Fassbender, H.; Benner, P.

Computing roots of matrix products

The problem of computing a kth root of a matrix product $W = \prod_{i=1}^{k} A_i$ is considered. The explicit computation of W may produce a highly inaccurate result due to rounding errors, such that the computed root $\widehat{W}^{\frac{1}{k}}$ is far from the actual root $W^{\frac{1}{k}}$. An algorithm for computing the square root of W is presented which avoids the explicit computation of W by employing the periodic Schur decomposition and therefore yields better accuracy in the computed root $\widehat{W}^{\frac{1}{2}}$. In principle, the techniques are applicable to $k > 2$ as well but lead to solving 2×2 polynomial matrix equations which are difficult to treat. The case $k = 3$ is also addressed briefly.

1. Introduction

Computing the square root of the product of two matrices can be used in model reduction methods based on the cross-Gramians; see [1] and the references therein. This lead us to consider the more general problem of computing the kth root of a matrix product $W = A_1 A_2 \cdots A_{k-1} A_k$ where $A_i \in \mathbb{R}^{n \times n}$. That is, a matrix $W^{\frac{1}{k}}$ is sought such that $W = (W^{\frac{1}{k}})^k$. A kth root of a matrix may not exist. For example, it is easy to verify that the matrix $\left[\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right]$ has no square root. Here we will not discuss existence results any further.

The direct approach for computing the kth root of W involves first the explicit computation of W followed by the computation of its kth root. Already the explicit computation of W may produce a highly inaccurate result due to rounding errors, such that the computed product $\widehat{W}$ is far from the actual matrix W. Then one can not expect that the computed root $\widehat{W}^{\frac{1}{k}}$ is close to the actual root $W^{\frac{1}{k}}$. Hence explicitly forming the matrix W should be avoided if possible. The approach suggested here was inspired by the work in [2]. There a fast and stable method for computing the square root X of a given matrix A is developed. The method is based on the Schur factorization $A = QSQ^H$ and uses a fast recursion to compute the upper triangular square root U of S. Then $X = QUQ^H$ is the desired square root. The fast recursion is obtained by comparing coefficients in the equation $U^2 = S$.

The algorithm for computing the kth root of W sketched in the following avoids the explicit computation of W by employing the periodic Schur decomposition. That is, the real Schur factorization $W = Q_1 R Q_1^T$ will be computed implicitly by simultaneously reducing all but the first A_j to upper triangular matrices R_j, A_1 can only be reduced to a quasi-upper triangular matrix R_1. Then the kth root of the product of the factors R_j is computed. Hence the computed root has the same quasi-upper triangular form as R_1. This root is then transformed back to the root of W. But note that even in case a kth root of $Q_1^T W Q_1$ exists, it must not necessarily have the form of R_1.

The process will be demonstrated for $k = 2$, the case $k = 3$ is briefly discussed. By avoiding to form the product W explicitly, this approach yields better accuracy in the computed root $\widehat{W}^{\frac{1}{k}}$ than the direct approach.

2. First step: The periodic Schur decomposition

In order to avoid explicitly computing the matrix W a two step approach for computing the kth root of W is proposed which implicitly computes the desired root without ever forming the explicit matrix product W. The first step of the algorithm consists of employing the periodic Schur decomposition.

Theorem 1. *Let $A_j \in \mathbb{R}^{n \times n}$, $j = 1, \ldots, k$. There exist orthogonal matrices Q_j, $j = 1, \ldots, k$, such that*

$$R_1 = Q_1 A_1 Q_2^T, \quad R_2 = Q_2 A_2 Q_3^T, \quad \ldots, \quad R_{k-1} = Q_{k-1} A_{k-1} Q_k^T, \quad R_k = Q_k A_k Q_1^T,$$

where $R_2, \ldots, R_k$ are upper triangular and R_1 is in quasi-upper triangular form with 1×1 and 2×2 diagonal blocks. Moreover, Q_1 puts W into real Schur form, i.e. $Q_1^T W Q_1 = R_1 R_2 \cdots R_{k-1} R_k$ is quasi-upper triangular.

Constructive proofs and algorithms for computing the periodic Schur decomposition without explicitly forming the product W can be found in [3] and [4].

3. Second step: The kth root

Here we will assume $k = 2$. Using the periodic Schur decomposition the product $W = A_1 A_2$ is transformed to

$\widehat{W} = Q_1^T W Q_1 = (Q_1^T A_1 Q_2)(Q_2^T A_2 Q_1)$ such that $H = Q_1^T A_1 Q_2$ is quasi-upper triangular and $R = Q_2^T A_2 Q_1$ is upper triangular. $\widehat{W}$ has the same quasi-triangular form as H. Comparing the coefficients in $\widehat{W} = X^2$ gives us formulae to compute the elements of the square root X of $\widehat{W}$. Then $W^{\frac{1}{2}} = Q_1 X Q_1^T$. In order to compute the elements of X we first have to consider the diagonal blocks. In case of a 1×1 block $h_{\ell\ell}$ with $h_{\ell+1,\ell} = h_{\ell,\ell-1} = 0$ we can directly read off $x_{\ell\ell} = \sqrt{h_{\ell\ell} r_{\ell\ell}}$. In case of a 2×2 block $\left[\begin{smallmatrix} h_{\ell\ell} & h_{\ell,\ell+1} \\ h_{\ell+1,\ell} & h_{\ell+1,\ell+1} \end{smallmatrix}\right]$ we follow the approach from [2]. First orthogonal matrices U, V are computed such that $\widetilde{R} = V^T \left[\begin{smallmatrix} r_{\ell\ell} & r_{\ell,\ell+1} \\ 0 & r_{\ell+1,\ell+1} \end{smallmatrix}\right] U$ is upper triangular, $\widetilde{H} = U^T \left[\begin{smallmatrix} h_{\ell\ell} & h_{\ell,\ell+1} \\ h_{\ell+1,\ell} & h_{\ell+1,\ell+1} \end{smallmatrix}\right] V$ and $\widetilde{H}\widetilde{R} = \left[\begin{smallmatrix} a & b \\ c & a \end{smallmatrix}\right]$ with $cb < 0$ (i.e., $\widetilde{H}\widetilde{R}$ is in standard real Schur form). Then

$$x_{\ell\ell} = x_{\ell+1,\ell+1} = \sqrt{\tfrac{1}{2}(a + \sqrt{a^2 - bc})}, \quad x_{\ell+1,\ell} = \tfrac{1}{2x_{\ell\ell}} c, \quad x_{\ell,\ell+1} = \tfrac{1}{2x_{\ell\ell}} b.$$

The transformation matrices U and V have to be applied to rows and columns $\ell, \ell+1$ of H, R and Q_1 in order to complete this part of the computation. Now the other elements of X can be computed one superdiagonal at a time. Four different cases have to be distinguished depending on the indices of $x_{j\ell}$:

1.: the indices j and ℓ are such that neither h_{jj} nor $h_{\ell\ell}$ belong to a 2×2 diagonal block, then

$$x_{j\ell} = \frac{\sum_{m=\max(1,j-1)}^{\ell} h_{jm} r_{m\ell} - \sum_{m=j+1}^{\ell-1} x_{jm} x_{m\ell}}{x_{jj} + x_{\ell\ell}}.$$

2.: index j is such that h_{jj} does not belong to a 2×2 diagonal block, while index ℓ is such that $h_{\ell\ell}$ belongs to a 2×2 diagonal block (assume w.l.o.g. that the 2×2 block is given by $\left[\begin{smallmatrix} h_{\ell\ell} & h_{\ell,\ell+1} \\ h_{\ell+1,\ell} & h_{\ell+1,\ell+1} \end{smallmatrix}\right]$), then we obtain a 2×2 system of linear equations for $x_{j\ell}$ and $x_{j,\ell+1}$ which yields with $s_p = \sum_{m=j}^{p} h_{jm} r_{mp} - \sum_{m=j+1}^{\ell-1} x_{jm} x_{mp}$

$$x_{j,\ell+1} = \frac{s_{\ell+1}(x_{jj} + x_{\ell\ell}) - s_\ell x_{\ell,\ell+1}}{(x_{jj} + x_{\ell\ell})(x_{jj} + x_{\ell+1,\ell+1}) - x_{\ell,\ell+1} x_{\ell+1,\ell}}, \quad x_{j\ell} = \frac{s_\ell - x_{j,\ell+1} x_{\ell+1,\ell}}{x_{jj} + x_{\ell\ell}}.$$

3.: index ℓ is such that $h_{\ell\ell}$ does not belong to a 2×2 diagonal block, while index j is such that h_{jj} belongs to a 2×2 diagonal block (assume w.l.o.g. that the 2×2 block is given by $\left[\begin{smallmatrix} h_{j-1,j-1} & h_{j-1,j} \\ h_{j,j-1} & h_{jj} \end{smallmatrix}\right]$), then we obtain a 2×2 system of linear equations for $x_{j\ell}$ and $x_{j-1,\ell}$ which yields with $s_p = \sum_{m=\max(1,j-1)}^{\ell} h_{pm} r_{m\ell} - \sum_{m=j+1}^{\ell-1} x_{pm} x_{m\ell}$

$$x_{j\ell} = \frac{s_j(x_{j-1,j-1} + x_{\ell\ell}) - x_{j,j-1} s_{j-1}}{(x_{jj} + x_{\ell\ell})(x_{j-1,j-1} + x_{\ell\ell}) - x_{j,j-1} x_{j-1,j}}, \quad x_{j-1,\ell} = \frac{s_{j-1} - x_{j-1,j} x_{j\ell}}{x_{j-1,j-1} + x_{\ell\ell}}.$$

4.: both indices j and ℓ are such that h_{jj} and $h_{\ell\ell}$ belong to 2×2 diagonal blocks (assume as before, that h_{jj} and $h_{\ell\ell}$ are the $(2,2)$, resp. the $(1,1)$ element of the respective 2×2 block), then the 4 elements of $X_{j\ell} = \left[\begin{smallmatrix} x_{j-1,\ell} & x_{j-1,\ell+1} \\ x_{j,\ell} & x_{j,\ell+1} \end{smallmatrix}\right]$ can be computed via solving a 2×2 Sylvester equation

$$\begin{bmatrix} x_{j-1,j-1} & x_{j-1,j} \\ x_{j,j-1} & x_{jj} \end{bmatrix} X_{j\ell} + X_{j\ell} \begin{bmatrix} x_{\ell\ell} & x_{\ell,\ell+1} \\ x_{\ell+1,\ell} & x_{\ell+1,\ell+1} \end{bmatrix} = C_{j\ell},$$

where $C_{j\ell} = H_{j-1:j,1:\ell+1} R_{1:\ell+1,\ell:\ell+1} - X_{j-1:j,j+1:\ell-1} X_{j+1:\ell-1,\ell:\ell+1}$ (employing standard MATLAB notation).

In case the square root does not exist as in the example described in the introduction, the above algorithm fails due to a zero denominator. Numerical examples show that the approach presented here for computing the square root of $W = A_1 A_2$ gives better accuracy than when applying the method proposed in [2] to the product W. In case $A_1 A_2$ is already in the form quasi-upper triangular/triangular, both algorithms behave similarly as 1.–4. are adapted from [2]. Similar to the approach above algorithms for computing the kth root of a product of k matrices can be derived. These formulae are much more involved, e.g., for a 3rd root, 2×2 equations of the form $A^2 X + AXB + XB^2 = C$ have to be solved. Stability issues and existence of kth roots also need further discussion.

4. References

1 Aldhaheri, R.W.: Model order reduction via real Schur-form decomposition. Int. J. Control **53** (1991), 709–716.

2 Björck, Å. and Hammarling, S.: A Schur Method for the Square Root of a Matrix. Lin. Alg. Appl. **52/53** (1983), 127–140.

3 Bojanczyk, A., Golub, G.H. and Van Dooren, P.: The periodic Schur decomposition, algorithms and applications. In Proc. SPIE Conf., **1770** (1992), 31–42.

4 Hench, J.J. and Laub, A.J.: Numerical Solution of the Discrete-Time Periodic Riccati Equations. IEEE Trans. Aut. Cont. **39** (1994), 1197–1210.

Addresses: Heike Fassbender, Technische Universität München, Fakultät für Mathematik, 80290 München, FRG.
Peter Benner, Universität Bremen, FB 3, Zentrum für Technomathematik, 28334 Bremen, FRG.