



SAPIENZA  
UNIVERSITÀ DI ROMA

Dipartimento di Scienze Statistiche  
Sezione di Statistica Economica ed Econometria

Massimo Franchi      Paolo Paruolo

## Inverting a matrix function around a singularity via local rank factorization

*DSS Empirical Economics and Econometrics  
Working Papers Series*

**DSS-E3 WP 2014/6**

Dipartimento di Scienze Statistiche  
Sezione di Statistica Economica ed Econometria  
"Sapienza" Università di Roma  
P.le A. Moro 5 – 00185 Roma - Italia

<http://www.dss.uniroma1.it>

# INVERTING A MATRIX FUNCTION AROUND A SINGULARITY VIA LOCAL RANK FACTORIZATION

MASSIMO FRANCHI\* AND PAOLO PARUOLO\*\*

December 11, 2014

**Abstract.** This paper proposes a recursive procedure that characterizes the order of the pole and the coefficients of the Laurent series representation of the inverse of a regular analytic matrix function. The algorithm consists in performing a finite sequence of rank factorizations of matrices of non-increasing dimension, at most equal to the dimension of the original matrix function. The order of the pole is established by a full rank condition and the Laurent coefficients  $B_n$  are calculated recursively as  $B_n = H_n + \sum_{k=1}^n F_k B_{n-k}$ , where  $H_n, F_k$  have simple closed form expressions in terms of the quantities generated by the algorithm. It is further shown that the complete reduction process in [1], which provides an efficient computational method for the Laurent coefficients, corresponds to this procedure; hence the present results also provide the explicit recursive formula to compute  $B_n$  when that complete reduction process is performed. Moreover, one finds that the number of reductions is equal to the number of distinct non-zero partial multiplicities and each reduction step decreases the dimension of the coefficients by the number of partial multiplicities that are equal to a given value. This links the characteristics of the reduction process to the structure of the local Smith form.

**Key words.** Matrix valued functions; Matrix inversion; Analytic perturbation; Laurent series expansion

**AMS subject classifications.** 15A09, 47A05 47A11 47A56

**1. Introduction.** Consider a regular analytic matrix function  $A(z)$  defined on an open set  $U \subseteq \mathbb{C}$  and let

$$A(z) = \sum_{n=0}^{\infty} A_n (z - z_0)^n, \quad A_n \in \mathbb{C}^{p \times p}, \quad A_0 \neq 0, \quad z \in U, \quad (1.1)$$

be its representation around a point  $z_0 \in U$ . Assume that  $A(z_0) = A_0$  is singular and let the Laurent representation of the inverse of  $A(z)$  be

$$A(z)^{-1} = \sum_{n=0}^{\infty} B_n (z - z_0)^{n-m}, \quad B_0 \neq 0. \quad (1.2)$$

This paper discusses a recursive procedure to determine  $m$ , the order of the pole of  $A(z)^{-1}$  at  $z_0$ , and the Laurent coefficients  $\{B_n\}_{n=0}^{\infty}$  given  $\{A_n\}_{n=0}^{\infty}$ .

A classical approach to characterize the relation between (1.1) and (1.2) is via the local spectral theory, based on the concepts of root functions, Jordan chains and local Smith form, see [3, 10, 14]. The case of matrix polynomials is an important special case, see [11, 12, 13, 23, 24, 26, 29] and [9, 25, 30, 32] for matrix polynomials of degree one. The tools derived from the local spectral theory are used in the study of similarity of matrices [9, 25, 30], for the solutions of systems of differential equations [12, 13], in linear

---

\*\* **Corresponding author:** M. Franchi, Sapienza University of Rome, P.le A. Moro 5, 00185 Rome, Italy; tel. +39 06 49910784; e-mail: [massimo.franchi@uniroma1.it](mailto:massimo.franchi@uniroma1.it). \*\* P. Paruolo, European Commission, DG JRC IPSC, Economics and Applied Statistics Unit, Via E.Fermi 2749, I-21027 Ispra (VA), Italy; tel. +39 0332 785642; e-mail: [paolo.paruolo@jrc.ec.europa.eu](mailto:paolo.paruolo@jrc.ec.europa.eu)

control theory [2, 19, 22], as well as in time series econometrics [5, 6, 7, 16, 21, 31]. The same tools are also employed in numerical algorithms, such as the ones in [33, 34], for calculating the global Smith form of matrix polynomials and the Laurent represent of the inverse.

A different approach to the calculation of the Laurent coefficients  $B_n$  is found [1], see also [20]: building on the results of [19] and on the reduction technique developed in [17, 18], [1] provide efficient computational procedures for the Laurent series coefficients. In particular, they obtain recursive formulae to compute  $B_n$  when 0 or 1 reduction steps are performed on the system of equations

$$C_n = D_n = \delta_{n,m} I, \quad C_n := \sum_{k=0}^n A_k B_{n-k}, \quad D_n := \sum_{k=0}^n B_k A_{n-k}, \quad n \geq 0, \quad (1.3)$$

where  $\delta_{n,m}$  is Kronecker's delta and  $C_n, D_n$  are defined by convolution from  $A(z)A(z)^{-1} = \sum_{n=0}^{\infty} C_n(z - z_0)^{n-m}$  and  $A(z)^{-1}A(z) = \sum_{n=0}^{\infty} D_n(z - z_0)^{n-m}$  respectively. Moreover, [1] outline how the complete reduction process can be performed but do not provide a closed form expression for  $B_n$  in the general case.

The procedure presented in this paper consists in performing a sequence of rank factorizations of matrices of non-increasing dimension derived from (1.3). The order of the pole is established when a full rank condition is verified and the Laurent coefficients are then calculated recursively as

$$B_n = H_n + \sum_{k=1}^n F_k B_{n-k}, \quad n \geq 0, \quad (1.4)$$

where  $H_n, F_k$  have simple closed form expressions in terms of the quantities generated by the algorithm.

The present procedure is called ‘extended local rank factorization’ (ELRF) and it is an extension of the ‘local rank factorization’ (LRF) in [8]. In that paper it is shown that the LRF delivers the partial multiplicities and the number of partial multiplicities of a given value, i.e. the local Smith form of (1.1); it also shows how the LRF can be used to construct an extended canonical system of root functions and a canonical set of Jordan chains. Moreover, the LRF algorithm stops after a finite number of iterations by construction and this finite number is equal to the order of the pole.

In this paper we show that the extension contained in the ELRF allows to compute the Laurent coefficients  $B_n$  as in (1.4). Moreover, we find that the complete reduction process in [1] coincides with the ELRF procedure; hence the present results provide the explicit recursive formula to compute  $B_n$  when the complete reduction process is performed. Furthermore, the characteristics of the complete reduction process are linked to the structure of the local Smith form: the number of reductions is equal to the number of distinct non-zero partial multiplicities and each reduction step decreases the dimension of the coefficients by the number of partial multiplicities that are equal to a given value.

The paper is organized as follows: the rest of this introduction defines notational conventions and Section 2 motivates and defines the ELRF algorithm. Section 3 contains the main results of the paper, namely the recursive formula for the calculation of the Laurent coefficients, the relation of ELRF with the complete reduction process in [1] and the link between the characteristics of reduction process and the structure of the local Smith form. Section 5 contains an example and Section 6 concludes. Proofs are collected in Appendix A and a **MATLAB** script that implements the ELRF is provided in the Additional Material.

**1.1. Notation.** The following notation will be used throughout:  $a := b$  and  $b =: a$  indicate that  $a$  is defined by  $b$ ; any sum in which the lower limit is greater than the upper one is defined equal to 0, i.e.  $\sum_{h=a}^b c_h := 0$  if  $a > b$ . For any matrix  $\varphi \in \mathbb{C}^{p \times q}$ ,  $\varphi'$  denotes its conjugate transpose. We indicate by  $\text{col } \varphi := \{\varphi v, v \in \mathbb{C}^q\}$  the column space of  $\varphi$  and by  $\text{col } \varphi'$  the row space of  $\varphi$ ; this is in line with current use, see [28] p. 170.  $\varphi_\perp$  indicates a basis of  $\text{col}^\perp \varphi$ , the orthogonal complement of  $\text{col } \varphi$  in  $\mathbb{C}^p$ , where orthogonality is with respect to the standard inner product in  $\mathbb{C}^p$ ,  $\langle x, y \rangle := y'x$ . The matrix rank factorization of  $\varphi$  is written as  $\varphi = -\xi\eta'$ , where  $\xi$  and  $\eta$  are bases of  $\text{col } \varphi$  and  $\text{col } \varphi'$ , see Theorem 1 in [27] and the following section; the negative sign is chosen here for convenience in the calculations. When  $\varphi$  has full column rank, the following definition is used  $\bar{\varphi} := \varphi(\varphi'\varphi)^{-1}$  and  $\bar{\varphi}' := (\bar{\varphi})' = (\varphi'\varphi)^{-1}\varphi'$  while when  $\varphi = 0$ , one sets  $\bar{\varphi} := 0$ ; with this notation the orthogonal projection matrix onto  $\text{col } \varphi$  can be written as  $P_\varphi := \bar{\varphi}\varphi' = \varphi\bar{\varphi}'$  and  $P_{\varphi_\perp} := I - P_\varphi$  indicates the orthogonal projection matrix onto  $\text{col}^\perp \varphi$ . Horizontal concatenation of  $a$  and  $b$  is indicated by  $(a, b)$  and  $\#\mathcal{A}$  indicates cardinality of the set  $\mathcal{A}$ .

**2. Extended local rank factorization.** This section contains a motivation and the definition of the ‘extended local rank factorization’ (ELRF) algorithm. As the ELRF makes repeated use of matrix rank factorizations, preliminaries are first reviewed, see e.g. [27].

Consider a square non-zero  $p \times p$  matrix  $\varphi$  of rank  $r$ , and its rank decomposition  $\varphi = -\xi\eta'$  where  $\xi$  and  $\eta$  are  $p \times r$  full column rank matrices. The existence of the rank decomposition can be proven simply by *i*) observing that the column space of  $\varphi$  has dimension  $r = \text{rank } \varphi$ , *ii*) obtaining a basis  $\xi$  for this space (e.g. by Gram-Schmidt orthogonalization of the columns of  $\varphi$ ) and *iii*) by representing each column of  $\varphi$  in terms of the basis  $\xi$  (up to a change of sign); this step can be performed computing  $\eta' = -\bar{\xi}'\varphi$ . When  $\varphi = 0$  one has  $r = 0$  and one can take  $\xi = \eta = \bar{\xi} = \bar{\eta} = 0$  and  $\xi_\perp = \eta_\perp = \bar{\xi}_\perp = \bar{\eta}_\perp = I$ . In the following  $\{r, \xi, \eta\}$  are said to be given by the matrix rank factorization of  $\varphi$ ; it is also assumed that  $\bar{\xi}, \bar{\eta}, \xi_\perp, \eta_\perp, \bar{\xi}_\perp, \bar{\eta}_\perp$  are simultaneously computed, as illustrated in the following remark.

**REMARK 2.1** (Rank factorization via SVD). *Several standard matrix procedures can be used perform the matrix rank factorization  $\{r, \xi, \eta\}$  of  $\varphi$ ; here computations are illustrated using the Singular Value Decomposition (SVD), which is the standard preferred numerical method to compute the rank of a matrix, given its numerical stability, see e.g. [15].*

*Let  $\varphi = USV'$  represent the SVD of  $\varphi$ , where  $U'U = V'V = I$  and  $S = \text{diag}(s_1^2, \dots, s_p^2)$ , with  $s_1^2 \geq \dots \geq s_p^2 \geq 0$ . The rank of  $\varphi$  is numerically computed as the largest integer  $r$  for which  $s_1^2 \geq \dots \geq s_r^2 > 0$  and  $s_{r+1}^2 = \dots = s_p^2 = 0$ . Given  $r$ , one can define  $\xi = -U_1$ ,  $\eta = V_1 S_1$ ,  $\xi_\perp = U_2$ ,  $\eta_\perp = V_2$ , where  $U = (U_1, U_2)$  and  $V = (V_1, V_2)$  are partitioned into blocks of the first  $r$  columns (with subscript 1) and the last  $p-r$  columns (with subscript 2), and  $S_1 = \text{diag}(s_1^2, \dots, s_r^2)$ .*

*With this choice, one has  $\xi'\xi = I_r$ ,  $\xi'_\perp \xi_\perp = \eta'_\perp \eta_\perp = I_{p-r}$  so that  $\bar{\xi} = \xi$ ,  $\bar{\xi}_\perp = \xi_\perp$ ,  $\bar{\eta}_\perp = \eta_\perp$ ; that is, no matrix inversion is involved when computing the ‘bar’ operation  $\bar{\varphi} := \varphi(\varphi'\varphi)^{-1}$  in these cases. Moreover, one has  $\bar{\eta} = V_1 \text{diag}(s_1^{-2}, \dots, s_r^{-2})$ . This requires the inversion of the diagonal matrix  $S_1$  which is just a diagonal matrix with reciprocal entries on the main diagonal, i.e. it can be computed element-wise. Note that this is one possible choice of bases of the various spaces; this specific choice is convenient, because no matrix inversion is involved.*

The rank conditions in the ELRF are a generalization of the so called  $I(1)$  and  $I(2)$  conditions in [21], which are necessary and sufficient rank conditions for a pole of order 1 or 2; see also [19] for results similar to the first order case. Here the  $I(1)$  condition is introduced in order to provide the intuition behind the Definition 2.2 of the ELRF algorithm below. Because  $A(z_0) = A_0 \neq 0$  is singular,  $0 < r_0 := \text{rank } A_0 < p$  and  $\alpha_0, \beta_0$  in the matrix rank factorization  $A_0 = -\alpha_0 \beta'_0$  are full-column-rank matrices of dimension  $p \times r_0$ .

Consider  $C_0, D_0$  in (1.3); from  $C_0 = D_0 = 0$ , one has  $A_0 B_0 = B_0 A_0 = 0$  and because  $B_0 \neq 0$  one finds  $B_0 = \beta_{0\perp} \varphi_1 \alpha'_{0\perp}$  for some  $\varphi_1 \neq 0$ . Now consider  $C_1, D_1$  in (1.3); substituting  $A_0 = -\alpha_0 \beta'_0$  and  $B_0 = \beta_{0\perp} \varphi_1 \alpha'_{0\perp}$  in  $C_1$ , one then finds  $\alpha'_{0\perp} C_1 \bar{\alpha}_{0\perp} = \alpha'_{0\perp} (A_0 B_1 + A_1 B_0) \bar{\alpha}_{0\perp} = \alpha'_{0\perp} A_1 \beta_{0\perp} \varphi_1$  and similarly  $\bar{\beta}'_{0\perp} D_1 \beta_{0\perp} = \bar{\beta}'_{0\perp} (B_1 A_0 + B_0 A_1) \beta_{0\perp} = \varphi_1 \alpha'_{0\perp} A_1 \beta_{0\perp}$ . If  $m = 1$ ,  $C_1 = D_1 = I$  implies  $\alpha'_{0\perp} A_1 \beta_{0\perp} \varphi_1 = I$  and hence  $\text{rank } \alpha'_{0\perp} A_1 \beta_{0\perp} = p - r_0$ ,  $\varphi_1 = (\alpha'_{0\perp} A_1 \beta_{0\perp})^{-1}$  and  $B_0 = \beta_{0\perp} (\alpha'_{0\perp} A_1 \beta_{0\perp})^{-1} \alpha'_{0\perp}$ . Conversely, if  $\text{rank } \alpha'_{0\perp} A_1 \beta_{0\perp} = p - r_0$  and  $m > 1$ , the equation  $\alpha'_{0\perp} A_1 \beta_{0\perp} \varphi_1 = 0$  implies  $\varphi_1 = 0$  and leads to a contradiction. This shows that  $m = 1$  if and only if the so-called  $I(1)$  condition  $r_1 := \text{rank } \alpha'_{0\perp} A_1 \beta_{0\perp} = p - r_0$  holds. This condition is hence necessary and sufficient for the pole to be of order 1.

When the  $I(1)$  condition fails but a further full rank condition (the  $I(2)$  condition) holds the pole is of order 2. The generalization of this idea leads to the definition of the ELRF algorithm given below.

DEFINITION 2.2 (Extended local rank factorization (ELRF) algorithm).

INPUT: The inputs are the  $p \times p$  matrices  $\{A_n\}_{n=0}^\infty$  and the number  $q$  of Laurent coefficients  $B_0, \dots, B_{q-1}$  to be computed.<sup>1</sup>

OUTPUT: The outputs are the scalar  $\mu$  and the  $p \times p$  matrices  $\{F_{\mu+1,k}\}_{k=1}^{q-1}$ ,  $\{H_{\mu+1,k}\}_{k=0}^{q-1}$ .

INITIALIZATION: Set  $j = 0$ ,  $r_0^{\max} := p$ ,  $\mathcal{J}_0 = 0$  and  $a_0 = b_0 = 0$ . Compute  $\{r_0, \xi_0, \eta_0\}$  as the matrix rank factorization of  $A_0$ ,

$$A_0 = -\xi_0 \eta'_0, \quad (2.1)$$

and set  $a_1 := \alpha_0 := \xi_0$ ,  $b_1 := \beta_0 := \eta_0$ ,  $\theta_0 := \bar{\beta}_0 \bar{\alpha}'_0$ . Go to RECURSION.

RECURSION: If  $r_j = r_j^{\max}$ , then go to FINAL LOOPS; else increase  $j$  by 1 and perform all the following computations. Set  $r_j^{\max} := r_{j-1}^{\max} - r_{j-1}$ , compute  $A_{1,j} := A_j$ ,  $F_{1,j} := \theta_0 A_{1,j}$ , and for  $s = 2, \dots, j$  compute  $A_{s,j-s+1}$  and  $F_{s,j-s+1}$  using

$$A_{s,k} := A_{s-1,k+1} + A_{s-1,1} F_{s-1,k}, \quad F_{s,k} := F_{s-1,k} + \theta_{s-1} A_{s,k}. \quad (2.2)$$

Next calculate  $\{r_j, \xi_j, \eta_j\}$  as the matrix rank factorization of  $a'_{j\perp} A_{j,1} b_{j\perp}$ ,

$$a'_{j\perp} A_{j,1} b_{j\perp} = -\xi_j \eta'_j, \quad (2.3)$$

where  $a_{j\perp} = a_{j-1\perp} \xi_{j-1\perp}$  and  $b_{j\perp} = b_{j-1\perp} \eta_{j-1\perp}$ . If  $r_j = 0$ , define  $\mathcal{J}_j := \mathcal{J}_{j-1}$ ,  $a_{j+1} := a_j$ ,  $b_{j+1} := b_j$ , and  $\theta_j := 0$ ; else (i.e.  $1 \leq r_j \leq r_j^{\max}$ ) set  $\mathcal{J}_j := (\mathcal{J}_{j-1}, j)$ ,  $\alpha_j := \bar{a}_{j\perp} \xi_j$ ,  $\beta_j := \bar{b}_{j\perp} \eta_j$ ,  $\theta_j := \bar{\beta}_j \bar{\alpha}'_j$ ,  $a_{j+1} := (a_j, \alpha_j)$ , and  $b_{j+1} := (b_j, \beta_j)$ .

<sup>1</sup>Because  $A_j$  is used in RECURSION  $j$  and the ELRF stops after a finite number of iterations, in practise only a finite number of coefficients is used as input.

FINAL LOOPS: Set  $\mu := j$ ,  $\mathcal{J} := \mathcal{J}_j$ ,  $a := a_{j+1}$ ,  $b := b_{j+1}$  and compute  $F_{\mu+1,k}$  using (2.2) for  $k = 1, \dots, q-1$ . Next let  $G_{1,k} := -\delta_{k,\mu}I$ ,  $H_{1,k} := -\theta_0\delta_{k,\mu}$  and compute  $H_{\mu+1,k}$  for  $k = 0, \dots, q-1$  using the following recursions for  $s \geq 2$ :

$$G_{s,k} := G_{s-1,k+1} + A_{s-1,1}H_{s-1,k}, \quad H_{s,k} := H_{s-1,k} + \theta_{s-1}G_{s,k}. \quad (2.4)$$

The initialization and the main recursions of the ELRF correspond to the LRF in [8] and allow to determine the order of the pole  $m$ . The extension is contained in the FINAL LOOPS and allows to compute the  $q$  coefficients  $B_0, \dots, B_{q-1}$  as shown in Theorem 3.1 below.

The procedure determines the order of the pole  $m$  by checking the ranks of the  $r_j^{\max} \times r_j^{\max}$  matrices  $a'_{j\perp}A_{j,1}b_{j\perp}$  in (2.3) until full rank is found. This stopping condition terminates the recursion and determines the index  $\mu$  of the ELRF, which in [8] is shown to be equal to the order of the pole  $m$ .

We note that successive rank decompositions are performed on matrices of non-increasing dimension, i.e.  $r_j^{\max} \leq r_{j-1}^{\max} \leq p - r_0$ , where  $p$  is the dimension of  $A_n$  and  $r_0$  is the rank of  $A_0$ . At each iteration the ELRF defines the orthogonal subspaces  $\text{col } \xi_j$ ,  $\text{col } \xi_{j\perp}$  ( $\text{col } \eta_j$ ,  $\text{col } \eta_{j\perp}$ ), see (2.3); a basis of the first subspace is used to construct  $\alpha_j = \bar{a}_{j\perp}\xi_j$  ( $\beta_j = \bar{b}_{j\perp}\eta_j$ ) and the remaining orthogonal subspace is used to define  $\alpha_{j+1}$  ( $\beta_{j+1}$ ). This construction implies that  $a = (\alpha_0, \dots, \alpha_\mu)$  and  $b = (\beta_0, \dots, \beta_\mu)$  are  $p \times p$  matrices with orthogonal blocks.

Because in a rank decomposition the factors are not unique, one of them can be chosen to be orthonormal, for instance the first one as in Remark 2.1. In this case one has  $\xi_j = \bar{\xi}_j$  ( $\alpha_j = \bar{\alpha}_j$ ) so that only  $\bar{\eta}_j$  ( $\bar{\beta}_j$ ) needs to be computed. Similarly, because  $\xi_{j\perp}$ ,  $\eta_{j\perp}$  are any bases of the orthogonal complements of  $\text{col } \xi_j$  and  $\text{col } \eta_j$ , one can choose them to be orthonormal. In this case, because  $a_{j\perp}$  can also be chosen orthonormal, one finds  $a_{j+1\perp} = a_{j\perp}\xi_{j\perp}$ . Similar remarks apply to  $\eta_{j\perp}$  and  $b_{j\perp}$ . Finally note that the outputs of the ELRF are invariant with respect to the choice of bases.

REMARK 2.3 (The ELRF and Moore-Penrose inverses). *Note that*

$$F_{s,k} = \sum_{j \in \mathcal{J}_{s-1}} \bar{\beta}_j \bar{\alpha}'_j A_{j+1,k}, \quad H_{s,k} = \sum_{j \in \mathcal{J}_{s-1}} \bar{\beta}_j \bar{\alpha}'_j G_{j+1,k}.$$

These expressions include  $\bar{\beta}_j \bar{\alpha}'_j = (\alpha_j \beta'_j)^+$ , where  $^+$  denotes the Moore-Penrose inverse, see e.g. Theorem 5, p. 48, in [4].

The dimension of the Moore-Penrose inverses  $\bar{\beta}_j \bar{\alpha}'_j = (\alpha_j \beta'_j)^+$  is  $p$ . It may be noted that in fact one can compute Moore-Penrose inverses of smaller matrices. Because  $\bar{\alpha}_j = a_{j\perp} \bar{\xi}_j$  and  $\bar{\beta}_j = b_{j\perp} \bar{\eta}_j$ , the terms  $\bar{\beta}_j \bar{\alpha}'_j$  that appear in these expressions can be written as  $\bar{\beta}_j \bar{\alpha}'_j = b_{j\perp} \bar{\eta}_j \bar{\xi}_j a'_{j\perp}$ . Here  $\bar{\eta}_j \bar{\xi}'_j = (\xi_j \eta'_j)^+$ , with dimension of the Moore-Penrose inverses equal to  $r_j^{\max} = p - \sum_{i \in \mathcal{J}_{j-1}} r_i$ .

It can also be noted that no matrix inversion is required in the computation of these Moore-Penrose inverses, if one performs the matrix rank factorizations in (2.1) and (2.3) as illustrated in Remark 2.1, where the only matrix inversion is that of a  $r_j^{\max} \times r_j^{\max}$  diagonal matrix which can be performed element-wise.

REMARK 2.4 (Simplifications). *Applying the definition in (2.4), it is straightforward to verify that  $G_{s,k} = H_{s,k} = 0$  for  $s + k < \mu + 1$  and*

$$H_{\mu+1,0} = -\theta_\mu = -\bar{\beta}_m \bar{\alpha}'_m.$$

*Moreover, it can also be observed that  $G_{s,k} = H_{s,k} = 0$  for  $k > \mu$ , which yields*

$$H_{\mu+1,k} = 0, \quad k \geq \mu + 1.$$

*All these zero entries do no need to be computed, and the only relevant nonzero coefficients  $G_{s,k}$ ,  $H_{s,k}$  are found in the triangle  $0 \leq k \leq \mu$ ,  $1 \leq s \leq \mu + 1 - k$ .*

*Finally, if  $A(z)$  is a matrix polynomial of degree  $d$ , i.e.  $A_k = 0$  for  $k > d$ , one has  $A_{s,k} = 0$  for  $k > d$ , which implies  $F_{s,k} = 0$  for  $k > d$  and hence*

$$F_{\mu+1,k} = 0, \quad k \geq d + 1.$$

**3. Main results.** This section contains the main results of the paper. Theorem 3.1 proves that the ELRF delivers the coefficients of the Laurent representation and Theorem 3.3 shows that the complete reduction process in [1] coincides with the ELRF. Finally Theorem 3.4 links the characteristics of the reduction process to the structure of the local Smith form of  $A(z)$  at  $z_0$ .

THEOREM 3.1 (Laurent coefficients). *Let  $\mu$ ,  $\{F_{\mu+1,k}\}_{k=1}^{q-1}$ ,  $\{H_{\mu+1,k}\}_{k=0}^{q-1}$  be the outputs of the ELRF with inputs  $\{A_n\}_{n=0}^\infty$  and  $q$ . Then  $m = \mu$  and the Laurent coefficients in (1.2) satisfy the recursion*

$$B_n = H_n + \sum_{k=1}^n F_k B_{n-k}, \quad 0 \leq n \leq q-1, \quad (3.1)$$

*with  $H_n = H_{m+1,n}$  and  $F_k = F_{m+1,k}$ , i.e.*

$$F_k = \sum_{j \in \mathcal{J}} \theta_j A_{j+1,k}, \quad H_k = \sum_{j \in \mathcal{J}} \theta_j G_{j+1,k}, \quad \theta_j = \bar{\beta}_j \bar{\alpha}'_j. \quad (3.2)$$

*Proof.* See Appendix A.  $\square$

Remark that the coefficients of the Laurent representation are calculated using rank factorizations and Moore-Penrose inverses of matrices of decreasing dimensions, see Remark 2.1 and 2.3, and not by stacking matrices in large-dimensional systems.

REMARK 3.2 (Simplifications of Laurent coefficients). *As direct consequences of Remark 2.4 and Theorem 3.1, one finds*

$$B_0 = -\bar{\beta}_m \bar{\alpha}'_m, \quad B_n = \begin{cases} H_n + \sum_{k=1}^g F_k B_{n-k} & \text{if } 1 \leq n \leq m \\ \sum_{k=1}^g F_k B_{n-k} & \text{if } m+1 \leq n \leq q-1 \end{cases}, \quad g = n. \quad (3.3)$$

*When  $A(z)$  is a matrix polynomial of degree  $d$ , one has (3.3) with  $g = \min(n, d)$ .*



Next attention is turned to the relation between the ELRF and the reduction process in [1]. Given that in Theorem 3.3 below the two procedures are shown to coincide, it follows that (3.1) provides the explicit recursive formula to compute the Laurent coefficients when the complete reduction process in [1] is performed.

**THEOREM 3.3** (Complete reduction in [1] and ELRF). *The complete reduction process in [1] coincides with the ELRF.*

*Proof.* See Appendix A.  $\square$

Inspection of the proof of Theorem 3.3 reveals that the equations at the basis of the ELRF can be written in the format of equations (8.0)-(8.t) in [1], i.e. as the reducible system<sup>2</sup>

$$C_0 V_n + \sum_{k=1}^n C_k V_{n-k} = R_n,$$

where the dimension of  $C_k$  is  $r_j^{\max} \times r_j^{\max}$  and  $C_0 = \xi_j \eta'_j$ ; after applying a reduction step, the reduced system can be rewritten in the format of equations (10.0)-(10.t-1) in [1], i.e. as the reduced system

$$D_0 W_s + \sum_{k=1}^s D_k W_{s-k} = S_s,$$

where the dimension of  $D_k$  is  $r_{j+1}^{\max} \times r_{j+1}^{\max}$  and  $D_0 = a'_{j+1\perp} A_{j+1,1} b_{j+1\perp}$ . Because  $r_{j+1}^{\max} = r_j^{\max} - r_j$  and  $r_j = \text{rank } C_0$ , this shows that a reduction occurs if and only if  $r_j > 0$  and the dimension of the coefficients is decreased by  $r_j$ .

Next the characteristics of the reduction process are linked to the structure of the local Smith form of  $A(z)$  at  $z_0$ , see e.g. [10], with form

$$\begin{pmatrix} (z - z_0)^{\kappa_s} I_{\ell_s} & & & \\ & \ddots & & \\ & & (z - z_0)^{\kappa_2} I_{\ell_2} & \\ & & & I_{\ell_1} \end{pmatrix},$$

where  $\kappa_s > \dots > \kappa_2 > 0$ ,  $\ell_i > 0$ ,  $\sum_{i=1}^s \ell_i = p$ , and the empty elements are equal to 0. We set  $\kappa_1 = 0$  and say that  $\kappa_i \geq 0$  is a partial multiplicity of  $A(z)$  at  $z_0$  and that there are  $\ell_i > 0$  partial multiplicities that are equal to  $\kappa_i$ .

**THEOREM 3.4** (Partial multiplicities and reduction steps). *Let  $\kappa_s > \dots > \kappa_2 > \kappa_1 = 0$  be the partial multiplicities of  $A(z)$  at  $z_0$  and for  $i = 1, \dots, s$  let  $\ell_i > 0$  be the number of partial multiplicities that are equal to  $\kappa_i$ . Then  $s = \#\mathcal{J} \leq m + 1$  and the complete reduction process in [1] consists of  $s - 1$  reduction steps; reduction step  $i = 1, \dots, s - 1$  decreases the dimension of the coefficients by  $\ell_i$ , the number of partial multiplicities that are equal to  $\kappa_i$ .*

*Proof.* See Appendix A.  $\square$

The result follows from the fact that each and only  $j \in \mathcal{J}$  is a partial multiplicity of  $A(z)$  at  $z_0$  and that there are exactly  $r_j$  partial multiplicities that are equal to  $j$ , see [8]. That is,  $s = \#\mathcal{J} \leq m + 1$  and for

<sup>2</sup>Here the letters  $C$  and  $D$  are used to match the notation in [1] and they do not refer to (1.3) as in the rest of the paper.

$i = 1, \dots, s$  one has  $\kappa_i = j_i$  and  $\ell_i = r_{j_i}$ , where  $\mathcal{J} = (j_1, \dots, j_s) = (0, \dots, m)$ ; that is, the local Smith form of  $A(z)$  at  $z_0$  is equal to

$$\text{diag}((z - z_0)^j I_{r_j})_{j \in \mathcal{J}_\downarrow} = \begin{pmatrix} (z - z_0)^m I_{r_m} & & & \\ & \ddots & & \\ & & (z - z_0)^{r_{j_2}} I_{r_{j_2}} & \\ & & & I_{r_0} \end{pmatrix}, \quad (3.4)$$

where  $\mathcal{J}_\downarrow = (j_s, \dots, j_1) = (m, \dots, 0)$  indicates the vector of indices  $\mathcal{J}$  in reversed order.

The structure of the local Smith form is fully characterized by the ELRF; via Theorem 3.3, the characteristics of the reduction process are thus linked to the structure of the local Smith form.

**4. Computational complexity.** In this section we evaluate the computational complexity of the ELRF in terms of floating point operations (flops); because of Theorem 3.3, this corresponds to the computational complexity of the complete reduction process in [1]. In particular it is shown that the flops associated to the one-step reduction process are always greater or equal to those of the complete reduction process, where the former requires previous knowledge of  $m$ , unlike the ELRF.

The  $AB + C$  operation, where  $A, B$  and  $C$  are  $p \times p$  matrices, requires  $O(p^3)$  flops; the same order of complexity holds for the rank decomposition of a  $p \times p$  matrix via SVD, see e.g. p. 18 and p. 253 in [15]. In each recursion,  $j$  operations of the type  $AB + C$  are performed to compute  $A_{s,k}$  in (2.2) and the same number of  $AB + C$  operations is required for  $F_{s,k}$  in (2.2). Hence the total number of  $AB + C$  operations is  $2 \sum_{j=1}^m j = m(m+1)$ , corresponding to  $O(m^2 p^3)$  flops.

The total complexity of the rank decompositions is always less than  $O(mp^3)$  flops, because it consists of  $O(p^3)$  flops for (2.1) and of  $O((r_j^{\max})^3)$  flops for (2.3), where  $r_j^{\max} \leq p - r_0$ . Next consider final loops; each iteration involves  $AB + C$  operations to compute  $A_{s,k}$  and  $F_{s,k}$  in (2.2) and  $G_{s,k}$  and  $H_{s,k}$  in (2.4). Hence this requires  $O(m^2 p^3)$  flops. Because there are  $q - 1$  final loops, this leads to a total  $O((q - 1)m^2 p^3)$  flops.

Summing up, it can be seen that the ELRF computes  $m$  and  $B_0, \dots, B_{q-1}$  with  $O(qm^2 p^3)$  flops, see (3.1). Remark that this complexity is determined by the  $AB + C$  operations and not by the matrix rank decompositions. Note also that this estimate of the complexity does not include the simplifications due to the presence of zero matrices, see Remarks 2.4 and 3.2.

In [1] it is shown that (for known order of the pole) the one-step reduction process computes  $B_0$  with  $O(\max\{m^2 p^3, m^3(p - r_0)^3\})$  flops. Hence when  $\max\{m^2 p^3, m^3(p - r_0)^3\} = m^2 p^3$ , i.e. if  $m \leq p^3/(p - r_0)^3$ , the computational complexity of the one-step reduction coincides with the one of the ELRF (which however also provides the order of the pole), as can be seen by setting  $q = 1$ . When  $m > p^3/(p - r_0)^3$  there is a computational gain in using the ELRF, i.e. the complete reduction processes, with respect to the one-step reduction process. This arises because stacking matrices in a large-dimensional system and performing a Moore-Penrose inverse on it dominates the computational complexity of the  $AB + C$  operations as  $m$  increases.

**5. Example.** This section illustrates results using a numerical example. Consider the matrix polynomial

$$A(z) = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{A_0} + \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix}}_{A_1} z + \underbrace{\begin{pmatrix} 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix}}_{A_2} z^2 + \underbrace{\begin{pmatrix} 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{4} \end{pmatrix}}_{A_3} z^3$$

and observe that  $A(0) = A_0$  is singular. We use the ELRF at  $z_0 = 0$  to determine the order of the pole of  $A(z)^{-1}$  at  $z_0 = 0$  and to compute the coefficients of its principal part.

INITIALIZATION delivers  $r_0^{\max} = 3$ ,  $\mathcal{J}_0 = 0$  and

$$A_0 = - \underbrace{\begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}}_{\xi_0} \underbrace{\begin{pmatrix} 1 & 0 & 0 \end{pmatrix}}_{\eta'_0}, \quad a_1 = \alpha_0 = \xi_0, \quad b_1 = \beta_0 = \eta_0, \quad \theta_0 = \bar{\beta}_0 \bar{\alpha}'_0 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Given that  $1 = r_0 < r_0^{\max} = 3$ , the counter is increased to  $j = 1$  and RECURSION 1 delivers  $r_1^{\max} = 2$ ,

$$A_{1,1} = A_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix}, \quad F_{1,1} = \theta_0 A_{1,1} = 0, \quad a'_{1\perp} A_{1,1} b_{1\perp} = - \underbrace{\begin{pmatrix} 1 \\ 0 \end{pmatrix}}_{\xi_1} \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{\eta'_1}, \quad \mathcal{J}_1 = (0, 1),$$

$$\alpha_1 = \bar{a}_{1\perp} \xi_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \beta_1 = \bar{b}_{1\perp} \eta_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \theta_1 = \bar{\beta}_1 \bar{\alpha}'_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

and  $a_2 = (\alpha_0, \alpha_1)$ ,  $b_2 = (\beta_0, \beta_1)$ . Since  $1 = r_1 < r_1^{\max} = 2$ , the counter is incremented to  $j = 2$  and RECURSION 2 delivers  $r_2^{\max} = 1$ ,

$$A_{1,2} = A_2 = \begin{pmatrix} 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix}, \quad F_{1,2} = \theta_0 A_{1,2} = \begin{pmatrix} 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

and because  $F_{1,1} = 0$ ,

$$A_{2,1} = A_{1,2} = \begin{pmatrix} 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix}, \quad F_{2,1} = \theta_1 A_{2,1} = 0.$$

Next one finds  $a'_{2\perp} A_{2,1} b_{2\perp} = 0$  and hence  $\mathcal{J}_2 = \mathcal{J}_1$ ,  $a_3 = a_2$ ,  $b_3 = b_2$ , and  $\theta_2 = 0$ . Because  $0 = r_2 < r_2^{\max} = 1$ , the counter is upgraded to  $j = 3$  and RECURSION 3 delivers  $r_3^{\max} = 1$ ,

$$A_{1,3} = A_3 = \begin{pmatrix} 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{4} \end{pmatrix}, \quad F_{1,3} = \theta_0 A_{1,3} = \begin{pmatrix} 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$A_{2,2} = A_{1,3} + A_{1,1}F_{1,2} = \begin{pmatrix} 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad F_{2,2} = F_{1,2} + \theta_1 A_{2,2} = \begin{pmatrix} 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

and

$$A_{3,1} = A_{2,2} + A_{2,1}F_{2,1} = \begin{pmatrix} 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad F_{3,1} = F_{2,1} + \theta_2 A_{3,1} = 0.$$

Hence one finds  $a'_{3\perp} A_{3,1} b_{3\perp} = 0$  so that  $\mathcal{J}_3 = \mathcal{J}_2$ ,  $a_4 = a_3$ ,  $b_4 = b_3$ , and  $\theta_3 = 0$ . Because  $0 = r_3 < r_3^{\max} = 1$ , the counter is raised to  $j = 4$  and RECURSION 4 delivers  $r_4^{\max} = 1$ ,  $A_{1,4} = A_4 := 0$ ,  $F_{1,4} = \theta_0 A_{1,4} = 0$ ,

$$A_{2,3} = A_{1,4} + A_{1,1}F_{1,3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} \end{pmatrix}, \quad F_{2,3} = F_{1,3} + \theta_1 A_{2,3} = \begin{pmatrix} 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$A_{3,2} = A_{2,3} + A_{2,1}F_{2,2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}, \quad F_{3,2} = F_{2,2} + \theta_2 A_{3,2} = \begin{pmatrix} 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$A_{4,1} = A_{3,2} + A_{3,1}F_{3,1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}, \quad F_{4,1} = F_{3,1} + \theta_3 A_{4,1} = 0.$$

Hence one has  $a'_{4\perp} A_{4,1} b_{4\perp} = \frac{1}{2}$ ,  $\xi_4 = 1$ ,  $\eta_4 = -\frac{1}{2}$ ,  $\mathcal{J}_4 = (0, 1, 4)$ , and

$$\alpha_4 = \bar{a}_{4\perp} \xi_4 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \beta_4 = \bar{b}_{4\perp} \eta_4 = \begin{pmatrix} 0 \\ 0 \\ -\frac{1}{2} \end{pmatrix}, \quad \theta_4 = \bar{\beta}_4 \bar{\alpha}'_4 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -2 \end{pmatrix},$$

$a_5 = (\alpha_0, \alpha_1, \alpha_4)$ ,  $b_5 = (\beta_0, \beta_1, \beta_4)$ .

Given that the full rank condition  $1 = r_4 = r_4^{\max} = 1$  is reached, the algorithm enters the FINAL LOOPS and defines  $\mu = j = 4$ ,  $\mathcal{J} = \mathcal{J}_4 = (0, 1, 4)$ ,  $a = a_5$ ,  $b = b_5$ . Setting  $q = \mu = 4$ , one can then compute  $F_k = F_{5,k}$  for  $k = 1, 2, 3$  and  $H_k = H_{5,k}$  for  $k = 0, 1, 2, 3$  using (2.2) and (2.4); one finds

$$F_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} \end{pmatrix}, \quad F_2 = F_3 = \begin{pmatrix} 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$H_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad H_1 = H_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \quad H_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Hence, see Theorem 3.1,  $A(z)$  has a pole of order  $m = \mu = 4$  at  $z_0 = 0$  and the coefficients of the principal part of  $A(z)^{-1}$  at  $z_0 = 0$  are given by

$$B_0 = H_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad B_1 = H_1 + F_1 B_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & -1 \end{pmatrix},$$

$$B_2 = H_2 + F_1 B_1 + F_2 B_0 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}, \quad B_3 = H_3 + F_1 B_2 + F_2 B_1 + F_3 B_0 = \begin{pmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & -1 & 0 \\ \frac{1}{4} & 0 & -\frac{1}{4} \end{pmatrix}.$$

A direct computation shows that (1.3) is satisfied. Because  $m = 4$ ,  $p = 3$  and  $p - r_0 = 2$ , one has  $m > \frac{p^3}{(p-r_0)^3}$  and thus a computational gain arises by performing the complete reduction processes instead of the one-step reduction. Also recall that the latter requires previous knowledge of  $m$  which on the contrary is determined within the ELRF.

Finally consider the application of Theorem 3.4. Given the ELRF, the local Smith form of  $A(z)$  at  $z_0 = 0$  can be computed using (3.4); because  $\mathcal{J}_\downarrow = (4, 1, 0)$  and  $r_0 = r_1 = r_4 = 1$ , one finds

$$\text{diag}((z - z_0)^j I_{r_j})_{j \in \mathcal{J}_\downarrow} = \begin{pmatrix} z^4 & 0 & 0 \\ 0 & z & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The complete reduction process consists of  $\#\mathcal{J} - 1 = 2$  reduction steps: reduction step 1 occurs because  $j_1 = 0$  is a partial multiplicity of  $A(z)$  at  $z_0$  and the dimension of the system is decreased by the number of partial multiplicities that are equal to 0,  $r_0 = 1$  in this case. Reduction step 2 occurs because  $j_2 = 1$  is a partial multiplicity of  $A(z)$  at  $z_0$  and the dimension of the system is decreased by the number of partial multiplicities that are equal to 1,  $r_1 = 1$  in this case.

**6. Conclusion.** The ELRF delivers a recursive formula to compute the order of the pole and the Laurent coefficients of the inverse of a regular analytic matrix function, without stacking matrices in large-dimensional systems. The procedure consists in performing a finite sequence of rank factorizations of matrices of non-increasing dimension at most equal to the dimension of the original matrix function. The complete reduction process in [1] coincides with the ELRF; hence the latter provides the explicit recursive formula to compute the Laurent coefficients when that complete reduction process is performed. The present paper also shows that the number of reductions is equal to the number of distinct non-zero partial multiplicities and each reduction step decreases the dimension of the coefficients by the number of partial multiplicities that are equal to a given value. This links the characteristics of the reduction process to the structure of the local Smith form. Finally it is shown that the computational complexity of the ELRF compares favourably with the one of the one-step reduction process.

### Appendix A. Proofs.

The proof of Theorem 3.1 is based on the following lemma.

LEMMA A.1. *Let  $\mu, \mathcal{J}, \alpha_j, \beta_j, A_{s,k}, G_{s,k}$  be defined as in the ELRF algorithm; then for  $j \in \mathcal{J}$  and  $n \geq 0$  one has*

$$\alpha_j \beta'_j B_n = P_{a_{j\perp}} \sum_{k=1}^n A_{j+1,k} B_{n-k} + P_{a_{j\perp}} G_{j+1,n}. \quad (\text{A.1})$$

*Proof.* Pre and post-multiplying (2.3) by  $\bar{a}_{j\perp}$  and  $\bar{b}'_{j\perp}$  respectively, the rank factorization (2.3) can be rewritten as  $P_{a_{j\perp}} A_{j,1} P_{b_{j\perp}} = -\alpha_j \beta'_j$ . This shows that the initialization and main recursions of the ELRF coincide with the LRF in [8]; hence, by Theorem 3.2 in [8] one has that the index of the ELRF equals the order of the pole, i.e.  $\mu = m$ . Next observe that  $0 = P_{a_{j\perp}} (C_n - \delta_{n,m} I)$  by definition. We wish to show that for  $j \in \mathcal{J}$  and  $n \geq j$  one has

$$0 = P_{a_{j\perp}} (C_n - \delta_{n,m} I) = -\alpha_j \beta'_j B_{n-j} + P_{a_{j\perp}} \sum_{k=1}^{n-j} A_{j+1,k} B_{n-j-k} + P_{a_{j\perp}} G_{j+1,n-j}; \quad (\text{A.2})$$

remark that (A.2) implies that (A.1) holds. The proof of (A.2) is by induction. Let  $\mathbb{J}_\ell := (j_1, \dots, j_\ell)$ , with  $\mathbb{J}_u = \mathcal{J} = (j_1, \dots, j_u) = (0, \dots, m)$ . We first consider (A.2) for  $j_1 = 0$ . Because  $P_{a_{0\perp}} = I$ ,  $A_0 = -\alpha_0 \beta'_0$ ,  $A_{1,k} = A_k$ , and  $G_{1,n} = -\delta_{n,\mu} I$ , by definition of  $C_n$  one can rewrite (1.3) as

$$0 = -\alpha_0 \beta'_0 B_n + P_{a_{0\perp}} \sum_{k=1}^n A_{1,k} B_{n-k} + P_{a_{0\perp}} G_{1,n}$$

and this shows that (A.2) holds for  $j = j_1 = 0$ . Next assume that (A.2) holds for  $j \in \mathbb{J}_\ell$  for  $\ell < u$ ; one wishes to show that it also holds for  $j = j_{\ell+1} \in \mathbb{J}_{\ell+1}$  for  $\ell + 1 \leq u$ . Let  $t := j_\ell + 1$ ; pre-multiply (A.2) by  $P_{a_{t\perp}}$  and re-arrange terms to find

$$0 = P_{a_{t\perp}} A_{t,1} B_{n-t} + P_{a_{t\perp}} \sum_{k=1}^{n-t} A_{t,k+1} B_{n-t-k} + P_{a_{t\perp}} G_{t,n-t+1} =: U + V, \quad (\text{A.3})$$

where  $U := P_{a_{t\perp}} A_{t,1} B_{n-t}$  and  $V$  is defined accordingly. Next use projections, inserting  $I = P_{b_{t\perp}} + P_{b_t}$  between  $A_{t,1}$  and  $B_{n-t}$  in  $U$ ; one finds

$$U = P_{a_{t\perp}} A_{t,1} P_{b_{t\perp}} B_{n-t} + P_{a_{t\perp}} A_{t,1} P_{b_t} B_{n-t} =: U_1 + U_2.$$

The term  $U_2$  involves  $P_{b_t} = \sum_{h \in \mathbb{J}_\ell} \bar{\beta}_h \beta'_h$ , and one has  $U_2 = P_{a_{t\perp}} A_{t,1} \sum_{h \in \mathbb{J}_\ell} \bar{\beta}_h \beta'_h B_{n-t}$ .

We next observe that for  $h \in \mathbb{J}_\ell$  one has

$$\beta'_h B_{n-t} = \bar{\alpha}'_h \sum_{k=1}^{n-t} A_{h+1,k} B_{n-t-k} + \bar{\alpha}'_h G_{h+1,n-t}.$$

This is derived from (A.2) choosing  $j$  equal to  $h \in \mathbb{J}_\ell$  and replacing  $n$  with  $n - t + j$ . Substituting in the expression of  $U_2$ , one finds

$$U_2 = P_{a_{t\perp}} \sum_{k=1}^{n-t} \left( A_{t,1} \sum_{h \in \mathbb{J}_\ell} \bar{\beta}_h \bar{\alpha}'_h A_{h+1,k} \right) B_{n-t-k} + P_{a_{t\perp}} A_{t,1} \sum_{h \in \mathbb{J}_\ell} \bar{\beta}_h \bar{\alpha}'_h G_{h+1,n-t}.$$

Summing  $U + V$  and using (2.2), one finds

$$0 = U_1 + P_{a_{t\perp}} \sum_{k=1}^{n-t} A_{t+1,k} B_{n-t-k} + P_{a_{t\perp}} G_{t+1,n-t}, \quad (\text{A.4})$$

where  $G_{t+1,n-t} = G_{t,n-t+1} + A_{t,1} \sum_{h \in \mathbb{J}_\ell} \bar{\beta}_h \bar{\alpha}'_h G_{h+1,n-t}$ .

If  $t < j_{\ell+1}$  then  $P_{a_{t\perp}} A_{t,1} P_{b_{t\perp}} = 0$  and hence  $U_1 = 0$ ; in this case (A.4) reduces to (A.3) with the counter  $t$  increased by one and the process is repeated increasing the counter again until  $t = j_{\ell+1}$ . If  $t = j_{\ell+1}$  then  $U_1 = -\alpha_{j_{\ell+1}} \beta'_{j_{\ell+1}} B_{n-j_{\ell+1}}$  which makes (A.4) equal to (A.2). This completes the proof of (A.2) and hence of (A.1).  $\square$

*Proof of Theorem 3.1.* Pre-multiply (A.1) by  $\bar{\alpha}'_j$  to find

$$\beta'_j B_n = \bar{\alpha}'_j \sum_{k=1}^n A_{j+1,k} B_{n-k} + \bar{\alpha}'_j G_{j+1,n};$$

using projections, one then has

$$\begin{aligned} B_n &= \sum_{j \in \mathcal{J}} \bar{\beta}_j \beta'_j B_n = \sum_{j \in \mathcal{J}} \bar{\beta}_j \left( \bar{\alpha}'_j \sum_{k=1}^n A_{j+1,k} B_{n-k} + \bar{\alpha}'_j G_{j+1,n} \right) \\ &= \sum_{k=1}^n \left( \sum_{j \in \mathcal{J}} \bar{\beta}_j \bar{\alpha}'_j A_{j+1,k} \right) B_{n-k} + \sum_{j \in \mathcal{J}} \bar{\beta}_j \bar{\alpha}'_j G_{j+1,n} = \sum_{k=1}^n F_{\mu+1,k} B_{n-k} + H_{\mu+1,n}, \end{aligned}$$

where the last equality follows by definition from (2.2) and (2.4), see also (3.2).  $\square$

*Proof of Theorem 3.3.* Pre-multiply (A.1) by  $a'_{j\perp}$  and use the definitions  $\alpha_j = \bar{a}_{j\perp} \xi_j$ ,  $\beta_j = \bar{b}_{j\perp} \eta_j$  to find

$$\begin{aligned} \xi_j \eta'_j \bar{b}'_{j\perp} B_n &= a'_{j\perp} \sum_{k=1}^n A_{j+1,k} B_{n-k} + a'_{j\perp} G_{j+1,n} \\ &= \sum_{k=1}^n a'_{j\perp} A_{j+1,k} P_{b_{j\perp}} B_{n-k} + \sum_{k=1}^n a'_{j\perp} A_{j+1,k} P_{b_j} B_{n-k} + a'_{j\perp} G_{j+1,n}, \end{aligned} \quad (\text{A.5})$$

where the last equality follows from inserting the projection identity  $I = P_{b_{j\perp}} + P_{b_j}$  between  $A_{j+1,k}$  and  $B_{n-k}$ .

We note that (A.5) for  $j = 0$  gives the original system (1.3). We next show that, given the system (A.5), the application of one reduction step in the sense of [1] leads to the next matrix rank factorization in the ELRF. This shows that the complete reduction process in [1] coincides with the ELRF.

First observe that (A.5) can be written in the format of equations (8.0)-(8.t) in [1],<sup>3</sup>

$$C_0 V_n + \sum_{k=1}^n C_k V_{n-k} = R_n, \quad (\text{A.6})$$

<sup>3</sup>In this proof the letters  $C$  and  $D$  are used to match the notation in [1] and they do not refer to (1.3) as in the rest of the paper.

by setting  $C_0 = \xi_j \eta'_j$ ,  $C_k = -a'_{j\perp} A_{j+1,k} b_{j\perp}$ ,  $V_n = \bar{b}'_{j\perp} B_n$  and  $R_n = \sum_{k=1}^n a'_{j\perp} A_{j+1,k} P_{b_j} B_{n-k} + a'_{j\perp} G_{j+1,n}$ .

We next apply a reduction step to (A.5) pre-multiplying it by a basis of the left null space of  $C_0$  to find

$$\sum_{k=1}^n \xi'_{j\perp} C_k V_{n-k} = \xi'_{j\perp} R_n;$$

because  $\xi'_{j\perp} C_k V_{n-k} = -\xi'_{j\perp} a'_{j\perp} A_{j+1,k} P_{b_j\perp} B_{n-k} = -a'_{j+1\perp} A_{j+1,k} P_{b_{j+1\perp}} B_{n-k} - a'_{j+1\perp} A_{j+1,k} P_{\beta_j} B_{n-k}$ , where the last equality follows by definition from  $a_{j+1\perp} = a_{j\perp} \xi_{j\perp}$  and  $P_{b_{j\perp}} = P_{b_{j+1\perp}} + P_{\beta_j}$ , rearranging terms and setting  $s = n - 1$  one has

$$-a'_{j+1\perp} A_{j+1,1} P_{b_{j+1\perp}} B_s - \sum_{k=1}^s a'_{j+1\perp} A_{j+1,k+1} P_{b_{j+1\perp}} B_{s-k} = S_s,$$

where  $S_s = \xi'_{j\perp} R_{s+1} + \sum_{k=1}^{s+1} a'_{j+1\perp} A_{j+1,k} P_{\beta_j} B_{s+1-k}$ . This can be rewritten in the format of equations (10.0)-(10.t-1) in [1],

$$D_0 W_s + \sum_{k=1}^s D_k W_{s-k} = S_s, \quad (\text{A.7})$$

where  $D_k = -a'_{j+1\perp} A_{j+1,k+1} b_{j+1\perp}$ ,  $W_{s-k} = \bar{b}'_{j+1\perp} B_{s-k}$  for  $k = 0, \dots, s$  and  $s = 0, \dots, m-1$ . Because the reduced system (A.7) is again reducible if and only if  $D_0 = a'_{j+1\perp} A_{j+1,1} b_{j+1\perp}$  is of reduced rank, which is the rank condition in (2.3), this proves that the complete reduction process in [1] coincides with the ELRF.  $\square$

*Proof of Theorem 3.4.* In the proof of Theorem 3.3 it is shown that dimension of the  $C_k$  coefficients of the reducible system (A.6) is  $r_j^{\max} \times r_j^{\max}$ , where  $r_j^{\max} = p - \sum_{h \in \mathcal{J}_{j-1}} r_h$ , and the dimension of the  $D_k$  coefficients of the reduced system (A.7) is  $r_{j+1}^{\max} \times r_{j+1}^{\max}$ , where  $r_{j+1}^{\max} = r_j^{\max} - r_j$  and  $r_j = \text{rank } C_0$ . Hence a reduction occurs if and only if  $r_j > 0$ , i.e.  $j \in \mathcal{J}$ , and the dimension of the coefficients is decreased by  $r_j$ . Because each and only  $j \in \mathcal{J}$  is a partial multiplicity of  $A(z)$  at  $z_0$  and there are  $r_j$  partial multiplicities that are equal to  $j$ , see (3.4), the statement is proved.  $\square$

## REFERENCES

- [1] K. AVRACHENKOV, M. HAVIV, AND P. HOWLETT, *Inversion of analytic matrix functions that are singular at the origin*, SIAM Journal of Matrix Analysis and Applications, 22 (2001), pp. 1175–89.
- [2] H. BART, I. GOHBERG, M. KAASHOEK, AND A. RAN, *Factorization of Matrix and Operator Functions: The State Space Method*, Birkhäuser, Basel-Boston-Berlin, 2008.
- [3] H. BAUMGÄRTEL, *Analytic Perturbation Theory for Matrices and Operators*, Birkhäuser, Basel-Boston-Berlin, 1985.
- [4] A. BEN-ISRAEL AND T. GREVILLE, *Generalized Inverses: Theory and Applications*, 2nd ed., Springer, 2003.
- [5] R. ENGLE AND C. GRANGER, *Co-integration and Error Correction: Representation, Estimation, and Testing*, Econometrica, 55 (1987), pp. 251–276.
- [6] M. FRANCHI, *A representation theory for polynomial cofractionality in vector autoregressive models*, Econometric Theory, 26 (2010), pp. 1201–1217.
- [7] M. FRANCHI AND P. PARUOLO, *A characterization of vector autoregressive processes with common cyclical features*, Journal of Econometrics, 163 (2011), pp. 105–117.
- [8] ———, *Inversion of regular analytic matrix functions: local Smith form and subspace duality*, Linear Algebra and its Applications, 435 (2011), pp. 2896–2912.



- [9] F. GANTMACHER, *The Theory of Matrices*, vol. I, AMS Chelsea Publishing, 1959.
- [10] I. GOHBERG, M. KAASHOEK, AND F. VAN SCHAGEN, *On the local theory of regular analytic matrix functions*, Linear Algebra and its Applications, 182 (1993), pp. 9–25.
- [11] ———, *Partially Specified Matrices and Operators: Classification, Completion, Applications*, Birkhäuser, Basel-Boston-Berlin, 1995.
- [12] I. GOHBERG, P. LANCASTER, AND L. RODMAN, *Matrix polynomials*, Academic Press, New York, 1982.
- [13] ———, *Invariant subspaces of matrices with applications*, Wiley, New York, 1986.
- [14] I. GOHBERG AND E. SIGAL, *On operator generalizations of the logarithmic residue theorem and the theorem of Rouché*, Sbornik: Mathematics, 13 (1971), pp. 603–625.
- [15] G. GOLUB AND C. VAN LOAN, *Matrix computations*, The Johns Hopkins University Press, Baltimore and London, 1996.
- [16] N. HALDRUP AND M. SALMON, *Representation of  $I(2)$  cointegrated systems using the Smith-McMillan form*, Journal of Econometrics, 44 (1998), pp. 303–325.
- [17] M. HAVIV AND Y. RITOV, *On series expansions and stochastic matrices*, SIAM Journal on Matrix Analysis and Applications, 14 (1993), pp. 670–676.
- [18] M. HAVIV, Y. RITOV, AND U. ROTHBLUM, *Taylor expansions of eigenvalues of perturbed matrices with applications to spectral radii of nonnegative matrices*, Linear Algebra and its Applications, 168 (1992), pp. 159 – 188.
- [19] P. HOWLETT, *Input retrieval in finite dimensional linear systems*, Journal of Australian Mathematical Society (Series B), 23 (1982), pp. 357–82.
- [20] P. HOWLETT, K. AVRACHENKOV, C. PEARCE, AND V. EJOV, *Inversion of analytically perturbed linear operators that are singular at the origin*, Journal of Mathematical Analysis and Applications, 353 (2009), pp. 68–84.
- [21] S. JOHANSEN, *Likelihood-based Inference in Cointegrated Vector Auto-Regressive Models*, Oxford University Press, 1996.
- [22] T. KAILATH, *Linear systems*, Prentice Hall, Englewood Cliffs, N.J., 1980.
- [23] M. KELDYSH, *On eigenvalues and eigenfunctions of certain classes of non self-adjoint operators*, Dokl. Akad. Nauk SSSR (Russian), 77 (1951), pp. 11–14, English transl. in [26].
- [24] P. LANCASTER AND M. TISMENETSKY, *The theory of matrices*, Academic Press, 1985.
- [25] C. LANGENHOP, *The Laurent expansion for a nearly singular matrix*, Linear Algebra and its Applications, 4 (1971), pp. 329–40.
- [26] A. MARKUS, *Introduction to the spectral theory of polynomial operator pencils*, *Translations of Mathematical Monographs*, vol. 71, American Mathematical Society, Providence, 1988.
- [27] G. MARSAGLIA AND G. STYAN, *Equalities and inequalities for ranks of matrices*, Linear and Multilinear Algebra, 2 (1974), pp. 269–292.
- [28] C. MEYER, *Matrix Analysis and Applied Linear Algebra*, Society for Industrial and Applied Mathematics (SIAM), 2001.
- [29] L. RODMAN, *An Introduction to Operator Polynomials*, Birkhäuser, Basel-Boston-Berlin, 1989.
- [30] P. SCHWEITZER AND G. STEWART, *The Laurent expansion of pencils that are singular at the origin*, Linear Algebra and its Applications, 183 (1993), pp. 237–54.
- [31] F. VAHID AND R. ENGLE, *Codependent cycles*, Journal of Econometrics, 80 (1997), pp. 199–221.
- [32] M. VISHIK AND L. LYUSTERNIK, *The solution of some perturbation problems for matrices and selfadjoint or non-selfadjoint differential equations*, Uspekhi Mat. Nauk (Russian), 15 (1960), pp. 3–80, English transl. in Russian Math. Surveys, 1960, vol. 15, pp. 1–73.
- [33] J. WILKENING, *An algorithm for computing Jordan chains and inverting analytic matrix functions*, Linear Algebra and its Applications, 427 (2007), pp. 6–25.
- [34] J. WILKENING AND J. YU, *A local construction of the Smith normal form of a matrix polynomial*, Journal of Symbolic Computation, 46 (2011), pp. 1–22.

**Additional material for:  
Inverting a matrix function around a singularity  
via local rank factorization**

MASSIMO FRANCHI\* AND PAOLO PARUOLO\*\*

ABSTRACT. This file contains MATLAB routines that implement the ELRF.



1. DESCRIPTION

This file describes and includes the following MATLAB routines:

- main.m: call to the ELRF with the example in the paper;
- ELRF.m: performs the ELRF;
- recursionAF.m: recursions to compute  $\{A_{s,k}\}$  and  $\{F_{s,k}\}$ ;
- recursionGH.m: recursions to compute  $\{G_{s,k}\}$  and  $\{H_{s,k}\}$ ;
- MatRnkDecSvd.m: function that performs a matrix rank decomposition, based on the SVD function of MATLAB.

The scripts are embedded as PDF ‘file attachment annotations’, which requires PDF version 1.4 or higher (Adobe Acrobat 4.0 or higher). The embedding was generated with the attachfile LaTeX package, see [Pakin \(2011\)](#).

To open the scripts, you need to either

- right click on the icons  and choose ‘Save Embedded File to Disk...’
- or double-click on the icons .

Note that in Adobe Acrobat, annotations never print unless the Annotations box is checked in the Print dialog.

2. SCRIPT MAIN.M



This function calls the ELRF for a specific example in Section 4 of the paper.

```
%% Main file -----  
clear all; clc;  
aA(:, :, 1) = [1 0 0; 0 0 0; 0 0 0];
```

---

*Date:* December 12, 2014.

\* **Corresponding author:** M. Franchi, Sapienza University of Rome, P.le A. Moro 5, 00185 Rome, Italy; e-mail: massimo.franchi@uniroma1.it.

\*\* P. Paruolo, European Commission, Joint Research Centre, Via E.Fermi 2749, I-21027 Ispra (VA), Italy; e-mail: paolo.paruolo@jrc.ec.europa.eu.

```

aA(:,:,2)=[0 0 0; 0 -1 0; 0.5 0 0];
aA(:,:,3)=[0 0 -0.5; 0 0 0; 0.5 0 0];
aA(:,:,4)=[0 0 -0.5; 0 0 0; 0 0 -1/4];
%% ELRF -----
[out,all] = ELRF(aA,10);

```

### 3. FUNCTION ELRF.M



This function perform the ELRF.

```

%% ELRF function -----
function [out,all] = ELRF(aA,q)
% PURPOSE: compute the order of the pole and the first q Laurent
%           coefficients via ELRF. If q is left empty return the
%           coefficients of the principal part.
%-----
% USAGE:    [out,all] = ELRF(aA,q)
% where:    aA = array {A_i} i=0:d of size p x p x (d+1) for
%            A(z)=A_0+A_1(z-z_0)+...+A_{d}(z-z_0)^{d}
%            q = scalar >=0 the first q Laurent coefficients are saved in
%            all.B. If left empty, all.B contains the coefficients of
%            the principal part.
%-----
% RETURNS:
% out = structure with
% out.m = scalar >=0, the order of the pole
% out.B = array {B_{n}} n=0:q-1 of size p x p x q,
%         the first q Laurent coefficients. If q is left empty,
%         all.B contains the coefficients of
%         the principal part B_0,...,B_{m-1}
%
% all = structure with additional output
% all.a = p x p matrix a=[alpha_0,...,alpha_{mu}]
% all.b = p x p matrix b=[beta_0,...,beta_{mu}]
% all.vr = 1 x mu+1 vector vr=[r_0,...,r_mu]=[r_j]
% all.vJ = 1 x mu+1 vector vJ=[0,...,mu]
% all.theta = p x p x mu+1 array {bar(beta_j)bar(alpha_j)'} j=0:mu
% all.aA = p x p x h+1 array {A_{n}} n=0:h

```

```

% all.aB = p x p x h+1 array {B_{n}} n=0:h
% all.daA = p x p x mu+1 x h double array {A_{s,i}} s=1:mu+1,i=1:h
% all.daF = p x p x mu+1 x h double array {F_{s,i}} s=1:mu+1,i=1:h
% all.daG = p x p x mu+1 x h double array {G_{s,i}} s=1:mu+1,i=1:h
% all.daH = p x p x mu+1 x h double array {H_{s,i}} s=1:mu+1,i=1:h
% all.aF = p x p x h array {F_{mu+1,i}} i=1:h
% all.aH = p x p x h array {H_{mu+1,i}} i=1:h
% -----
% NOTES: in all arrays where one the theoretical index starts from 0,
%         entries are shifted by 1
%% Define dimensions -----
% and define maximum order of the pole, zero and identity
% and initialize matrices before the proper 'ELRF Inizialization' step
[p,p,w]=size(aA); d=w-1; mumax=d*p; all.mu=mumax; Z=zeros(p,p); I=eye(p);
all.minusI=-I; all.aA=zeros(p,p,mumax+1); all.aA(:,:,1:w)=aA;
all.daA=zeros(p,p,mumax,mumax); all.daF=all.daA;
all.theta=zeros(p,p,mumax+1); all.a=Z; all.b=all.a;
all.vJ=zeros(1,mumax+1); vuJ=all.vJ; all.vr=all.vJ; vcumr=all.vJ;
%% ELRF Inizialization -----
j=0;
[xi0,eta0,r0,xi0ort,eta0ort,eta0bar]= MatRnkDecSvd(aA(:,:,1));
rj=r0; aj=xi0; bj=eta0; aj_1ort=I; bj_1ort=I; rjmax=p;
all.a(:,1:r0)=xi0; all.b(:,1:r0)=eta0;
xij_1ort=xi0ort; etaj_1ort=eta0ort;
all.theta(:,:,1)=eta0bar*xi0';
all.vr(1)=r0; vcumr(1)=r0;
vuJ(1)=1;
%% ELRF Recursion -----
while rj < rjmax;
    j=j+1;
    rjmax=rjmax-rj;
    all=reursionAF(all,j); % F(s,j-s+1) and A(s,j-s+1)
    ajort =aj_1ort*xij_1ort;
    bjort =bj_1ort*etaj_1ort;
    mM=ajort'*all.daA(:,:,j,1)*bjort;
    [xij,etaj,rj,xijort,etajort,etajbar]= MatRnkDecSvd(mM);
    all.vr(j+1)=rj; vcumr(j+1)=vcumr(j)+rj; % r and cumr (they are shifted)

```

```

    if rj > 0;
        alphaj=ajort*xij; all.a(:,vcumr(j)+1:vcumr(j+1))=alphaj; % a
        betaj=bjort*etaj; all.b(:,vcumr(j)+1:vcumr(j+1))=betaj; % b
        all.theta(:,:,j+1)=(bjort*etajbar)*(xij'*ajort'); % theta (shifted)
        all.vJ(vuJ(j)+1)=j; vuJ(j+1)=vuJ(j)+1; % vJ and vuJ (shifted)
    else
        all.vJ(vuJ(j)+1)=all.vJ(vuJ(j)); % vJ (shifted)
        vuJ(j+1)=vuJ(j); % vuJ (shifted)
    end; % rj > 0;
    aj_1ort=ajort; bj_1ort=bjort; xij_1ort=xijort; etaj_1ort=etajort;
end; % while r < rjmax;
%% Empty un-needed entries -----
% and initialize matrices after ELRF Recursion and before ELRF Final loops
all.theta(:,:,j+2:end)=[]; all.vJ(:,j+2:end)=[]; vuJ(:,j+2:end)=[];
all.vr(:,j+2:end)=[]; vcumr(:,j+2:end)=[]; all.daA(:,:,j+1:end,:)=[];
all.daA(:,:,:,j+1:end)=[];
all.daF(:,:,j+1:end,:)=[]; all.daF(:,:,:,j+1:end)=[];
if nargin == 1
    q=j;
end
h=max(2*j+2,j+q-1); all.aA=zeros(p,p,max(h,d)+1); all.aA(:,:,1:w)=aA;
daZ=zeros(p,p,j+1,h); all.daG=daZ; all.daH=daZ;
foo=daZ; foo(:,:,1:j,1:j)=all.daA; all.daA=foo;
foo=daZ; foo(:,:,1:j,1:j)=all.daF; all.daF=foo;
all.aF=zeros(p,p,h); all.aH=zeros(p,p,h); all.aB=zeros(p,p,h+1);
%% ELRF Final loops -----
all.mu=j; % Index of the ELRF
all.H0=-all.theta(:,:,all.mu+1); % H_0=H(mu+1,0)=-theta_mu
all=recursionGH(all,all.mu); % G(s,j-s+1) and H(s,j-s+1)
for j=all.mu+1:h;
    all=recursionAF(all,j); % F(s,j-s+1) and A(s,j-s+1)
    all=recursionGH(all,j); % G(s,j-s+1) and H(s,j-s+1)
end;
all.aF=squeeze(all.daF(:,:,all.mu+1,:)); % F(mu+1,k)
all.aH=squeeze(all.daH(:,:,all.mu+1,:)); % H(mu+1,k)
%% compute B_k -----
all.aB(:,:,1)=all.H0;

```

```

for n=2:h+1
    all.aB(:,:,n)=all.aH(:,:,n-1);
    for k=2:n
        all.aB(:,:,n)=all.aB(:,:,n)+all.aF(:,:,k-1)*all.aB(:,:,n+1-k);
    end % for k=2:n
end % for n=2:h+1
%% save output -----
out.m=all.mu; out.B=all.aB(:,:,1:q);

```

#### 4. FUNCTION RECURSIONAF



This function performs recursions to compute  $\{A_{s,k}\}$  and  $\{F_{s,k}\}$ .

```

%% recursionAF function -----
function all=recursionAF(all,j)
%% PURPOSE: computes A(s,k) and F(s,k) along the j-th diagonal
%-----
% USAGE:      all=recursionAF(all,j)
% where:      j =1, 2, ...
%             all = structure
%-----
% RETURNS:
%             all = structure
%-----
% NOTES: see ELRF
%% A_{1,j} and F_{1,j} -----
all.daA(:,:,1,j)=all.aA(:,:,j+1);
all.daF(:,:,1,j)=all.theta(:,:,1)*all.daA(:,:,1,j);
%% A_{s,j-s+1} and F_{s,j-s+1} -----
foo=min(j,all.mu+1); for s=2:foo;
    all.daA(:,:,s,j+1-s)=all.daA(:,:,s-1,j+2-s)+...
        all.daA(:,:,s-1,1)*all.daF(:,:,s-1,j+1-s);
    all.daF(:,:,s,j+1-s)=all.daF(:,:,s-1,j+1-s)+...
        all.theta(:,:,s)*all.daA(:,:,s,j+1-s);
end

```

#### 5. FUNCTION RECURSIONGH



This function performs recursions to compute  $\{G_{s,k}\}$  and  $\{H_{s,k}\}$ .

```

%% recursionGH function -----
function all=recursionGH(all,j)
%% PURPOSE: computes G(s,k) and H(s,k) along the j-th diagonal
%-----
% USAGE:      all=recursionGH(all,j)
% where:      j =1, 2, ...
%             all = structure
%-----
% RETURNS:    all = structure
%-----
% NOTES: see ELRF
%% G_{1,j} and H_{1,j} -----
all.daG(:,:,1,j)=(j==all.mu)*all.minusI;
all.daH(:,:,1,j)=all.theta(:,:,1)*all.daG(:,:,1,j);
%% G_{s,j-s+1} and H_{s,j-s+1} -----
foo=min(j,all.mu+1);
for s=2:foo;
    all.daG(:,:,s,j+1-s)=all.daG(:,:,s-1,j+2-s)+...
        all.daA(:,:,s-1,1)*all.daH(:,:,s-1,j+1-s);
    all.daH(:,:,s,j+1-s)=all.daH(:,:,s-1,j+1-s)+...
        all.theta(:,:,s)*all.daG(:,:,s,j+1-s);
end

```

## 6. FUNCTION MATRNKDECSVD



This script performs a Matrix Rank Decomposition using the SVD.

```

%% MatRnkDecSvd function -----
function [mA,mB,r,mAort,mBort,mBbar]= MatRnkDecSvd(mP)
% PURPOSE: computes Matrix Rank Decomposition, mP=-mA*mB' based on svd
%-----
% USAGE:      [mA,mB,r,mAort,mBort]= MatRnkDecSvd(mP)
% where:      mP = m x n matrix
%-----
% RETURNS:    mA = m x r matrix
%             mB = n x r matrix,
%             r = rank(mP)
%             mAort = m x (m-r) matrix, basis of ort.complement of col(mA)

```

```

%           mBort = n x (n-r) matrix, basis of ort.complement of col(mB)
%-----
[U,S,V] = svd(mP); vs=diag(S); tol = max(size(mP))*eps(max(vs));
r = sum(vs > tol);
mA=-U(:,1:r); mAort=U(:,(r+1):end);
mV=V*S';
mB=mV(:,1:r);
if r > 0
    mBbar=V(:,1:r)*diag(ones(r,1)./vs(1:r));
else % this is to capture errors when rank is 0
    mBbar=mB;
end;
mBort=V(:,(r+1):end);

```

#### REFERENCES

Pakin, S. (2011). The attachfile package,  
<http://www.ctan.org/tex-archive/macros/latex/contrib/attachfile>.