KYBERNETIKA — VOLUME 30 (1994), NUMBER 6, PAGES 745 - 753

POLYNOMIAL OPERATIONS: NUMERICAL PERFORMANCE IN MATRIX DIOPHANTINE EQUATION

FERDINAND KRAFFER, SOŇA PEJCHOVÁ AND MICHAEL ŠEBEK

In the absence of any theoretical work on polynomial operations, the paper suggests possible evaluation of particular numerical algorithms to solve linear equation in polynomial matrices and draws some preliminary conclusions with respect to the use of polynomial operations.

1. INTRODUCTION

This paper is motivated by misproportion between theoretical developments in polynomial control and their computer implementation. Evolved from early attempts to use polynomial equations in discrete-time linear systems design, currently, polynomial control also covers continuous-time, infinite-dimensional, time-varying and nonlinear systems. However, popularity of polynomial control is sometimes undermined by pointing at its numerical weakness. No doubt, a lot of performance would be improved had there been numerical mathematics for polynomial operations.

The ultimate goal of this paper is one step forward in exploring polynomial operations impact on control system design. It is addressed by choosing a typical design equation and comparing various numerical algorithms to solve this equation. Due to the persistent lack of numerical mathematics, we are almost unable to compare the algorithms *theoretically*, thus we pursue the way of *experiment*. Quantities compared are solution error and computation time.

Usually, a number of polynomial equations are to be solved in the course of every design procedure. These may have different nature, however, in every design procedure, the most frequently encountered equation[1] reads

$$ax + by = c, \tag{1}$$

with a, b, c given and x, y unknown polynomials in one indeterminate. In multivariable control system design, (1) turns into a matrix version, such as

$$AX + BY = C, \tag{2}$$

[1] It is often called Diophantine for its similarity to the equation over integers.

which is the subject of our paper. Here A, B, C are given while X, Y are unknown polynomial matrices at compatible sizes. Every design procedure for multivariable systems usually includes, a number of equations (2) and alike. Hence, an effective tool to crack (2) is a must which certifies our preference.

In the last two decades, various numerical algorithms for (2) have appeared. They remain uncompared due to weak numerical background for polynomial equations. Therefore, we have started a project of *experimental* comparison. In the project, a number of numerical examples are computed using our software package [6] which is made up of MATLAB[1] and C programs for basic polynomial operations. As far as the polynomial matrix representation is concerned, we identify ourselves with [4] and recall that no standard is available in MATLAB© [5].

In our experimental work we have exercised various algorithms based on polynomial operations and/or constant matrix operations. To investigate polynomial operations we chose

EOM	*elementary operations method*
SSM	*state-space realization method*
PIM	*polynomial interpolations method.*

These are ordered by amount of polynomial operations they use from exclusively polynomial to constant matrix only. Based on diophantine equation theory, we chose a standard to test the methods on. This benchmark triple A, B, C constitutes ill-conditioned equation.

The paper is structured followingly: First, diophantine equation theory is briefly surveyed. Second, listed methods are explained with particular emphasis on polynomial operations involved. PIM is not based on polynomial operations, therefore it is covered only by brief list of consecutive steps. Third, the benchmark is discussed and its simple realization is set along with experiment conditions. Last, results are described and conclusions drawn.

2. BRIEF REVIEW OF THEORY

Equation (2) has a solution pair X, Y over polynomials if and only if every common left divisor of A, B is a left divisor of C.

As a consequence, if A, B are relatively left prime, then (2) has a solution pair for any C.

Further, let S_1, R_1 be relatively right prime polynomial matrices, such that

$$AR_1 + BS_1 = 0.$$

Then if X^0, Y^0 is a particular solution pair of (2), any solution pair X, Y is of the form

$$\begin{aligned} X &= X^0 + R_1 T \\ Y &= Y^0 + S_1 T3, \end{aligned} \tag{3}$$

where T is a polynomial matrix parameter.

[1] MATLAB is a trademark of The MathWorks, Inc.

3. THE ELEMENTARY OPERATIONS METHOD

This algorithm originates from [3]. Its ideas can be traced to the previous section.

1. We postmultiply the composite matrix $[\,A\;B\,]$ by unimodular matrix U to bring it into a quasi-triangular form, i.e.,

$$[\;A \quad B\;]\,U = [\;G_1 \quad 0\;], \tag{4}$$

where G_1 is a triangular matrix. This represents elementary column operations.

2. We partition U according to numbers of columns in A, B and G_1 calling the partitions

$$U = \begin{bmatrix} P_1 & R_1 \\ V_1 & S_1 \end{bmatrix}.$$

It has been proved that P_1, V_1 and R_1, S_1 are couples of right coprime polynomial matrices. To appreciate this fact, note that (4) is equivalent to

$$\begin{aligned} AP_1 + BV_1 &= G_1 \\ AR_1 + BS_1 &= 0. \end{aligned} \tag{5}$$

Clearly, G_1 is a greatest left divisor of A, B.

3. We extract G_1 from C by $C = G_1\,C_1$ backsolve, calling the result C_1.

Then (5.a) postmultiplied by C_1 recovers (2) with a particular solution pair

$$\begin{aligned} X^0 &= P_1C_1 \\ Y^0 &= V_1C_1. \end{aligned}$$

4. We compute this particular solution pair.

Note, that G_1 extracted from (5.b) recovers (3) which certifies the U-partition notation.

4. THE STATE-SPACE REALIZATION METHOD

This algorithm is adopted from [2]. Here, A is assumed nonsingular which implies existence of observable state-space realizations to strictly proper parts of $A^{-1}B$, $A^{-1}C$ such that these realizations[1] differ only in the input matrix G.

$$\begin{aligned} \mathcal{B} &\equiv \{F, G_B, H\} \equiv A^{-1}V_B \\ \mathcal{C} &\equiv \{F, G_C, H\} \equiv A^{-1}V_C, \end{aligned}$$

[1] $\begin{aligned} x(t+1) &= Fx(t) + Gu(t) \\ y(t) &= Hx(t) \end{aligned}$

where $A^{-1}V_B$ and $A^{-1}V_C$ are strictly proper parts of $A^{-1}B$ and $A^{-1}C$, respectively. This fact is utilized as follows: Premultiplied by A^{-1}, the diophantine can be teared apart into polynomial and strictly proper parts only in two consecutive stages. Indeed, one cannot split $A^{-1}BY$ without knowledge of Y which reveals the first stage be about computing Y.

1. We start the first stage by mod$A$ factorization on B and C

$$B = AU_B + V_B$$
$$C = AU_C + V_C,$$

to find polynomial pairs U_B, V_B and U_C, V_C such that $A^{-1}V_B$ and $A^{-1}V_C$ are strictly proper.

These factorizations can be inserted into A^{-1}-premultiplied diophantine to get

$$X + (U_B + A^{-1}V_B)Y = U_C + A^{-1}V_C, \tag{6}$$

where infinite sequences can be teared apart from polynomials. Note, that the former constitute equation in unknown polynomial matrix Y, i.e.,

$$A^{-1}V_BY = A^{-1}V_C.$$

It has been proved that this equation can be given a state-space interpretation which leads to

$$\begin{aligned} \mathcal{R}_n \left[\begin{array}{cccc} Y_0' & Y_1' & \dots & Y_n' \end{array} \right]' &= G_C \\ \mathcal{R}_n = \left[\begin{array}{cccc} G_B & FG_B & \dots & F^nG_B \end{array} \right], & \end{aligned} \tag{7}$$

where $Y_0, Y_1, \dots, Y_n$ are coefficient matrices[2] of Y. $\mathcal{R}_n$ has $n+1$ blocks, where n is the degree of Y and, hence, is to be determined.

2. We finish the first stage by solving (7) which is equation in constant matrices. Note that (7) is solvable if and only if $\mathrm{Sp}[G_C] \subset \mathrm{Sp}[\mathcal{R}_n]$. Left-to-right deletion of linearly dependent columns in $\mathcal{R}_n$ uniquely defines not only n but also Y. Of course, one may adopt other strategies to compute Y from (7). Clearly, our strategy solves for minimum column degrees of Y.

In the second stage, given the knowledge of Y, the diophantine split is finished. This enables to complete the particular solution pair with X.

3. We start the second stage by mod$A$ factorization

$$V_BY = AU + V$$

to find polynomial U, V such that $A^{-1}V$ is strictly proper.

This factorization can be inserted into (6), and polynomials can be equated apart from causal sequences. Note, that the former constitute equation in unknown X, i.e.,

$$X + U_BY + U = U_C. \tag{8}$$

4. We finish the second stage by computing X which follows from (8) after simple polynomial operations.

[2] Polynomial matrix is treated as matrix polynomial.

5. THE POLYNOMIAL INTERPOLATION METHOD

For full detail, the reader is referred to [1]. Here, A is considered to be a square row reduced $(l \times l)$ matrix. The solution degree r depends on the reachability index ν of the system represented by $A^{-1}B$, where A, B are left coprime matrices with A row reduced with $d_i := \deg_{ri}[A]$. Solution of rth degree exists if r satisfies $\deg_{ri}[C] \leq d_i + r$ and $r \geq \nu - 1$ for $i = 1, 2, \ldots, l$. Hence, for a sufficiently high r, the equation is solved as follows:

1. Compute the number $k = \Sigma d_i + l(r+1)$ of interpolation couples (z_j, α_j) for $j = 1, 2, \ldots, k$, where z_j are distinct scalars and α_j are nonzero constant l-vectors.

2. Choose interpolation couples. To ensure X is of the rth degree, assume X_r, the matrix coefficient at z^r of X, to be identity.

3. For all interpolation couples, consider (1) in z_j-s and multiply it from the left by $\alpha(j)$. Solve linear equation in constant matrices.

4. Recover X, Y from the partitioned matrix $\left[\begin{array}{cccccc} X_0' & Y_0' & X_1' & Y_1' & \ldots & X_r' \quad Y_r' \end{array} \right]'$, where X_i, Y_i are matrix coefficients at z^i of matrix polynomials X, Y, respectively.

6. EXPERIMENTS

From the theory follows that A, B, C triple constitutes ill-conditioned equation anytime a left common divisor of A, B is "close" to violate the necessary and sufficient condition on solution existence. Recall, the condition demands the divisor be a left divisor of C. This fact motivates our choice of benchmark. For simplicity, C is chosen identity and property of "being close" is simulated by parameter over reals the zero value of which brings a left common divisor of A, B into the scene.

Well over thousand experimental examples have been computed thus far by the use of the software packages [6] and [7]. To illustrate the same, two typical representatives have been chosen in the paper.

Both tested examples posses the same

$$A = \begin{bmatrix} 3+3s & -1+s^2 \\ 0 & -1+s^2 \end{bmatrix}$$

and

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

while the matrix B is different:

$$B_{\text{first}} = \begin{bmatrix} (1+e)s + (1+e)^2 s^2 & -2 + 2(1+e)^2 s^2 \\ -1 + (1+e)^2 s^2 & -2 + 2(1+e)^2 s^2 \end{bmatrix}$$

$$B_{\text{second}} = \begin{bmatrix} e+(1+e)^2s+(1+e)^2s^2 & -2+2(1+e)^2s^2 \\ (e-1)+e(1+e)s+(1+e)^2s^2 & -2+2(1+e)^2s^2 \end{bmatrix}$$

Clearly, for $e = 0$ there exist nontrivial greatest common left divisor (the same for both examples):

$$gcld(A, B_{e=0}) = \begin{bmatrix} 3+3s & 0 \\ 0 & -1+s^2 \end{bmatrix}$$

which makes the equation unsolvable. Hence, the equation becomes ill-conditioned when
$e \to 0$. In the experiments, we set $e = 2^{-x}$ and let it to approach zero by substituting $x = 0, 1, 2, \ldots$.
As we have noted earlier in the paper, there exist an infinite number of solutions to the Diophantine equation, which are parameterized by (3). Unfortunately, each of the investigated methods yields by its nature different type of particular solution. Using EOM, the degrees of solution pair are a priori unknown in contrast to PIM which solves for conveniently chosen values. As implemented, SSM yields a solution with minimum column degrees in Y. Of course, by (3), particular solutions can be brought to the same type, however, this creates additional operations which should not be included. Clearly, different control problems require different particular solution choice (e. g. minimum row degrees for LQ optimization in contrast to minimum column degrees for deadbeat) and fair comparison is possible only for a specific problem in hand.
To overcome this difficulty we simply substitute every computed solution X^0, Y^0 to the original equation and enumerate the norm $\|AX^0 + BY^0 - C\|$.

Thus, the first quantity to evaluate the experiment is

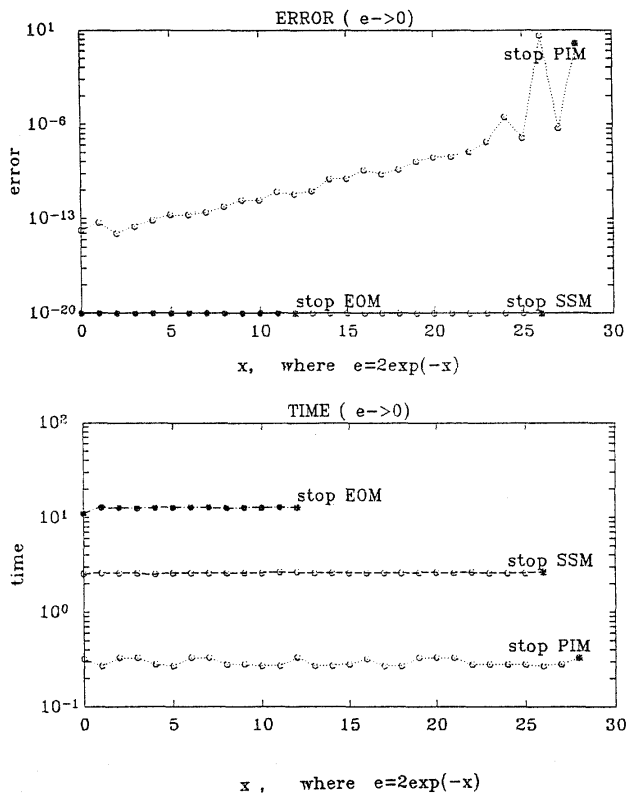
error – the MATLAB norm(Z, inf), where $Z = AX^0 + BY^0 - I$.

Quite naturally, the second quantity is

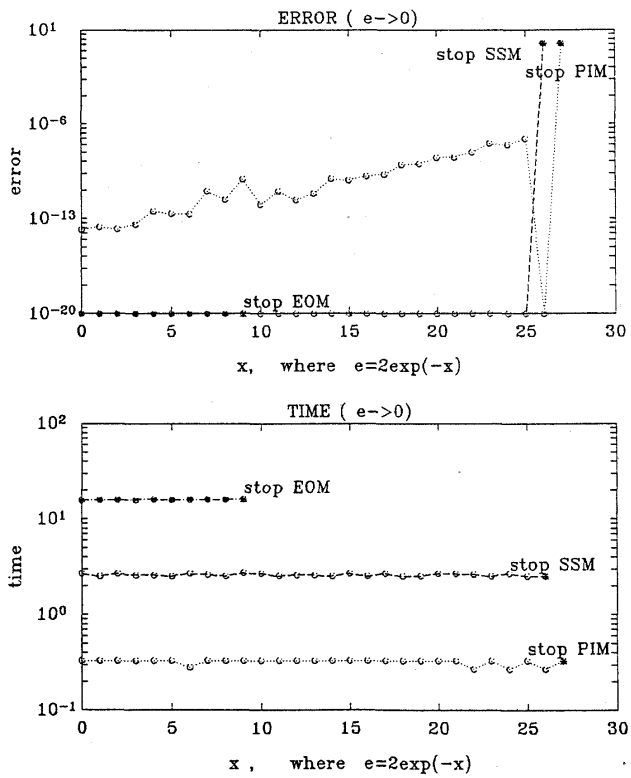
time – computational time of one run in MATLAB on a 486 PC under MS DOS.

Both quantities are plotted as functions of x for each method considered. As a result, each example is described by a couple of graphs, where the "stop" prefix is used to denote the point of numerical breakdown.

First example



Second example



7. PRELIMINARY CONCLUSIONS

The whole experimental work performed thus far must be taken only as a first step on the way to fairly evaluate and modify all existing procedures. Yet we can draw some preliminary conclusions:

EOM is of a genuine polynomial nature and the only one to provide a general solution in one shot. In addition, no initial settings are required. Last but not least, built directly on fundamental theorems, it is of pedagogical value. On the other hand, the method is rather computationally involved and usually the slowest one within the competition. Needless to say, this drawback is partly due to the nature of MATLAB, which is really not well suited for polynomial manipulations. In addition, after each polynomial operation, it demands to determine the degree of the product. This critical step must be accomplished by a specific program module.

SSM is based on both matrix and polynomial operations. It is particularly flexible with regard to particular solution choice. Particular solution Y is picked from a prespecified family, here it is chosen such that its column degrees are minimal.

PIM is based exclusively on constant matrix operations, which are well-understood and already implemented in MATLAB. The method provides one particular solution from a prespecified class of solutions such that the solution degree is less than a chosen parameter.

Provided the last two methods are performed also for homogeneous equation, (3) may be used to construct a parameterization of all solutions.

(Received November 30, 1993.)

REFERENCES

[1] P. J. Antsaklis and Z. Gao: Polynomial and rational matrix interpolation: Theory and control application. Control Systems Technical Report No. 71, University of Notre Dame, Indiana, U. S. A. 1992.

[2] F. Kraffer: State-space method to solve polynomial matrix diophantine equation. Young Scientist Contest 1993, Institute of Information Theory and Automation, Prague 1993.

[3] V. Kučera: Discrete Linear Control. Academia, Prague 1979.

[4] H. Kwakernaak: MATLAB Macros for Polynomial $\mathcal{H}_\infty$ Control System Optimization. Memorandum No. 881, University of Twente, The Netherlands 1990.

[5] MATLAB^{TM} for MS-DOS Personal Computers, User's Guide, The Math. Works Inc., South Natick, MA.

[6] S. Pejchová: Software Package for Polynomial Operations I. Research Report No. 1765, Institute of Information Theory and Automation, Prague 1993.

[7] M. Šebek: An algorithm for spectral factorization of polynomial matrices with any signature. Memorandum No. 912, University of Twente, The Netherlands 1990.

Ing. Ferdinand Kraffer, Ing. Soňa Pejchová, Ing. Michael Šebek, CSc., Ústav teorie informace a automatizace AV ČR (Institute of Information Theory and Automation – Academy of Sciences of the Czech Republic), Pod vodárenskou věží 4, 18208 Praha 8. Czech Republic.