}
p(j\omega_1) \\
p(j\omega_3) \\
\vdots
\end{bmatrix}
\]

and

\[
\begin{bmatrix}
\bar{p}_2 \\
\bar{p}_4 \\
\vdots
\end{bmatrix}
= \text{Im} \begin{bmatrix}
p(j\omega_1)/\omega_1 \\
p(j\omega_3)/\omega_3 \\
\vdots
\end{bmatrix}
\]

Recovering Determinant from Zeros (RDZ)

The last (but not least) method of computing the determinant consists of the following steps: At first, \( k \) zeros \( s_1, \ldots, s_k \) 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

\[
p(s) = c(s-s_1)(s-s_2)\ldots(s-s_k).
\]

The constant \( c = \text{det} P(z)/\prod (z-s_i) \) is to accomplish proper scaling. This approach is know to be very efficient for calculating the characteristic polynomial of a constant matrix [6]. The classical approach which characterizes roots 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)

\[\text{In the original version, } d_i \text{'s and } Q_i(s) \text{'s are constant scalars and matrices, respectively.}\]

\[\text{That is } p(s) = \text{det}(sI - A).\]
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

\[
\text{rank } P(s_i) < \text{rank } P(s) = n, \quad (3)
\]

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

\[
Cv = \lambda Dv, \quad (4)
\]

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

\[
C = \begin{bmatrix}
0 & I & 0 & \cdots & 0 \\
0 & 0 & I & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-\lambda P_0 & -\lambda P_1 & \cdots & -\lambda P_{m-1} \\
\end{bmatrix}
\]

and

\[
D = \begin{bmatrix}
I & 0 & \cdots & 0 \\
0 & I & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & P_m \\
\end{bmatrix}
\]

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 \( \det N(s) = cs^t \) (where \( c \) is a real constant) and that makes the leading coefficient of the product \( P(s) = P(s)N(s) \) nonsingular. Then the zeros of \( P(s) \) 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 function eig(C,D) and then the finite ones among them are exactly the desired zeros of \( P(s) \) (that is, the infinite ones 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\(^4\): 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])

\[
A = \begin{bmatrix}
A_{11} & 0 \\
A_{21} & A_{22}
\end{bmatrix}, \quad B = \begin{bmatrix}
0 \\
B_2
\end{bmatrix}
\]

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 mentioned techniques, the MATLAB function pdet from the Polynomial Control Toolbox allows the user to specify the algorithm to compute the determinant

\[^4\text{Here } P(s) \text{ is assumed to be row reduced with a row leading coefficient matrix } P_l = [D_L, 0] \text{ where } D_L \text{ square and nonsingular. So a general matrix must be first transformed into such a form. Alternatively, a singular system realization could be constructed directly and the desired zeros could be computed as (the finite) generalized eigenvalues of its uncontrollable part.}
\]

475
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 $5 \times 5$ polynomial matrix $P(s)$ of degree 3 given above.

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

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

The lower bound of $10^{-4}$ leads to the correct degree of 13 for the determinant. This value is then given to the function $p_{MFL}$ 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

$$
p_{TRI}(s) = (p_0 - 2.33 \cdot 10^{-10}) + (p_1 - 3.73 \cdot 10^{-2})s + \ldots + (p_{13} + 6.18 \cdot 10^{-11})s^{13} + (p_{13} + 3.54 \cdot 10^{-11})s^{13}.
$$

References


