These methods of reduction however have a disadvantage, namely, that of *stability preservation*, i.e., the reduced order model may be unstable even though the original high-order system is stable. To overcome this problem a number of alternative techniques have recently been proposed [1], [5], [6], [8]-[10], [12], [13]. All of these recently proposed methods were very casually extended to reducing the order of multivariable systems.

Our purpose in this note is to show that even though these methods do produce stable reduced order models for single-input/single-output systems, they do not necessarily produce reduced order models when applied to multivariable systems. This is illustrated by the following example where the "reduced order" model derived by a number of these methods have orders that are higher than the original system.

Example

Consider the two-input/two-output system described by the matrix transfer function

$$G(s) = \frac{\begin{bmatrix} 20 + 12s + s^2 & 20 + 40s + 11s^2 \\ 20 + 12s + s^2 & 80 + 122s + 33s^2 \end{bmatrix}}{s^3 + 13s^2 + 32s + 20}.$$

The order of a multivariable system is defined as the minimum number of state variables needed to represent the system in state-vector form. To determine the order of the system described by $G(s)$, the matrix transfer function is expanded into a partial fraction expansion, thus

$$[G(s)] = \frac{[K_1]}{(s+1)} + \frac{[K_2]}{(s+2)} + \frac{[K_3]}{(s+10)}$$

where

$$[K_1] = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}; \quad [K_2] = \begin{bmatrix} 0 & 2 \\ 0 & 4 \end{bmatrix}; \quad [K_3] = \begin{bmatrix} 0 & 10 \\ 0 & 30 \end{bmatrix}.$$

The order of the system is given by [7]

$$n = \text{rank } [K_1] + \text{rank } [K_2] + \text{rank } [K_3] = 1 + 1 + 1 = 3$$

hence the system is of order 3.

Applying the method of [9], [10] to $[G(s)]$, a "reduced" order model $[R_1(s)]$ is obtained where

$$[R_1(s)] = \frac{\begin{bmatrix} 1.5385 + 0.8047s & 1.5385 + 2.9586s \\ 1.5385 + 0.8047s & 6.1538 + 8.6036s \end{bmatrix}}{s^2 + 2.3432s + 1.5385}.$$

By expanding this transfer function into a partial fraction expansion, the order k of the supposed reduced model is easily shown to be 4. Thus the order of the reduced model is larger than the order of the original system.

Also if the method of [1], [8], [12] is applied to $[G(s)]$, a "reduced" order model $[R_2(s)]$ is obtained where

$$R_2(s) = \frac{\begin{bmatrix} 1.61616 + 0.9697s & 1.61616 + 3.232324s \\ 1.61616 + 0.9697s & 6.46464 + 9.53536s \end{bmatrix}}{s^2 + 2.58586s + 1.61616}.$$

Again the order k of the supposed reduced model is 4, which is larger than the order of the original system! The same result is obtained if the method of [5], [6] is used.

Comments

The above example illustrates clearly that methods that are developed for reducing single-input/single-output systems do not necessarily produce the desired results when applied to multivariable systems. In fact, whenever the residue matrices in the partial fraction expansion of $[G(s)]$ are not of full rank, then we can expect these methods of reduction to lead to reduced order models that have orders that are equal or even greater than the order of the original system.

References

[1] R. K. Appiah, "Linear model reduction using Hurwitz polynomial approximation," *Int. J. Control*, vol. 28, pp. 476-488, 1978.

[2] M. Bosley and F. Lees, "A survey of simple transfer function derivation from high-order state-variable models," *Automatica*, vol. 8, pp. 765-775, 1972.

[3] M. Calfe and M. Healey, "Continued-fraction model reduction techniques for multivariable systems," *Proc. Inst. Elec. Eng.*, vol. 121(5), pp. 393-395, 1974.

[4] C. F. Chen and L. S. Shieh, "An algebraic method for control system design," *Int. J. Control*, vol. 8, pp. 561-470, 1970.

[5] T. C. Chen, C. Y. Chang, and K. W. Han, "Model reduction using the stability-equation method and the continued-fraction method," *Int. J. Control*, vol. 32, pp. 81-94, 1980.

[6] T. C. Chen, C. Y. Chang, and K. W. Han, "Stable reduced-order Pade approximants using stability equation method," *Electron. Lett.*, vol. 16, pp. 345-346, 1980.

[7] E. G. Gilbert, "Controllability and observability in multivariable control systems," *SIAM J. Contr.*, vol. 1, pp. 128-151, 1963.

[8] M. F. Hutton and B. Friedland, "Routh approximations for reducing order of linear time-invariant systems," *IEEE Trans. Automat. Contr.*, vol. AC-20, pp. 320-337, 1975.

[9] J. Pal, "Stable reduced-order Pade approximants using the Routh-Hurwitz array," *Electron. Lett.*, vol. 15, pp. 225-226, 1979.

[10] J. Pal and L. M. Ray, "Stable Pade approximants to multivariable systems using a mixed method," *Proc. IEEE*, vol. 68, pp. 176-178, 1980.

[11] Y. Shamash, "Order reduction of linear systems by Pade approximation methods," Ph.D. dissertation, Imperial College of Science and Technology, Univ. London, London, England, 1973.

[12] ——, "Model reduction using the Routh stability criterion and the Pade approximation techniques," *Int. J. Contr.*, vol. 21, pp. 475-484, 1975.

[13] F. F. Shoji, R. R. Mohler, and T. C. Hsia, "Simplification of large linear systems using a two-step iterative method," *J. Franklin Inst.*, vol. 310, pp. 155-188, 1980.

An Iterative Algorithm for Testing Two-Asummability of Boolean Functions

AJIT PAL

***Abstract*—An iterative algorithm for testing two-asummability of any given Boolean function is described in this letter. The method is based on the decomposition of Boolean functions in terms of reduced functions, and it is suitable for machine computation.**

I. Introduction

Two-asummability is a necessary and sufficient condition for linear separability of Boolean functions up to 8 variables [1]. The testing of two-asummability is, in general, much less involved than other methods of testing two-asummability [2]. Moreover, the two-asummability has many useful applications in the synthesis of threshold logic networks [1]. This has motivated a number of workers to develop efficient methods for the testing of two-asummability [2]-[10]. This letter presents an iterative algorithm for testing two-asummability. The algorithm is based on the decomposition of Boolean functions in terms of reduced functions. This method can be advantageously used in the synthesis of logic functions and realization of sequential machines using threshold gates [9].

II. Formulation of the Problem

Consider a function F of n variables expressed in terms of reduced functions: $F = I_1J_1 + I_2J_2 + \cdots + I_mJ_m \cdots$ (1), where $m = 2^k$, k being the number of variables along which F has been expanded. Any $I_i (1 \leq i \leq m)$ is a vector of k variables and J_i is a function of the remaining variables. J_i's are called reduced functions of F.

The necessary and sufficient conditions of two-asummability of F in terms of the reduced functions can be expressed in terms of the following three theorems. Necessary conditions are expressed in Theorems 1 and 2, and a sufficient condition is expressed in Theorem 3.

Manuscript received March 7, 1980; revised April 27, 1981.

The author is with the Indian Statistical Institute, Electronics Unit, Calcutta 700035 India.

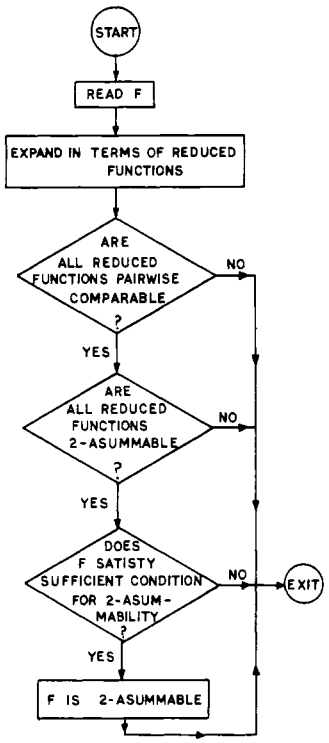

Fig. 1. Flowchart for testing two-asummability.

Theorem 1: A function is not two-asummable, if there exist two reduced functions J_i and J_j, $(1 \leqslant i, j \leqslant m)$ of F such that they are not comparable. The proof of this theorem follows from the unateness property of linearly separable functions [1].

Theorem 2: A function F is not two-asummable if any one of its reduced functions is not two-asummable.

Since the reduced functions of a threshold function are also threshold functions, the proof immediately follows.

Theorem 3: Let a function F be expressed in the form of (1). F is two-asummable iff, whenever $I_i \dot{+} I_j = I_k \dot{+} I_l$, $1 \leqslant i, j, k, l \leqslant m$, there does not exist four vectors t_o, t_p, t_q, and $t_r, t_o \in J_i, t_p \in J_j, t_q \in \bar{J}_k$ and $t_r \in \bar{J}_l$ such that $t_o \dot{+} t_p = t_q \dot{+} t_r$. Here $\dot{+}$ stands for vector sum [7]. This theorem is actually a restatement of the definition for two-asummability in terms of expression (1) and so the proof follows.

Based on the above discussion the two-asummability of a function can be tested by the following steps.

Step 1: Expansion of the given function in terms of reduced functions.

Step 2: Checking of comparability of the reduced functions.

Step 3: Testing of two-asummability of the reduced functions.

Step 4: Checking of the sufficient condition of two-asummability as stated in Theorem 3.

The flow diagram is shown in Fig. 1.

III. Algorithms

Though our method is valid up to 8 variables, a computer program has been developed to test two-asummability for functions up to 6 variables. In this section the actual algorithms followed in the program are discussed. Assuming a function is represented by the decimal values of its input vectors, any function of $n \geqslant 4$ is decomposed in terms of reduced functions of 3 variables.

Algorithm 1: Expansion of a n-variable function, $4 \leqslant n \leqslant 6$, in terms of reduced functions can be done by the following algorithm.

Step 1: Group the input vectors of F in 2^{n-3} blocks, the jth block, $1 \leqslant j \leqslant 2^{n-3}$, containing input vectors having decimal values from $8(j-1)$ to $(8j-1)$.

Step 2: Subtract $8(j-1)$ from each element in the jth block, $1 \leqslant j \leqslant 2^{n-3}$ to get the (J_j)th reduced function.

Example 1: Consider $F_1 = \Sigma(0, 1, 2, 3, 4, 5, 6, 9, 10, 13, 14, 15, 16, 20, 21, 22, 23, 28, 29, 30, 31)$. Following algorithm 1 we get:

Step 1: (0, 1, 2, 3, 4, 5, 6); (9, 10, 13, 14, 15); (16, 20, 21, 22, 23); (28, 29, 30, 31)

Step 2: $J_1 = \Sigma(0, 1, 2, 3, 4, 5, 6)$, $J_2 = \Sigma(1, 2, 5, 6, 7)$, $J_3 = \Sigma(0, 4, 5, 6, 7)$, and $J_4 = \Sigma(4, 5, 6, 7)$

Here $F_2 = I_1 J_1 + I_2 J_2 + I_3 J_3 + I_4 J_4$, where $I_1 = 0$, $I_2 = 1$, $I_3 = 2$ $I_4 = 3$.

Algorithm 2: Comparability of the reduced functions can be tested by the following algorithm.

Step 1: For any i and j, $1 \leqslant i, j \leqslant 2^{n-3}$ and $i \neq j$ perform step 2.

Step 2: Check whether there exist any elements $t_k, t_k \in J_i$ but $t_k \notin J_j$. If yes, perform step 3, otherwise, they are comparable.

Step 3: Check whether there exist any elements t_1, $t_1 \in J_j$ but $t_1 \notin J_i$. If yes, J_i and J_j are not comparable, otherwise they are comparable.

Example 2: Consider J_1 and J_2 of example 1. By step 2, we find that $3 \in J_1$, but $3 \notin J_2$ and by step 3 we find that $7 \in J_2$ but $7 \notin J_1$. So, J_1 and J_2 are not comparable and F_2 is not two-asummable.

Algorithm 3: To test two-asummability of the reduced functions, all possible two-asummable pairs are stored in the form of the following two arrays and used to check the existence of two-summable pairs.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 2 & 4 \\ 1 & 1 & 2 & 1 & 3 & 3 & 5 \\ \cdots & & 2 & & \cdots & & \\ \cdots & & 3 & & \cdots & & \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 5 & 6 & 7 & 7 & 7 & 7 \\ 2 & 4 & 4 & 6 & 5 & 6 & 6 \\ \cdots & & & 5 & & \cdots & \\ \cdots & & & 4 & & \cdots & \end{bmatrix}.$$

The elements of ith row and jth column of A and B are represented by a_{ij} and b_{ij}, respectively. Note that, excepting void elements, any pair (a_{ik}, b_{ik}) is two-summable with (a_{jk}, b_{jk}), $1 \leqslant i, j \leqslant 4$ and $1 \leqslant k \leqslant 7$. Now, two-asummability of any reduced function can be tested by the following steps.

Step 1: Start with $k = 1$, perform step 2.

Step 2: Check whether $a_{1k} \in J_i$ and $b_{1k} \in J_i$ for any l, $1 \leqslant l \leqslant 4$. If yes, perform step 3, otherwise perform step 4.

Step 3: Check whether $a_{nk} \in J_i$ or $b_{nk} \in J_i$ for all n, $l \neq n$, $1 \leqslant n \leqslant 4$. If yes, perform step 4, otherwise J_i is not two-asummable.

Step 4: If k is less than 7, increase k by 1 and perform step 2, otherwise J_i is two-asummable.

Example 3: Consider the reduced function J_2 of example 1. In step 2 we find $a_{21} \in J_2$ and $b_{21} \in J_2$ and in step 3 we find $a_{11} \notin J_2$ or $b_{11} \notin J_2$. So J_2 is not two-asummable.

Algorithm 4: The condition $I_i \dot{+} I_j = I_k \dot{+} I_l$ of Theorem 3 gives rise to the following possibilities: $i = j = k = 1$ *ii)* $i = k, j = 1, i \neq j$ and iii)

$i \neq j \neq k \neq 1$. The possibility (i) corresponds to the checking of two-asummability of the individual reduced functions which has been already tested.

Step 1: For $a\,I_i \dot{+} I_j = I_k \dot{+} I_1$, where $i \neq j$, $1 \leqslant i, j \leqslant m$, $1 \leqslant k \leqslant m-1$ and $(k+1) \leqslant 1 \leqslant m$ perform step 2.

Step 2: Check for the condition $a_{pr} \in J_i$, $b_{pr} \in J_i$, $a_{qr} \notin J_k$ and $b_{qr} \notin J_1$ where $1 \leqslant p, q \leqslant 4$ and $1 \leqslant r \leqslant 7$. If satisfied, F is not two-asummable. Otherwise perform step 3.

Step 3: Check whether all the values of i, j, k, and l have been exhausted. If yes, F is two-asummable, otherwise perform step 1 with other values of i, j, k, and l.

Example 4: Consider F_1 of example 1.

Step 1: With $i = k = 2$ and $j = l = 3$ perform step 2.

Step 2: We find $a_{14} \in J_2$, $b_{14} \in J_3$ and $a_{44} \notin J_3$ and $b_{44} \notin J_2$. So, F_1 is not two-asummable.

Conclusion: The iterative method discussed here is well suited for machine computation. A computer programme in Fortran IV has been written and tested up to 6 variables. (The program is available with the author.) It is extendable up to 8 variables, in which case the two-summable pairs for 4 variable functions have to be stored. It is difficult to compare the efficiency of this method with other existing methods, because none of them are computer oriented. If they are implemented on digital computer, all possible two-summable pairs have to be stored in the memory. For a n-variable function, the number of pairs to be so stored are

$$2^n \cdot (2^n - 1)/2 - n \cdot 2^{(n-1)} = 2^{n-1} \cdot (2^n - n - 1).$$

The number of pairs increases sharply with the number of variables. The number of pairs to be stored for $n = 6$ are 1824. But we need to store only 16 pairs (for $n = 3$), for testing two-asummability of 6-variable functions. Similarly, for $n = 8$, the number of pairs to be stored are only 88 (for $n = 4$), instead of 31 616. So, very efficient utilization of memory space is achieved.

Secondly, the time taken to test two-asummability varies from one function to another. If any given function does not pass through one of the necessary conditions stated in Theorems 1 and 2, then the testing time is very small. It has been found that most of the nonthreshold functions belong to this category. As the number of variables increases, the ratio of the threshold to nonthreshold functions decreases sharply. So, the average testing time over a large number of functions is rather small.

Acknowledgment

The author wishes to thank H. K. Worah for his programming assistance. Thanks are also due to the unknown reviewer for bringing this letter in its present form.

References

[1] S. Muroga, *Threshold Logic and Its Applications*. New York: Wiley-Interscience, 1971, 478 pp.

[2] J. F. Hopcroft and R. L. Mattson, "Synthesis of minimal threshold logic networks," *IEEE Trans. Electron. Comput.*, vol. EC-14, pp. 552-560, Aug. 1965.

[3] J. D. Bargainer and C. L. Coates, "Decimal numbers for checking summability," *IEEE Trans. Electron. Comput.*, vol. EC-15, p. 372, June 1966.

[4] P. K. Sinha Roy, "A slide rule device for checking 2-summability," *IEEE Trans. Comput.*, vol. C-17, pp. 279-283, Mar. 1968.

[5] R. O. Fontao, "A graphical method for checking complete monotonicity," *IEEE Trans. Comput.*, vol. C-20, pp. 461-464, Apr. 1971.

[6] S. Ghosh, S. Bhattacharyya, S. K. Mitra and A. K. Choudhury, "Simple methods for the testing of 2-summability of Boolean function and isobaricity of threshold functions," *IEEE Trans. Comput.*, vol. C-21, pp. 503-507, May 1972.

[7] Sureschander, "An algorithm for testing assummability of Boolean functions," *IEEE Trans. Comput.*, vol. C-23, pp. 188-191, Feb. 1974.

[8] P. De, A. Pal, A. Sen, D. Sarma, and A. K. Choudhury, "A tabular method for finding 2-summable and mutually 2-summable pairs of minterms," *Int. J. Electron.*, vol. 37, pp. 409-427, Mar. 1974.

[9] A. Pal, "Studies on the synthesis of digital logic circuits using threshold gates and MOS networks," Ph.D. dissertation, Calcutta University, Calcutta, India, 1975.

[10] A. K. Sarje and N. N. Biswas, "An algorithm for testing 2-assummability of Boolean functions," *IEEE Trans. Comput.*, vol. C-26, pp. 1049-1052, Oct. 1977.

Adaptive Prediction Applied to Seismic Event Detection

GREGORY A. CLARK AND PETER W. RODGERS

***Abstract*—Adaptive prediction was applied to the problem of detecting small seismic events in microseismic background noise. The Widrow-Hoff LMS adaptive filter [1], [2] used in a prediction configuration is compared with two standard seismic filters as an onset indicator. Examples demonstrate the technique's usefulness with both synthetic and actual seismic data.**

I. Introduction

Small seismic events, such as low magnitude earthquakes and low yield nuclear explosions are difficult to detect and locate because of corruption by additive ambient seismic background noise. This letter is concerned with detecting event onsets and their polarities, which can be done with varying degrees of success at the expense of distorting the event waveform [3], [4].

The microseismic noise is nonstationary, has a narrow-band spectrum centering around $\frac{1}{6}$ Hz, and is oscillatory in appearance. It is due primarily to small propagating surface and body waves generated by oceanic waves and atmospheric effects.

Typically, fixed-parameter bandpass filters are used to improve signal-to-noise ratio (SNR) before applying a detection algorithm [3], [4]. Because the noise is nonstationary, we used an adaptive predictor, which automatically chooses the correct passband and tracks the noise statistics. In the following experiments, we compare the SNR improvement ability of the adaptive predictor with that of computer simulations of two standard seismographic filters. These two filters, the WWSSN-SP (short period) and WWSSN-LP (long period), model the passbands of the World-Wide Standardized Seismograph Network [5].

II. The Adaptive Prediction Approach

Fig. 1 shows the seismic event detection scheme proposed in this letter. The low-pass filter removes high-frequency noise. The adaptive predictor is simply a noise canceller [2] with the reference noise equal to a delayed version of the input.

When no event is present, the canceller trys to predict the future value of the noise and force z_k toward zero, thus reducing the background noise entering the event detector.

When an event occurs, it represents data that is not correlated with the background noise. The canceller does not predict the event, so a larger output at this point in time indicates that an event has occurred. From another point of view, the event represents a sharp nonstationarity in data statistics, so the error (output of the canceller) becomes larger very quickly. Of course, the consequence of such processing is that the signal becomes distorted, but not until the time equal to the prediction distance (Δ) after the onset of the event. Because accurate prediction is more important than signal distortion for the event detection problem, we used a delay of one sample interval in all experiments.

Given the predictor size, the key to the success of the adaptive prediction technique is the choice of the convergence constant μ. It must be large enough so the predictor can track microseismic noise nonstationarities, but small enough to guarantee good prediction (small misadjustment [1]). Experiments show that both conditions can be met because

Manuscript received April 20, 1981. This work was supported by the U.S. Department of Energy by the Lawrence Livermore Laboratory under Contract W-7405-ENG-48.

G. A. Clark is with the Electronics Engineering Department, Lawrence Livermore National Laboratory, P.O. Box 808, L-156, Livermore, CA 94550.

P. W. Rodgers is with the Earth Sciences Division, Lawrence Livermore National Laboratory, P.O. Box 808, L-224, Livermore, CA 94550.