An Algorithm for the Machine Computation of Partial-Fractions Expansion of Functions Having Multiple Poles

SADASHIVA S. GODBOLE

Abstract—The partial-fractions expansion of a function $F(s)/(s-a)^m$, $m > 1$, involves the computation of m coefficients, namely $(1/i!)(d^iF(a)/ds^i)$, $0 \leq i \leq m-1$. Wehrhahn [1] and Karni [3] have provided a method for computing these coefficients algebraically. A new approach is taken here which involves approximating a multiple pole by neighboring simple poles. The theory developed turns out to have a very interesting resemblance to the FFT algorithm. The algorithm is illustrated by several examples. Typical applications are finding the inverse Laplace transform of a function having multiple poles and the evaluation of higher order derivatives of an arbitrary function $H(z)$ at some arbitrary $z = z_0$.

Index Terms—Discrete Fourier transform (DFT), fast Fourier transform (FFT), multiple pole, partial-fractions expansion.

Introduction

IT IS quite easy to perform machine computation of partial-fraction expansion (PFE) on functions having simple poles. Similar computations in the case of functions having multiple poles, e.g., $F(s)/(s-a)^m$ involve evaluation of m PFE coefficients, namely, $(1/i!)(d^iF(a)/ds^i)$, $0 \leq i \leq (m-1)$. Wehrhahn [1] provided a method for computing these coefficients for the special case $F(s)=1$. Karni [3] extended this method to arbitrary $F(s)$.

This paper describes a new algorithm for computing such PFE coefficients. The general theory is developed in the following section.

Theory

Consider a function $F(s)/(s-a)^m$, where $F(s)$ includes any other poles and a is complex in general. The PFE of this function with respect to the pole at a is given by

$$\frac{F(s)}{(s-a)^m} = \frac{F(a)}{(s-a)^m} + \frac{F^{(1)}(a)}{(s-a)^{m-1}} + \frac{F^{(2)}(a)/2!}{(s-a)^{m-2}} + \cdots + \frac{F^{(m-1)}(a)/(m-1)!}{(s-a)} \quad (1)$$

where $F^{(i)}(a)$ denotes $(d^iF(s)/ds^i)\big|_{s=a}$. It is desired to find expressions for the PFE coefficients, i.e., the numerators appearing on the right-hand side of (1).

Consider an auxiliary function $G(s)$ given by

$$G(s) = \frac{F(s)}{(s-a-\epsilon)(s-a-\epsilon r)(s-a-\epsilon r^2)\cdots(s-a-\epsilon r^{n-1})} \quad (2)$$

where ϵ is a positive number, $n \geq m$ and $r = e^{j2\pi/n}$. Let n_i, $1 \leq i \leq q$, be nontrivial integer prime factors of n so that

$$n = n_1 n_2 \cdots n_q. \quad (3)$$

The following identities concerning r will be used frequently in the subsequent development:

$$r^n = 1 \quad (4)$$

$$1 + r + r^2 + \cdots + r^{n-1} = 0 \quad (5)$$

$$1 + r^p + r^{2p} + \cdots + r^{(n-p)} = 0 \quad (6)$$

where p is a nontrivial integer factor (not necessarily prime) of n. For $n=m$, $G(s)$ approaches the original function, i.e., $F(s)/(s-a)^m$, in the limit as $\epsilon \to 0$ (see Fig. 1).

Let the residue of $G(s)$ at $s = a + \epsilon r^{i-1}$, $1 \leq i \leq n$, be denoted by R_i and let the numerator and denominator of R_i be denoted by N_i and D_i, respectively. Then R_i is given by

$$R_i = \frac{F(a+\epsilon r^{i-1})}{\epsilon^{n-1}(r^{i-1}-1)(r^{i-1}-r)\cdots(r^{i-1}-r^{i-2})(r^{i-1}-r^i)\cdots(r^{i-1}-r^{n-1})} = \frac{N_i}{D_i}. \quad (7a)$$

Only N_i plays the key role in the following development. The Taylor's series expansion of N_i about a will converge provided ϵ satisfies the inequality

Manuscript received September 24, 1970; revised January 13, 1971.
The author is with the Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Va. 24061.

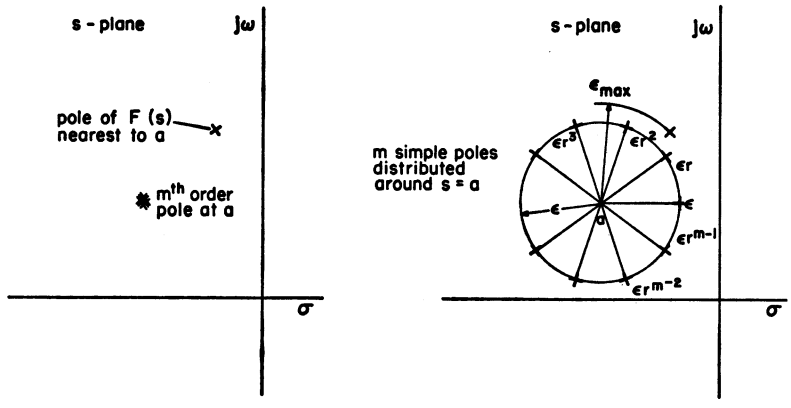

Fig. 1. s-plane representation of a multiple pole and its approximation.

$\epsilon <$ distance from $s=a$ to the nearest singularity of $F(s)$, say $\epsilon_{\max}$. (7b)

Moreover the series can be approximated by its first n terms if ϵ is chosen suitably. (More will be said about this choice in the next section.) Under these conditions, N_i may be expressed as

$$N_i = F(a + \epsilon r^{i-1}) \simeq F(a) + F^{(1)}(a)\epsilon r^{i-1} + \frac{F^{(2)}(a)}{2!}\epsilon^2 r^{2(i-1)} + \cdots + \frac{F^{(n-1)}(a)}{(n-1)!}\epsilon^{n-1} r^{(n-1)(i-1)}. \quad (8)$$

Expressing (8) in matrix form and using (4) yields

$$\boldsymbol{A}_n \boldsymbol{g} = \boldsymbol{N} \quad (9)$$

where

$$\boldsymbol{A}_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & r & r^2 & \cdots & r^{n-1} \\ 1 & r^2 & r^4 & \cdots & r^{n-2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & r^{n-1} & r^{n-2} & \cdots & r \end{bmatrix} n \times n$$

or simply $A_n(i,j) = r^{(i-1)(j-1)} \bmod n$

$$\boldsymbol{g} = \begin{bmatrix} F(a) \\ F^{(1)}(a)\epsilon \\ \frac{F^{(2)}(a)}{2!}\epsilon^2 \\ \vdots \\ \frac{F^{(n-1)}(a)}{(n-1)!}\epsilon^{n-1} \end{bmatrix}, \qquad \boldsymbol{N} = \begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_n \end{bmatrix}.$$

To solve (9) for $\boldsymbol{g}$ requires the knowledge of $\boldsymbol{A}_n^{-1}$. The Appendix shows the derivation of $\boldsymbol{A}_n^{-1}$ and some other interesting results. As proved there, $\boldsymbol{A}_n^{-1}$ is $(1/n)\boldsymbol{C}_n$ where $\boldsymbol{C}_n$ is given by

$$\boldsymbol{C}_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & r_1 & r_1^2 & \cdots & r_1^{n-1} \\ 1 & r_1^2 & r_1^4 & \cdots & r_1^{n-2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & r_1^{n-1} & r_1^{n-2} & \cdots & r_1 \end{bmatrix} \quad (10)$$

and $r_1 = 1/r = e^{-j2\pi/n}$. Identities (4)–(6) apply to r_1 also.

Solving (9) for $\boldsymbol{g}$ yields

$$\boldsymbol{g} = \frac{1}{n}\boldsymbol{C}_n \boldsymbol{N}. \quad (11)$$

It is very interesting to note the resemblance between (11) and the general equation describing the DFT (discrete Fourier transform) [2]

$$\begin{bmatrix} X^*(0) \\ X^*(2\pi/nT) \\ \vdots \\ X^*(2\pi(n-1)/nT) \end{bmatrix} = \boldsymbol{C}_n \cdot \begin{bmatrix} x(0) \\ x(T) \\ \vdots \\ x((n-1)T) \end{bmatrix}$$

or

$$\boldsymbol{X}^* = \boldsymbol{C}_n \boldsymbol{x}. \quad (12)$$

Thus $\boldsymbol{g}$ may be interpreted as the vector of DFT coefficients of a discrete time function described by $(1/n)\boldsymbol{N}$. The inverse DFT is given by

$$\boldsymbol{x} = \boldsymbol{C}_n^{-1}\boldsymbol{X}^* = \frac{1}{n}\boldsymbol{A}_n \boldsymbol{X}^*. \quad (13)$$

Another interesting fact about $\boldsymbol{C}_n$ is that for $n=3$, $\boldsymbol{C}_n^{-1}$ represents the matrix transformation used to resolve three unsymmetrical phase vectors into Fortescue's symmetrical components [5].

To perform the computation $\boldsymbol{C}_n\boldsymbol{N}$ in (11) fast, $\boldsymbol{C}_n$ may be split up into q factors [see (3)] as done in the FFT (fast Fourier transform) algorithm [2], [4]. Thus,

$$C_n = F_1F_2 \cdots F_q. \tag{14}$$

Incidentally, since A_n is of the same form as C_n, similar factoring may be used to compute the inverse FFT defined by (13). Substituting (14) into (11) yields

$$g = \frac{1}{n} F_1F_2 \cdots F_qN. \tag{15}$$

The desired PFE coefficients can now be readily obtained from the relation

$$\frac{F^{(i-1)}(a)}{(i-1)!} = \frac{g_i}{\epsilon^{i-1}}, \qquad 1 \le i \le m. \tag{16}$$

The algorithm based on the above theory is presented next with several examples.

ALGORITHM

The algorithm may now be stated as follows.

1) Choose a suitable n and ϵ. The criterion for doing this is discussed later in this section.
2) Compute $N_i = F(a+\epsilon r^{i-1})$, $1 \le i \le n$.
3) Form the product $F_1F_2 \ldots F_qN$ as in the FFT algorithm [2] and obtain g using (15).
4) Compute the desired PFE coefficients by using (16).

Scaling: While dealing with some functions, it may be necessary or desirable to scale them before applying the above algorithm. Suppose the given function, $F(s)/(s-a)^m$, is to be magnified (or reduced) by a factor M. Let the scaled s plane be called S plane such that

$$Ms = S. \tag{17}$$

Also let $F(s)$ have the following general form:

$$F(s) = \prod_{j=1}^{u} (s + z_j) \Big/ \prod_{j=1}^{v} (s + p_j). \tag{18}$$

The scaled $F(s)$ is denoted by $\Phi(S)$ and is given by

$$\Phi(S) = \left(\prod_{j=1}^{u} (S + Z_j) \Big/ \prod_{j=1}^{v} (S + P_j) \right) \cdot M^{(v-u)} \tag{19}$$

where $Z_j = Mz_j$ and $P_j = Mp_j$. Also a becomes A in the S plane.

Now, suppose the algorithm is applied to $\Phi(S)$ to compute $\Phi(A)$, $\Phi^{(1)}(A)$, . . . , $\Phi^{(n-1)}(A)/(n-1)!$. It can be shown that the corresponding coefficients in the s plane are given by

$$\frac{F^{(i)}(a)}{i!} = \frac{M^i\Phi^{(i)}(A)}{i!}, \qquad 0 \le i \le (n-1). \tag{20}$$

Example 1: Suppose the PFE of the following function is desired:

$$\frac{s+1}{(s+2)(s+4)^6}.$$

The PFE is given by

$$\frac{(s+1)}{(s+2)(s+4)^6} = \frac{-1/64}{(s+2)} + \frac{F(-4)}{(s+4)^6} + \frac{F^{(1)}(-4)}{(s+4)^5} + \cdots + \frac{F^{(5)}(-4)}{5!(s+4)}$$

where $F(s) = s + 1/s + 2$. The derivatives for this simple $F(s)$ can be readily calculated.

Here $m=6$ and if n is chosen equal to m, then $q=2$, $n_1=3$, and $n_2=2$. The corresponding factors are given by

$$F_1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & r_1 & r_1^2 \\ 1 & r_1^2 & r_1^4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & r_1^3 & 1 \\ 1 & r_1^4 & r_1^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & r_1^5 & r_1^4 \end{bmatrix}, \quad F_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & r_1^3 & 0 & 0 \\ 0 & 1 & 0 & 0 & r_1^3 & 0 \\ 0 & 0 & 1 & 0 & 0 & r_1^3 \end{bmatrix}.$$

The PFE coefficients were computed for several choices of n and ϵ. To compare the degree of accuracy achieved in each case, a cost function J was defined as follows:

$$J = \frac{100}{(m-1)} \sum_{i=2}^{m} l_i \tag{21}$$

where

$$l_i = \begin{cases} |(f_i - \hat{f}_i)/\hat{f}_i|, & |\hat{f}_i| \neq 0 \\ |f_i|, & |\hat{f}_i| = 0 \end{cases}$$

$$f_i = F^{(i-1)}(a)/(i-1)!$$

and $\hat{f}_i$ is the correct value of f_i. l_1 is not included in the summation on the right-hand side of (21) since $f_1 = F(a)$ can be computed accurately by direct substitution. This same cost function was used in the following examples also. For this example, $f_i \neq 0$ for any i and so J represents the average percent absolute variation from the correct coefficients.

Figs. 2, 3, and 4 show the results in graphical form. Fig. 2 shows that, for a given m, as n is varied from $n=m$ to higher values, the maximum accuracy (which corresponds to minimum J) tends to improve and the optimal value of ϵ tends to increase. Additional computations for $n=m=2$, 3, 4, 5 were also carried out and the results appear in Fig. 2. It can be seen therefrom that as m increases, the optimal ϵ corresponding to $n=m$ also increases. Fig. 3 shows the optimal combinations of ϵ and n while Fig. 4 shows ths improvement in the highest accuracy as n is increased. A sample of the computed coefficients is shown below along with the correct ones.

Coefficient	Computed Value, $n=12$, $\epsilon=0.8$ (i.e., 40 percent) Real	Imaginary	Correct Value
f_2	0.2500037E 00	−0.2270099E−06	0.25
f_3	0.1250017E 00	−0.3346941E−06	0.125
f_4	0.6250048E−01	−0.5513055E−06	0.0625
f_5	0.3125093E−01	−0.3880513E−06	0.03125
f_6	0.1562534E−01	−0.3031651E−06	0.015625

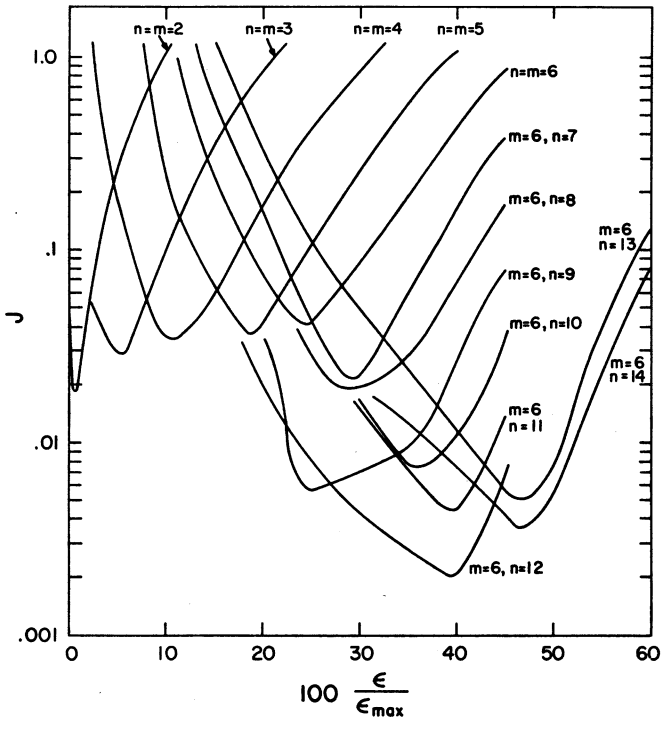

Fig. 2. Plot of cost function J versus percent ϵ. (Refer to Example 1.)

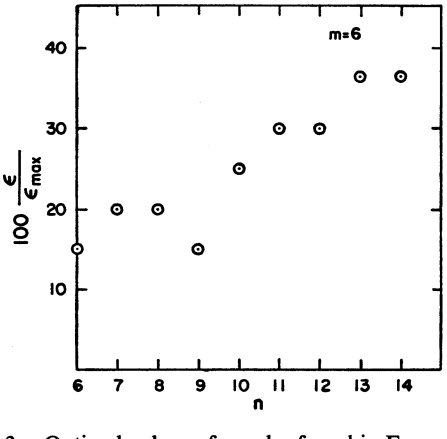

Fig. 3. Optimal values of ϵ and n found in Example 1.

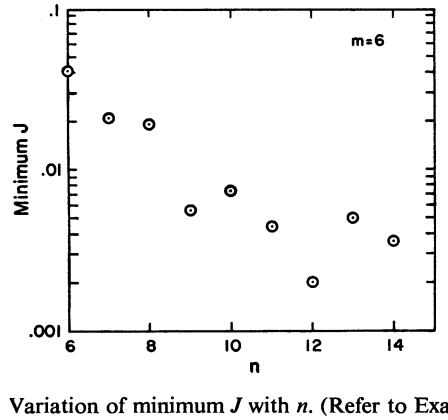

Fig. 4. Variation of minimum J with n. (Refer to Example 1.)

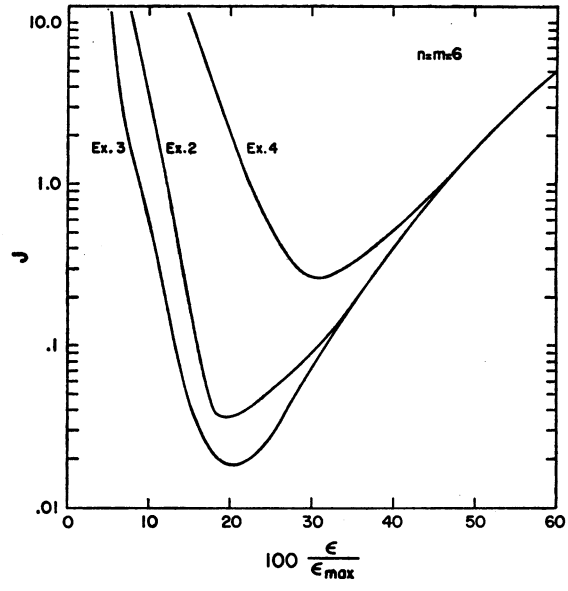

Fig. 5. Plot of J versus percent ϵ. (Refer to Examples 2, 3, and 4.)

Example 2: Assume we are given the following function:

$$\frac{s+1}{(s+2)(s+2.1)^6}.$$

In this case, n was chosen equal to m (i.e., 6) and various values of ϵ were tried. The results are plotted in Fig. 5. The highest accuracy occurred at $\epsilon=0.02$ (i.e., 20 percent).

Example 3: The function of the previous example was magnified by a factor of 100. The scaled-up $F(s)$ and a are given by

$$\Phi(S) = \frac{S+100}{S+200}, \qquad A = 210.$$

The results in this case were slightly better, in general, than those of the previous example and are shown in Fig. 5. The best accuracy occurred, however, at the same percent ϵ.

Example 4: Assume we are given the following function:

$$\frac{s+1}{(s+2)(s+102)^6}.$$

The result of computations for $n=m=6$ and various values of ϵ is shown in Fig. 5. The best accuracy occurred at $\epsilon=30$ (i.e., 30 percent).

Example 5: Assume we are given the following function:

$$\frac{(s+2+j8)(s+2-j8)(s+18+j8)(s+18-j8)}{(s+10)^6}.$$

In this case, ϵ_{max} given by (7b) is infinite. The PFE coefficients, $F^{(i)}(a)/i!$, $i \geq 5$ are all zero. The result of computations for $n=m=6$ and various values of ϵ is shown in Fig. 6. As expected (since (8) does not involve any approximation for the $F(s)$ of this example), there is a wide range of ϵ which gives good accuracy.

Choice of ϵ and n: The only approximation used while developing the theory in the preceding section was to truncate the Taylor's series in (8). Intuitively, therefore, it

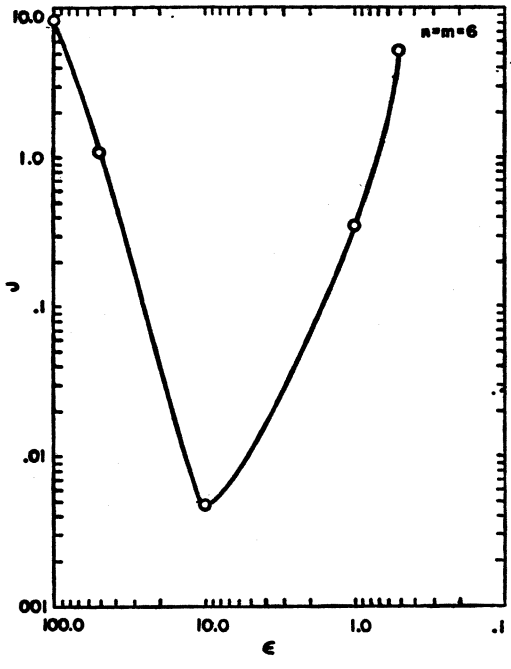

Fig. 6. Plot of J versus ϵ. (Refer to Example 5.)

would seem that computational accuracy should improve as ϵ is reduced and n increased. The results obtained in Example 1, however, indicate that a good choice of ϵ and n is as follows.

Case 1: ϵ_{max} given by (7b) is finite.

ϵ: Between 0.25 and 0.50 times ϵ_{max} commensurate with n.

n: $n>m+4$.

Case 2: $\epsilon_{max}=\infty$ (which implies $v=0$ [refer to (18)]). In this case, the coefficients f_i, $i>(u+1)$, must be all zero. The Taylor's series expansion of $F(a+\epsilon r^{i-1})$ thus has only $(u+1)$ nonzero terms at most. Equation (8), therefore, does not involve any approximation for $n=u+1$ and there is no advantage in letting n exceed $(u+1)$.

ϵ: Any reasonable value, say $\epsilon=1$.

n: $m+4<n\leq(u+1)$ if $u>m+4$, otherwise $n\leq u+1$.

The scaling technique described earlier should be used, when necessary, to ensure that $\epsilon^{n-1}>(1/L)$, L being the largest number that will not cause overflow in the computer.

Conclusions

The algorithm developed allows fast, efficient, and fairly accurate machine computation of PFE coefficients of functions having multiple poles. The accuracy of the results can be controlled by suitable choice of ϵ and n. Further investigation is necessary to substantiate/improve the guidelines described in the preceding section for choosing ϵ. The algorithm has several applications, typical ones being the following.

1) Inverse Laplace and Fourier transforms.
2) Inverse FFT.
3) Evaluation of derivatives of any order of an arbitrary function $H(z)$ at some arbitrary $z=z_0$.

The resemblance of this algorithm with the FFT algorithm is very interesting.

Appendix

Derivation of A_n^{-1}

Lemma 1: The sum of the elements of ith row of $\boldsymbol{A}_n$ and $\boldsymbol{C}_n$ (also ith column), $2\leq i\leq n$, is equal to zero.

Proof: The sum under consideration is the left-hand side of (5) or p times that of (6).

Theorem 1: $\boldsymbol{A}_n^{-1}=(1/n)\boldsymbol{C}_n$

Proof: Let $\boldsymbol{V}=\boldsymbol{A}_n\cdot(1/n)\boldsymbol{C}_n$. Using Lemma 1, it can be easily proved that

$$\begin{aligned} V(1,1) &= 1 \\ V(1,j) &= 0, \qquad 2\leq j\leq n \\ V(i,1) &= 0, \qquad 2\leq i\leq n. \end{aligned}$$

For $2\leq i\leq n$, $2\leq j\leq n$, the following relations hold:

$$\begin{aligned} V(i,j) &= \frac{1}{n}\sum_{k=1}^{n} A_n^{(i,k)}\, C_n^{(k,j)} \\ &= \frac{1}{n}\sum_{k=1}^{n} r^{(i-1)(k-1)}\, r_1^{(k-1)(j-1)} \\ &= \frac{1}{n}\sum_{k=1}^{n} r^{(k-1)(i-j)}. \end{aligned} \tag{22}$$

For $i=j$, (22) becomes

$$V(i,i) = 1.$$

For $i\neq j$, (22) may be rewritten as

$$V(i,j) = \frac{1}{n}\left[1 + r^{(i-j)} + r^{2(i-j)} + \cdots + r^{(n-1)(i-j)}\right]. \tag{23}$$

But $(i-j)$ must satisfy the relation

$$1\leq (i-j)\leq (n-2).$$

Hence the right-hand side of (23) is $1/n$ times the sum of the elements of mth row of $\boldsymbol{A}_n$, $2\leq m\leq n-1$. By Lemma 1, $V(i,j)=0$.

Thus $\boldsymbol{V}$ is an identity matrix. This and the fact that $\boldsymbol{A}_n$ and $\boldsymbol{C}_n$ are symmetrical assure that $\boldsymbol{A}_n^{-1}=(1/n)\boldsymbol{C}_n$.

Corollary 1: $\boldsymbol{C}_n^{-1}=(1/n)\boldsymbol{A}_n$.

Other Interesting Results

1) Given $\boldsymbol{A}_n$, $\boldsymbol{C}_n$ can be readily obtained by turning the last $(n-1)$ rows of $\boldsymbol{A}_n$ upside down. Expressed mathematically, this means

$$\boldsymbol{C}_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \boldsymbol{A}_n.$$

2) It can be proved that the demoninator of R_1, D_1, is given by

$$D_1 = \begin{cases} \epsilon^{n-1}(-1)^{(n-1)/2} r^{(3n^2-1)/4}(1-r)^2(1-r^2)^2 \cdots \\ \qquad \cdot(1-r^{(n-1)/2})^2, & n \text{ odd} \\ \epsilon^{n-1}(-1)^{(n-2)/2} r^{3n(n-2)/4}(1-r)^2(1-r^2)^2 \cdots \\ \qquad \cdot(1-r^{(n-1)/2})^2(1-r^{n/2}), & n \text{ even.} \end{cases}$$

3) The third interesting result is given by Theorem 2.

Theorem 2: $D_k = (1/r)D_{(k-1)}$, $2 \le k \le n$.

Proof:

$$\begin{aligned} D_k &= \epsilon^{n-1}(r^{k-1}-1)(r^{k-1}-r)\cdots(r^{k-1}-r^{k-2}) \\ &\quad \cdot(r^{k-1}-r^k)\cdots(r^{k-1}-r^{n-1}) \\ &= \epsilon^{n-1}(r^{k-1}-1)r(r^{k-2}-1)\cdots r(r^{k-2}-r^{k-3}) \\ &\quad \cdot r(r^{k-2}-r^{k-1})\cdots r(r^{k-2}-r^{n-2})\frac{(r^{k-2}-r^{n-1})}{(r^{k-2}-r^{n-1})} \\ &= \frac{(r^{k-1}-1)r^{n-2}}{(r^{k-2}-r^{n-1})} D_{(k-1)} \\ &= \frac{1}{r} D_{(k-1)}. \end{aligned}$$

Corollary 2: $D_k = (1/r^{k-1})D_1$.

References

[1] E. Wehrhahn, "On partial fraction expansion with high-order poles," *IEEE Trans. Circuit Theory* (Corresp.), vol. CT-14, Sept. 1967, pp. 346–347.

[2] J. A. Glassman, "A generalization of the fast Fourier transform," *IEEE Trans. Comput.*, vol. C-19, Feb. 1970, pp. 105–116.

[3] S. Karni, "Easy partial fraction expansion with multiple poles," *Proc. IEEE* (Lett.), vol. 57, Feb. 1969, pp. 231–232.

[4] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, Apr. 1965, pp. 297–301.

[5] C. L. Fortescue, "Method of symmetrical coordinates applied to the solution of polyphase networks," *AIEE Trans.*, vol. 37, 1918, pp. 1027–1140.

Checking Experiments for Sequential Machines

EDWARD P. HSIEH, MEMBER, IEEE

Abstract—Some new procedures for designing efficient checking experiments for sequential machines are described. These procedures are based on the use of four types of sequences introduced, namely, the compound DS, the resolving sequence (RS), the compound RS, and the simple I/O sequence. Significant reduction in the bound on the length of checking experiments is achieved. Along a parallel line of development, a new procedure, called the state counting method, is presented for detecting faults that can cause an increase in the number of states. For an n-state, m-input symbol machine, this procedure gives a bound on the length of checking experiments that is approximately $m^{\Delta n} \cdot n$ times the bound for conventional checking experiments designed strictly under the assumption that no faults increase the number of states, where $m > 1$ and Δn is the maximum anticipated increase in the number of states due to faults.

Index Terms—Bounds, checking experiments, fault detection, sequential machines, state-increasing faults, synchronous machines.

I. Introduction

ONE of the most studied and important theoretical aspects in the area of fault detection for sequential machines is concerned with the design of checking experiments [1]–[7]. In a checking experiment, the machine under test is considered as a black box with only input and output terminals accessible. By observing the output response to the input sequence constituting the checking experiment, one verifies whether the machine under test is operating in accordance with the given state table specification.

The earliest work on the fault detection problem was done by Moore [8]. His method requires the enumeration of all possible faulty machines, and yields an experiment that not only detects faults, but also identifies which fault has occurred. Along the same "machine identification" approach, Poage and McCluskey [9] gave a method for designing optimal experiments which checks against a small, selected set of faults. An efficient approach to the design of checking experiments, called the transition checking approach, was first introduced by Hennie [1]. His method does not require the enumeration of the faulty machines and yields most effective results 1) for machines that have a distinguishing sequence (an input sequence whose application makes it possible to determine the initial state of the machine by observing the output response), and 2) for machines that are reduced, strongly connected, and such that no malfunction causes an increase in the number of states.

For machines that do not have any distinguishing sequences (DS), Hennie's procedure yields very long experiments with a general bound on length being proportional to the factorial of the number of states. For this class of machines, a great deal of attention has been given to the

Manuscript received March 29, 1970; revised February 1, 1971. This work was supported in part by NSF Grant GK1341.

The author is with the IBM T. J. Watson Research Center, Yorktown Heights, N. Y. 10598.