Bidiagonal Factorization of Fourier Matrices and Systolic Algorithms for Computing Discrete Fourier Transforms

PAUL D. GADER, MEMBER, IEEE

Abstract—**An algorithm for factoring Fourier matrices into products of bidiagonal matrices is presented. These factorizations have the same structure for every n and make possible DFT computation via a sequence of local, regular computations. A parallel pipeline technique for computing sequences of k-point DFT's, for every $k \leq n$, on a systolic array is proposed.**

I. Introduction

LET $\omega_n = \exp[-2\pi i/n]$ where $i = \sqrt{-1}$. The Fourier matrix of order n is the $n \times n$ matrix F_n having ω_n^{ij} as the i, j entry, for $i, j = 0, 1, \cdots, n-1$. The discrete Fourier transform (DFT) is given by $\vec{x} \rightarrow (1/\sqrt{n})F_n\vec{x}$. Most, if not all, FFT algorithms can be expressed in terms of factorizations of Fourier matrices [3], [4]. In this paper, we present a novel numerical method for factoring F_n. This method is based on a theorem, proved in [1], that shows that F_n can be written as a product of $2n-2$ bidiagonal matrices. The bidiagonal factorizations of F_n have the same, highly regular structure for every n, and make DFT computation possible via a sequence of local computations. Due to the similarity of the factorizations for all block sizes and the regularity for each block size, a simple method can be used to compute sequences of k-point DFT's, for every $k \leq n$, on a triangular systolic or wave front array of n^2 processors, avoiding the high dependence on n and data reorganization overhead of FFT's. The theory behind the factorizations has been presented [1]. The purposes of this paper are to inform a wider audience of their existence, to provide an algorithm for computing them, and to suggest one application (a systolic algorithm for DFT computation).

II. Bidiagonal Factorization of F_n

The bidiagonal factorizations of F_n, for $n \geq 3$, are

$$F_n = PL_1L_2 \cdots L_{n-2}ABU_{n-2}U_{n-3} \cdots U_1 \quad (1)$$

where P is a permutation matrix, and $L_1, L_2, \cdots, L_{n-2}$, A and $U_1, U_2, \cdots, U_{n-2}, B$ are lower and upper bidiagonal matrices, respectively. The method used to obtain the factorization is called minimal variable oblique elimination (MVOE). MVOE can be applied to any square matrix but it is often unsuccessful. It always works for F_n. The idea is to factor an input matrix, M, into a product of lower and upper triangular matrices. The obliques, or bands, of L and U are successively zeroed out in a careful way. More precisely (recall that a matrix $A = (a_{ij})$ has lower bandwidth r if $a_{ij} = 0$ whenever $i > j + r$),

1) determine a permutation matrix P such that $PM = LU$ is an LU decomposition of PM; and

2) (MVOE) construct matrices $L_1^{-1}, L_2^{-1}, \cdots, L_{n-2}^{-1}$ such that for every $j \in [1, n-2]$, the matrix L_j is unit lower bidiagonal and the matrix $L_j^{-1}L_{j-1}^{1} \cdots L_1^{-1}L$ has lower bandwidth $n-j$. Similarly, construct matrices $U_1^{-1}, U_2^{-1}, \cdots, U_{n-2}^{-1}$ such that for every $j \in [1, n-2]$, the matrix U_j^t is a unit lower bidiagonal and the matrix $(U_j^t)^{-1}(U_{j-1}^t)^{-1} \cdots (U_1^t)^{-1}U^t$ has lower bandwidth $n-j$. Then

$$A = L_{n-2}^{-1}L_{n-3}^{-1} \cdots L_1^{-1}L \quad \text{and}$$

$$B = UU_1^{-1}U_2^{-1} \cdots U_{n-2}^{-1}$$

are lower and upper bidiagonal matrices, respectively. Thus,

$$M = P^{-1}L_1L_2 \cdots L_{n-2}ABU_{n-2}U_{n-3} \cdots U_1$$

is a bidiagonal factorization of M (see Fig. 1). The permutation matrix occurs because computing the LU decomposition in a numerically stable way may require Gaussian elimination (GE) with partial pivoting [2]. The

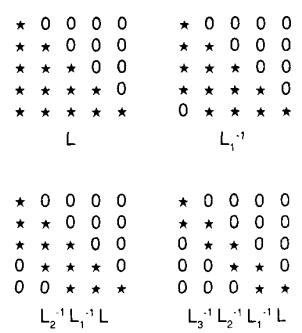


Fig. 1. Steps in oblique elimination of a lower triangular matrix L. The *'s represent possible nonzero entries.

Manuscript received August 29, 1987; revised October 13, 1988.

The author is with the Environmental Research Institute of Michigan, Ann Arbor, MI 48107.

IEEE Log Number 8928743.

0096-3518/89/0800-1280$01.00 © 1989 IEEE

L_i^{-1} are of the form

$$L_i^{-1} = \left[\begin{array}{c|cccccc} I_{n-i-1} & & & 0 & & & \\ \hline & 1 & 0 & & \cdots & & 0 \\ & x_{n-i} & 1 & 0 & \cdots & & \cdot \\ 0 & x_{n-i}x_{n-i+1} & x_{n-i+1} & 1 & \cdots & & \cdot \\ & \cdot & \cdot & \cdots & 1 & & \cdot \\ & \prod_{j=0}^{i-1} x_{n-i+j} & \prod_{j=1}^{i-1} x_{n-i+j} & \cdots & & x_{n-1} & 1 \end{array}\right]$$

and the L_i are of the form

$$L_i = \left[\begin{array}{ccc|cccccc} & I_{n-i} & & 0 & 0 & & & 0 \\ \hline 0 & \cdots & -x_{n-i}^{(i)} & 1 & 0 & & & 0 \\ & 0 & & -x_{n-i+1}^{(i)} & 1 & 0 & & 0 \\ & 0 & & 0 & -x_{n-i+2}^{(i)} & 1 & & \cdot \\ & \cdot & & \cdot & 0 & & & \cdot \\ & \cdot & & \cdot & \cdot & & & 0 \\ & 0 & & 0 & 0 & & -x_{n-1}^{(i)} & 1 \end{array}\right].$$

The U_i have the form of the transposes of L_i, $A = L_{n-1}$, and B is as described in the theorem. For example, by applying GE with partial pivoting and MVOE to F_5, we obtain $F_5 = PL_1L_2L_3ABU_3U_2U_1$ where (using $y_j^{(i)}$ to denote the jth element of the superdiagonal of U_i) the entries of L_1, L_2, and L_3 are

$$x_4^{(1)} = -1;$$

$$x_3^{(2)} = -1,\ x_4^{(2)} = 0.809017 - 0.587785i;$$

$$x_2^{(3)} = -1,\ x_3^{(3)} = 1.30902 + 0.951057i,\ x_4^{(3)} = 1.$$

The diagonal entries of A are all 1's. The subdiagonal entries are given by

$$a_{21} = -1, \quad a_{32} = 0.190983 + 0.587785i,$$
$$a_{43} = -0.809017 - 0.587785i,$$
$$a_{54} = -1.61803.$$

The diagonal entries of B are

$$b_{11} = 1, \quad b_{22} = -1.80902 - 0.587785i,$$
$$b_{33} = -0.690983 - 2.12663i,$$
$$b_{44} = -2.5 + 0.8123i,$$
$$b_{55} = 1.54508 - 4.75528i.$$

The superdiagonal elements of B are

$$b_{12} = 1, \quad b_{23} = 1.11803 + 1.53884i,$$
$$b_{34} = 1.80902 + 1.31433i,$$
$$b_{45} = 2.62866i.$$

The entries of U_1, U_2, U_3 are given by

$$y_4^{(3)} = +0.809017 - 0.587785i,$$
$$y_3^{(3)} = 0.809017 + 0.587785i,$$
$$y_2^{(3)} = -1;$$
$$y_4^{(2)} = 0.809017 + 0.587785i, \quad y_3^{(2)} = -1;$$
$$y_4^{(1)} = -1.$$

P is a permutation matrix representing the permutation $1 \to 1$, $2 \to 3$, $3 \to 4$, $4 \to 2$, and $5 \to 5$, i.e., $P = (p_{ij})$ where $p_{ij} = 1$, if $i \to j$ and $p_{ij} = 0$ otherwise. The 5-point DFT of $\vec{z} = (z_0, z_1, z_2, z_3, z_4)^t$ can be computed by the following algorithm.

Algorithm 1:

for $i = 1, n-2$; for $j = 1, i$ $(z_{4-j} := z_{4-j} + y_{4-j+1}^{(i)} z_{4-j})$

for $i = 0, n-1$ $(z_i := b_{i+1,i+1}z_i + b_{i+1,i+2}z_{i+1})$

where $b_{n,n+1} \equiv z_n \equiv 0$

for $i = n-1, 1$; for $j = 1, i$ $(z_j := z_j + x_j^{(i)} z_{j-1})$

permute $(t := z_1;\ z_1 := z_2;\ z_2 := z_3;\ z_3 := t)$.

Note that Algorithm 1 can be used for any n once the factorization (1) has been computed. Algorithm 1 is no more or less efficient than direct computation of DFT's with respect to number of floating-point operations. Any advantages will probably be due to the matching of the local, regular structure to processor arrays.

The algorithm for computing the factorization (1) consists of two steps given above. Step (1) is standard. Step (2) (MVOE) consists of constructing the matrices L_i^{-1} and U_i^{-1} and then computing the matrix multiplications $L_i^{-1}(L_{i-1}^{-1} \cdots L_1^{-1}L_0^{-1})$ and $(U_i^t)^{-1}[(U_{i-1}^t)^{-1} \cdots (U_0^t)^{-1}]$ where $L_0^{-1} \equiv L$ and $U_0^{-1} = U$. We describe how to do so for the lower triangular case, the upper triangular is similar. The matrix L_{n-1}^{-1} has $x_{n-1}^{(1)} = -1$. Let i be an integer between 2 and $n-2$ and assume that $A_i = L_{i-1}^{-1}L_{i-1}^{-1} \cdots L_1^{-1}L$ has the first $i-1$ obliques eliminated, that is, A_i is a lower triangular, banded matrix with the lower $i-1$ bands all 0.

The ith oblique of A_i is eliminated by constructing L_i^{-1} and computing $L_i^{-1}A_i$. The entries $x_{n-i+k}^{(i)}$, $k = 0, 1, \cdots, i-1$ of L_i^{-1} (and of L_i) are

$$x_{n-i+k}^{(i)} = -a_{n-i+k+1,k+1}^{(i)}/d_{n-i+k}$$

where

$$\begin{aligned} d_{n-i+k} &= \left(\prod_{j=0}^{k-1} x_{n-i+j}\right)a_{n-i,k+1} \\ &\quad + \left(\prod_{j=1}^{k-1} x_{n-i+j}\right)a_{n-i+1,k+1} \\ &\quad + \cdots + x_{n-i+k-1}a_{n-i+k-1,k+1} \\ &\quad + a_{n-i+k,k+1} \\ &= x_{n-i+k-1}\Big(x_{n-i+k-2}\big(\cdots\big)x_{n-i+1} \\ &\quad \cdot (x_{n-i}a_{n-i,k+1} \\ &\quad + a_{n-i+1,k+1}) \\ &\quad + a_{n-i+2,k+1} \\ &\quad + a_{n-i+k-1,k+1}\Big) + a_{n-i+k,k+1}. \end{aligned}$$

The algorithm for computing (1) is as follows.

Step 0: Input $Z_4^{(0)} \equiv Z_4$ to PE(1,1)
Shift $Z_4^{(1)} \equiv Z_4^{(0)}$ to PE(1,2) and PE(2,2)
Input $Z_3^{(1)} \equiv Z_2$ to PE(1,2)

For $i = 1,3$

Step $2i-1$: For $j = 2,1+i$
Execute $Z_{n-j}^{(1+i)} = Z_{n-j}^{(i)} + y_{n-j+1}^{(i)}Z_{n-j+1}^{(i)}$ in PE($2+i-j,1+i$)

Step $2i$: Input $Z_{1+i}^{(1+i)} \equiv Z_{i+1}$ to PE($1,2+i$)
For $j = 1, 1+i$
Shift $Z_{n-j}^{(1+i)}$ to PE($2+i-j,2+i$) and PE($3+i-j,2+i$)

Step 7: For $j = 1,5$
Execute $Z_{j-1}^{(5)} = b_{jj}Z_{j-1}^{(4)} + b_{j,j+1}Z_j^{(4)}$ in PE($j,5$)

For $i = 1,4$

Step $2i+6$: Output $Z_{i-1}^{(i+4)}$ from PE($1,4+i$)
Shift $Z_{i-1}^{(i+4)}$ to PE($1,5+i$)
Shift $Z_4^{(i+4)}$ to PE($n-i,5+i$)
For $j = i,3$ (not for $i = 4$)
Shift $Z_j^{(i+4)}$ to PE($1+j-i,5+i$) and PE($2+j-i,5+i$)

Step $2i+7$: For $j = i,4$
Execute $Z_j^{(5+i)} = Z_j^{(4+i)} + x_j^{(n-i)}Z_{j-1}^{(4+i)}$ in PE($1+j-i,5+i$)

Step 16: Output $Z_4^{(9)}$ from PE(1,9).

Algorithm 2—Computing Bidiagonal Factorization of F_n:

compute LU decomposition, $PF_n = LU$
$A_2 = L_1^{-1}L$
for $i = 2,n-2$
  compute entries of L_i^{-1}
    for $k = 0,1,\cdots,i-1$
      compute d_{n-i+k}
      compute $x_{n-i+k}^{(i)}$
  $A_{i+1} = L_i^{-1}A$
  compute entries of U_i^{-1} and B_i similarly
continue; $A = A_{n-1}$; $B = B_{n-1}$.

Observe that the bidiagonal factorizations need only be computed once for each n and can therefore be computed to a high degree of accuracy.

III. Systolic Computation of DFT's

One potential application of (1) is using it to compute sequences of DFT's, each of length $k \leq n$, in a parallel/pipeline fashion using a triangular array of n^2 simple processing elements (PE's) each having very little local memory and capable of computing $y := y + wx$ where x, y, and w are complex numbers and w is a weight stored in a local register of the PE. We describe the technique for one DFT of length $n = 5$ and then show how cases of length $k < n$ and sequences of DFT's can be handled. The processor array (together with the data flow which we explain now) is depicted in Fig. 2. The systolic algorithm for $n = 5$ with input data $\vec{Z} = (Z_0, Z_1, Z_2, Z_3, Z_4)^t$ is (letting $L_{n-1} \equiv A$ and letting $Z_i^{(k)}$ denote the value of the ith element of the input sequence after k steps; in the figure, we take $Z_{ik} \equiv Z_i^{(k)}$) as follows.

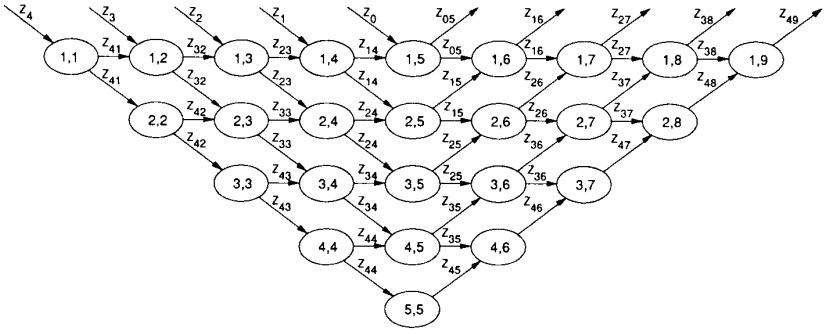


Fig. 2. Systolic array with data flow for computing 5-point DFT.

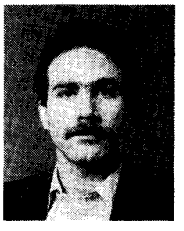
The output vector $(Z_0^{(5)}, Z_1^{(6)}, Z_2^{(7)}, Z_3^{(8)}, Z_4^{(9)})'$ must be permuted according to the permutation used to compute the LU decomposition.

Examination of the structure of the factorizations for general n as compared to a triangular array with n^2 processors shows that the systolic algorithm can be used for any n. DFT's of length $k \leq n$ can be computed by ignoring the outer layers of PE's, changing the weights stored in the PE's, and changing the column in which the diagonal shifts change directions. This can be accomplished by having two state bits in each PE—one to disable the PE and one to indicate the direction of the diagonal shift. Sequences of DFT's can be implemented by taking advantage of the fact that processing of an input vector proceeds column by column through the array. Thus, for example, if we wish to compute DFT's of the vectors $\vec{Z}_1, \vec{Z}_2, \vec{Z}_3, \cdots$, we can execute processing of $\vec{Z}_1$ in column 3 simultaneously with that of $\vec{Z}_2$ in column 2, horizontal and diagonal shifts can then each be performed concurrently, processing of $\vec{Z}_1$ in column 4, $\vec{Z}_2$ in column 3, and $\vec{Z}_1$ in column 2 can be executed concurrently, etc. The capability of each PE to perform data transfers concurrently with processing would result in greater speedup.

A theoretical comparison can be made to a systolic algorithm for DFT computation developed by Kung [5]. Define a cycle to be the time it takes a PE to perform its function. Kung's method (which we will refer to as the linear method) requires $2n - 1$ cycles to compute an n-point DFT as does the method presented in this paper (which we will refer to as the triangular method). Hence, since the linear method requires far fewer processors, the linear method is preferable if there is only a need for a single DFT. If we need to compute many DFT's at a high data rate, then we should use several independent linear arrays for the linear method. If we use n linear arrays to compute n DFT's in parallel, then it takes $2n - 1$ cycles, which is the same number of cycles and PE's as for the triangular method. If we want to compute more DFT's, the advantages of the triangular method become apparent. With the triangular method, we can compute $n + 1$ DFT's on n^2 processors using one more cycle than for n DFT's but it takes either n more processors or at least $2n - 1$ more cycles for the linear method. Furthermore, the triangular method will also be better if the DFT's should be computed in sequence, for example, if they are DFT's of rows of an image acquired by a line scanning device. Hence, if many DFT's need to be computed, it appears that the triangular method is preferable to the linear method.

References

[1] P. D. Gader, "Tridiagonal factorizations of Fourier matrices and applications to parallel computations of discrete Fourier transforms," *Linear Algebra Appl.*, vol. 145, pp. 169-211, Apr. 1988.
[2] G. H. Golub and C. F. van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 1983.
[3] B. N. Parlett, "Winograd's Fourier transform via circulants," *Linear Algebra Appl.*, vol. 45, pp. 137-155, 1982.
[4] D. J. Rose, "Matrix identities of the fast Fourier transform," *Linear Algebra Appl.*, vol. 29, pp. 423-443, 1980.
[5] H. T. Kung, "Special purpose devices for signal and image processing: An opportunity in VLSI," in *Proc. SPIE*, vol. 241: Real-Time Signal Processing III, Society of Photo-Optical Instrumentation Engineers, July 1980, pp. 77-84.



Paul D. Gader (M'87) was born in Jamestown, ND, on September 11, 1956. He received the B.S. degree in mathematics from the University of Central Florida in 1981 and the M.S. and Ph.D. degrees in mathematics from the University of Florida in 1983 and 1986, respectively.

Since graduating he has served as a Senior Research Scientist at Honeywell Systems and Research Center and as an Assistant Professor of Mathematics at the University of Wisconsin. He is currently a Research Engineer at the Environmental Research Institute of Michigan. His research interests include computer vision, image algebra, and applied linear algebra.

Correspondence

Conditions for the 2-D Characteristic Polynomial of a Matrix to be Very Strict Hurwitz

P. AGATHOKLIS

Abstract—**Conditions for the bivariate characteristic polynomial of a matrix to be very strict Hurwitz are presented. These conditions are based on the necessary and sufficient conditions for the existence of positive definite solutions to the 2-D continuous Lyapunov equation. It is shown that such an existence is only sufficient but not necessary for the characteristic polynomial to be very strict Hurwitz. Further, the testing of zeros at infinite distant points [6] requires the use of a class of very strict positive real functions.**

I. Introduction

In the stability analysis of 2-D continuous systems, it is required to test whether the 2-D characteristic polynomial of a matrix is Hurwitzian. Although this can be tested in the frequency domain using any of the existing tests [1], [6], [11], there is considerable interest in stability conditions involving the 2-D Lyapunov equation. This equation for continuous 2-D systems has been introduced in [2], while for the 2-D discrete case it has been studied in [3]–[5]. The question considered in this context is what are the necessary and sufficient conditions for the existence of positive definite solutions to the 2-D Lyapunov equation and how are these conditions related to stability. In [4], it has been shown that the existence of positive definite solutions to the discrete 2-D Lyapunov equation is a sufficient but not necessary condition for the 2-D characteristic polynomial of the system matrix to have no zeros inside the unit bidisc. These results for the discrete case prompted the question as to how the very strict Hurwitzian property is related to the conditions for the existence of positive definite solutions to the continuous 2-D Lyapunov equation. This question is being considered in this correspondence.

The correspondence starts in Section II with the formulation of the stability problem to be considered. Further, in Section II, the positive real [7], [8] and the strictly positive real properties [9] are considered and the very strict positive real functions are introduced in order to satisfy the requirements for testing zeros at the infinite distant points [6], [11]. These functions are then used to formulate a more strict version of the positive real lemma and to obtain the necessary and sufficient conditions for the existence of positive definite solutions to the continuous 2-D Lyapunov equation in Section III. These conditions are compared to the stability condition, and it is shown that they are sufficient but not necessary for the very strict Hurwitz property. Some interesting special cases are discussed in Section III.

Notation: $\overline{\overline{D}}_i$ is defined by

$$\overline{\overline{D}}_i = \{s_i \,|\, \mathrm{Re}\,[s_i] > 0,\ |s_i| < \infty\} \cup \{s_i \,|\, \mathrm{Re}\,[s_i] = 0,\ |s_i| < \infty\} \cup \{\infty\}$$

Manuscript received September 19, 1987; revised November 28, 1988. This work was supported by NSERC.

The author is with the Department of Electrical and Computer Engineering, University of Victoria, P.O. Box 1700, Victoria, B.C., Canada V8W 2Y2.

IEEE Log Number 8928741.

and the Cartesian product of $\overline{\overline{D}}_i$ is defined as

$$\overline{\overline{D}}^2 = \overline{\overline{D}}_1 \times \overline{\overline{D}}_2.$$

Further, s^* denotes the complex conjugate of s, $\lambda_i\{A\}$ denotes the eigenvalues of A, and Re $[s]$ is the real part of s. $W > 0$, where W is a matrix, stands for W symmetric and positive definite.

II. Preliminaries

In the stability analysis of 2-D continuous systems, it is required to consider the zeros of the 2-variable characteristic polynomial of a matrix A given by

$$C(s_1, s_2) = \det \begin{bmatrix} s_1 I - A_{11} & -A_{12} \\ -A_{21} & s_2 I - A_{22} \end{bmatrix}. \tag{1}$$

$C(s_1, s_2)$, the bivariate characteristic polynomial of A, is said to be very strict Hurwitz polynomial (VSHP) [10] iff

$$C(s_1, s_2) \neq 0 \quad \text{for } (s_1, s_2) \in \overline{\overline{D}}^2, \tag{2}$$

i.e., if it does not have any zeros in the closed right half of the biplane including infinite distant points. In this correspondence, conditions for a polynomial to be VSHP will be developed based on the Lyapunov equation. The extension of the 1-D Lyapunov equation to the two- and higher-dimensional case has been first studied in [2], and the continuous 2-D Lyapunov equation was introduced as

$$A^T W + WA = -Q \tag{3}$$

where W and Q are symmetric and W is block diagonal of appropriate dimensions.

Consider an $m \times m$ transfer matrix $H(s)$ of an m-input m-output system having entries which are rational functions in the complex variable s. The positive real and the strictly positive real properties are defined as follows.

Definition 1 [7]: A transfer matrix $H(s)$ is positive real iff

i) the elements of $H(s)$ are analytic for Re $(s) > 0$
ii) $H^*(s) = H^T(s^*)$
iii) $H^T(s^*) + H(s) \geq 0$ for all Re $[s] > 0$.

Definition 2 [8], [9]: A transfer matrix $H(s)$ is strictly positive real iff $H(s - \epsilon)$ is positive real for some real $\epsilon > 0$.

For the development of the necessary and sufficient conditions for the existence of positive definite matrices satisfying the 2-D Lyapunov equation, a more strict type of positive real matrix is needed. Let us call them very strict positive real functions and define them as follows.

Definition 3: A transfer matrix $H(s)$ is called very strictly positive real iff

i) $H(s)$ is strictly positive real
ii) $H^T(\infty) + H(\infty) > 0$.

It can be easily seen that the above definition implies the following lemma.

Lemma 1: A transfer matrix $H(s)$ is very strictly positive real iff

i) $H(s)$ is analytic in Re $[s] \geq 0$
ii) $H^*(s) = H^T(s^*)$
iii) $H^T(-j\omega) + H(j\omega) > 0$ for all $\omega \in (-\infty, \infty)$
iv) $H^T(\infty) + H(\infty) > 0$.

0096-3518/89/0800-1284$01.00 © 1989 IEEE