<table>
<thead>
<tr>
<th>REPORT NUMBER</th>
<th>17636.13-MA</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE</td>
<td>Algorithms for Solvents and Spectral Factors of Matrix Polynomials</td>
</tr>
<tr>
<td>AUTHOR(s)</td>
<td>Leang S. Shieh, Yih T. Tsay, Norman P. Coleman</td>
</tr>
<tr>
<td>PERFORMING ORGANIZATION NAME AND ADDRESS</td>
<td>University of Houston, Houston, TX 77004</td>
</tr>
<tr>
<td>CONTROLLING OFFICE NAME AND ADDRESS</td>
<td>U. S. Army Research Office, F. 0. Box 12211, Research Triangle Park, NC 27709</td>
</tr>
<tr>
<td>DISTRIBUTION STATEMENT (of this Report)</td>
<td>Submitted for announcement only.</td>
</tr>
<tr>
<td>SECURITY CLASS. (of this report)</td>
<td>Unclassified</td>
</tr>
<tr>
<td>DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)</td>
<td></td>
</tr>
<tr>
<td>KEY WORDS</td>
<td></td>
</tr>
<tr>
<td>ABSTRACT</td>
<td></td>
</tr>
</tbody>
</table>

**Abstract**: Algorithms for Solvents and Spectral Factors of Matrix Polynomials.
Algorithms for solvents and spectral factors of matrix polynomials

LEANG S. SHIEH†, YIH T. TSAY† and NORMAN P. COLEMAN‡

A generalized Newton method, based on the contracted gradient of a matrix polynomial, is derived for solving the right (left) solvents and spectral factors of matrix polynomials. Two methods of selecting initial estimates for rapid convergence of the newly developed numerical method are proposed. Also, new algorithms for solving complete sets of the right (left) solvents and spectral factors without directly using the eigenvalues of matrix polynomials are derived. The proposed computer-aided method can be used to determine the spectral factorization of a matrix polynomial for optimal control, filtering and estimation problems.

1. Introduction

Linear time-invariant multi-input multi-output systems are often described by a set of coupled high-degree multivariable differential equations. The Laplace transform descriptions of such systems are the matrix fraction descriptions or polynomial matrix descriptions (Kailath 1980). The non-singular denominator matrix polynomial of the matrix fraction description, called a lambda (λ) matrix (Frazer et al. 1952, Lancaster 1966), characterizes the properties of a multivariable (multi-input-output) system. The stability property of a λ-matrix has been investigated by Lancaster (1966), Jury (1974), Anderson and Bitmead (1977), Papaconstantinou (1975), Shieh et al. (1978), Shieh and Sacheti (1978), Shieh and Tajvari (1980). Recently, the algebraic theory of λ-matrices has been investigated by Dennis et al. (1978), Gohberg et al. (1978), Markus and Mereuca (1978), and Tsay et al. (1981). Also, numerical methods for solving solvents and spectral factors of a matrix polynomial have been proposed by Dennis et al. (1978) and Shieh and Chahin (1981).

In this paper, a new numerical method, based on the contracted gradient of a matrix polynomial, is proposed for solving complete sets of right (left) solvents and spectral factors of a λ-matrix. The proposed method can be applied to determine the spectral factorization of a matrix polynomial for optimal control, filtering and estimation problems.

2. Solvents and spectral factorization of λ-matrices

An nth-degree mth-order monic λ-matrix is defined as

\[ A(\lambda) = \sum_{i=0}^{n} A_i \lambda^{n-i} \]  

where \( A_i, i = 1, \ldots, n \) are \( m \times m \) complex matrices and \( A_n(=I_m) \) is an \( m \times m \)
identity matrix. \( \lambda \) is a complex variable. Let \( X \) be an \( m \times m \) complex matrix. Two matrix polynomials, defined by

\[
A_R(X) = \sum_{i=0}^{n} A_i X^{n-i}
\]

and

\[
A_L(X) = \sum_{i=0}^{n} X^{n-i} A_i
\]

are referred to as the right and left matrix polynomials associated with \( \lambda \)-matrix \( A(\lambda) \), respectively. A right solvent \( R \) of \( A(\lambda) \) is defined by

\[
A_R(R) = \sum_{i=0}^{n} A_i R^{n-i} = 0_m
\]

and the left solvent \( L \) of \( A(\lambda) \) is defined by

\[
A_L(L) = \sum_{i=0}^{n} L^{n-i} A_i = 0_m
\]

where \( 0_m \) is an \( m \times m \) null matrix. The right (left) solvents play an important role in the analysis of the \( \lambda \)-matrix. An important set of solvents is the complete set of regular solvents (Gohberg et al. 1978, Markus and Mereuca 1978). The complete regular right solvents \( R_i, i=1, \ldots, n \) have the following properties:

\[
\rho(R_i) \cap \rho(R_j) = \emptyset, \quad i \neq j, \quad i, j = 1, \ldots, n
\]

and

\[
\bigcup_{i=1}^{n} \rho(R_i) = \rho(A(\lambda))
\]

where \( \rho(R_i) \) is the spectrum of \( R_i \) and \( \rho(A(\lambda)) \) is the spectrum of \( A(\lambda) \).

Similarly, a complete set of regular left solvents \( L_i, i=1, \ldots, n \) has the following properties:

\[
\rho(L_i) \cap \rho(L_j) = \emptyset, \quad i \neq j, \quad i, j = 1, \ldots, n
\]

and

\[
\bigcup_{i=1}^{n} \rho(L_i) = \rho(A(\lambda))
\]

The existence and uniqueness of the complete set of solvents have been investigated by Lancaster (1966), Dennis et al. (1976), Gohberg et al. (1978) and Markus and Mereuca (1978).

If a monic \( \lambda \)-matrix can be decomposed into the product of first-degree linear \( \lambda \)-matrices:

\[
A(\lambda) = (\lambda I_m - S_n)(\lambda I_m - S_{n-1}) \ldots (\lambda I_m - S_1)
\]

then \( (\lambda I_m - S_i), i=1, \ldots, n \), are referred to as the complete set of linear spectral factors. The \( m \times m \) complex matrices \( S_i, i=1, \ldots, n \), are the spectral factors of the \( \lambda \)-matrix \( A(\lambda) \). Notice that \( S_1 \) is a right solvent of \( A(\lambda) \), whereas \( S_n \) is the left solvent of \( A(\lambda) \). Other spectral factors are, in general, not the right or left solvents of \( A(\lambda) \).
Since $S_1$ is a right solvent $(\neq R_1)$ of $A(\lambda)$, a deflated $\lambda$-matrix can be obtained by factoring out the linear spectral factor $(\lambda I_m - R_1)^{-1}$ from $A(\lambda)$, as follows:

$$A^{(j)}(\lambda) \triangleq A(\lambda) (\lambda I_m - R_1)^{-1} = (\lambda I_m - S_1) \ldots (\lambda I_m - S_k) \quad (8\ b)$$

Determining a right solvent $(\neq R_1^{(j)})$ of $A^{(j)}(\lambda)$ yields the spectral factor $S_1$ in $(8\ b)$. By continuing the same procedure, we can determine all spectral factors of $A(\lambda)$ in $(8\ a)$. Notice that $R_1^{(j)}$ is a right solvent of $A^{(j)}(\lambda)$ but not that of $A(\lambda)$. In the same fashion, the spectral factors of $A(\lambda)$ can be computed from the successive left solvents of $A(\lambda)$ and the deflated $A(\lambda)$.

In optimal control and estimation problems, $A(\lambda)$ is an even degree real $\lambda$-matrix with the property

$$A(\lambda) = A^T(-\lambda) \quad (9\ a)$$

where $T$ designates transpose. The spectral factorization (Youla 1961, Anderson 1967) of $A(\lambda)$ is described by

$$A(\lambda) = A_1(\lambda) A_1^T(-\lambda) \quad (9\ b)$$

where $A_1(\lambda)$ contains all latent roots of $A(\lambda)$ in the LHP of the complex plane and $A_1(-\lambda)$ in the RHP of the complex plane. To determine $A_1(\lambda)$ the procedure for obtaining the spectral factors of $A(\lambda)$ in $(8\ a)$ can be applied. However, by using the property shown in $(9\ a)$, the computations can be simplified. In other words, if $R$ is a right solvent of $A(\lambda)$ in $(9\ b)$ with all eigenvalues in the RHP, then

$$A(\lambda) = A^{(j)}(\lambda) (\lambda I_m - R) \quad (10\ a)$$

where $A^{(j)}(\lambda)$ is an $(n-1)$ degree $\lambda$-matrix. From the property of $(9\ b)$ we have

$$A(\lambda) = A^T(-\lambda) = (-\lambda I_m - RT) A^{(j)}(-\lambda) \quad (10\ b)$$

From $(10\ b)$ we observe that $-RT$ is a left solvent of $A(\lambda)$. Thus, the factorization of $A(\lambda)$ in $(9\ b)$ can be simplified by factoring out $(\lambda I_m - R)$ and $(\lambda I_m + RT)$ from the right and left of $A(\lambda)$, respectively, if a right solvent $R$ is found. The deflated $A(\lambda)$ becomes

$$A^{(j)}(\lambda) = (\lambda I_m + RT)^{-1} A(\lambda) (\lambda I_m - R)^{-1} \quad (10\ c)$$

where $A^{(j)}(\lambda)$ is an $(n-2)$ degree $\lambda$-matrix.

Repeating the same procedure for $(10\ c)$ gives the desired spectral factors. Thus, the spectral factorization of $A(\lambda)$ can be determined from the obtained spectral factors.

In this paper, a new numerical method for solving the right (left) solvents $R_1(L_1)$ and spectral factors $(S_1)$ of $A(\lambda)$ is derived by using the contracted gradient of a matrix polynomial.

### 3. Gradient of a matrix polynomial

The gradient of an $n$th-degree $m$th-order matrix polynomial is defined as a fourth-order tensor $\nabla A(X)$ with components (Vetter 1973, Fong 1971)

$$\{\nabla A(X)\}_{k,l,k,l} = \frac{\partial}{\partial X_{ki}} \{A(X)\}_{ki}; \quad i, j, k, l = 1, \ldots, m \quad (11)$$
where \( X_{k,t} \) denotes the \((k, t)\) element of \( X \), \( \{A(X)\}_{i,j} \) designates the \((i, j)\) element of \( A(X) \) and \( \{\nabla A(X)\}_{i,j,k,l} \) denotes the \((i, j, k, l)\) element of \( \nabla A(X) \).

Direct use of the gradient for the purposes of this paper involves the inversion of a fourth-order tensor, thus causing computational difficulty. A contraction operation (Fong 1971) on \( \nabla A(X) \) with respect to an arbitrary \( m \times m \) square matrix \( Y \) is defined as

\[
\{\nabla A(X)Y\}_{i,j} = \sum_{k=1}^{n} \sum_{l=1}^{m} \{\nabla A(X)\}_{i,k,j,l} Y_{k,l} \quad i, j = 1, \ldots, m
\]

from which it follows that

\[
\nabla A(X)Y = \frac{d}{d\eta} (A(X + \eta Y))|_{\eta=0}
\]

where \( \eta \) is a scalar variable. Having the contraction operation defined in (12), the contracted gradient \( \nabla A(X)Y \) becomes an \( m \times m \) matrix. Thus, computational algorithms can be easily derived and the computational difficulty can be greatly reduced. The contraction operation in (12) can also be extended to the right (left) matrix polynomial in (2) and (3), as follows.

Given a \( \lambda \)-matrix in (1) and the right matrix polynomial in (2), the contraction operation on \( \nabla A_R(X) \) with respect to a matrix \( Y \) is

\[
\nabla A_R(X)Y = \frac{d}{d\eta} (A_R(X + \eta Y))|_{\eta=0}
\]

Each term inside the summation of (13a) can be computed as

\[
\frac{d}{d\eta} (A_t(X + \eta Y))|_{\eta=0} = A_t \sum_{s=0}^{r-1} X^s Y X^{r-s-1}
\]

Substituting (13b) into (13a) and rearranging the indexes gives

\[
\nabla A_R(X)Y = \sum_{s=0}^{n-1} A_t \sum_{r=0}^{s-1} X^s Y X^{r-s-1}
\]

Performing index transformations or letting \( k = i + q + 1 \) and \( j = i \), (14) becomes

\[
\nabla A_R(X)Y = \sum_{i=1}^{s} B_{kr}(X)Y X^{s-i}
\]

where \( B_{kr}(X) \) is the right matrix polynomial of the following \( \lambda \)-matrix \( B_k(\lambda) \):

\[
B_k(\lambda) = \sum_{j=0}^{s-1} A_j \lambda^{s-j-1}
\]

Similarly, for the left matrix polynomial \( A_L(X) \), we have

\[
\nabla A_L(X)Y = \sum_{i=1}^{s} X^{s-i} Y B_{kl}(X)
\]

where \( B_{kl}(X) \) is the left matrix polynomial of the \( \lambda \)-matrix \( B_k(\lambda) \) defined in (15b).
4. A generalized Newton method for solving solvents

The right matrix polynomial \( A_R(X) \) can be expanded around an \( m \times m \) matrix \( X_0 \) as

\[
A_R(X) = A_R(X_0) + \nabla A_R(X_0)(X - X_0) + O((X - X_0))
\]

where \( O((X - X_0)) \) is a matrix polynomial with high-degree terms of \((X - X_0)\), and \( \nabla A_R(X_0)(X - X_0) \) is a contracted gradient of dimension \( m \times m \).

The first-degree approximation of (17) with \( \| \Delta X \| < 1 \) becomes

\[
A_R(X) \approx A_R(X_0) + \nabla A_R(X_0)\Delta X
\]

where \( \Delta X \triangleq X - X_0 \).

Define a recursive formula

\[
X_{i+1} = X_i + \Delta X_{i+1}
\]

Thus, (18) becomes

\[
A_R(X_{i+1}) \approx A_R(X_i) + \nabla A_R(X_i)\Delta X_{i+1}
\]

If \( X_{i+1} \) is the right solvent of \( A(\lambda) \), or \( A_R(X_{i+1}) = 0_m \), then

\[
A_R(X_i) + \nabla A_R(X_i)\Delta X_{i+1} = 0_m
\]

Solving \( \Delta X_{i+1} \) from (20) and substituting it into (19 a) gives the recursive formula for solving the right solvent of \( A(\lambda) \). To solve the \( \Delta X_{i+1} \) we use the contracted gradient derived in (15), as follows.

From (15) we have

\[
\nabla A_R(X_i)\Delta X_{i+1} = \sum_{k=1}^{n} B_{kr}(X_i)\Delta X_{i+1}X_i^{s-k}
\]

Substituting (21) into (20) yields

\[
\sum_{k=1}^{n} B_{kr}(X_i)\Delta X_{i+1}X_i^{s-k} = -A_R(X_i)
\]

\( \Delta X_{i+1} \) can be solved by using Lancaster's Kronecker-product expansion method (Tsay et al. 1981, Lancaster 1970) as follows:

\[
\text{Vec} \{ \Delta X_{i+1} \} = -G(X_i)^{-1} \text{Vec} \{ A_R(X_i) \}
\]

where

\[
G(X_i) = \sum_{k=1}^{n} (X_i^{s-k})^T \otimes B_{kr}(X_i)
\]

\( \otimes \) designates the Kronecker product (Barnett 1971) and \( \text{Vec} \{ M \} \) is a column vector consisting of the column vectors \( M_j \) of an \( m \times m \) matrix \( M \) or

\[
\text{Vec} \{ M \} \triangleq [M_1^T, M_2^T, \ldots, M_m^T]^T
\]

Note that the dimension of \( G(X_i) \) is \( m^2 \times m^2 \). \( B_{kr}(X_i) \) is a constant matrix of the right matrix polynomial of the \( \lambda \)-matrix \( B_{kr}(\lambda) \) shown in (15 b) evaluated at \( X_0 \).

Similarly, the recursive formula for solving the left solvent of \( A(\lambda) \) is

\[
X_{i+1} = X_i + \Delta X_{i+1}
\]
where $\Delta X_{i+1}$ is the solution of the following linear matrix equation:

$$\nabla A_L(X_i)\Delta X_{i+1} = \sum_{k=1}^{n} X_i^{n-k} \Delta X_{i+1} B_{kl}(X_i) = -A_L(X_i) \tag{25}$$

or

$$\text{Vec} \{\Delta X_{i+1}\} = -H(X_i)^{-1} \text{Vec} \{A_L(X_i)\} \tag{26a}$$

where

$$H(X_i) = \sum_{k=1}^{n} B_{kl}(X_i)^T \otimes X_i^{n-k} \tag{26b}$$

The convergent criterion in both (19) and (24) is

$$\|\Delta X_{i+1}\| < \varepsilon \tag{27}$$

where $\varepsilon$ is an assigned small value.

Note that when $m=1$, $A(\lambda)$ in (1) becomes a scalar $\lambda$-polynomial. The recursive algorithm in (19) and (24) is the well-known Newton method. Thus, the proposed algorithm in this section is a generalized Newton method.

5. Algorithms for solving complete sets of solvents

To construct a similarity block transformation (Dennis et al. 1976, Tsay et al. 1981) for the analysis of a large-scale multivariable system, it is necessary to determine a complete set of solvents.

Given a $\lambda$-matrix $A(\lambda)$ in (1), if a right solvent $R$ is obtained, then $A(\lambda)$ can be described as

$$A(\lambda) = B(\lambda)(\lambda I_m - B) \tag{28a}$$

where

$$B(\lambda) = \sum_{i=0}^{n-1} B_i \lambda^{n-1} \tag{28b}$$

and the coefficients $B_i$ can be determined by

$$B_i = A_i + B_{i-1} R_i, \quad i = 1, \ldots, n-1 \tag{28c}$$

The right solvent of $B(\lambda)$ is a spectral factor of $A(\lambda)$ but in general not the right solvent of $A(\lambda)$. To resolve this problem, we have to determine the left solvent of $A(\lambda)$ associated with $R_1$ by using the following relationship (Tsay et al. 1981):

$$L_1 = Q_1^{-1} R_1 Q_1 \quad \text{if rank} \quad Q_1 = m \tag{29}$$

where $Q_1$ is the solution of the following linear matrix equation (Tsay et al. 1981, Lancaster 1970):

$$\sum_{i=0}^{n-1} R_i Q_1 = I_m \tag{30}$$

or

$$\text{Vec} \{Q_1\} = \left[ \sum_{i=0}^{n-1} B_i^T \otimes (R_i^{n-1}) \right]^{-1} \text{Vec} \{I_m\} \tag{31}$$
Equations (29) and (30) can be easily verified by substituting $L_1$ of (29) into the left matrix polynomial of $A(\lambda)$ in (29), or $A_1(L_1)$, and using the result of (30). Also, following the similar formulation in (25) and solution in (26), we can solve (30) for $Q_1$, which is shown in (31). Once $L_1$ is found, $A(\lambda)$ can be expressed by

$$A(\lambda) = (\lambda I_n - L_1)A^{(1)}(\lambda) = \lambda A^{(1)}(\lambda) - L_1 A^{(1)}(\lambda)$$

where

$$A^{(1)}(\lambda) = \sum_{i=0}^{n-1} A_i^{(1)} \lambda^{n-1-i}$$

and

$$A_i^{(1)} = A_i + L_1 A_{i-1}^{(1)}, \quad i = 1, 2, \ldots, n - 1$$

Assuming that $R_2$ is a right solvent of $A^{(1)}(\lambda)$, or $A^{(1)}(R_2) = 0$, then from (32a) we have the right matrix polynomial

$$A_2(R_2) = A_2^{(1)}(R_2)R_2 - L_1 A_2^{(1)}(R_2) = 0$$

Therefore, a right solvent of $A^{(1)}(\lambda)$ is a right solvent of $A(\lambda)$. Repeating the same procedures, we can determine the complete set of right solvents. The algorithm can be summarized as follows.

**Algorithm 1—Complete set of right solvents**

1. Set an index $i = 1$.
2. Find a right solvent $R_i$ of $A(\lambda)$ by using the recursive algorithm in (19a) and (23).
3. Find

$$B(\lambda) = \sum_{j=0}^{n-1} B_j \lambda^{n-1-j}$$

where

$$B_j = A_j + B_{j-1} R_i, \quad j = 1, 2, \ldots, n - i, \quad \text{and} \quad B_n = A_n$$

4. Find $L_i = Q_i^{-1} R_i Q_i$ by using (31)

$$\text{Vec}(Q_i) = \left\{ \sum_{j=0}^{n-1} B_j \otimes (R_i^{n-1-j}) \right\}^{-1} \text{Vec}(I_n)$$

5. Construct

$$A^{(1)}(\lambda) = \sum_{j=0}^{n-1} A_j^{(1)} \lambda^{n-1-j}$$

where

$$A_j^{(1)} = A_j + L_1 A_{j-1}^{(1)}, \quad j = 1, 2, \ldots, n - i, \quad \text{and} \quad A_n^{(1)} = A_n$$

6. Replace $A(\lambda)$ by $A^{(1)}(\lambda)$ and raise the index $i$ by 1.
7. If $i > n$, then go to (2); otherwise stop.

Similarly, the algorithm for determining the complete set of left solvents is described in Algorithm 2 as follows.
Algorithm 2—Complete set of left solvents

1. Set an index $i = 1$.
2. Find a left solvent $L_i$ of $A(\lambda)$ by the recursive algorithm in (24) and (26).
3. Find
   \[ B(\lambda) = \sum_{j=0}^{n-i} B_j \lambda^{n-i-j} \]
   where
   \[ B_j = A_j + L_i B_{j-1}, \quad j = 1, \ldots, n-i, \quad \text{and} \quad B_n = A_n \]
4. Find
   \[ R_i = P_i L_i P_i^{-1} \]
   where the $m \times m$ matrix $P_i$ can be constructed by
   \[ \text{Vec} \{ P_i \} = \left\{ \sum_{j=0}^{n-i} (L_i)^{n-i-j} \otimes B_j \right\}^{-1} \text{Vec} \{ I_m \} \]
5. Construct $A^{(i)}(\lambda)$ of degree $n-i$ by
   \[ A^{(i)}(\lambda) = \sum_{j=0}^{n-i} A_j^{(i)} \lambda^{n-i-j} \]
   where
   \[ A_j^{(i)} = A_j + A_{j-1}^{(i)} R_i, \quad j = 1, \ldots, n-i, \quad \text{and} \quad A_n^{(i)} = A_n \]
6. Replace $A(\lambda)$ by $A^{(i)}(\lambda)$ and raise the index $i$ by 1.
7. If $i \leq n$, then go to (2); otherwise stop.

If a set of linear spectral factors is desired, then the following algorithm which is derived from (8 a) and Algorithm 1 can be applied.

Algorithm 3—Linear spectral factors

1. Set an index $i = 1$.
2. Find a right solvent $R_i (= S_i)$ of $A(\lambda)$ by the recursive algorithm in (19 a) and (23).
3. Find
   \[ B(\lambda) = \sum_{j=0}^{n-i} B_j \lambda^{n-i-j} \]
   where
   \[ B_j = A_j + B_{j-1} S_i, \quad j = 1, \ldots, n-i, \quad \text{and} \quad B_n = A_n \]
4. Replace $A(\lambda)$ by $B(\lambda)$ and raise the index $i$ by 1.
5. If $i \leq n$, then go to (2); otherwise stop.

When the $\lambda$-matrix of interest has the property that $A(\lambda) = A^T(\lambda)$ of $2n$ degree and the spectral factorization is desired, we have the following algorithm.
Algorithm 4—Spectral factorization

(1) Set an index \( i = 1 \).
(2) Find a right solvent \( R_i = S_i \) of \( A(\lambda) \) by the recursive algorithm in (19 a) and (23).
(3) Find
\[
B(\lambda) = \sum_{j=0}^{n-2i+1} B_j \lambda^{n-2i-j}
\]
where
\[
B_j = A_j + B_{j-1} S_j, \quad j = 1, \ldots, n-2i+1, \quad \text{and} \quad B_n = A_n
\]
(4) Find
\[
A^{(1)}(\lambda) = \sum_{j=0}^{n-2i} A_j^{(1)} \lambda^{n-2i-j}
\]
where
\[
A_j^{(1)} = B_j - S_j^T B_{j-1}, \quad j = 1, \ldots, n-2i, \quad \text{and} \quad A_n^{(1)} = B_n
\]
(5) Replace \( A(\lambda) \) by \( A^{(1)}(\lambda) \) and raise the index \( i \) by 1.
(6) If \( i \leq n/2 \), then go to (2); otherwise stop.

The resulting spectral factorization is
\[
A(\lambda) = D^T(-\lambda)D(\lambda)
\]
where
\[
D(\lambda) = (\lambda I_m - S_{n/2}) \ldots (\lambda I_m - S_2)(\lambda I_m - S_1)
\]

6. Normalization and initial guesses

The rate of convergence of the proposed numerical method heavily depends upon the initial guesses. In order to derive a systematic method for determining the initial guesses, the \( \lambda \)-matrix is normalized such that the latent roots of the \( \lambda \)-matrix are uniformly distributed around a unit circle with centre at \( \lambda = 0 \).

The arithmetic mean of the latent roots of a given \( \lambda \)-matrix in (1) is
\[
k_1 = \frac{\sum_{i=1}^{n} \lambda_i - t_A}{nm} = \frac{n}{n}
\]
where \( \lambda_i \) are the latent roots of \( A(\lambda) = 0 \) and \( t_A \) is the trace of a matrix \( A_1 \).

Substituting
\[
\lambda = y + k_1
\]
into (1) results in
\[
A^{(1)}(y) = I_m y^n + A_1^{(1)} y^{n-1} + \ldots + A_n^{(1)}
\]
where
\[
t_A A_1^{(1)} = 0
\]
The geometric mean of the latent roots of \( A^{(2)}(y) \) is

\[
k_2 = \left( \prod_{i=1}^{m} a_i \right)^{1/m} = (\det A^{(2)})^{1/m}
\]  \hspace{1cm} (37)

where \( a_i \) are the latent roots of \( A^{(1)}(y) \).

Assuming that \( k_2 \neq 0 \) and substituting

\[
y = |k_2|z
\]  \hspace{1cm} (38)

into (36), we have

\[
A^{(2)}(z) = I_m z^n + A_2^{(2)} z^{n-1} + \ldots + A_{n-1}^{(2)} z + A_n^{(2)}
\]  \hspace{1cm} (39)

where \(|\det A^{(2)}| = 1\).

\( A^{(2)}(z) \) is referred to as the normalized \( \lambda \)-matrix of \( A(\lambda) \). The initial guesses for Algorithms 1 to 4 can be set in the following two ways.

1. Lin’s method (Lin 1943)

   The scalar version of Lin’s method is extended to the matrix case for determining the initial guess. The initial guess for the recursive algorithm of the right solvent of \( A^{(2)}(z) \) in (39) is

\[
R_0 = (A_{n-1}^{(2)})^{-1} A_n^{(2)} \quad \text{if} \quad \text{rank} \ A_{n-1}^{(2)} = m \hspace{1cm} (40)
\]

The initial guess for the recursive algorithm of the left solvent of \( A^{(2)}(z) \) becomes

\[
L_0 = A_n^{(2)}(A_{n-1}^{(2)})^{-1} \quad \text{if} \quad \text{rank} \ A_{n-1}^{(2)} = m \hspace{1cm} (41)
\]

2. Unitary matrix method

   Since the latent roots of the normalized \( \lambda \)-matrix has its arithmetic mean at zero and the geometric mean at unity, the initial guess can be set to a matrix with the magnitudes of all eigenvalues equal to unity. The matrix is a unitary matrix.

   The desired solvents or spectral factors of the \( \lambda \)-matrix can be determined by denormalizing the solvents or spectral factors obtained from \( A^{(2)}(z) \) via the transformations in (38) and (35).

7. Examples

Two examples are illustrated for finding the solvents, spectral factors and spectral factorization of \( \lambda \)-matrices. The computations were carried out on a PDP-11/70 using single precision floating point only.

Example 1

Given the \( \lambda \)-matrix

\[
A(\lambda) = I_4 \lambda^3 + \begin{bmatrix} 4 & 2 \\ -2 & 7 \end{bmatrix} \lambda^2 + \begin{bmatrix} 12 & 11 \\ -2 & 28 \end{bmatrix} \lambda + \begin{bmatrix} 19 & 14 \\ 16 & 36 \end{bmatrix}
\]  \hspace{1cm} (42)

find the complete set of regular right solvents and its linear spectral factors.
Using the normalization scheme in § 6, we have

\begin{align*}
  k_1 &= -1.83333 \\  k_2 &= 1.071363
\end{align*}

and the normalized matrix polynomial \( A^{(z)}(z) \) in (39) becomes

\[
A^{(z)}(z) = I z^2 + \begin{bmatrix}
-1.40009 & 1.866098 \\
-1.866098 & 1.40009
\end{bmatrix} z^3 + \begin{bmatrix}
6.46154 & 3.19447 \\
4.64650 & 10.8176
\end{bmatrix} z + \begin{bmatrix}
3.48240 & 0.45177 \\
10.5263 & 1.65273
\end{bmatrix}
\]

The initial guess by using Lin's method is

\[
\hat{R}_0 = \begin{bmatrix}
-0.0724806 & 0.0071291 \\
-0.041503 & -0.155843
\end{bmatrix}
\]

Applying Algorithm 1 to (44) and using the initial guess \( \hat{R}_0 \) gives the right solvent of \( A^{(z)}(z) \) at the fourth iteration with an error tolerance of \( \epsilon = 10^{-4} \) in (27). Denormalizing the obtained solvent by using (35) and (38) yields the desired right solvent

\[
\hat{R}_1 = \begin{bmatrix}
-2.00000 & -0.37887 \\
-1.00000 & -2.00000
\end{bmatrix}
\]

The corresponding left solvent, which can be determined from (29) and (31), is

\[
\hat{L}_1 = \begin{bmatrix}
-2.46154 & 0.115385 \\
-1.84615 & -0.153846
\end{bmatrix}
\]

Now factoring out \((\lambda I_2 - L_1)\) from the left of \(A(\lambda)\), or using (32), we have

\[
A^{(z)}(\lambda) = I z^2 + \begin{bmatrix}
1.53846 & 2.11538 \\
-3.84615 & 5.46154
\end{bmatrix} \lambda + \begin{bmatrix}
7.76923 & 6.42308 \\
10.76922 & 15.69233
\end{bmatrix}
\]

The normalized \( \lambda \)-matrix with \( k_1 = -1.750 \) and \( k_2 = 2.70594 \) is

\[
\hat{A}^{(z)}(z) = I z^2 + \begin{bmatrix}
-0.724901 & 0.781756 \\
-1.42138 & 0.72490
\end{bmatrix} z + \begin{bmatrix}
1.11162 & 0.371635 \\
1.06632 & 1.25607
\end{bmatrix}
\]

The initial guess by Lin's method is

\[
\hat{R}_0 = \begin{bmatrix}
0.0474374 & 1.21960 \\
-1.37797 & 0.852733
\end{bmatrix}
\]
The right solvent of \( A^{(1)}(\lambda) \) can be obtained at the thirteenth iteration with the same error tolerance of \( \epsilon = 10^{-6} \). After denormalization we have the desired right solvent of \( A^{(1)}(\lambda) \) or another right solvent of \( A(\lambda) \), as follows:

\[
R_3 = \begin{bmatrix}
-0.999976 & 1.49996 \\
-2.00004 & -2.00001
\end{bmatrix}
\]  
(46d)

The corresponding left solvent is

\[
L_3 = \begin{bmatrix}
-6.94525 & 3.35920 \\
-9.64537 & 3.94526
\end{bmatrix}
\]  
(46e)

After factoring out \((\lambda I_2 - L_3)\) from the left of \( A^{(1)}(\lambda) \) we have

\[
A^{(1)}(\lambda) = I_4 \lambda + \begin{bmatrix}
-5.40679 & 5.47459 \\
-13.4915 & 9.40880
\end{bmatrix}
\]  
(47a)

From (47a) we have the desired right solvent of \( A^{(3)}(\lambda) \) or the last right solvent of \( A(\lambda) \) as

\[
R_3 = \begin{bmatrix}
5.40679 & 5.47459 \\
13.4915 & 9.40880
\end{bmatrix}
\]  
(47b)

The complete set of linear spectral factors of \( A(\lambda) \) in (42) can be determined by using Algorithm 3. The result is

\[
A(\lambda) = (\lambda I_2 - S_3)(\lambda I_2 - S_4)(\lambda I_2 - S_3)
\]  
(48)

where

\[
S_1 = \begin{bmatrix}
-2.00000 & -0.37887 \times 10^{-7} \\
-1.00000 & -2.00000
\end{bmatrix}
\text{ (iteration number = 4)}
\]

\[
S_2 = \begin{bmatrix}
-0.99996 & 2.99991 \\
-1.00002 & -2.00004
\end{bmatrix}
\text{ (iteration number = 13)}
\]

\[
S_3 = \begin{bmatrix}
-1.00004 & -4.99991 \\
4.00002 & -2.99986
\end{bmatrix}
\text{ (iteration number = 0)}
\]

It might be interesting to investigate the relationship between the eigenvalues of \( A(\lambda) \) and the corresponding solvents and spectral factors. The eigenvalues of \( A(\lambda) \) or solvents and spectral factors are

\[
\sigma(S_1) = \sigma(R_3) = -2, -2
\]
\[
\sigma(S_2) = \sigma(R_3) = -1.5 \pm j/11
\]
\[
\sigma(S_3) = \sigma(R_3) = -2 \pm j/7
\]  
(49)
From (49) we observe that, although we have repeated latent roots with a Jordan chain of length 2 and two pairs of complex roots, the corresponding solvents and spectral factors are all real. This peculiarity has important applications in the analysis and synthesis of large-scale multivariable control systems.

**Example 2**

Given a \( A \)-matrix

\[
A(\lambda) = I_4 \lambda^4 + \begin{bmatrix} 0 & -6 \\ -6 & 0 \end{bmatrix} \lambda^2 + \begin{bmatrix} -37 & 3 \\ 3 & -18 \end{bmatrix} \lambda^2 
\]

\[
+ \begin{bmatrix} 0 & 49 \\ -49 & 0 \end{bmatrix} \lambda + \begin{bmatrix} 68 & 2 \\ 2 & 26 \end{bmatrix}
\]

and

\[
A(\lambda) = A^T(-\lambda)
\]

it is desirable to find the spectral factorization of \( A(\lambda) \) such that

\[
A(\lambda) = A_1(\lambda) A^T_2(-\lambda)
\]

where \( A_1(\lambda) \) contains all latent roots of \( A(\lambda) \) in the LHP of the complex plane and \( A^T_2(-\lambda) \) in the RHP of the complex plane.

Using Algorithm 4 and the initial guess \( I_4 \) (a unitary matrix), we have

\[
A_1(\lambda) = (\lambda I_4 - S_1)(\lambda I_4 - S_2)
\]

where

\[
S_1 = \begin{bmatrix} -0.99999 & 2.00001 \\ -1.00000 & -0.99999 \end{bmatrix}
\]

Iteration number: 11
Error tolerance: \( 10^{-4} \)

\[
S_2 = \begin{bmatrix} -0.99999 & 2.00001 \\ -1.00000 & -0.99999 \end{bmatrix}
\]

Iteration number: 8
Error tolerance: \( 10^{-4} \)

Thus, \( A_1(\lambda) \) becomes

\[
A_1(\lambda) = I_4 \lambda^4 + \begin{bmatrix} 4.99997 & -3.99997 \\ 1.99999 & 3.99998 \end{bmatrix} \lambda + \begin{bmatrix} 1.99996 & -7.99996 \\ 4.99996 & 0.99990 \end{bmatrix}
\]

Observe that, although the latent roots of \( A(\lambda) \) are all complex, i.e.

\[
\lambda = -1 \pm j\sqrt{2}; \quad \frac{1}{4}(-7 \pm j\sqrt{7}); \quad 1 \pm j\sqrt{2}; \quad \frac{1}{4}(7 \pm j\sqrt{7})
\]

\( S_1 \) and \( S_2 \) are real matrices.
8. Conclusions

A generalized Newton method, which is based on the contracted gradient of a matrix polynomial, has been derived for solving the complete sets of solvents and spectral factors of a matrix polynomial without directly using the eigenvalues of the matrix polynomial. Lin's method and the unitary matrix method have been proposed for determining the initial estimates of the solvents and spectral factors. Thus, the convergence speed of the new numerical method is improved. The proposed method can be used to determine the spectral factorization of a matrix polynomial for optimal control, filtering and estimation problems. The obtained solvents can be used to construct a block similarity transformation matrix for the decoupling of a large-scale multivariable control system.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. Army Research Office, under Grant DAAG 29-80-K-0077 and by the U.S. Army Missile Research and Development Command, under Contract DAAH 01-80-C-0323.

REFERENCES