Journal of
Linear and Topological Algebra
Vol. 07*, No.* 01*,* 2018*,* 53- 62

Symbolic computation of the Duggal transform

D. Pappas[a,*], V. N. Katsikis[b], I. P. Stanimirović[c]

[a] *Department of Statistics, University of Economics and Business*
76 Patission Str, 10434, Athens, Greece.

[b] *Department of Economics, Division of Mathematics and Informatics, University of Athens*
Athens, Greece

[c] *Department of Computer Science, Faculty of Science and Mathematics, University of Niš*
Višegradska 33, 18000 Niš, Serbia.

Received 9 November 2017; Revised 9 January 2018; Accepted 10 January 2018.

Communicated by Ghasem Soleimani Rad

Abstract. Following the results of [6], regarding the Aluthge transform of polynomial matrices, the symbolic computation of the Duggal transform of a polynomial matrix A is developed in this paper, using the polar decomposition and the singular value decomposition of A. Thereat, the polynomial singular value decomposition method is utilized, which is an iterative algorithm with numerical characteristics. The introduced algorithm is proven and illustrated in numerical examples. We also represent symbolically the Duggal transform of rank-one matrices using cross products of vectors and show that the Duggal transform of such matrices can be given explicitly by a closed formula and is equal to its Aluthge transform.

Keywords: Duggal transform, symbolic computation, polar decomposition, PSVD algorithm, polynomial matrices, rank 1 matrices.

2010 AMS Subject Classification: 15A60, 15A69, 15A23.

1. Introduction

Two transforms of operators and matrices that have been studied over the recent years are the Duggal and the Aluthge transforms. Especially, the Aluthge transform has been extensively studied. In [2], Foias, Jung, Ko and Pearcy introduced the concept of the

*Corresponding author.
E-mail address: dpappas@aueb.gr (D. Pappas); vaskatsikis@econ.uoa.gr (V. N. Katsikis)
ivan.stanimirovic@gmail.com (I. P. Stanimirović).

Print ISSN: 2252-0201
Online ISSN: 2345-5934

Duggal transformation, and proved several analogous results for both the Aluthge and Duggal transformations.

Definition 1.1 For an $n \times n$ (complex) matrix T with polar representation $T = U|T|$ where $|T| = (T^*T)^{\frac{1}{2}}$ and U is unitary (or a partial isometry), let us consider its Duggal transform defined by

$$\widehat{T} = |T|U$$

This transform was first introduced in [2].
Let us also restate the definition of Aluthge transform from [1]:

Definition 1.2 For an $n \times n$ (complex) matrix T with the standard polar decomposition $T = U|T|$ where $|T| = (T^*T)^{\frac{1}{2}}$ and U is a unitary matrix (or a partial isometry) satisfying $\mathcal{N}(U) = \mathcal{N}(T)$, the Aluthge transform of T is defined by

$$\Delta(T) = |T|^{\frac{1}{2}}U|T|^{\frac{1}{2}}$$

The partial isometry U in the polar representation is unique under the condition $\mathcal{N}(U) = \mathcal{N}(T)$. However the partial isometry U is not unique without the condition $\mathcal{N}(U) = \mathcal{N}(T)$. Especially, there exists a unitary matrix U such that $T = U|T|$.
Other known and important facts are the following:

$$N(U) = N(T) = N(|T|) = N(|T|^{\frac{1}{2}}), \quad N(U^*) = N(T^*), \quad R(U) = R(T), \quad U^*U = P$$

where P stands for the orthogonal projection on $\mathcal{N}(T)^{\perp}$.

Our basic motivation is the representation and the efficient symbolic computation of the Duggal transform, as a sequel of [6] where a similar work has been presented for the Aluthge transform. We will compute symbolically two specific types of matrices: the polynomial matrices and rank-one matrices using cross products of vectors. The polar decomposition as well as the singular value decomposition are often required to compute the Duggal transform. Some algorithms for the SVD and the QR decomposition for polynomial matrices are presented in [4, 5], and will be considered in the present paper.

2. The Duggal transform of a polynomial matrix

Let us propose a method for the symbolic computation of the Duggal transform of a polynomial matrix, based on the polar decomposition. The algorithm is implemented in the symbolic computational language MATHEMATICA and the corresponding codes are given in the Appendix.
Consider the polar decomposition of T given by $T = U|T|$ and its singular value decomposition (SVD) $T = W\Sigma V$. Then

$$U = WV, \quad |T| = V^*\Sigma V.$$

Calculating the SVD of a polynomial matrix $A(x)$ is clearly a more complex problem than formulating the same decomposition of a scalar matrix, as each element of A now consists of a series of polynomial coefficients. In order to drive one element of the matrix to zero, all coefficients of this element must be driven to zero and this can no longer be achieved using only Givens rotations.

Iterative nulling of coefficients in elements of a polynomial matrix is a general concept in multiple polynomial decomposition algorithms (see [5]). The process of nulling of coefficients usually involves Polynomial Givens Rotations (PGRs). The Polynomial SVD is then computed by iteratively performing consecutive PGRs on the given polynomial matrix. However, the decompositions produced by the PSVD algorithm will be approximative, since as it is shown in [8], an accurate decomposition of a polynomial matrix is practically not achievable.

McWhirter in [9] suggests another algorithm for computing a singular value decomposition of a polynomial matrix, where a sequential best rotation (SBR2) method is introduced. It considers generalized Kogbetliantz transformations and it is proven to have better performance than other similar methods.

In every step of PSVD algorithm, there is an iterative sub-routine used to change the coefficients in all polynomial elements allocated beneath the main diagonal of a particular column in the matrix $A(x)$ close to zero. The algorithm operates as a series of steps, where at each step all coefficients associated with the polynomial elements positioned beneath the diagonal of one column of the polynomial matrix $A(x)$ are driven sufficiently small [9].

In [4], a similar approach is implemented to that used when generating the paraunitary transformation matrix required within the SBR2 algorithm and so the paraunitary polynomial matrix Q(z) is formulated as a series of elementary rotation matrices interspersed with delay matrices. Thereat, an iterative algorithm for computing the singular value decomposition of a polynomial matrix is provided in [4]. It is based on iterative computations of the QR decomposition of the given matrix and its transformation to an approximately diagonal polynomial matrix.

Theorem 2.1 The Duggal transform of a rational matrix T can be expressed as

$$\widehat{T} = V^*\Sigma VWV,$$

where $T = W\Sigma V^*$ is the SVD decomposition of T.

Proof. Proceeds from the product of the matrix

$$|T| = V^*\Sigma V$$

and the matrix U which can be represented as $U = WV$. ■

Let us now derive the algorithm for computing the Duggal transform of a polynomial matrix. Polynomial SVD of the input matrix $T(x)$ is computed using the polynomial eigenvalue decomposition explained in [5], where the SBR2 algorithm is being applied to the matrix products $T(x)\widetilde{T}(x)$ and $\widetilde{T}(x)T(x)$, where $\widetilde{T}(x) = T^*(1/x)$, in order to generate the paraunitary matrices $V(x)$ and $W(x)$ and the approximately diagonal matrix $\Sigma(x)$. The elements outside of the diagonal of the matrix $\Sigma(x)$ could be driven smaller by setting smaller convergence parameters when applying the SBR2 algorithm to the matrix products $T(x)\widetilde{T}(x)$ and $\widetilde{T}(x)T(x)$.

Algorithm 2.1 Computing the polynomial matrix Duggal transform **(PMDT)**

[1] A polynomial matrix $T(x) \in \mathbb{C}(x)_r^{n\times n}$ of the normal rank r, the convergence parameter ϵ and the truncation parameter μ.

Compute the matrix

$$A_1(x) = T(x)T^*(1/x)$$

Apply the SBR2 algorithm from [5] to $A_1(x)$ and parameters ϵ and μ to compute the matrix $V(x)$.

Compute the matrix

$$A_2(x) = T^*(1/x)T(x)$$

Apply the SBR2 algorithm from [5] to $A_2(x)$ and parameters ϵ and μ to compute the matrix $W(x)$.

Compute the diagonal matrix $\Sigma(x) = V(x)T(x)V^T(1/x)$.

Denote the entries of the polynomial matrices W, Σ and V, generated in Step 1 as

$$w_{i,j}(x) = \sum_{k=\overline{w}_{min}}^{\overline{w}_{max}} w_{k,i,j}x^k, \quad i,j = \overline{1,n},$$

$$\Sigma_{i,i}(x) = \sum_{k=\overline{\sigma}_{min}}^{\overline{\sigma}_{max}} \sigma_{k,i,i}x^k, \quad i = \overline{1,n},$$

$$v_{i,j}(x) = \sum_{k=\overline{v}_{min}}^{\overline{v}_{max}} v_{k,i,j}x^k, \quad i,j = \overline{1,n},$$

For $i,j = \overline{1,n}$ perform Step 3.1 – Step 3.3.

3.1: Perform the following computation:

$$u_{t,i,j} = \sum_{k=1}^{n} \sum_{t_1=-2\overline{v}_{max}}^{t} v_{t_1,i,j}\sigma_{t_1,j,j}v^*_{t-t_1,k,j},$$
$$-2\overline{v}_{max} \leqslant t \leqslant -2\overline{v}_{min}.$$

3.2: For $k = \overline{1,n}$ perform the following computations:

$$\delta_{t,i,j,k} = \sum_{l=1}^{n} \sum_{t_1=\overline{u}_{min}+\overline{v}_{min}+\overline{w}_{min}}^{t} \sum_{t_2=\overline{u}_{min}+\overline{v}_{min}+\overline{w}_{min}}^{t-t_1} u_{t_1,i,k}w_{t-t_1-t_2,k,l}v^*_{t_2,j,l},$$
$$\overline{u}_{min} + \overline{v}_{min} + \overline{w}_{min} \leqslant t \leqslant \overline{u}_{max} + \overline{v}_{max} + \overline{w}_{max}$$

3.3: **Return** the (i,j)-th entry of the Aluthge transform matrix $\Delta(T)$ as

$$\sum_{t=\overline{u}_{min}+\overline{v}_{min}+\overline{w}_{min}}^{\overline{u}_{max}+\overline{v}_{max}+\overline{w}_{max}} \left(\sum_{k=1}^{n} \delta_{t,i,j,k} \right) x^t.$$

The convergence of PMDT algorithm can be obtained from the convergence of PSVD algorithm presented in [5], and the complexity of PMDT is the same as the complexity of PSVD algorithm.

***Example* 2.2** Consider the polynomial matrix T with the following polynomial elements:

$$T(x) = \begin{bmatrix} 0.93 - \frac{0.08}{x} & 0.17x + 0.57 & \frac{0.40}{x} - 0.05 \\ -0.35 - \frac{0.09}{x} & 0.19x + 0.67 & \frac{0.47}{x} - 0.34 \\ -0.33 - \frac{0.12}{x} & 0.87 - 0.26x & 0.29 + \frac{0.62}{x} \end{bmatrix}.$$

given in [6], with its SVD decomposition obtained from PSVD method, using $\mu = 10^{-5}$ and $\varepsilon = 10^{-3}$. The matrices W, Σ, U from $T = W\Sigma U$ are

$$W = \begin{bmatrix} -0.887 & -\frac{0.461}{x} & 0.006 \\ 0.277 & -\frac{0.543}{x} & -0.793 \\ 0.368 & -\frac{0.702}{x} & 0.610 \end{bmatrix}, \Sigma \approx \begin{bmatrix} -1.077 & 0 & 0 \\ 0 & -1.539 & 0 \\ 0 & 0 & 0.555 \end{bmatrix},$$

$$V = \begin{bmatrix} 0.982 & 0.182x & -0.059 \\ -0.116 & 0.811x & 0.574 \\ 0.151 & -0.556x & 0.817 \end{bmatrix}$$

By performing Steps 3.1 and 3.2 of the Algorithm PMDT for $i, j = \overline{1, n}$, the Duggal transform matrix has the following form:

$$\widehat{T} = \begin{bmatrix} 0.0295058x^2 + 0.976302 & 0.183589x^2 + 0.22979 & 0.209653x^2 - 0.3396 \\ 0.131479x^2 - 0.00704274 & 0.818081x^2 - 0.257083 & 0.934225x^2 + 0.225589 \\ 0.307877 - 0.0901385x^2 & -0.560854x^2 - 0.299649 & 0.217986 - 0.64048x^2 \end{bmatrix}$$

3. Rank one operators and matrices

In this section we will present the symbolic computation of the Duggal transform of rank-one operators and matrices, using cross products of vectors. This notion comes from operator theory and finite rank operators, and when applied in the finite dimensional case, we get the set of rank one matrices. We will show that the Duggal transform of such matrices can be given explicitly by a closed formula and is equal to its Aluthge transform.

In the infinite dimensional case, an operator $T \in \mathcal{B}(\mathcal{H})$, where $\mathcal{H}$ is a Hilbert space, is called a finite rank operator if the dimension of its range is finite. The number $n = \dim \mathcal{R}(T)$ is called the rank of T and it is denoted by $r(T)$. For every rank one operator T there are vectors $e, f \in H$ such that $Tx = \langle x, e\rangle f$, for every $x \in H$. Without loss of generality, the vector e can be assumed to be a unit vector. The rank one operator T is denoted by $e \otimes f$. The adjoint T^* of T is the rank one operator $T^* = f \otimes e$. When the space is finite dimensional then these operators can be represented by matrices having rank equal to one. For more on finite rank operators see [7].

In [6], we showed that the Aluthge transform of a rank one operator or matrix A is given by the formula

$$\Delta(T) = \langle x, e\rangle\langle f, e\rangle \overrightarrow{e}.$$

We will now examine the same problems for the Duggal transform, $\widehat{T}$. Without loss of generality we may assume that one of the two vectors is a unit vector.

Theorem 3.1 Let $T = e \otimes f$ be a rank one operator on a Hilbert space $\mathcal{H}$, where $\| e \| = 1$. Then $\widehat{T}$ is also rank one and it holds that $\widehat{T} = \Delta(T)$.

Proof. At first we need to find the form that the operator $|T| = (T^*T)^{\frac{1}{2}}$ has. Since

$$T^*Tx = T^*((e \otimes f)x) = \langle x, e\rangle T^*(f) = \langle x, e\rangle \| f \|^2 e,$$

we may deduce that to

$$|T|x = \langle x, e\rangle \parallel f \parallel e,$$

for more details see [6].

Since U is a partial isometry having the same range as $\mathcal{R}(T)$ we have

$$Ux = \frac{\langle x, e\rangle}{\parallel f \parallel}\overrightarrow{f}.$$

Therefore, the partial isometry on $\mathcal{R}(T^*)$ is U^* which can be found using the relation $\langle Ux, y\rangle = \langle x, U^*y\rangle$ for all $x, y \in \mathcal{H}$, to see that $U^*x = \frac{\langle x,f\rangle}{\|f\|}\overrightarrow{e}$.
We will also show that U^*U is the orthogonal projection on $\mathcal{N}(T)^{\perp} = \mathcal{R}(T)^*$.

$$U^*Ux = U^*\frac{\langle x, e\rangle}{\parallel f \parallel}\overrightarrow{f} = \frac{\langle x, e\rangle}{\parallel f \parallel}U^*\overrightarrow{f} = \frac{\langle x, e\rangle\langle f, f\rangle}{\parallel f \parallel^2}\overrightarrow{e} = \langle x, e\rangle\overrightarrow{e}$$

which is the orthogonal projection on $\mathcal{R}(T)^*$.
We will now find $\widehat{T}$:

$$\widehat{T}x = |T|Ux = |T|\frac{\langle x, e\rangle}{\parallel f \parallel}\overrightarrow{f} = \frac{\langle x, e\rangle}{\parallel f \parallel}|T|\overrightarrow{f} = \frac{\langle x, e\rangle}{\parallel f \parallel} \parallel f \parallel \langle f, e\rangle\overrightarrow{e} = \langle x, e\rangle\langle f, e\rangle\overrightarrow{e} = \Delta(T)x.$$

■

***Example* 3.2** Let $\overrightarrow{e} = \begin{bmatrix} \frac{6}{7} \\ -\frac{2}{7} \\ \frac{3}{7} \end{bmatrix}$ be a unit vector and $\overrightarrow{f} = \begin{bmatrix} 7 \\ 2 \\ -1 \end{bmatrix}$.

Then,

$$T = \begin{bmatrix} 6 & \frac{12}{7} & -\frac{6}{7} \\ -2 & -\frac{4}{7} & \frac{2}{7} \\ 3 & \frac{6}{7} & -\frac{3}{7} \end{bmatrix}$$

In addition, using the definition of the Duggal transform, we have that

$$T = U|T| = \begin{bmatrix} 0.8165 & 0.2333 & -0.1166 \\ -0.2722 & -0.0778 & 0.0389 \\ 0.4082 & 0.1166 & -0.0583 \end{bmatrix}\begin{bmatrix} 6.6681 & 1.9052 & -0.9526 \\ 1.9052 & 0.5443 & -0.2722 \\ -0.9526 & -0.2722 & 0.1361 \end{bmatrix}$$

and so,

$$\widehat{T} = |T|U = \begin{bmatrix} 4.5370 & 1.2963 & -0.6481 \\ 1.2963 & 0.3704 & -0.1852 \\ -0.6481 & -0.1852 & 0.0926 \end{bmatrix} = \Delta(T)$$

as expected.
For the computation of $\Delta(T)$ we used the definition of the Aluthge transform.

Remark 1 *Since in the relation* $\widehat{T} = \Delta(T)x = \langle x, e\rangle\langle f, e\rangle\overrightarrow{e}$ *the inner product* $\langle f, e\rangle$ *depends only on the choice of the two vectors we can say that* $\widehat{T} = \Delta(T)x = \langle f, e\rangle P$ *where* P *stands for the orthogonal projection on* $\mathcal{N}(T)^{\perp}$.

4. Conclusion

We have dealt with the symbolic computation of the Duggal transform of dual class of matrices - polynomial matrices and matrices of rank equal to one. For the class of polynomial matrices, the algorithm based on singular value decomposition of polynomial matrices was developed. The proposed procedure can be useful for large sparse matrices as well, since it is essentially an iterative method, and it is good idea to take benefit from the elements being zeros or nearly-zeros in a matrix and their positions in the matrix. The notion of exploring the sparsity patterns within the input matrix can be also combined with the proposed procedure.

In the case of rank one matrices we proved that the Duggal transform coincides with their Aluthge transform.

A future research may involve the expansion of this work for rational matrices and matrices of rank equal to 2 or higher. Further exploration of possible applications of the proposed method to large sparse matrices can be investigated in terms of determining sparsity patterns and positions of approximate zeros in the matrix.

References

[1] A. Aluthge, On p-hyponormal operators for $0 < p < 1$, Integ. Equ. Oper. Theory. 13 (1990), 307-315.
[2] C. Foias, B. Jung, E. Ko, C. Pearcy, Complete contractivity of maps associated with the Aluthge and Duggal transforms. Pacific J. Math. 209 (2) (2003), 249-259.
[3] J. Foster, J. Chambers, J. McWhirter, A novel algorithm for calculating the QR decomposition of a polynomial matrix, IEEE Acoustics, Speech and Signal Processing, ICICS, 2009.
[4] J. Foster, J. McWhirter, M. Davies, An algorithm for calculating the QR and singular value decompositions of polynomial matrices, IEEE Transactions on Signal Processing. 58 (3) (2009), 1263-1274.
[5] J. Foster, Algorithms and techniques for polynomial matrix decompositions, Ph.D. dissertation, School Eng, Cardiff Univ, U.K., 2008.
[6] D. Pappas, V. N. Katsikis, I. P. Stanimirović, Symbolic computation of the Aluthge transform. Mediterr. J. Math. (2017), doi:10.1007/s00009-017-0862-5.
[7] J. Ringrose, Compact non self adjoint operators, Van Nostrand London, 1971.
[8] R. Wirski, K. Wawryn, Decomposition of rational matrix functions, Information. Communications and Signal Processing, ICICS, 2009.
[9] J. G. McWhirter, An algorithm for polynomial matrix SVD based on generalized Kogbetliantz transformations, Proceedings of EUSIPCO, 2010.

Appendix

The implementation of PSVD algorithm using the PQRD-BS algorithm presented in [3] is provided. The complexity of PMDT is equal to the complexity of PSVD algorithm. Notice that PQRD algorithm is performed through executing a sequence of polynomial Givens rotations in order to modify an input matrix to a matrix which is upper-triangular. Thus, it represents a generalization of the baseline Givens technique for calculating the QR decomposition of a constant numerical matrix.

```
PQRDBS[A_List, eps_, mu_, MaxIter_] := Module[{Q, R, QBest, RBest,
        p = Length[A], q = Length[A[[1]]], i, ajkt, t, j1, k1, j, k,
        A1 = A, aij, g, gbest},
```

```
Q = IdentityMatrix[p];
QBest = Q;   RBest = A;
gbest = 100;   g = 100;

For[i = 0, (g > eps) && (i < MaxIter), i++,
 j = 2; k = 1; t = 1; ajkt = 0;
 For[m = 2, m <= p, m++,
  For[n = 1, n <= Min[m - 1, q], n++,
   coef = CoefficientList[Numerator[Together[A1[[m, n]]]], x];
   For[ind = 1, ind <= Length[coef], ind++,
    If[Abs[coef[[ind]]] > Abs[ajkt],
      j = m;        k = n;         t = ind;
      ajkt = coef[[ind]];
      ];      ];
  ]];
 t = t - Exponent[Denominator[A1[[j, k]]], x] - 1;
 g = Abs[ajkt];
 If[g < gbest,
  gbest = g;     QBest = Q;     RBest = A1;
  ];
 If[g > eps,
  G = IdentityMatrix[p];
  If[Abs[Coefficient[A1[[k, k]], x, 0]] != 0,
   teta = ArcTan[ Abs[Coefficient[A1[[j, k]], x, t]]/
   Abs[Coefficient[A1[[k, k]], x, 0] ]], teta = PI/2];
  fi = -Arg[Coefficient[A1[[j, k]], x, t]];
  alfa = -Arg[Coefficient[A1[[k, k]], x, 0]];
  G[[j, j]] = Cos[teta]*E^(I*alfa);
  G[[j, k]] = Sin[teta]*E^(I*fi);
  G[[k, j]] = -Sin[teta]*E^(-I*fi);
  G[[k, k]] = Cos[teta]*E^(-I*alfa);
  B = IdentityMatrix[p];
  B[[j, j]] = x^(-t);
  A1 = N[Simplify[G.B.A1]];     Q = N[Simplify[G.B.Q]];
  For[i1 = 1, i1 <= p, i1++,
   For[i2 = 1, i2 <= q, i2++,
     aij = 0;
     For[j1 = -10, j1 <= 10, j1++,
      If[N[Abs[Coefficient[A1[[i1, i2]], x, j1]]] > 0.001,
        aij += x^j1*Coefficient[A1[[i1, i2]], x, j1]];
      ];
     A1[[i1, i2]] = aij;
     ];
   ];
  For[i1 = 1, i1 <= p, i1++,
   For[i2 = 1, i2 <= p, i2++,
     aij = 0;
     For[j1 = -10, j1 <= 10, j1++,
      If[N[Abs[Coefficient[Q[[i1, i2]], x, j1]]] > 0.001,
        aij += x^j1*Coefficient[Q[[i1, i2]], x, j1]];
```

```
        ];
       Q[[i1, i2]] = aij;
       ];    ];    ];
    ];
   Return[{QBest, RBest}];
   ];
```

PSVD algorithm is performed through a sequence of QR decompositions calculations in order to modify the polynomial matrix into a diagonal matrix. An embedded try-error procedure is performed to determine appropriate values for the stop parameter and truncation parameter that lead to a nearly diagonal matrix with coefficients off the main diagonal are nearly equal to zero.

```
PSVDbyPQRD[A_List, eps_, mu_, MaxIter_] := Module[{U1, Ap, App, Appp, V1,
    R, p = Length[A], q = Length[A[[1]]], i, j, k, A1 = A, aij, n, g,
    gmin, ajk, ajk1, coef, coef1, br, mm, jj, m, U, V, Ubest, Vbest,
    SBest, nase, cuvam, sum, naset},
   U = IdentityMatrix[p];
   V = IdentityMatrix[q];
   Ubest = U;   Vbest = V;   SBest = A;
   g = 100;   gmin = 100;
   For[i = 0, (g > eps) && (i < MaxIter), i++,
    g = 0;
    For[j = 1, j <= p, j++,
     For[k = 1, k <= q, k++,
       If[k != j,
        ajk = Together[A1[[j, k]]];
        coef = CoefficientList[Numerator[ajk], x];
        br = 1;
        For[ind = 2, ind <= Length[coef], ind++,
         If[Abs[coef[[ind]]] > Abs[coef[[br]]], br = ind];
         ];
        t = br - Exponent[Denominator[ajk], x] - 1;
        g = Max[g, Abs[Coefficient[A1[[j, k]], x, t]]];
        ]];
    ];
    If[g < gmin,
     Ubest = U;     Vbest = V;     SBest = A1;     gmin = g;
     Print["U=", U // MatrixForm, "S=", A1 // MatrixForm, "V=",
      V // MatrixForm];
     ];
    If[g > eps,
     {U1, Ap} = PQRDBS[A1, eps, 0.01, 400];
     Ap += Simplify[U1.A1 - Ap];
     App = (Transpose[Ap] /. x -> x^-1);
     U = U1.U;
     {V1, Appp} = PQRDBS[App, eps, 0.01, 400];
     Appp += Simplify[V1.App - Appp];
     V = V1.V;
     A1 = (Transpose[Appp] /. x -> x^-1);
```

```
    For[i1 = 1, i1 <= p, i1++,
     For[i2 = 1, i2 <= q, i2++,
       aij = 0;
       For[j1 = -5, j1 <= 5, j1++,
        If[N[Abs[Coefficient[A1[[i1, i2]], x, j1]]] > 0.001,
          aij += x^j1*Coefficient[A1[[i1, i2]], x, j1]];
        ];
       A1[[i1, i2]] = aij;
    ];      ];
    For[i1 = 1, i1 <= p, i1++,
     For[i2 = 1, i2 <= p, i2++,
       aij = 0;
       For[j1 = -5, j1 <= 5, j1++,
        If[N[Abs[Coefficient[U[[i1, i2]], x, j1]]] > 0.001,
          aij += x^j1*Coefficient[U[[i1, i2]], x, j1]];
        ];
       U[[i1, i2]] = aij;
     ]; ];
    For[i1 = 1, i1 <= q, i1++,
     For[i2 = 1, i2 <= q, i2++,
       aij = 0;
       For[j1 = -5, j1 <= 5, j1++,
        If[N[Abs[Coefficient[V[[i1, i2]], x, j1]]] > 0.001,
          aij += x^j1*Coefficient[V[[i1, i2]], x, j1]];
        ];
       V[[i1, i2]] = aij;
       ];];
    ];
   ];
  Return[{Ubest, SBest, Vbest}];
];
```