POLYNOMIAL FACTORIZATION OF LARGE MIMO SYSTEMS USING SLICOT LIBRARY AND POLYX TOOLBOX

P.-O. Malaterre

Cemagref, 361 rue J.-F. Breton, BP 5095, 34033 Montpellier Cedex 1, France
e-mail: pom@montpellier.cemagref.fr

Keywords: polynomial factorization, Youla parametrization, ℓ_1 control, high-order systems, numerical tools

Abstract

The aim of this note is to present an application of the SLICOT Fortran Library and PolyX MatLab© ToolBox for polynomial factorization of large MIMO systems. The corresponding polynomial matrix fractions have been calculated for the H, U and V matrix transfer functions coming from the Youla parametrization of a high order system (an irrigation canal). The H (resp. U and V) matrix transfer function has 5 inputs, 10 outputs and 130 states (resp. 5, 10, 65 for U and 5, 5, 65 for V). The results proved to be very good in terms of singular values matching between the original and the factorized system (polynomial matrix fraction). Scaling has been introduced, reducing considerably the size of the polynomial coefficients, without losing precision for the singular values.

1 Introduction

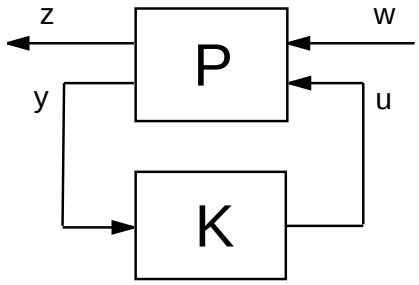

Figure 1: Standard framework

In many control design problems, the objective is to find a stabilizing linear time-invariant (LTI) discrete-time controller K which, in addition to stabilizing the closed loop system, minimizes some norm(s) of the transfer matrix $\Phi : w \to z$ (Figure 1). If the considered norm is the $\mathcal{H}_2$ or $\mathcal{H}_\infty$ norm of the transfer matrix Φ in the frequency domain, then a state-space solution exists for the controller. But in the case of the ℓ_1 norm of the impulse response, no state-space solution exists for the moment, and the problem must be solved by a linear programming approach [1]. This can be stated as solving:

$$\gamma^{opt} = \inf_{\text{K stabilizing}} \|F_l(P,K)\|_1 \tag{1}$$

where P represents the LTI discrete-time generalized plant (Figure 1), K the LTI discrete-time controller, $F_l(P,K) = \Phi$ the lower linear fractional transformation of P by K. We assume the dimensions of w, z, u, and y are n_w, n_z, n_u, and n_y respectively. It can be shown [1], that this problem can be formulated as that of finding:

$$\gamma^{opt} = \inf_{Q \in \ell_1^{n_u \times n_y}} \|H - U * Q * V\|_1 \tag{2}$$

where $*$ denotes convolution, $H \in \ell_1^{n_z \times n_w}$, $U \in \ell_1^{n_z \times n_u}$, and $V \in \ell_1^{n_y \times n_w}$ are fixed and depend on the problem data: P, n_w, n_z, n_u, and n_y.

In order to transform this problem into a Finite Dimensional Linear Programing problem (FD LP) one solution is to compute Finite Impulse Response (FIR) approximations of H, U and V, and to look for the optimal Q with a finite support of length N. It is proved [2], that this approach with one additional constraint provides converging upper and lower bounds for the original problem, when $N \to \infty$.

2 Incentive for polynomial factorization

The numerical approach used to solve the previous problem can lead to a large number of linear constraints in the LP problem if the FIR approximations of H, U and V are very long. Another approach can be to compute a right (resp. left) polynomial factorization of U (resp. V): $U = N_{ur}.D_{ur}^{-1}$ (resp. $V = D_{vl}^{-1}.N_{vl}$). A similar work can be done with H but will not be detailed in this note. The original problem is then transformed into:

$$\gamma^{opt} = \inf_{Q \in \ell_1^{n_u \times n_y}} \|H - N_{ur} * D_{ur}^{-1} * Q * D_{vl}^{-1} * N_{vl}\|_1 \tag{3}$$

$$= \inf_{\widetilde{Q} \in \ell_1^{n_u \times n_y}} \|H - N_{ur} * \widetilde{Q} * N_{vl}\|_1 \tag{4}$$

with $\widetilde{Q} = D_{ur}^{-1} * Q * D_{vl}^{-1}$

Proof: U and V are stable $\Rightarrow$ the λ-Transform $\widehat{D}_{ur}$ and $\widehat{D}_{vl}$

	without scaling	with scaling
from state-space	$2.8160\ 10^{-10}$	$2.8160\ 10^{-10}$
left factorization	$9.5064\ 10^{-6}$	$9.1206\ 10^{-6}$
right factorization	$9.5899\ 10^{-9}$	$9.5920\ 10^{-9}$

Table 1: TB03AD: Maximum difference of singular values to the ones obtained with the MatLab "Sigma" function, for U

	without scaling	with scaling	order
N_{ul}	$1.7894\ 10^{13}$	0.2767	7
D_{ul}	$6.4676\ 10^{13}$	1.0	7
N_{ur}	$5.1159\ 10^{11}$	0.6223	13
D_{ur}	$8.2215\ 10^{11}$	1.0	13

Table 2: TB03AD: Maximum absolute value of coefficients of the obtained polynomial matrices, and order of the polynomials, for U

have no unstable zeros $\Rightarrow \widehat{D}_{ur}^{-1}$ and $\widehat{D}_{vl}^{-1} \in \ell_1$ (Weiner Theorem) and therefore $Q \in \ell_1 \Leftrightarrow \widetilde{Q} \in \ell_1$, which means that searching over $\widetilde{Q}$ is equivalent to searching over Q.

3 Computation of the polynomial factorizations

At least two softwares or libraries can compute such polynomial factorizations. The open FORTRAN library SLICOT [3] proposes the TB03AD routine. The PolyX MatLab Toolbox [4] proposes the routines ss2lmf and ss2rmf.

In order to use the TB03AD SLICOT routine from MatLab, a Fortran program (MTB03AD.f) has been written. This source code has been compiled using the MicroSoft PowerStation 4.0 Fortran compiler, and linked to the SLICOT, LAPACK and BLAS libraries [3]. This compiled program (MTB03AD.exe) is then called from a TB03AD.m MatLab function. More recently, we tested another approach, consisting in generating a .mex file using the Digital Fortran Compiler 6.0. In terms of computational speed this second approach is 3 to 5 time faster.

In PolyX the routines ss2lmf and ss2rmf are called directly from MatLab.

In the following sections the two packages are tested on the same examples.

4 Results of the polynomial factorization of U

The left and right polynomial factorization $U = D_{ul}^{-1}.N_{ul} = N_{ur}.D_{ur}^{-1}$ have been computed using the SLICOT TB03AD subroutine [3]. The U system has 5 inputs, 10 outputs and 65 states. The order of the N_{ul} (resp. D_{ul}, N_{ur}, D_{ur}) matrix polynomial is 7 (resp. 7, 13, 13). The singular values of the original system are compared, over a large range of frequencies, to the ones of the left and right polynomial factorizations (Figure 2, Table 1). The matching is very good (the maximum deviation is $9.5064\ 10^{-6}$ for the left factorization). Nevertheless, one drawback of this factorization is that the coefficients of the polynomial matrices are very large in our example (Table 2). But this can be easily solved by scaling them (for example by dividing all coefficients by an appropriate scalar, which does not change the condition number of the matrices). The matching of the singular values is not affected (Table 1), and the maximum value of the coefficients of the polynomial matrices is greatly reduced (Table 2). This is important if these coefficients are then used in a Linear Programming software. It would be probably a good idea to add this scaling option to both packages.

	without scaling	with scaling
from state-space	$2.8160\ 10^{-10}$	$2.8160\ 10^{-10}$
left factorization	$2.7644\ 10^{-6}$	$2.9720\ 10^{-6}$
right factorization	$9.8655\ 10^{-9}$	$9.7450\ 10^{-9}$

Table 3: PolyX: Maximum difference of singular values to the ones obtained with the MatLab "Sigma" function, for U

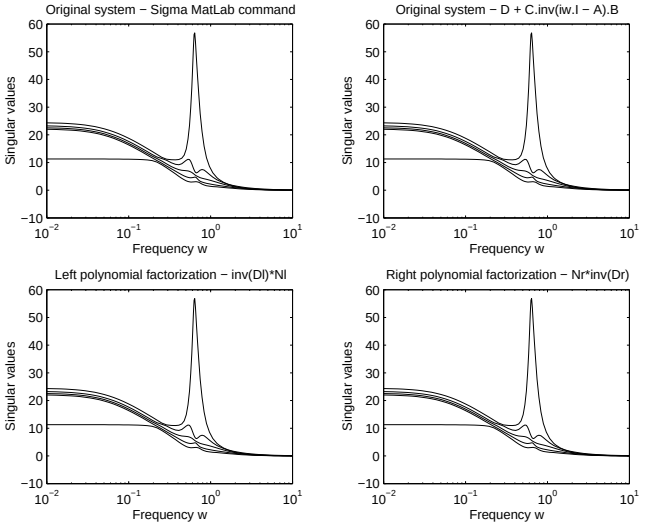

Figure 2: Comparison of the singular values for U

5 Comparison with the PolyX ToolBox

The results obtained with the SLICOT TB03AD routine are compared to the ones obtained with the MatLab Polynomial ToolBox PolyX (Tables 3, 4, 5, 6). These results have been kindly provided by PolyX, inc [4]. The results are very similar, as far as the singular values and the maximum polynomial coefficients are concerned. Except in one case (right factorization for U), PolyX gives slightly better results. Also, with both packages the computation time is about 1 to 2 seconds on a PC based on a Pentium II, 350 Mhz (and about 0.3 seconds on a Pentium III, 667 Mhz with the exe file method and about 0.06 seconds with the mex file method).

	without scaling	with scaling	order
N_{ul}	1.7575 10^{13}	0.2680	7
D_{ul}	6.5584 10^{13}	1.0	7
N_{ur}	3.2054 10^{11}	0.4997	13
D_{ur}	6.4140 10^{11}	1.0	13

Table 4: PolyX: Maximum absolute value of coefficients of the obtained polynomial matrices, for U

	without scaling	with scaling
from state-space	8.7041 10^{-14}	8.7041 10^{-14}
left factorization	2.9665 10^{-13}	2.9488 10^{-13}
right factorization	6.8212 10^{-13}	6.8301 10^{-13}

Table 5: PolyX: Maximum difference of singular values to the ones obtained with the MatLab "Sigma" function, for V

6 Conclusion

The SLICOT TB03AD subroutine proved to be very useful and numerically efficient, even on large systems. The results are close to the ones obtained using the PolyX MatLab ToolBox, although slightly not as good on 3 cases out of 4. But the errors are very small and far bellow the required precision.

The numerical precision obtained with the SLICOT subroutines proved to be very similar using .mat files and an external .exe program compiled with the PowerStation 4.0 Fortran Compiler compared to using a .mex file compiled with the Digital Fortran Compiler 6.0. But this latest option proved to be faster (about 3 to 5 times). Nevertheless this computational time is very small and if this calculation has to be made from MatLab, the PolyX is certainly much more convenient. The SLICOT package has the advantage to be freely available, and is useful in case the calculation has to be made from a Fortran Program.

The coefficients of the obtained polynomials proved to be very large. This is an important problem since these coefficients are then planned to be used into a Linear Programming software. But, this could be easily solved using a scaling factor. Neither the precision of the polynomial factorization nor the condition number of the matrices have been altered. It would be a good idea to add this option to both packages.

	without scaling	with scaling	order
N_{vl}	2.1523 10^{8}	0.0263	13
D_{vl}	8.1682 10^{9}	1.0	13
N_{vr}	5.6355 10^{6}	0.0264	13
D_{vr}	2.1380 10^{8}	1.0	13

Table 6: PolyX: Maximum absolute value of coefficients of the obtained polynomial matrices, for V

These polynomial factorizations are used and illustrated in a companion paper [5].

Acknowledgment

The material presented in this note was studied during my stay at ISU as a Visiting Scientist. I would like to express deep gratitude to Mustafa Khammash who kindly welcomed me in his research group. I would also like to thank Cemagref, Montpellier, France for its financial support and my colleagues there that accepted to do part of my share of work during this period.

I also want to thank Vasile Sima, from the Research Institute for Informatics, Bucharest, for his help in correcting bugs in the original version of the TB03AD SLICOT routine [3]. I also want to thank Michael Sebek from PolyX [4], for his help in showing that this problem could be solved numerically at the time the bug in the original TB03AD SLICOT routine was not identified, and proving the quality of the PolyX MatLab Toolbox.

References

[1] M. A. Dahleh and I. J. Diaz-Bobillo, *Control of uncertain systems: a linear programming approach*. Prentice-Hall, 1995.

[2] M. Khammash, "A new approach to the solution of the ℓ_1 control problem: the Scaled-Q method," *IEEE Transaction on Automatic Control*, vol. 45, pp. 180–187, February 2000.

[3] E. Barth, T. Beelen, P. Benner, C. Benson, R. Byers, R. Dekeyser, F. Delebecque, M. Denham, F. Dumortier, A. Emami-Naeini, D.-W. Gu, A. Geurts, S. Hammarling, G. Van Den Hurk, B. Kgstrm, C. Kliman, M. Konstantinov, D. Kressner, A. Laub, C. Oara, C. Paige, T. Penzl, P. Petkov, V. Sima, S. Steer, F. Svaricek, M. Vanbegin, P. Van Dooren, S. Van Huffel, A. Varga, M. Verhaegen, L. Westin, H. Willemsen, T. Williams, and X. Hongguo, "The control and systems library SLICOT." http://www.win.tue.nl/niconet/NIC2/slicot.html, 2000.

[4] H. Kwakernaak and M. Sebek, "Polynomial toolbox 2." PolyX Ltd, Prague, Czech Republic, www.polyx.com, www.polyx.cz, 2000.

[5] P.-O. Malaterre and M. Khammash, "Comparison of different polynomial factorization approaches as an alternative to FIR approximation to solve the ℓ_1 design problem," *ECC Porto, Portugal*, September 2001.

Appendix 1 - Results for H

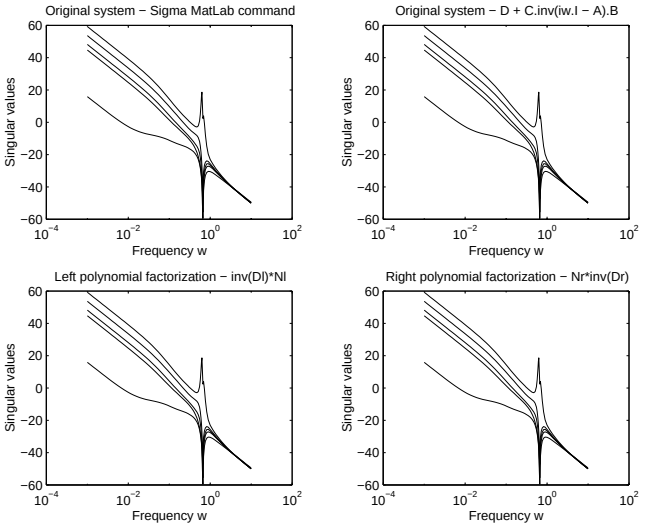

Figure 3: Comparison of the singular values for H

	without scaling	with scaling
from state-space	$4.7494\ 10^{-10}$	$4.7494\ 10^{-10}$
left factorization	$2.3334\ 10^{-8}$	$2.3342\ 10^{-8}$
right factorization	$1.2744\ 10^{-8}$	$1.2763\ 10^{-8}$

Table 7: TB03AD: Maximum difference of singular values to the ones obtained with the MatLab "Sigma" function, for H

	without scaling	with scaling	order
N_{hl}	$2.5911\ 10^{8}$	0.0028	13
D_{hl}	$9.1911\ 10^{10}$	1.0	13
N_{hr}	$1.5042\ 10^{21}$	0.0202	26
D_{hr}	$7.4527\ 10^{22}$	1.0	26

Table 8: TB03AD: Maximum absolute value of coefficients of the obtained polynomial matrices, and order of the polynomials, for H

Appendix 2 - Results for V

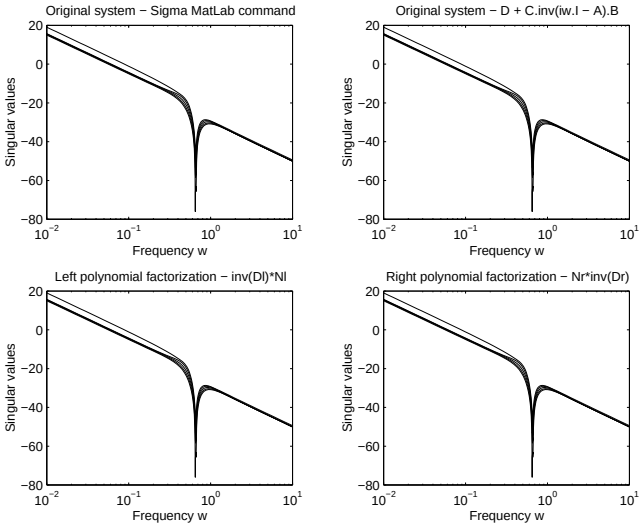

Figure 4: Comparison of the singular values for V

	without scaling	with scaling
from state-space	$8.7041\ 10^{-14}$	$8.7041\ 10^{-14}$
left factorization	$1.4602\ 10^{-12}$	$1.4619\ 10^{-12}$
right factorization	$1.0090\ 10^{-12}$	$1.0036\ 10^{-12}$

Table 9: TB03AD: Maximum difference of singular values to the ones obtained with the MatLab "Sigma" function, for V

	without scaling	with scaling	order
N_{vl}	$1.7564\ 10^{8}$	0.0265	13
D_{vl}	$6.6233\ 10^{9}$	1.0	13
N_{vr}	$4.9486\ 10^{6}$	0.0265	13
D_{vr}	$1.8684\ 10^{8}$	1.0	13

Table 10: TB03AD: Maximum absolute value of coefficients of the obtained polynomial matrices, and order of the polynomials, for V