The Normal Lattice—A Cascade Digital Filter Structure

CASPER W. BARNES, SENIOR MEMBER, IEEE, AND SHU HUNG LEUNG

Abstract—A normal lattice is defined as a cascade of two-input two-output second-order normal form digital filter modules. An analysis of this structure is presented along with an iterative algorithm for constructing the system eigenvectors, a scaling algorithm, and an explicit expression for roundoff noise gain. These expressions and algorithms provide an explicit design procedure for realizing arbitrary rational transfer functions for single-input single-output filters.

Comparisons with other digital filter structures show that the normal lattice has very low roundoff noise and low coefficient sensitivity in realizations of narrow-band filters. This structure is also well adapted to the realization of complex filters.

I. Introduction

In normal form realizations of digital filters, the basic building block for generating a conjugate pair of complex poles is the structure shown in Fig. 1. For the purposes of our discussion here

Manuscript received June 10, 1980; revised December 2, 1980 and July 2, 1981. This work was based on work supported by the National Science Foundation under Grant ENG 7818252 and by the Electronics Systems Group of Rockwell International Corporation under Contract AaWM-79070.

The authors are with the Electrical Engineering Department, University of California, Irvine, CA 92717.

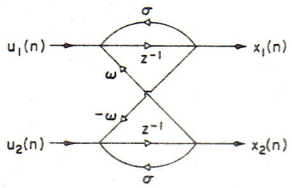


Fig. 1. The normal module.

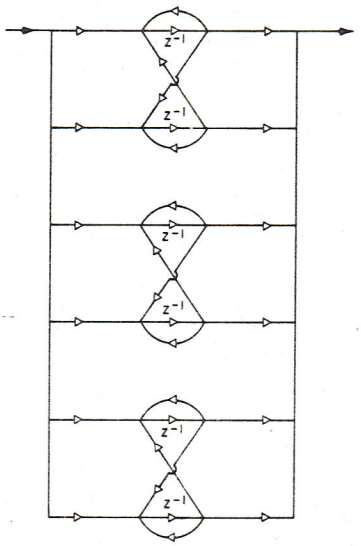


Fig. 2. Parallel structure of normal modules.

it will be convenient to regard this structure as two-input two-output module with state equation

$$\begin{pmatrix} x^1(n+1) \\ x^2(n+1) \end{pmatrix} = \begin{pmatrix} \sigma & \omega \\ -\omega & \sigma \end{pmatrix} \begin{pmatrix} x^1(n) \\ x^2(n) \end{pmatrix} + \begin{pmatrix} u_1(n) \\ u_2(n) \end{pmatrix} \tag{1}$$

and with matrix transfer function

$$H(z) = \frac{1}{(z-\sigma)^2+\omega^2} \begin{pmatrix} z-\sigma & \omega \\ -\omega & z-\sigma \end{pmatrix}. \tag{2}$$

As is well known, this structure will not support overflow limit cycles [1], has a uniform grid of possible pole locations for fixed-point realizations [2], and exhibits near optimal roundoff noise performance [3]–[6].

It has also been observed that this is a natural structure for realizing single complex poles in complex digital filters [7], with applications to complex demodulation. Thus if we interpret $u_1(n)$ and $u_2(n)$ as the real and imaginary parts of the input, and $x^1(n)$ and $x^2(n)$ as the real and imaginary parts of the state, then the complex state equation is

$$\begin{aligned} &\left[x^1(n+1)+ix^2(n+1)\right] \\ &\quad = (\sigma - i\omega)\left[x^1(n)+ix^2(n)\right] + \left[u_1(n)+iu_2(n)\right] \end{aligned} \tag{3}$$

and the complex transfer function is

$$H_c(z) = \frac{1}{z-(\sigma - i\omega)}. \tag{4}$$

These modules can be combined in various ways to realize either real or complex digital filters. For example, a parallel structure of normal modules is shown in Fig. 2.

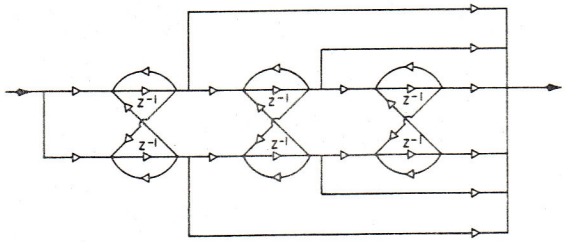


Fig. 3. Cascade structure of normal modules (normal lattice).

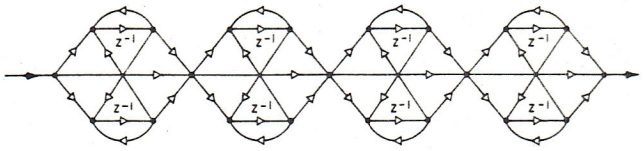


Fig. 4. Single-input single-output cascade of normal sections.

In this report we shall be concerned with single-input single-output filters realized as cascades of normal modules, as shown in Fig. 3. The distinctive feature of this cascade structure that distinguishes it from usual cascade realizations [8] is that *internally* it is a two-input two-output cascade, rather than a single-input single-output cascade. In the ensuing discussion, we shall refer to this structure as a normal lattice. It should be noted that scaled second-order normal filter sections require two input multipliers and two output multipliers and, therefore, naturally lend themselves to a two-input two-output cascade structure.

An alternative cascade structure of normal sections, proposed by Mullis and Roberts [9] and Jackson, Lindgren, and Kim [3], is the single-input single-output cascade shown in Fig. 4. We see that the normal lattice requires one fewer multiplier per stage than the single-input single-output cascade. An additional property of the normal lattice that simplifies design is that the optimal ordering of filter sections can be found by examining only pole pair orderings (rather than both pole pair and zero pair orderings, as required by single-input single-output cascades [8]) since the zeros of the filter are determined by the state variable tap weights feeding the output, as shown in Fig. 3.

In the following we shall provide a state variable description of the normal lattice, derive the design equations for realizing an arbitrary rational transfer function with simple poles, present an algorithm for scaling the structure, and derive an expression for the roundoff noise gain.

Finally, the roundoff noise performance of the normal lattice is compared with the noise performances of other filter structures. Over the range of parameters considered, the normal lattice exhibited the lowest noise gain. In addition, a specific narrow band filter design is used to illustrate the relatively low coefficient sensitivity of the normal lattice.

II. State Variable Description of the Normal Lattice

We consider a general cascade of N normal modules, as shown in Fig. 5. The 2×2 matrix transfer function for each normal module is

$$\boldsymbol{H}_n = (z\boldsymbol{I} - \boldsymbol{A}_n)^{-1} \tag{5}$$

where

$$\boldsymbol{A}_n = \begin{pmatrix} \sigma_n & \omega_n \\ -\omega_n & \sigma_n \end{pmatrix} \tag{6}$$

for $n=1,2,\cdots,N$. The normal modules are separated by 2×2 scaling matrices of the form

$$\boldsymbol{K}_n=\begin{pmatrix} K_n^{11} & K_n^{12} \\ K_n^{21} & K_n^{22} \end{pmatrix} \tag{7}$$

$n=2,3,\cdots,N$. For this general discussion we allow the scaling matrices to take the most general 2×2 form. As we shall later show, complete scaling can be accomplished with diagonal scaling matrices, as shown in Fig. 3. For complex filters the scaling multipliers would have the form: $K_n^{11}=K_n^{22}$ and $K_n^{21}=-K_n^{12}$.

The outputs of the normal modules (which are the outputs of the delay elements) form a convenient set of state variables with which to describe the behavior of the complete system in the standard form[1]

$$\begin{aligned} \boldsymbol{x}(n+1)&=\boldsymbol{A}\boldsymbol{x}(n)+\boldsymbol{b}u(n) \\ y(n)&=\boldsymbol{c}^{\dagger}\boldsymbol{x}(n) \end{aligned} \tag{8}$$

where

$$\boldsymbol{A}=\begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{K}_2 & \boldsymbol{A}_2 & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{K}_3 & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & & \boldsymbol{A}_{N-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{K}_N & \boldsymbol{A}_N \end{pmatrix} \tag{9}$$

$$\boldsymbol{b}=\operatorname{col}(\boldsymbol{b}_1,\boldsymbol{0},\boldsymbol{0},\cdots,\boldsymbol{0}) \tag{10}$$

$$\boldsymbol{c}=\operatorname{col}(\boldsymbol{c}_1,\boldsymbol{c}_2,\cdots,\boldsymbol{c}_N) \tag{11}$$

and

$$\boldsymbol{x}(n)=\operatorname{col}(\boldsymbol{x}_1(n),\boldsymbol{x}_2(n),\cdots,\boldsymbol{x}_N(n)) \tag{12}$$

where $\boldsymbol{A}_n$ and $\boldsymbol{K}_n$ are given by (6) and (7), respectively, and

$$\boldsymbol{b}_1=\operatorname{col}(b_1^1,b_1^2) \tag{13}$$

$$\boldsymbol{c}_k=\operatorname{col}(c_k^1,c_k^2) \tag{14}$$

$$\boldsymbol{x}_k(n)=\operatorname{col}(x_k^1(n),x_k^2(n)). \tag{15}$$

If this system has only simple poles, then the eigenvectors and reciprocal eigenvectors (left-handed eigenvectors) of the system A-matrix form a bi-orthonormal pair of bases for the state space. In the following, we shall use these eigenvector bases to formulate the design equations, and to derive a general expression for the roundoff noise. It is notationally convenient to group the eigenvectors and reciprocal eigenvectors in pairs, as follows:

$$\boldsymbol{A}\boldsymbol{\phi}_n^k=\lambda_n^k\boldsymbol{\phi}_n^k, \qquad n=1,2,\cdots, \quad N;\ k=1,2 \tag{16}$$

and

$$\boldsymbol{\psi}_n^{k\dagger}\boldsymbol{A}=\lambda_n^k\boldsymbol{\psi}_n^{k\dagger}, \qquad n=1,2,\cdots,N;\ k=1,2 \tag{17}$$

where

$$\lambda_n^1=\sigma_n+i\omega_n, \qquad n=1,2,\cdots,N \tag{18}$$

$$\lambda_n^2=\sigma_n-i\omega_n, \qquad n=1,2,\cdots,N \tag{19}$$

and where the eigenvectors and reciprocal eigenvectors are normalized so that

$$\boldsymbol{\psi}_n^{k\dagger}\boldsymbol{\phi}_m^l=\delta_{nm}\delta_{kl}. \tag{20}$$

In the preceding equations $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are column matrices of length $2N$. In Appendix A we describe a recursive procedure for constructing the eigenvectors and reciprocal eigenvectors for a normal cascade of any length; Fortran code is provided for the algorithm.

We can now expand the system matrix parameters on these bases as follows:

$$\boldsymbol{A}=\sum_{n=1}^{N}\sum_{k=1}^{2}\lambda_n^k\boldsymbol{\phi}_n^k\boldsymbol{\psi}_n^{k\dagger} \tag{21}$$

$$\boldsymbol{b}=\sum_{n=1}^{N}\sum_{k=1}^{2}\beta_n^k\boldsymbol{\phi}_n^k \tag{22}$$

and

$$\boldsymbol{c}^{\dagger}=\sum_{n=1}^{N}\sum_{k=1}^{2}\gamma_n^k\boldsymbol{\psi}_n^{k\dagger} \tag{23}$$

where

$$\beta_n^k=\boldsymbol{\psi}_n^{k\dagger}\boldsymbol{b} \tag{24}$$

and

$$\gamma_n^k=\boldsymbol{c}^{\dagger}\boldsymbol{\phi}_n^k. \tag{25}$$

The transfer function of the normal lattice is then given by

$$\begin{aligned} H(z)&=\boldsymbol{c}^{\dagger}(z\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{b} \\ &=\sum_{n=1}^{N}\sum_{k=1}^{2}\frac{\alpha_n^k}{z-\lambda_n^k} \end{aligned} \tag{26}$$

where

$$\alpha_n^k=\beta_n^k\gamma_n^k. \tag{27}$$

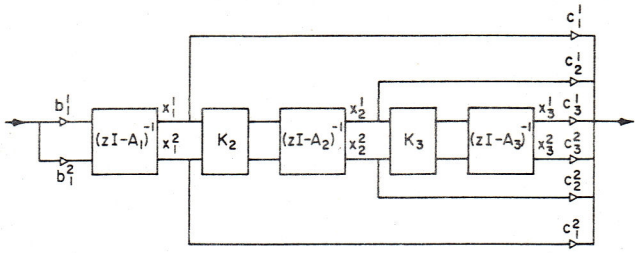


Fig. 5. Block diagram for normal lattice.

Scaling

Scaling of the normal lattice can be accomplished by the following sequential procedure (refer to Fig. 5). We first choose b_1^1 and b_1^2 to scale the dynamic range of the state variable pair x_1^1 and x_1^2. Then we choose $\boldsymbol{K}_2$ to scale the dynamic range of x_2^1 and x_2^2, and so forth, proceeding sequentially to the last normal module in the lattice. The equations that describe the scaling relations are cumbersome, but the numerical process can be accomplished easily on a digital computer.

An algorithm for the L_2-scaling of a special class of normal lattices is described in Appendix B, and Fortran code for the algorithm is provided.

[1]Here and throughout the superscript dagger denotes matrix conjugate transpose.

Design Procedure

A normal lattice realization of a rational transfer function can be obtained as follows.

1) Expand the transfer function into partial fractions as in (26).

2) Choose $A_1, A_2, \cdots, A_N$, as in (6), to obtain the required poles; i.e., $\lambda_n = \sigma_n + i\omega_n$.

3) Choose $b_1, K_2, \cdots, K_N$ to scale the dynamic ranges of the state variables (as shown in Appendix B). A and b, as given by (9) and (10), are now determined.

4) Calculate the reciprocal eigenvectors of A (using the recursive procedure described in Appendix A).

5) Calculate $\{\beta_n^k\}$ and $\{\gamma_n^k\}$, from (24) and (27), respectively.

6) Calculate $c^\dagger$ from (23).

III. Noise Gain

The matrix transfer function from the state variable nodes to output is given by

$$F(z) = z^{-1}c^\dagger(zI - A)^{-1} = z^{-1}\sum_{n=1}^{N}\sum_{k=1}^{2}\frac{\gamma_n^k}{z-\lambda_n^k}\psi_n^{k\dagger}. \tag{28}$$

If we model the roundoff errors at each summing node as uncorrelated white noise, each with variance σ_0^2, then the variance of the roundoff error at the filter output is given by

$$\sigma_y^2 = (1+\mu)\sigma_0^2 \tag{29}$$

where

$$\mu = \frac{1}{2\pi i}\oint\frac{dz}{z}F(z)F^\dagger(1/z^*) = \sum_{n,m=1}^{N}\sum_{k,l=1}^{2}\frac{\gamma_n^k\gamma_m^{l*}}{1-\lambda_n^k\lambda_m^{l*}}\psi_n^{k\dagger}\psi_m^l. \tag{30}$$

Equation (30) is suitable for numerical evaluation of the unit noise gain μ, since values for all of the required quantities can be computed from the transfer function.

To determine the *minimum* noise gain, (30) must be evaluated for all possible pole-pair sequences. For each new pole sequence, the filter must be rescaled, and the reciprocal eigenvectors recomputed.

IV. Numerical Comparisons

To illustrate some of the properties of the normal lattice structure, and compare its performance with other structures, we consider the design of an eighth-order elliptic low-pass filter with a 0.1-dB passband ripple, and −72-dB stopband gain. In Fig. 6 we plot the filter roundoff noise gain as a function of passband edge frequency for five different filter realizations—the direct form parallel [8], normal form parallel [4], the normalized orthogonal polynomial structure[2] described by Gray and Markel [10]–[12], block optimal parallel [9], and the normal lattice structure described in this paper. In all cases the filter was scaled for an L_2 norm of unity for the transfer functions from the input to each internal summing node. The normal lattice was found to have the lowest noise gain over the range of filter bandwidths from 0.005 to 0.45.

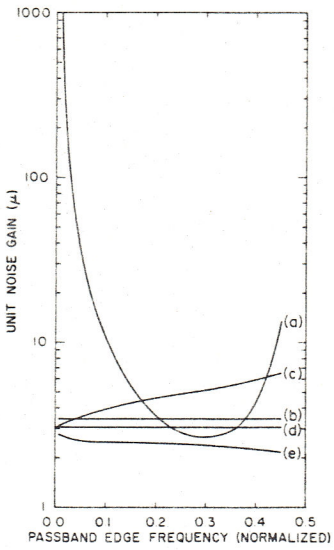


Fig. 6. Comparison of roundoff noise gains as a function of passband. Edge for five different realizations of the same eighth-order elliptic low-pass filter transfer function. Performance specifications: passband ripple = 0.1 dB; stopband ripple = −72 dB. (a) Direct form parallel realization. (b) Normal form parallel realization. (c) Normalized orthogonal polynomial structure. (d) Block-optimal parallel realization. (e) Normal lattice.

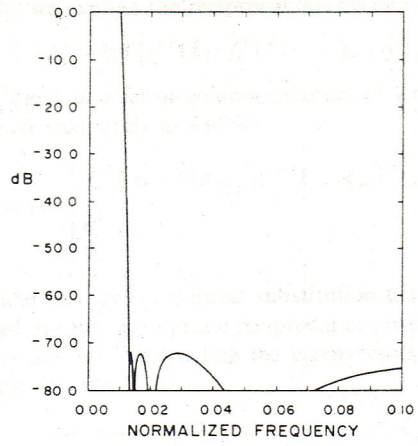
In addition we find that for very narrow band filters the multiplier coefficients have reasonable values (i.e., neither large nor small compared to unity), and that the filter response is relatively insensitive to coefficient quantization. To illustrate these properties we consider an eighth-order elliptic low-pass filter designed to meet the following performance specifications:

passband edge frequency = 0.01
stopband edge frequency = 0.013
passband ripple = 0.1 dB
stopband gain = −72 dB.

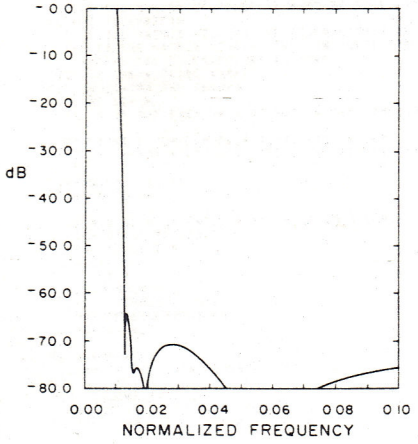
Following the design procedure given in Section II, we obtain the following matrix parameters for a normal lattice:

$$A = \begin{pmatrix}
0.97536457 & 0.016554894 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
-0.016554894 & 0.97536457 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
-0.0035905249 & 0.0 & 0.98241036 & 0.043118061 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.053246011 & -0.043118061 & 0.98241036 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & -0.026864306 & 0.0 & 0.99000867 & 0.057908279 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.046561198 & -0.057908279 & 0.99000867 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & -0.016569622 & 0.0 & 0.99555249 & 0.06397247 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.023718661 & -0.06397247 & 0.99555249
\end{pmatrix} \tag{31}$$

[2] The orthogonal polynomial filter noise gains were computed using a program written by T. Miyawaki.



(a)



(b)

Fig. 7. Magnitude of transfer function of eighth-order low-pass filter realized as a normal lattice. (a) Multiplier coefficients unrounded. (b) Multiplier coefficients rounded to sixteen bits.

TABLE I
COMPARISON OF ROUNDOFF NOISE VARIANCE FOR TENTH-ORDER ELLIPTIC FILTER

REALIZATION	PARALLEL NORMAL	CASCADE NORMAL	NORMAL LATTICE
σ_0^2/σ_0^2 (dB) ROUNDING AFTER SUMMATION	7.69		7.16
σ^2/σ_0^2 (dB) ROUNDING BEFORE SUMMATION	14	16.9	13.7

$$\boldsymbol{b} = \text{col}\,(0.14743664, 0.27393837, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) \tag{32}$$

$$\boldsymbol{c} = \text{col}\,(0.18535528, -0.099731184, 0.034813461, -0.070817758, -0.06813572, -0.030977459, -0.033505959, 0.029857621) \tag{33}$$

This design was scaled for an L_2 norm of unity from the input to each of the state variable nodes. The computed noise gain from (30) was given by

$$\mu = 2.71.$$

Fig. 7 is a plot of the magnitude of the transfer function for unrounded multiplier coefficients, and with all multiplier coefficients rounded to 16 bits. For this example, we see that the effects of coefficient quantization are small.

In Table I we compare the noise performance of the normal lattice with the noise performance of a normal cascade (i.e., a single-input single-output cascade) for the tenth-order elliptic filter used by Jackson, Lindgren, and Kim [3], using their performance figure for the normal cascade. For this case, we see that the difference between the parallel normal and normal lattice realization is insignificant, whereas the normal lattice is about 3 dB better than the normal cascade.

V. Concluding Remarks

Second-order normal modules can be combined in parallel or cascade to realize arbitrary rational transfer functions. The distinctive feature of the second-order normal module is that it is a two-input two-output structure. This leads to a unique form of cascade structure that we call the normal lattice. We have presented here an analysis of the normal lattice, an iterative algorithm for constructing the eigenvectors and reciprocal eigenvectors of the system matrix, an algorithm for scaling the structure, and an explicit expression for the roundoff noise gain. These relations provide an explicit procedure for designing digital filters in normal lattice form.

Filters realized in normal lattice form possess all of the desirable properties associated with the normal form; viz., no overflow limit cycles, low roundoff noise, and low coefficient sensitivity. They require the same number of multipliers, adders, and delays as the parallel normal form.

The noise performance of the normal lattice has been compared with the noise performances of other filter structures for a specific group of a low-pass filter transfer functions. Over the range of parameters considered, the normal lattice exhibited the lowest noise gain. In addition, a specific narrow-band low-pass filter design, realized as a normal lattice, was used to illustrate the relative insensitivity of this structure to coefficient quantization and to illustrate that, as in other low-noise structures, the multiplier coefficients do not take on extreme values for narrow-band filters.

Appendix A

Recursive Computation of the Eigenvectors and Reciprocal Eigenvectors of A

We consider the system matrix given by (9)
Define

$$\boldsymbol{u}^1 = 2^{-1/2}\text{col}\,(1, i)$$

$$\boldsymbol{u}^2 = 2^{-1/2}\text{col}\,(1, -i).$$

We observe that

$$\boldsymbol{A}_n \boldsymbol{u}^k = \lambda_n^k \boldsymbol{u}^k$$

for $n = 1, 2, \cdots, N$ and $k = 1, 2$.

We now express the eigenvectors of A in the form

$$\phi_n^k = \text{col}\,\{e_n^k(1), e_n^k(2), \cdots, e_n^k(N)\}$$

where $\{e_n^k(m)\}$ is a set of column matrices of length 2 that can be computed recursively as follows:

$$e_n^k(m) = \begin{cases} \boldsymbol{0}, & m < n \\ \boldsymbol{u}^k, & m = n \\ (\lambda_n^k \boldsymbol{I} - \boldsymbol{A}_m)^{-1} \boldsymbol{K}_m e_n^k(m-1), & m > n. \end{cases}$$

The reader can verify by direct substitution that the vectors so constructed are the appropriate eigenvectors of $\boldsymbol{A}$.

Similarly, we express the reciprocal eigenvectors in the form

$$\psi_n^k = \text{col}\{f_n^k(1), f_n^k(2), \cdots, f_n^k(N)\}$$

where $\{f_n^k(m)\}$ is a set of column matrices of length 2 that can be computed recursively as follows:

$$f_n^{k\dagger}(m) = \begin{cases} f_n^{k\dagger}(m+1)K_{m+1}(\lambda_n^k I - A_m)^{-1}, & m < n \\ u^{k\dagger}, & m = n \\ 0, & m > n. \end{cases}$$

The reader can verify by direct substitution that the vectors so constructed are the appropriate reciprocal eigenvectors of A.

Fortran code for constructing the eigenvectors and reciprocal eigenvectors follows.

```
00100  C
00200  C       SUBROUTINE TO COMPUTE EIGENVECTORS AND RECIPROCAL EIGENVECTORS
00300  C
00400  C       INPUT PARAMETERS :
00500  C         LAMBDA IS THE ARRAY STORING THE COORDINATES OF THE POLES
00600  C         A IS THE A-MATRIX
00700  C         N IS THE NUMBER OF POLES (N MUST BE EVEN AND LESS THAN 21)
00800  C       OUTPUT PARAMETERS :
00900  C         T IS THE MATRIX STORING THE EIGENVECTORS AS COLUMNS
01000  C         TPI IS THE MATRIX STORING THE CONJUGATE TRANSPOSE OF THE
01100  C           RECIPROCAL EIGENVECTORS AS ROWS
01200  C       EIG1 IS THE WORKING ARRAY
01300  C       IER INDICATES THE ERRORS :
01400  C         0= NO ERROR
01500  C         1=  N IS NOT EVEN OR N IS NOT IN THE RANGE OF 2 TO 20
01600  C
01700          SUBROUTINE EIGEN(LAMBDA,A,T,TPI,EIG1,N,IER)
01800          COMPLEX LAMBDA(N),T(20,N),TPI(20,N),EIG1(N),POL,DET,DIF,
01900         1 F1,F2
02000          DIMENSION A(20,N)
02100  C         ERROR CHECK
02200          IER=0
02300          IF(N-N/2.GT.0.OR.(N.LT.2.AND.N.GT.20)) IER=1
02400          IF(IER.GT.0) RETURN
02500  C         INITIALIZATION
02600          DO 5 I=1,N
02700          DO 5 J=1,N
02800          T(I,J)=CMPLX(0.,0.)
02900  5       TPI(I,J)=CMPLX(0.,0.)
03000  C         COMPUTE EIGENVECTORS
03100          DO 50 I=1,N,2
03200          K=I
03300          POL=LAMBDA(K)
03400  C         COMPUTE THE I-TH EIGENVECTOR
03500          DO 10 J=1,N
03600  10      EIG1(J)=CMPLX(0.,0.)
03700  C         ASSIGN VALUE TO I-TH SECTION
03800          EIG1(K)=CMPLX(1.,0.)/SQRT(2.)
03900          K1=K+1
04000          EIG1(K1)=CMPLX(0.,1.)/SQRT(2.)
04100          IF(K1.EQ.N) GO TO 25
04200          K2=K+2
04300  C         COMPUTE RECURSIVELY FOR THOSE SECTIONS AFTER I-TH SECTION
04400          DO 20 L=K2,N,2
04500          SIG=A(L,L)
04600          OME=A(L,L+1)
04700          S1=A(L,L-2)
04800          S2=A(L+1,L-1)
04900          DIF=POL-SIG
05000          DET=DIF*DIF+OME*OME
05100          F1=EIG1(L-2)
05200          F2=EIG1(L-1)
05300          EIG1(L)=(DIF*S1*F1+F2*S2*OME)/DET
05400          L1=L+1
05500  20      EIG1(L1)=(-F1*OME*S1+F2*DIF*S2)/DET
05600  C         ASSIGN I-TH EIGENVECTOR TO THE I-TH COLUMN OF T-MATRIX
05700  25      DO 30 M=1,N
05800          T(M,K)=EIG1(M)
05900  30      T(M,K1)=CONJG(EIG1(M))
06000  50      CONTINUE
06100  C         COMPUTE THE CONJUGATE TRANSPOSE RECIPROCAL EIGENVECTOR
06200          DO 120 I=N-1,1,-2
06300          K=I
06400          POL=LAMBDA(K)
06500  C         COMPUTE THE I-TH CONJUGATE TRANSPOSE OF RECIPROCAL EIGENVECTOR
06600          DO 70   J=1,N
06700  70      EIG1(J)=CMPLX(0.,0.)
06800  C         ASSIGN VALUE TO I-TH SECTION
06900          EIG1(K)=CMPLX(1.,0.)/SQRT(2.)
07000          K1=K+1
07100          EIG1(K1)=CMPLX(0.,-1.)/SQRT(2.)
07200          IF(K1.EQ.2) GO TO 90
07300          K2=K-2
07400  C         COMPUTE RECURSIVELY FOR THOSE SECTIONS AFTER I-TH SECTION
07500          DO 80 L=K2,1,-2
07600          SIG=A(L,L)
07700          OME=A(L,L+1)
07800          S1=A(L+2,L)
07900          S2=A(L+3,L+1)
08000          DIF=POL-SIG
08100          DET=DIF*DIF+OME*OME
08200          F1=EIG1(L+2)
08300          F2=EIG1(L+3)
08400          EIG1(L)=(F1*S1*DIF-F2*S2*OME)/DET
08500          L1=L+1
08600  80      EIG1(L1)=(F1*S1*OME+F2*S2*DIF)/DET
08700  C         ASSIGN THE I-TH CONJUGATE TRANSPOSE OF RECIPROCAL EIGENVECTOR
08800  C           TO THE I-TH ROW OF TPI-MATRIX
08900  90      DO 100 M=1,N
09000          TPI(K,M)=EIG1(M)
09100  100     TPI(K1,M)=CONJG(EIG1(M))
09200  120     CONTINUE
09300          RETURN
09400          END
```

APPENDIX B

SCALING ALGORITHM

Each state variable summing node is L_2-scaled by choosing $b_1, K_2, K_3, \cdots, K_N$ such that for unit variance white noise input, the variance of each state variable is equal to be unity. The transfer function from the input to the mth-state variable pair is given by

$$G_m(z) = (zI - A_m)^{-1} K_m (zI - A_{m-1})^{-1} \cdot K_{m-1} \cdots (zI - A_2)^{-1} K_2 (zI - A_1)^{-1} b_1$$

where

$$b_1 = \begin{pmatrix} b_1^1 \\ b_1^2 \end{pmatrix}$$

and

$$K_m = \begin{pmatrix} b_m^1 & 0 \\ 0 & b_m^2 \end{pmatrix}, \qquad \text{for } m = 2, \cdots, N.$$

For unit variance white noise input, the covariance matrix of the mth-state variable pair is

$$\Lambda_m = \frac{1}{2\pi i} \oint_{\substack{\text{unit} \\ \text{circle}}} \frac{dz}{z} G_m(z) G_m^{\dagger}(z) = \sum_j \sum_k \Lambda_m^{jk}$$

where Λ_m^{jk} is the matrix residue of Λ_m at $z = \lambda_j^k$. This matrix residue can be expressed in the form

$$\Lambda_m^{jk} = H_m^{jk} K_m \Delta_{m-1}^{jk} K_m E_m^{jk}$$

where

$$H_m^{jk} = \begin{cases} (\lambda_j^k)^{-1} (\lambda_j^k I - A_m)^{-1}, & j \neq m \\ \left(z^{-1}(zI - A_m)^{-1}(z - \lambda_j^k) \right)\Big|_{z=\lambda_j^k}, & j = m \end{cases}$$

$$E_m^{jk} = \left((\lambda_j^k)^{-1} I - A_m^{\dagger} \right)^{-1}$$

and

$$\Delta_{m-1}^{jk} = \begin{cases} \left(G_{m-1}(z) G_{m-1}^{\dagger}(z)(z - \lambda_j^k) \right)\Big|_{z=\lambda_j^k}, & j \neq m \\ G_{m-1}(\lambda_j^k) G_{m-1}^{\dagger}(\lambda_j^k), & j = m. \end{cases}$$

The diagonal elements of the matrix residues are given by

$$\begin{aligned} (\Lambda_m^{jk})_{11} = {} & (H_m^{jk})_{11} (E_m^{jk})_{11} (\Delta_{m-1}^{jk})_{11} (b_m^1)^2 \\ & + (H_m^{jk})_{12} (E_m^{jk})_{11} (\Delta_{m-1}^{jk})_{21} b_m^1 b_m^2 \\ & + (H_m^{jk})_{11} (E_m^{jk})_{21} (\Delta_{m-1}^{jk})_{12} b_m^1 b_m^2 \\ & + (H_m^{jk})_{12} (E_m^{jk})_{21} (\Delta_{m-1}^{jk})_{22} (b_m^2)^2 \end{aligned}$$

and

$$\begin{aligned} (\Delta_m^{jk})_{22} = {} & (H_m^{jk})_{21} (E_m^{jk})_{12} (\Delta_{m-1}^{jk})_{11} (b_m^1)^2 \\ & + (H_m^{jk})_{22} (E_m^{jk})_{12} (\Delta_{m-1}^{jk})_{21} b_m^1 b_m^2 \\ & + (H_m^{jk})_{21} (E_m^{jk})_{22} (\Delta_{m-1}^{jk})_{12} b_m^1 b_m^2 \\ & + (H_m^{jk})_{22} (E_m^{jk})_{22} (\Delta_{m-1}^{jk})_{22} (b_m^2)^2. \end{aligned}$$

Summing up all of the residues and we obtain

$$\begin{aligned} (\Lambda_m)_{11} &= \sum_{j=1}^{m} \sum_{k=1}^{2} (\Lambda_m^{jk})_{11} \\ &= \epsilon_m^1 (b_m^1)^2 + \zeta_m^1 b_m^1 b_m^2 + \eta_m^1 (b_m^2)^2 \end{aligned}$$

$$(\Lambda_m)_{22} = \sum_{j=1}^{m}\sum_{k=1}^{2}(\Lambda_m^{jk})_{22}$$
$$= \epsilon_m^2(b_m^1)^2 + \zeta_m^2 b_m^1 b_m^2 + \eta_m^2(b_m^2)^2$$

where

$$\epsilon_m^1 = \sum_{j=1}^{m}\sum_{k=1}^{2}(H_m^{jk})_{11}(E_m^{jk})_{11}(\Delta_{m-1}^{jk})_{11}$$

$$\zeta_m^1 = \sum_{j=1}^{m}\sum_{k=1}^{2}(H_m^{jk})_{12}(E_m^{jk})_{11}(\Delta_{m-1}^{jk})_{21} + (H_m^{jk})_{11}(E_m^{jk})_{21}(\Delta_{m-1}^{jk})_{12}$$

$$\eta_m^1 = \sum_{j=1}^{m}\sum_{k=1}^{2}(H_m^{jk})_{12}(E_m^{jk})_{21}(\Delta_{m-1}^{jk})_{22}$$

$$\epsilon_m^2 = \sum_{j=1}^{m}\sum_{k=1}^{2}(H_m^{jk})_{21}(E_m^{jk})_{12}(\Delta_{m-1}^{jk})_{11}$$

$$\zeta_m^2 = \sum_{j=1}^{m}\sum_{k=1}^{2}(H_m^{jk})_{22}(E_m^{jk})_{12}(\Delta_{m-1}^{jk})_{21} + (H_m^{jk})_{21}(E_m^{jk})_{22}(\Delta_{m-1}^{jk})_{12}$$

$$\eta^2 m = \sum_{j=1}^{m}\sum_{k=1}^{2}(H_m^{jk})_{22}(E^{jk}m)_{22}(\delta^{jk}m-1)_{22}.$$

For L_2 scaling,

$$(\epsilon_m^1+\epsilon_m^2)(b_m^1)^2 + (\zeta_m^1+\zeta_m^2)b_m^1 b_m^2 + (\eta_m^1+\eta_m^2)(b_m^2)^2 = 2$$

$$(\epsilon_m^1-\epsilon_m^2)(b_m^1)^2 + (\zeta_m^1-\zeta_m^2)b_m^1 b_m^2 + (\eta_m^1-\eta_m^2)(b_m^2)^2 = 0.$$

We obtain

$$b_m^1 = f b_m^2$$

where

$$f = \frac{-(\zeta_m^1-\zeta_m^2) \pm \sqrt{(\zeta_m^1-\zeta_m^2) - 4(\epsilon_m^1-\epsilon_m^2)(\eta_m^1-\eta_m^2)}}{2(\epsilon_m^1-\epsilon_m^2)}$$

and thus

$$b_m^2 = \sqrt{\frac{2}{(\epsilon_m^1+\epsilon_m^2)f^2 + (\zeta_m^1+\zeta_m^2)f + (\eta_m^1+\eta_m^2)}}.$$

repeat the algorithm for every section of the normal lattice to generate all of the scaling multipliers.

Fortran code for the scaling algorithm follows.

```
C       SCALING ROUTINE IS TO DETERMINE THE SCALING MULTIPLIERS
C
C       INPUT PARAMETERS:
C         POLE IS THE ARRAY STORING THE COORDINATES OF THE POLES
C         N IS THE NUMBER OF POLES ( N MUST BE EVEN AND LESS THAN 21 )
C         W1, W2, W3 ARE 2 BY 2 WORKING ARRAYS
C       OUTPUT PARAMETERS :
C         B IS THE ARRAY STORING THE SCALING MULTIPLIERS (IN POLE ORDER)
C       IER INDICATES THE ERROR
C         0= NO ERROR
C         1= N IS NOT EVEN OR N IS NOT IN THE RANGE OF 2 TO 20
C       SUBROUTINE CALLED : PHI ( TO EVALUATE THE COVARIANCE MATRIX )
C
      SUBROUTINE SCALE(POLE,B,N,W1,W2,W3,IER)
      COMPLEX POLE(N),W1(2,2),W2(2,2),W3(2,2),D,DPI,
     1 ALPHA1,ALPHA2,BETA1,BETA2,GEMA1,GEMA2,A,C,E,G
      DIMENSION B(N)
C       ERROR CKECK
      IER=0
      IF(N-N/2.GT.0.OR.(N.LT.2.AND.N.GT.20)) IER=1
      IF(IER.GT.0) RETURN
      DO 200 L=1,N/2
C         INITIALIZATION
      L2=L*2
      L1=L2-1
      ALPHA1=CMPLX(0.,0.)
      ALPHA2=CMPLX(0.,0.)
      BETA1=CMPLX(0.,0.)
      BETA2=CMPLX(0.,0.)
      GEMA1=CMPLX(0.,0.)
      GEMA2=CMPLX(0.,0.)
C       START TO COMPUTE THE SCALING PARAMETERS FOR EACH SECTION
      DO 160 I=1,L2
      NI=I
      W1(1,1)=CMPLX(1.,0.)
      W1(1,2)=CMPLX(1.,0.)
      W1(2,1)=CMPLX(1.,0.)
      W1(2,2)=CMPLX(1.,0.)
      I1=L-1
      IF(I1) 100,100,30
C         COMPUTE THE (L-1)-TH COVARIANCE MATRIX AT NI-TH POLE
30    CALL PHI(POLE,B,NI,I1,N,W1,W2,W3)
C
100   IF(L-(NI+1)/2) 120,110,120
C         THE DENOMINATOR OF H-MATRIX WHEN I=L
110   D=(POLE(NI)-CONJG(POLE(NI)))*POLE(NI)
      GO TO 140
C         THE DENOMINATOR OF H-MATRIX WHEN I<>L
120   D=(POLE(NI)-POLE(L1))*(POLE(NI)-POLE(L2))
      DPI=(1/POLE(NI)-POLE(L1))*(1/POLE(NI)-POLE(L2))
      SIGMA=REAL(POLE(L1))
      OMEGA=AIMAG(POLE(L1))
C         COMPUTE THE COMPONENTS OF H-MATRIX AND E-MATRIX
      A=(POLE(NI)-CMPLX(SIGMA,0.))/D
      C=-CMPLX(OMEGA,0.)/D
      E=(CMPLX(1.,0.)/POLE(NI)-CMPLX(SIGMA,0.))/DPI
      G=CMPLX(OMEGA,0.)/DPI
C         SUM ALL THE RESIDUES
      ALPHA1=ALPHA1+A*E*W1(1,1)
      ALPHA2=ALPHA2-C*G*W1(1,1)
      BETA1=BETA1-C*E*W1(2,1)+A*G*W1(1,2)
      BETA2=BETA2-A*G*W1(2,1)+C*E*W1(1,2)
      GEMA1=GEMA1-C*G*W1(2,2)
      GEMA2=GEMA2+A*E*W1(2,2)
160   CONTINUE
C         COMPUTE SCALING MULTIPLIERS OF THE L-TH BLOCK
      P1=REAL(ALPHA1+ALPHA2)/2.
      P2=REAL(ALPHA1-ALPHA2)/2.
      Q1=REAL(BETA1+BETA2)/2.
      Q2=REAL(BETA1-BETA2)/2.
      R1=REAL(GEMA1+GEMA2)/2.
      R2=REAL(GEMA1-GEMA2)/2.
      FRACT=(SQRT(Q2**2-4.*P2*R2)-Q2)/2./P2
      CONST=P1*FRACT**2+Q1*FRACT+R1
      B(L2)=1/SQRT(CONST)
      B(L1)=FRACT*B(L2)
200   CONTINUE
      RETURN
      END
C
C
C       SUBROUTINE TO EVALUATE L-TH COVARIANCE MATRIX AT NI-TH POLE
C       TO FACILITATE THE SCALE ROUTINE
C       INPUT PARAMETERS :
C         POLE IS THE ARRAY STORING THE COORDINATES OF THE POLES
C         B IS THE ARRAY STORING THE SCALING MULTIPLIERS UP TO L-TH
C           SECTION
C         W1 IS THE 2 BY 2 MATRIX CONSISTING UNIT ELEMENTS
C         NI INDICATES THE POLE AT WHICH COVARIANCE MATRIX EVALUATED
C         L INDICATES L-TH COVARIANCE MATRIX
C       OUTPUT PARAMETERS :
C         W1 IS ALSO STORING THE EVALUATED VALUES OF THE L-TH COVARIANCE
C           MATRIX AT NI-TH POLE
C       W2, W3 ARE WORKING MATRIX OF 2 BY 2
C       SUBROUTINE CALLED : CGMPRD (COMPLEX MATRIX MULTIPLICATION)
C
      SUBROUTINE PHI(POLE,B,NI,L,N,W1,W2,W3)
      COMPLEX POLE(N),W1(2,2),W2(2,2),W3(2,2),D,DPI,S,O
      DIMENSION B(N)
C
      DO 100 K=1,L
      K2=K*2
      K1=K2-1
      IF(K-(NI+1)/2) 20,10,20
C         DENOMINATOR OF K-TH SECTION WHEN K=NI
10    D=(POLE(NI)-CONJG(POLE(NI)))*POLE(NI)
      GO TO 30
C         DENOMINATOR OF K-TH SECTION WHEN K<>NI
20    D=(POLE(NI)-POLE(K1))*(POLE(NI)-POLE(K2))
30    DPI=(1/POLE(NI)-POLE(K1))*(1/POLE(NI)-POLE(K2))
      S=(POLE(K1)+POLE(K2))/2.
      O=CMPLX(AIMAG(POLE(K1)),0.)
C         EVALUATE THE TRASFER FUNCTION TO K-TH SECTION AT NI-TH POLE
      W2(1,1)=(POLE(NI)-S)/D*B(K1)
      W2(1,2)=O/D*B(K2)
      W2(2,1)=-W2(1,2)*B(K1)/B(K2)
      W2(2,2)=W2(1,1)*B(K2)/B(K1)
C         2 BY 2 MATRIX MULTIPLICATION TO CONCATENATE THE K-TH SECTION
C           TO THE (K-1)-TH COVARIANCE MATRIX
      CALL CGMPRD(W2,W1,W3,2,2,2)
C         EVALUATE THE TRANSPOSE OF THE TRANSFER FUNCTION TO K-TH
C           SECTION AT NI-TH POLE
      W2(1,1)=(1/POLE(NI)-S)/DPI*B(K1)
      W2(2,1)=O/DPI*B(K2)
      W2(1,2)=-W2(2,1)*B(K1)/B(K2)
      W2(2,2)=W2(1,1)*B(K2)/B(K1)
C         2 BY 2 MULTIPLIATION TO EVALUATE THE K-TH COVARIANCE MATRIX
      CALL CGMPRD(W3,W2,W1,2,2,2)
100   CONTINUE
      RETURN
      END
C
C       SUBROUTINE CGMPRD IS TO MULTIPLY TWO GENERAL COMPLEX MATRIX
C       TO FORM A RESULTANT GENERAL COMPLEX MATRIX
C
C       INPUT PARAMETERS:
C         A IS THE NAME OF THE FIRST INPUT MATRIX
C         B IS THE NAME OF THE SECOND INPUT MATRIX
C       OUTPUT PARAMETERS:
C         C IS THE NAME OF THE OUTPUT MATRIX
C       N IS THE NUMBERS OF ROWS IN A
C       M IS THE NUMBER OF COLUMNS IN A AND ROWS IN B
C       L IS THE NUMBER OF COLUMNS IN B
C
C       REMARKS: NUMBER OF COLUMNS OF MATRIX A MUST BE EQUAL TO NUMBER
C         OF ROWS OF MATRIX B  AND N CANNOT BE GREATER THAN 20
C
      SUBROUTINE CGMPRD(A,B,R,N,M,L)
      COMPLEX A(20,M),B(20,L),R(20,L)
C
      DO 20 I=1,N
      DO 20 J=1,L
C         INITIALIZATION
      R(I,J)=CMPLX(0.,0.)
      DO 20 K=1,M
      R(I,J)=R(I,J)+A(I,K)*B(K,J)
20    CONTINUE
      RETURN
      END
C
```

REFERENCES

[1] C. W. Barnes and A. T. Fam, "Minimum norm recursive digital filters that are free of overflow limit cycles," *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 569–574, Oct. 1977.

[2] C. M. Rader and B. Gold, "Effects of parameter quantization on the poles of a digital filter," *Proc. IEEE*, vol. 55, pp. 688–689, May 1967.

[3] L. B. Jackson, A. G. Lindgren, and Y. Kim, "Optimal synthesis of second-order state-space structures for digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 149–153, Mar. 1979.

[4] C. W. Bârnes, "Roundoff noise and overflow in normal digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 154–159, Mar. 1979.
[5] W. L. Mills, C. T. Mullis, and R. A. Roberts, "Normal realizations of IIR digital filters," in *1979 IEEE Int. Conf. Acoustics, Speech, Signal Processing Rec.*, pp. 340–343, Apr. 1979.
[6] M. Arjmand and R. A. Roberts, "Reduced multiplier, low roundoff noise digital filters," in *1979 IEEE Int. Conf. Acoustics, Speech, Signal Processing Rec.*, pp. 344–346, Apr. 1979.
[7] T. H. Crystal and L. Erhman, "The design and application of digital filters with complex coefficients," *IEEE Trans. Audio Electroacoust.*, vol. AU-16, pp. 315–320, Sept. 1968.
[8] L. B. Jackson, "Roundoff noise analysis for fixed-point digital filters realized in cascade or parallel form," *IEEE Trans. Audio Electoacoust.*, vol. AU-18, pp. 107–122, June 1970.
[9] C. T. Mullis and R. A. Roberts, "Synthesis of minimum roundoff noise fixed point digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-23, Sept. 1976.
[10] A. H. Gray, Jr. and J. D. Markel, "A normalized digital filter structure," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-23, pp. 268–277, June 1975.
[11] J. D. Markel and A. H. Gray, Jr., "Roundoff noise characteristics of a class of orthogonal polynomial structures," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-23, pp. 473–485, Oct. 1975.
[12] J. D. Markel and A. H. Gray, Jr. "Fixed-point implementation algorithms for a class of orthogonal polynomial filter structures," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-23, pp. 486–494, Oct. 1975.

Conditions for Finiteness of a Constructive Algorithm for Determining Stability

JOHN N. TSITSIKLIS

Abstract—**In earlier papers [1], [2], a methodology for deciding on the stability of a system of nonlinear differential equations was proposed. This methodology reduced the problem to a test of boundedness of all finite products of a given finite set of matrices. This paper investigates further the issue of whether the algorithm will terminate in finitely many steps and obtains some new conditions, concentrating on the case where the system being tested is unstable.**

I. Introduction

Brayton and Tong have proposed in [1] and [2] a constructive algorithm for determining the stability properties of a nonlinear time-invariant differential equation. They show that under certain conditions the problem may be reduced to the study of the "stability of a finite set of matrices", a concept to be defined below. They then proposed an algorithm that solves the latter problem and which is equivalent to constructing a Lyapunov function for the original differential equation. The range of potential applications of this algorithm is quite broad. It can be used, for example, in the study of switching systems, meaning linear systems such that the A matrix may undergo sudden changes (either random or deterministic).

Given the importance of this algorithm, it is natural to ask under what conditions it will be constructive (i.e., it terminates after a finite number of steps) and is, therefore, implementable in a computer. Some sufficient conditions are given in [1] and [2]. In this paper we obtain more conditions for the finiteness of the algorithm.

The termination of the algorithm is very closely linked to the properties of subsets of $\mathcal{C}^n$ that are left invariant under multiplication by a set of matrices. Most of the results in this paper involve characterizations of such invariant subsets.

Definition 1: A set of n by n complex matrices $\mathcal{A}$ is stable if for every neighborhood of the origin $U \subseteq \mathcal{C}^n$, there exists another neighborhood of the origin V such that $MV \subseteq U, \forall M \in \mathcal{A}^*$, where $\mathcal{A}^*$ is the semigroup of matrices generated by $\mathcal{A}$ (i.e., the set of all finite products of matrices in $\mathcal{A}$).

If $W \subseteq \mathcal{C}^n$, let $\mathcal{H}[W]$ be the convex hull of W.

Theorem 1: Given a finite set $\mathcal{A} = \{M_0, M_1, \cdots, M_{m-1}\}$ of m distinct n by n complex matrices. Let $W_0 \subseteq \mathcal{C}^n$ be a bounded, convex and symmetric polyhedral neighborhood of the origin. Define, for $k > 0$

$$W_k = \mathcal{H}\left[\bigcup_{i=0}^{\infty} M_j^i W_{k-l}\right] \tag{1.1}$$

where $j \equiv (k-1) \pmod m$. Then $\mathcal{A}$ is stable if and only if $W^* \equiv \cup_{i=0}^{\infty} W_i$ is bounded.

Proof: See [1], p. 227.

Brayton and Tong's algorithm consists of constructing the set W^*. We should point out that if W is a convex polyhedron it is uniquely determined by the finite set $E[W]$ of its extreme points. The algorithm is finite if and only if W^* can be constructed in a finite number of steps. In particular, a necessary condition for finiteness is that W^*, as well as $W_i, i = 1, 2 \cdots$ are convex polyhedra. Concerning the first step of the algorithm (equation (1.1)), the following result is proved in [1]:

Theorem 2: Let M be a matrix whose eigenvalues have magnitude $|\lambda(M)| < 1$; then, for any bounded neighborhood of the origin R, there exists some J such that $\cup_{i=0}^{\infty} M^i R = \cup_{i=0}^{J} M^i R$.

In order to complete the discussion, we must consider what happens if M is stable but has eigenvalues with $|\lambda_i| = 1$. In that case, it is necessary for finiteness that M has no eigenvalue equal to $e^{i\theta} (\theta \in [0, 2\pi])$ with $\theta/2\pi$ irrational. However, this condition is not sufficient. (For an example, see [2, p. 1124].)

Concerning the second step of the algorithm, we have, when $\mathcal{A}$ is stable [2]:

Theorem 3: If a set $\mathcal{A}$ of matrices is stable and there exists a positive k such that $|\lambda_i(M)| \leq k < 1$ for all $M \in \mathcal{A}^*$, then the algorithm will terminate "stable" in a finite number of steps.

Similarly with the case of the first step, it is easy to prove the following necessary condition for finiteness: If there exists some $M \in \mathcal{A}^*$, and an eigenvalue λ of M such that $\lambda = \exp[i\theta]$ and $\theta/2\pi$ is irrational, then the algorithm is not finite.

Now define a set Θ as follows: $\theta \in \Theta$ if and only if $0 \leq \theta \leq 2\pi$ and $\exp[i\theta]$ is an eigenvalue of some $M \in \mathcal{A}^*$. Then, if the algorithm terminates, $\theta/2\pi$ has to be rational, $\forall \theta \in \Theta$. We also have the following result:

Proposition 1: If the algorithm is finite, then Θ is a finite set.

Proof: Suppose that $\theta \in \Theta$ and $\theta/2\pi$ is rational and has been written as an irreducible ratio n/d of two integers. It is then easy to show that W^* has at least d extreme points. On the other hand, finiteness of the algorithm means that W^* has finitely many extreme points. Because there are only finitely many rational numbers in $[0, 1]$ whose denominator is less or equal than a finite number, it follows that Θ is a finite set. ■

Manuscript received May 11, 1981; revised November 13, 1981, and December 21, 1981.

The author is with the Laboratory for Information and Decision Systems and the Department of Electrical Engineering and Computer Science, Massachussetts Institute of Technology, Cambridge, MA 02139.

0098-4094/82/0600-0400$00.75