TITLE: **COMPUTATIONAL BALANCE IN REAL-TIME CYCLIC SPECTRAL ANALYSIS**

AUTHOR(S): **Randy S. Roberts, Herschel H. Loomis**

SUBMITTED TO: **International Acoustics, Speech and Signal Processing
Conference, Adelaide, Australia, April 19-22, 1994**

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

By acceptance of this article the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes.

The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.

Los Alamos

MASTER
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

COMPUTATIONAL BALANCE IN REAL-TIME CYCLIC SPECTRAL ANALYSIS*

Randy S. Roberts

Herschel H. Loomis, Jr.

Group ESA-6, Mail Stop J580
Los Alamos National Laboratory
Los Alamos, NM 87545 USA

Dept. of ECE, Code EC/Lm
Naval Postgraduate School
Monterey, CA 93943 USA

ABSTRACT

Real-time cyclic spectral analysis is useful in many applications, but is difficult to achieve because of its computational complexity. This paper studies the distribution of complex multipliers in multiprocessor cyclic spectrum analyzers, with the objective of obtaining computational balance. Computationally balanced implementations efficiently use hardware so that computational bottlenecks are reduced and a smooth flow of data between computational sections of the analyzer is maintained. Tables are presented that give the number of complex multipliers required in each section of the analyzer to obtain computational balance.

1 INTRODUCTION

Cyclic spectral analysis is a useful signal analysis tool in a variety of applications [1]. In applications such as signal detection, signal recognition and parameter estimation, estimation of the cyclic spectrum is often a fundamental component of the algorithm. Unfortunately, the computational complexity involved in cyclic spectral analysis often prohibits its use in many real-time applications. Generally speaking, cyclic spectral analysis is far more computationally complex than conventional spectral analysis, and real-time cyclic spectral analysis is far more difficult than real-time spectral analysis [2].

*This work was supported in part by a grant from ESL Inc. with partial matching support from the California State MICRO Program (PI: W. A. Gardner), in part by the U. S. Department of the Air Force (PI: H. H. Loomis, Jr.), and in part by the Los Alamos National Laboratory, which is operated by the University of California for the United States Department of Energy.

The key to real-time cyclic spectral analysis is parallel processing. The parallel structure of several computationally efficient algorithms has been determined [2]. Still, many implementation issues such as processor memory requirements, interprocessor communications, processor allocation and computational balance are unresolved. Furthermore, while signal flow graph descriptions of the algorithms are useful in designing multiprocessor architectures, direct implementation of the graphs leads to redundancy. Efficient implementations can be obtained by computationally balancing the multiprocessor architectures suggested by the graphs. By computational balance, we mean distributing the computational resources in such a way as to maintain a smooth flow of data in the implementation. Analysis of the computational balance leads to efficient distribution of arithmetic units as well as providing a means of analyzing implementation trade-offs.

Computational balance is achieved when each section of an implementation takes approximately the same amount of time to complete its computations. Data flows smoothly in a balanced implementation, and computational bottlenecks are minimized. In order to determine the number of arithmetic units needed to obtain computational balance, a performance metric is required. A useful metric for quantifying the notion of real-time performance is the factor-of-real-time, F_T [3]. It is defined as

$$F_T = \frac{T_{cmp}}{T_{col}} \quad (1)$$

where T_{cmp} is the time required to perform the calculation and T_{col} is the time required to collect all data needed for the calculation. The computation time is

given by

$$T_{cmp} = \left\lceil \frac{N_{ao}}{N_{au}} \right\rceil \times T_c + T_l \quad (2)$$

where N_{ao} is the number of arithmetic operations to be performed, N_{au} is the number of arithmetic units available to perform the operations, T_c is the time interval between operand input to the pipelined arithmetic units, and T_l is the latency or overall delay of the arithmetic unit. The latency term, T_l , of (2) is normally very small compared to the first term and is ignored in the rest of the paper. Address computations are ignored in this analysis, and it is assumed that memory and processors are connected so that operands are available to the processors every T_c seconds.

In the cyclic spectral analysis algorithms discussed in this paper, the time required to collect data for a computation is $\Delta t = NT_s$, where T_s is the sampling interval of the input sequences. Since we are interested in determining N_{au} for a given F_T instead of finding F_T for a given N_{au} , we can drop the ceiling function in (2). Combining (1) and (2), and substituting $T_{col} = NT_s$, we arrive at

$$N_{au} = \frac{N_{ao}}{N} \left(\frac{T_c}{T_s} \right) \frac{1}{F_T} \quad (3)$$

In cyclic spectrum analyzers, it is useful to balance the number of complex multipliers required for a particular value of F_T . Complex multipliers are used as the fundamental unit of balance since complex multiplication is the most time consuming and pervasive operation in cyclic spectral analysis algorithms. Complex additions enter the calculations in Fast Fourier Transform (FFT) operations. Each butterfly computation in an FFT consists of one complex multiplication and two complex additions. Thus, if an implementation requires N complex multiplications in an FFT calculation for computational balance, then $2N$ complex adders are required.

In this paper, we analyze the computational balance of implementations of the FFT Accumulation Method (FAM) and the Strip Spectral Correlation Algorithm (SSCA). These computationally efficient algorithms have been described in a number of papers [2], [3], [4]. In particular, we consider application of these algorithms to two problems of interest: estimating the cyclic cross spectrum of two complex-valued signals and estimating the cyclic auto spectrum of a single real-valued signal. For brevity, we refer to the first problem as a Type C computation, and the second problem as a Type A computation. The resulting analysis describes the distribution of complex multipliers in parallel implementations and indicates how

trade-offs can be made without a negative influence on the computation.

2 COMPUTATIONAL BALANCE IN THE FAM

Before describing computational balance in the FFT Accumulation Method (FAM), the computational aspects of the algorithm are reviewed. The first step in the FAM is to compute the complex demodulates of the input sequences. This computation is performed by N' -channel channelizer and consists of multiplying a block of input data by a data-tapering window, Fourier transforming the block of data, and then frequency shifting the output streams of the Fourier transformer to baseband. In the FAM, the input samples are grouped into blocks of N' samples, multiplied by a data-tapering window and Fourier transformed by an N' -point FFT. The output sequences from each channel of the FFT are then frequency shifted to baseband by multiplying the output streams by the appropriate complex exponential. To improve the efficiency of the computation, input samples are shifted into the channelizer L samples at a time. Typically, the decimation factor is set to $L = N'/4$. In all, a total of N' complex demodulate sequences are computed, and each sequence contains $P = N/L$ samples.

After channelization, the complex demodulate sequences are cross multiplied to form product sequences, and each product sequence is Fourier transformed by a P -point FFT. For a Type C computation, N'^2 P -point FFTs are performed, and for a Type A computation, $N'^2/4$ P -point FFTs are performed. The computational complexity of the FAM, in terms of complex multiplications, is given in Table 1 for Type C and Type A computations. For reference, the frequency resolution of the FAM is $\Delta f \approx f_s/N'$, where $f_s = 1/T_s$ and the cycle frequency resolution is $\Delta \alpha = f_s/N$. The time-frequency resolution product of the FAM, a measure of the statistical reliability of the estimates, is $\Delta t \Delta f = PL/N'$.

Using Table 1, the number of complex multiplications, N_{cm} , for a Type C computation is

$$N_{cm} = 2PN' + PN' \log_2 N' + 2PN' + P(N')^2 + (N')^2 \frac{P}{2} \log_2 P \quad (4)$$

Recalling that $P = N/L$ and $L = N'/4$, equation (4) simplifies to

$$N_{cm} = 8N + 4N \log_2 N' + 8N + 4NN' + 2NN' \log_2 P \quad (5)$$

Operation	Computation	
	Type C	Type A
Data Tapering	$2N'P$	†
N' -point FFT	$PN' \log_2 N'$	$P \frac{N'}{2} \log_2 N'$
Frequency Shift	$2N'P$	$N'P$
Product Sequences	$P(N')^2$	$P \frac{(N')^2}{4}$
P -point FFTs	$(N')^2 \frac{P}{2} \log_2 P$	$\frac{(N')^2}{4} \frac{P}{2} \log_2 P$

Table 1: Complex multiplications in the FAM for Type C and Type A computations. † $N'P$ real multiplications required for data tapering.

Since we are interested in the number of complex multipliers necessary for balance, we substitute (5) for N_{au} in (3) and interpret N_{au} as N_{CM} , the number of complex multipliers. Performing the substitutions yields

$$N_{CM} = (8 + 4 \log_2 N' + 8 + 4N' + 2N' \log_2 P) \times \left(\frac{T_c}{T_s} \right) \frac{1}{F_T} \quad (6)$$

From Table 1, the number of complex multiplications for a Type A computation is

$$N_{cm} = P \frac{N'}{2} \log_2 N' + N'P + P \frac{(N')^2}{4} + \frac{(N')^2}{4} \frac{P}{2} \log_2 P \quad (7)$$

Multiplications for the initial data tapering are not included in this expression since it calls for $N'P$ real multiplications. Simplifying (7) by substituting $P = N/L$ and $L = N'/4$, substituting the resulting equation into (3), and interpreting N_{au} as N_{CM} yields

$$N_{CM} = \left(2 \log_2 N' + 4 + N' + \frac{N'}{2} \log_2 P \right) \times \left(\frac{T_c}{T_s} \right) \frac{1}{F_T} \quad (8)$$

The interpretation of equations (6) and (8) is summarized in Table 2. The table specifies the number of complex multipliers required to estimate the cyclic spectrum in real-time ($F_T = 1$) for $T_s = T_c$ with the FAM. Although Table 2 indicates that no complex multipliers are required to taper the input sequence for a Type A computation, it is easily shown that four real multipliers are required for computational balance.

It may seem odd that a constant number of multipliers is used in the data-tapering and frequency-translation sections. In those calculations, the number

Operation	Computation	
	Type C	Type A
Data Tapering	8	†
N' -point FFT	$4 \log_2 N'$	$2 \log_2 N'$
Frequency Shift	8	4
Product Sequences	$4N'$	N'
P -point FFTs	$2N' \log_2 P$	$(N'/2) \log_2 P$

Table 2: Distribution of complex multipliers in balanced FAM implementations for $F_T = 1$ and $T_s = T_c$. † Four real multipliers required for balance.

of operations and the amount of time allotted to perform the computations scale proportionately so that the number of multipliers is constant.

Table 2 can be used to illustrate the redundancy found in implementations based on the signal flow graphs presented in [2]. For example, implementations derived from the signal flow graph for a Type C computation require $(N')^2$ complex multipliers for the product sequence computation, whereas the balanced implementation requires only $4N'$. The reason for this discrepancy is that the signal flow graph of the algorithm indicates the connectivity of the data while the computational balance accounts for the timing of data movement.

Table 2 also indicates how implementation trade-offs can be made without affecting the overall performance of the implementation. For example, consider the product sequence calculation of a Type C computation. By replacing multipliers in that part of the implementation with multipliers that operate at $T_c^* = 4T_c$, the number of multipliers used in that section can be reduced by a factor of four.

3 COMPUTATIONAL BALANCE IN THE SSCA

The Strip Spectral Correlation Algorithm (SSCA) is computationally similar to the FAM, except that only one channelizer is required and the product sequences (and therefore the smoothing FFTs) are larger. Using a simple interpolation scheme, the same channelizer that is used in the FAM can be used in the SSCA [4]. The computational complexity of the SSCA is summarized in Table 3. For reference, the frequency resolution of the SSCA is $\Delta f = f_s/N'$, the cycle frequency resolution $\Delta \alpha = f_s/N$, and the time-frequency resolution product is $\Delta t \Delta f = N/N'$.

Operation	Computation	
	Type C	Type A
Data Tapering	$4N$	$\uparrow$
N' -point FFT	$2N \log_2 N'$	$2N \log_2 N'$
Frequency Shift	$4N$	$2N$
Product Sequences	NN'	$N(N'/2)$
N -point FFT	$N' \frac{N}{2} \log_2 N$	$\frac{N'}{2} \frac{N}{2} \log_2 N$

Table 3: Complex multiplications in the SSCA for Type C and Type A computations. $\uparrow$ $4N$ real multiplications required for data tapering.

From Table 3, the number of complex multiplications for a Type C computation is

$$N_{cm} = 4N + 2N \log_2 N' + 4N' + NN' + N' \frac{N}{2} \log_2 N \quad (9)$$

Substituting N_{cm} for N_{ao} in (3), and interpreting N_{ao} as N_{CM} gives

$$N_{CM} = \left(4 + 2 \log_2 N' + 4 + N' + \frac{N'}{2} \log_2 N \right) \times \left(\frac{T_c}{T_s} \right) \frac{1}{F_T} \quad (10)$$

The number of complex multiplications for a Type A computation is

$$N_{cm} = 2N \log_2 N' + 2N + N \frac{N'}{2} + \frac{N'}{2} \frac{N}{2} \log_2 N \quad (11)$$

Like the FAM Type A computation, the SSCA Type A computation requires $4N$ real multiplications for the initial data tapering. Performing the necessary substitutions and manipulations yields

$$N_{CM} = \left(2 \log_2 N' + 2 + \frac{N'}{2} + \frac{N'}{4} \log_2 N \right) \times \left(\frac{T_c}{T_s} \right) \frac{1}{F_T} \quad (12)$$

The distribution of complex multipliers for this architecture is given in Table 4 for $T_c = T_s$ and $F_T = 1$. Although Table 4 indicates that no complex multipliers are required to taper the input sequence, it is easily shown that four real multipliers are necessary for computational balance. The comments at the end of Section 2 regarding computational balance in the FAM also apply to computational balance in the SSCA.

Operation	Computation	
	Type C	Type A
Data Tapering	4	$\uparrow$
N' -point FFT	$2 \log_2 N'$	$2 \log_2 N'$
Frequency Shift	4	2
Product Sequences	N'	$N'/2$
N -point FFTs	$(N'/2) \log_2 N$	$(N'/4) \log_2 N$

Table 4: Distribution of complex multipliers in balanced SSCA implementations for $F_T = 1$ and $T_s = T_c$. $\uparrow$ Four real multipliers required for balance.

4 SUMMARY

Parallel architectures are a key element of real-time cyclic spectral analysis. Using the tables presented here, together with the signal flow graphs in [2], efficient cyclic spectrum analyzers can be designed. The computational balancing technique illustrated here can be useful in analyzing design alternatives as well as determining performance trade-offs.

References

- [1] W. A. Gardner, "Exploitation of spectral redundancy in cyclostationary signals," *IEEE Signal Processing Magazine*, April 1991, pp. 14-36.
- [2] R. S. Roberts, W. A. Brown, and H. H. Loomis, Jr., "Computationally efficient algorithms for cyclic spectral analysis," *IEEE Signal Processing Magazine*, April 1991, pp. 38-49.
- [3] W. A. Brown and H. H. Loomis, Jr., "Digital implementations of spectral correlation analyzers," *IEEE Transactions on Signal Processing*, Vol. 41, No. 2, February 1993, pp. 703-720.
- [4] R. S. Roberts, W. A. Brown, and H. H. Loomis, Jr., "A Review of Digital Spectral Correlation Analysis: Theory and Implementation," to appear in *Cyclostationarity in Communications and Signal Processing*, IEEE Press, 1993.

END

DATE

FILMED

3/4/94