美国摄影光学仪器工程师学会—国际光学工程学会
会议文集 第431卷：实时信号处理 VI

PROCEEDINGS

Of SPIE-The International Society for Optical Engineering

Volume 431

Real Time Signal Processing VI

Keith Bromley
Chairman/Editor

August 23-25, 1983
San Diego, California

Proceedings of SPIE—The International Society for Optical Engineering

Volume 431

Real Time Signal Processing VI

Keith Bromley
Chairman/Editor

August 23-25, 1983
San Diego, California

Published by
SPIE—The International Society for Optical Engineering
P.O. Box 10, Bellingham, Washington 98227-0010 USA
Telephone 206/676-3290 (Pacific Time) • Telex 46-7053

SPIE (The Society of Photo-Optical Instrumentation Engineers) is a nonprofit society dedicated to advancing engineering and scientific applications of optical, electro-optical, and photo-electronic instrumentation, systems, and technology.

Please use the following format to cite material from this book:
Author(s), "Title of Paper," *Real Time Signal Processing VI*, Keith Bromley, Editor, Proc. SPIE 431, page numbers (1983).

Library of Congress Catalog Card No. 83-051064
ISBN 0-89252-466-9

Printed in the United States of America.

Volume 431

Conference Committee

Chairman
Keith Bromley
U.S. Naval Ocean Systems Ctr.

Co-Chairmen
H. John Caulfield
Aerodyne Research, Inc.

Theo Kooij
U.S. Office of Naval Technology

J. P. Letellier
U.S. Naval Electronic Systems Command

Jeffrey M. Speiser
U.S. Naval Ocean Systems Center

Session Chairmen
Session 1—Parallel Processing Algorithms, Architectures, and Applications I
J. P. Letellier, U.S. Naval Electronic Systems Command

Session 2—Parallel Processing Algorithms, Architectures, and Applications II
Jeffrey M. Speiser, U.S. Naval Ocean Systems Center

Session 3—Electronic Systolic Processors
Theo Kooij, U.S. Office of Naval Technology

Session 4—Optical Systolic Processors
H. John Caulfield, Aerodyne Research, Inc.

Session 5—Some Recent Signal Processing Research
Keith Bromley, U.S. Naval Ocean Systems Ctr.

INTRODUCTION

The world of signal processing is today in a period of rapid revolutionary growth. The soon-to-come capability to put over a million transistors on a single integrated-circuit chip opens up a frontier of advanced highly parallel architectures. The design of configurations of large arrays of processing elements with interconnections and control structures optimized for efficient arithmetic computation is an extremely active research area. These developments in advanced architectures have in turn challenged mathematicians to develop more efficient stable parallel algorithms. This flurry of activity is all to the benefit of systems engineers who soon will no longer have to discard the *preferred* algorithmic solution to their problem due to lack of computational capability.

Highly parallel processing has long been the exclusive domain of *optical* signal processing technology. The optical community has impressively responded to this "invasion of territory" with techniques for performing advanced linear algebra operations with digital precision, yet which take full advantage of the two-dimensional parallelism of optics.

In organizing this year's Real Time Signal Processing conference, emphasis was placed on the algorithms and architectures for parallel implementation of matrix operations. Matrix algorithms tend to provide increased performance by permitting more realistic modeling of the signal processing problem and incorporation of prior knowledge concerning signal and/or noise distribution. Systolic architectures are well matched to efficiently implementing these algorithms due to their modular parallelism, local interconnects, and regular data flow.

Highlights of the papers on recent algorithmic developments include (a) a parallel approach to recursively updated least-squares solutions which avoids numerical instability without requiring column pivoting, (b) algorithms for partitioning the singular value decomposition when the matrix size is larger than the processor-array size, and (c) algorithms for the parallel computation of the generalized singular value decomposition by a combination of simple systolic architectures.

Discussion of electronic implementation issues ranges from the design of a bit-level systolic-array chip to the completion of the 16,384-element Massively Parallel Processor. A novel concept for designing fault tolerance into systolic processors looks extremely promising. After an overview of electro-optical and acousto-optical implementations of systolic processors, methods for optically performing *digital* operations and advanced matrix decompositions are presented.

I wish to thank those authors and co-authors who contributed to the program, as well as the co-chairmen, particularly my colleague Jeffrey M. Speiser, who contributed their time and expertise in organizing and implementing the conference.

Keith Bromley
U.S. Naval Ocean Systems Center

REAL TIME SIGNAL PROCESSING VI

Volume 431

Contents

Session 1

Parallel Processing Algorithms, Architectures, and Applications I

Chairman
J. P. Letellier
U.S. Naval Electronic Systems Command

A review of signal processing with systolic arrays

J. M. Speiser
H. J. Whitehouse

Naval Ocean Systems Center
San Diego, CA 92152

Abstract

This paper reviews recent developments in signal processing and surveys recent progress in parallel processing algorithms and architectures for their real-time implementation.

It has previously been shown[1-2] that the major computational requirements for many important real-time signal processing tasks can be reduced to a common set of basic matrix operations including matrix-vector multiplication, matrix-matrix multiplication and addition, matrix inversion, solution of systems of linear equations, least squares approximate solution of linear systems, eigensystem solution, generalized eigensystems solution, and singular value decomposition (SVD) of matrices. To this list, we would now add the generalized singular value decompositions of Van Loan[3,4] and Paige-Saunders[5].

The first five matrix operations listed above may be computed non-iteratively, and systolic array architectures and algorithms are available which provide modular parallelism, local interconnects, regular data flow, and high efficiency, with the efficiency essentially constant as the parallelism is increased[6-8].

Parallel computation of eigensystems, generalized eigensystems, the singular value decomposition, and the generalized singular value decomposition is more difficult, since the computation is necessarily iterative, and it is difficult to utilize only local communication between processing elements while maintaining high efficiency. Algorithms for the latter problems are therefore still the subject of intensive research.

Introduction

Digital signal processing architectures underwent a revolutionary change in the mid-1960's with the development of the dedicated FFT processor, followed by an evolutionary change as further flexibility was added in the more general array processor. The requirements of modern signal processing together with the availability of VLSI/VHSIC technology are leading to another revolutionary change in signal processing architectures: the systolic[6] and wavefront[9] cellular processors with vector and matrix parallelism for matrix operations.

Signal Processing Operations

Much of present day signal processing is based on assumptions of wide-sense stationarity for noise and random signals, together with linear, time-invariant models for signal propagation. Since sinusoids are the eigenfunctions of shift-invariant kernels, the Fourier transform is important both for implementing convolutions and providing eigensystem expansions for spectrum analysis.

Recent signal processing algorithms tend to incorporate additional prior information concerning the signal structure and more realistic non-stationary noise models. This more realistic modelling will generally provide improved performance at the expense of a greater computational load. Typical examples are algorithms for spectrum analysis[10], adaptive cancellation of noise and interference[11], and beamforming. Maximum entropy spectrum analysis provides improved resolution compared to classical windowed spectrum analysis, when an all-pole spectral model is applicable, but requires solution of a linear prediction problem. Adaptive cancellation of noise or interference by direct least squares techniques provides faster convergence[11], and hence better tracking behavior than gradient based adaptive filtering, but requires the orthogonal triangularization[12] of a matrix or algorithms of comparable complexity. Recent beamforming/direction finding techniques provide improved resolution by incorporating a spatial multiple point source model for the signals, but require the solution of an eigensystem or generalized eigensystem problem at each resolved temporal frequency[13-15]. The common trend in these and many other recent signal processing algorithms is their extensive utilization of linear algebra operations, both conceptually and computationally. For the conventional computer, there are available libraries of high quality (robust, numerically stable) numerical linear algebra software: EISPACK[16] for eigensystems problems, and LINPACK[17] for linear equation solution and least squares problems. Real time signal processing will increasingly require the parallel hardware equivalent of LINPACK and EISPACK, together with the more recent generalized singular value decompositions of Van Loan[3,4] and Paige-Saunders[5]. Signal processing algorithms frequently require the solution of the eigensystem of an estimated covariance matrix, where the covariance matrix is estimated as the product of a data matrix and its transpose. Roundoff error in the formation of this matrix product can cause a significant, irreversible loss of accuracy. The computational wordlength requirement for such a problem can be reduced by about a factor of two by solving the eigensystems problem indirectly via computing the singular value decomposition of the data matrix. This is similar to the improvement in solving least squares problems via orthogonal triangularization of the data matrix, rather than forming the Gauss normal equations[18,12]. The "MUSIC" high resolution direction finding algorithm of R. Schmidt[13] requires the solution of a generalized eigensystem involving two estimated covariance matrices. For such computations, the generalized singular value decompositions of Van Loan and Paige-Saunders, permit a similar reduction in computational word length by applying a reduction directly to the data matrices.

Parallel Architectures

For applications in which the sampling rate approaches the computational cycle time, parallel architectures are required for real-time implementation of tasks requiring a number of operations proportional to the square or the cube of the number of points per data block, such as typical matrix operations[19]. Representative parallel architectures include the digital tapped delay line transversal filter, the array processor, the bus organized multiprocessor, and the systolic array[6,7]. This paper emphasizes the systolic array because of its promising combination of characteristics for utilizing VLSI/VHSIC technology for real-time signal processing: modular parallelism with throughput directly proportional to the number of cells, simple control, synchronous data flow, local interconnects, and sufficient versatility for implementing the matrix operations needed for signal processing. The authors view the systolic array as an adjunct to the general purpose computer and the array processor rather than as a replacement for either, since the general purpose computer excels in unstructured computation and decision-making and the array processor excels in fast Fourier transform computation.

Systolic Arrays

The systolic array[6], introduced by H.T. Kung, is a regular geometric array of identical or nearly identical computational cells with common timing and control and synchronous data flow. Representative systolic architectures are shown in Fig. 1 for a variety of matrix computations. Although systolic arrays have also been proposed for sorting, searching, and coding operations[6], this paper will only address matrix computations for signal processing. Systolic arrays may be viewed as distributed, space-time implementations of recursion relations. Occasionally the operation required to initiate or terminate a set of recursions will be slightly different from the intermediate recursive steps, necessitating one or more boundary cells which differ from the interior cells of the array. For example, the linear systolic array shown in Fig. 1 may be used without an exceptional boundary cell to perform matrix-vector multiplication using only scalar multiplications and additions[20]. However, the solution of triangular linear systems of equations by back substitution also requires one division per equation, and therefore an exceptional cell with division capability is then required at one end of the array[20]. Similarly, the simple hexagonal array in Fig. 1 may be used for the multiplication of banded matrices, but must be augmented by an exceptional cell with division capability for use in factoring banded matrices into a product of lower and upper triangular matrices[20]. These architectures are particularly attractive when the matrices have only a few bands occupied about the main diagonal. For the dense matrices more commonly encountered in signal processing applications, a rectangular systolic array called the engagement processor provides more efficient matrix multiplication[7]. The systolic architectures described so far are based primarily on multiplications and additions. An important class of methods for the high accuracy solution of the linear least squares problem require the factorization of the data matrix into the product of an orthogonal matrix and a triangular matrix[12,21]. This may be performed by the triangular systolic array shown at the right side of Fig. 1[8]. This array implements Givens' rotations in its interior cells, so the fundamental operation here is replacing a pair of rows by a pair of linear combinations of the two rows. The triangular systolic array for orthogonal-triangular factorization may be combined with a linear array for triangular backsubstitution to solve the least squares problem of finding the vector x which minimizes the norm of Ax-y. In many adaptive noise cancellation or adaptive interference cancellation applications, it is not necessary to find the optimum weight vector explicitly, since all that is ultimately needed is the residual Ax-y corresponding to the optimum weight vector. In this case, McWhirter has shown that the backsubstitution may be avoided[8a]. Avoiding the backsubstitution has two important consequences. It not only permits a reduction in the amount of hardware needed, but it also allows one to guarantee numerical stability of the algorithm without introducing column pivoting.

In all of the above systolic architectures, the fraction of the cells performing useful computations at any time is independent of the size of the array, so constant efficiency may be maintained as additional cells are added to increase throughput. It has also been shown that for the operations of matrix multiplication and matrix inversion, constant efficiency may be maintained when a large matrix is partitioned to fit on a small systolic array, if memory is provided, conceptually orthogonal to the plane of the array, so that submatrices may be stored or loaded in a single memory cycle time[7]. An inner product is unchanged by a cyclic permutation of the products to be summed. This idea can be used to provide a matrix multiplication using an NxN enhanced engagement processor in such a way that all of the cells are active all of the time - a result ordinarily achieved only when pipelining is possible[7].

Systolic Hardware Development

In order to explore the hardware potential of systolic arrays, TRW/ESL has developed a processor called the Phoenix which incorporates 28 of the TRW fixed point multiplier/accumulators in a linear systolic array[22]. This processor has a throughput of up to 200 million 16 bit fixed point operations per second, and has been used to provide two-dimensional finite impulse response filters for image processing. In order to evaluate systolic algorithms and architectures, the Naval Ocean System Center has developed a reconfigurable one/two dimensional systolic array with 64 processing elements[23-25]. The individual processing elements use commercial VLSI components and include a general purpose microprocessor, a floating point arithmetic unit, RAM, and ROM in each cell. This systolic testbed processor may be reconfigured under software control to perform

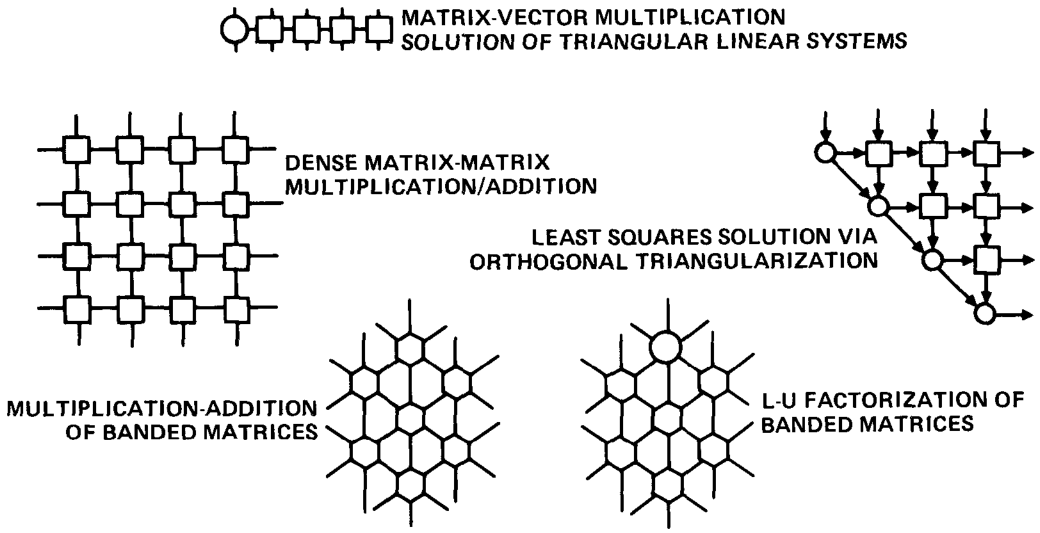

Figure 1. Systolic array solutions

as a rectangular, hexagonal, or linear systolic array, or as 8 small linear systolic arrays. It was designed primarily for flexibility in algorithm evaluation, and the cells are much slower than those used in the TRW/ESL Phoenix processor or ones which could be implemented in a future dedicated VLSI design.

Current Systolic Research

Current research on systolic algorithms and architectures for matrix computation is directed primarily toward the singular value decomposition and the symmetric eigensystem problem. Both computations are necessarily iterative for general matrices larger than four by four. The simplest systolic approach to eigensystem computation would be to use the power method, since it requires primarily matrix-vector multiplications. However, this method is not being actively pursued because of its severe numerical difficulties for closely spaced eigenvalues and small eigenvalues and their associated eigenvectors. Modern, numerically stable algorithms for the symmetric eigenvalue problem and the singular value decomposition are based on real orthogonal transformations are used in the QR algorithm[26] and the Jacobi algorithm[27]. For the singular value decomposition, one sided orthogonal transformations are used in the Nash-Hestenes method, and a modification of the QR eigensystem algorithm is used in the method of Golub and Reinsch[28]. In trying to find systolic parallelizations of these algorithms, the difficulty is not so much in providing a parallel implementation of the required transformations as in computing the transformation to be used at each stage of the algorithm. The QR type algorithms proceed in two stages. The first is a preliminary reduction to tridiagonal form for the symmetric eigenvalue problem or a reduction to bidiagonal form for the singular value decomposition. Successive transformations then perform a reduction to diagonal form. This approach has been investigated at Pennsylvania State University[29] and at Stanford University[30]. The principal difficulty is that of performing the reduction to tridiagonal form for the eigensystem problem or to bidiagonal form for the SVD in time proportional to N using about N squared processors. The Jacobi algorithms for the symmetric eigensystem problem and the corresponding one-sided orthogonalization algorithms[21] for the SVD lend themselves readily to vector parallelism[31]. Systolic versions with both matrix and vector parallelism are being studied at Cornell University. The initial method using matrix parallelism attempted to compute all N(N-1)/2 rotations for a sweep in advance, and then perform the rotations[32]. The principal difficulty with this approach is that rotations proceed simultaneously in several planes, and the classical methods and corresponding proofs of convergence are designed for computing optimal or nearly optimal rotation choices in one plane at a time. In effect, some of the rotations are performed using old information to compute the rotation angles. New methods are needed to understand the convergence behavior of such algorithms. Two alternative systolic architectures avoid the use of old data in computing the rotations. One uses vector parallelism to implement in exact Nash-Hestenes method with a modified order of performing the rotations[33]. The data movements for this architecture are shown in Fig. 2. A second method uses two-sided orthogonalization to permit computation of the SVD in $O(n \log n)$ time using $O(n^2)$ processors[34].

Open Problems

Several issues are still not well understood. Signal processing tasks frequently require repeatedly solving the same numerical problem with updated matrices as new observations are added. Although efficient recursive updating is possible for least squares solution, there remains a need for similarly efficient updating of eigensystem solutions and singular value decompositions. A second difficult issue is the choice of a control structure for a systolic array. For a fixed type of computation on matrices matched to the

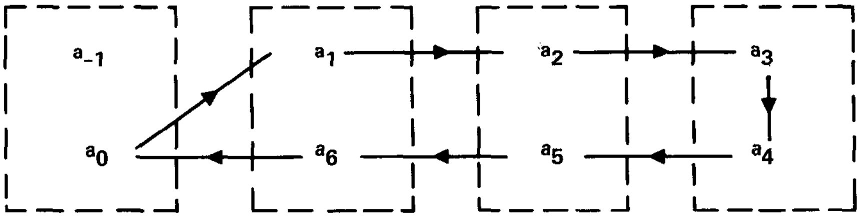

Figure 2. Data movement in the Brent-Luk linear systolic array for singular value decomposition via one-sided orthogonalization

size of the array, the control structure is minimal, but the control needed grows rapidly for partitioned computations or reconfigurable use of the processors.

Conclusions

As a result of the concurrent advances in numerical linear algebra, parallel architectures for computation, and VLSI hardware implementations, it is less frequently necessary for the signal processing engineer to use suboptimal computations when optimal computations will provide improved performance. Direct least squares minimization and iterative eigensystem calculation can be considered for future signal processing systems. The use of the singular value decomposition (SVD) and the generalized SVD make possible the use of single precision computation in many applications where double precision computation was previously required. Although much research still needs to be done in applying these new techniques to situations with time varying statistics, it is clear that the ability to implement computational macros in silicon and to connect them together without conflicts for access to resources will make possible many new and important advances in signal processing.

References

1. Speiser, J. M., H. J. Whitehouse, and K. Bromley, "Signal Processing Applications for Systolic Arrays", Record of the 14th Asilomar Conference on Circuits, Systems, and Computers, held at Pacific Grove, California, 17-19 Nov., 1980, IEEE Catalog No. 80CH1625-3, pp. 100-104.

2. Whitehouse, H. J. and J. M. Speiser, "Sonar Applications of Systolic Array Technology", presented at IEEE EASCON, Washington, D.C., Nov. 17-19, 1981.

3. Van Loan, C. F., "Generalizing the Singular Value Decomposition", SIAM Journal on Numerical Analysis, Vol. 13, No. 1, March 1976, pp. 76-83.

4. Van Loan, C. F., "Generalized Singular Values, with Algorithms and Applications", Ph.D. Thesis, Univ. of Michigan, Ann Arbor, Michigan, 1973.

5. Paige, C. C. and M. A. Saunders, "Towards a Generalized Singular Value Decomposition", SIAM Journal on Numerical Analysis, Vol. 18, No. 3, June 1981, pp. 398-405.

6. Kung, H. T., "Why Systolic Architectures?", IEEE Computer, January, 1982, pp. 37-46.

7. Speiser, J. M. and H. J. Whitehouse, "Parallel Processing Algorithms and Architectures for Real-Time Signal Processing", SPIE Vol. 298, Real-Time Signal Processing IV (1981), paper 298-01.

8. Gentleman, W. M. and H. T. Kung, "Matrix Triangularization by Systolic Arrays", SPIE Vol. 298, Real-Time Signal Processing IV (1981), paper 298-03.

8a. McWhirter, J. G., "Recursive least-squares minimization using a systolic array", to appear in Real-Time Signal Processing VI, SPIE Vol. 431, paper 431-15, San Diego, CA, Aug 23-24, 1983.

9. Kung, S. Y., R. J. Gal-Ezer, and K. S. Arun, "Wavefront Array Processors: Architecture, Language, and Applications", Proceedings, Conference on Advanced Research in VLSI, edited by Paul Penfield, Jr., MIT, January 25-27, 1982.

10. Kay, S. M. and S. L. Marple, Jr., "Spectrum Analysis — A Modern Perspective", Proc. IEEE, Vol. 69, No. 11, Nov. 1981, pp. 1380-1418.

11. Monzingo, R. A. and T. W. Miller, Introduction to Adaptive Arrays, John Wiley & Sons, New York, 1980.

12. Lawson, C. L. and R. J. Hanson, Solving Least Squares Problems, Prentice-Hall, 1974.

13. Schmidt, R. O., "A Signal Subspace Approach to Multiple Emitter Location and Spectral Estimation", Ph.D. Dissertation, Dept. of Electrical Engineering, Stanford University, November, 1981.

14. Mermoz, H. F., "Spatial Processing Beyond Adaptive Beamforming", Journal of the Acoustical Society of America, Vol. 70, No. 1, July, 1981, pp. 74-79.

15. Johnson, D. H. and S. R. DeGraaf, "Improving the Resolution of Bearing in Passive Sonar Arrays by Eigenvalue Analysis", IEEE Transactions on Acoustics Speech, and Signal Processing, Vol. ASSP-30, No. 4, August 1982, pp. 638-647.

16. Garbow, B. S. et al, "Matrix Eigensystem Routines — EISPACK Guide Extensions", Springer-Verlag, New York, 1977.

17. Dongarra, J. J. et al, "LINPACK Users' Guide", Society for Industrial and Applied Mathematics, Philadelphia, 1979.

18. Chambers, J. M., Computational Methods for Data Analysis, Chapter 5, "Linear Models", John Wiley and Sons, New York, 1977.

19. Bromley, K. and H. J. Whitehouse, "Signal Processing Technology Overview", SPIE Vol. 298, Real-Time Signal Processing IV (1981), paper 298-14.

20. Kung, H. T., and C. E. Leiserson, "Algorithms for VLSI Processor Arrays", Section 8.3 of Introduction to VLSI Systems by C. Meade and L. Conway, Addisson-Wesley Publishing Co., Reading, MA, 1980.

21. Nash, J. C., Compact Numerical Methods for Computers, Halsted Press, John Wiley & Sons, New York, 1979.
22. Blackmer, J., G. Franks, and P. Keukes, "A 200 Million Operations Per Second (MOPS) Systolic Processor", SPIE Vol. 298, Real Time Signal Processing IV (1981), paper 298-02.
23. Symanski, J. J., "A Systolic Array Processor Implementation", SPIE Vol. 298, Real-Time Signal Processing IV (1981), paper 298-04.
24. Bromley, K., J. J. Symanski, J. M. Speiser, and H. J. Whitehouse, "Systolic Array Processor Developments", in proceeding of the CMU Conference on VLSI Systems and Computation, Carnegie-Mellon University, Pittsburgh, PA, 19-21 October, 1981.
25. Symanski, J. J., "Progress on a Systolic Processor Implementation", SPIE Vol. 341 (1982), Real-Time Signal Processing V, paper 341-01.
26. Wilkinson, J. H., The Algebraic Eigenvalue Problem, Oxford University Press, 1965.
27. Schwarz, H. R., H. Rutishauser, and E. Steifel, Numerical Analysis of Symmetric Matrices, Prentice-Hall, Englewood Cliffs, NJ, 1973.
28. Golub, G. H. and C. Reinsch, "Singular Value Decomposition and Least Squares Solutions", Numer. Math. Vol. 14 (1970), pp. 403-420.
29. Heller, Don E. and Ilse C. F. Ipsen, "Systolic Networks for Orthogonal Decompositions, with Applications", Report No. CS-81-18, August 1981, Dept. of Computer Science, Pennsylvania State University (also in Pennfield, P., Proceedings of the Conference on Advanced Research in VLSI, Artech House, Dedham, MA, 1981.
30. Schreiber, R., "Systolic Arrays for Eigenvalue Computation", in Real-Time Signal Processing V, SPIE Vol. 341, paper 341-50.
31. Luk, F. T., "Computing the Singular Value Decomposition on the Illiac IV", ACM Transactions on Mathematical Software, Vol. 6, 1980, pp. 524-539.
32. Luk, F. T., A. M. Finn, and C. Pottle, "Systolic Array Computation of the Singular Value Decomposition", SPIE Vol. 341, Real Time Signal Processing V (1982), paper 341-05.
33. Brent, R. P. and Luk, F. T., "A Systolic Architecture for the Singular Value Decomposition", Report TR 82-522, Dept. of Computer Science, Cornell University, Sept. 1982.
34. Brent, R. P., F. T. Luk, and C. Van Loan, "Computation of the Singular Value Decomposition using Mesh-Connected Processors", Report TR 82-528, Department of Computer Science, Cornell University, March 1983.

Acknowledgement

The authors would like to acknowledge the support of the Office of Naval Research (Mathematics and Information Sciences Division).

New Mathematical Tools in Direction Finding and Spectral Analysis

Ralph O. Schmidt

ESL,Inc., a Subsidiary of TRW,Inc.
495 Java Dr., Sunnyvale, CA 94088-3510

Abstract

Linear Algebra (*i.e.*, the algebra of vector spaces) provides widely used mathematical tools and concepts which are today being considered for implementation in special compute architectures. It seems that so many signal processing problems can be expressed and, more importantly, implemented efficiently as a sequence of vector and matrix operations, that a signal processing system with a capability for high speed linear algebra is necessary if the more advanced signal processing algorithms are to be implemented to operate in real time. The purpose of this paper is to support the notion that linear algebra is a sound basis for important signal processing system implementations and, further, to suggest that **multilinear algebra** (*i.e.*, the algebra of vector, bivector, trivector, *etc.* spaces) offers an even broader set of signal processing "tools". Examples and ideas from direction finding and time series analysis are discussed.

§1. Introduction

Many physical problems in sonar, radar, seismic, *etc.* signal processing utilize the records of multiple sensors to estimate the location of one or more sources of coherent energy—often assumed to be point source emitters. Typically, the sensor outputs contain the source waveforms as modified in amplitude and phase by the medium between the sources and the array elements. Essentially all of the geometric information is encoded in the set of pairwise cross correlations associated with the sensor outputs and arising from the propagation of the directional wavefronts across the array. Estimating the source locations and the emission parameters constitutes, in a general sense, spectral estimation.

Alternatively, a tapped delay line may be used to provide the multichannel version of a single channel time series. The parameters of multiple cisoids (pole location, amplitude, and phase) may be estimated using the same mathematical as well as algorithmic tools as in the direction finding(DF) or emitter location problem. This also is a problem in spectral estimation.

In the modern (*e.g.,* high resolution) spectral analysis implementations, the predominant computations are vector dot products, projections of vectors onto subspaces, and least squares estimates of subspaces. But, as we shall see, there is a vector continuum (*i.e.*, the "array manifold") of very practical as well as theoretical use to the DF problem and there is at least one manifold of importance consisting of a continuum of 2D-spaces (a *bivector* manifold) which permits the solution of DF for polarizationally diverse antenna arrays. Ordinary vectors alone appear insufficient to treat the problem in all quite practical generality. The purpose of this paper is to discuss the introduction of additional geometrical objects and the associated tools for dealing with them into the general spectral analysis problem and in particular into the DF problem.

Specifically, we wish to show that wedge products between vectors yield useful bivectors, trivectors, *etc.* —graphically interpretable but not representable in the same space as the original vectors since they are of a different dimensionality—which permit a new viewpoint of the problem as well as new implementations and extensions of its solution.

Thus, the mathematics of spectral analysis applies to both temporal and spatial spectra, and the tools of multilinear algebra can apply to both. However, there is a significant class of spectral analysis methods which will not be discussed here. To establish the range of application, consider the general state variable approach with a difference (or differential) equation embodying known or assumed structure in the observed data and an equation relating the vector of observations $\mathbf{x}$ to the state vector $\mathbf{z}$.

$$\begin{aligned} \mathbf{z}_{n+1} &= \mathbf{F}\mathbf{z}_n + \mathbf{u}_n \\ \mathbf{x}_n &= \mathbf{H}\mathbf{z}_n + \mathbf{w}_n \end{aligned}$$

There are two distinct types of "noise" associated with this model—the input noise $\mathbf{u}$ and the output or measurement noise $\mathbf{w}$. Since the inclusion of both types of noise is quite general and leads to implementations (*e.g.,* the Kalman filter) which may be more *costly* than necessary, the quest for simplicity arises. We may simplify the model and perhaps, thereby, simplify implementations, by setting either of the two noises to zero.[1]

Setting the measurement noise to zero by assumption leads to the approach known as autoregressive, moving average (ARMA) modelling with its many variations, special cases, and alternatives, and descriptions—pole/zero modelling, Maximum Entropy analysis, *etc.* . Speech is often modelled as the output of a linear system which is driven by (input) noise. Economic time series are also often modelled as systems driven by random forces.

However, for data processing problems where the data disturbance is an additive noise, it would be more appropriate to set, by assumption, the input noise to zero leaving free the output or measurement noise to account for the random component in the data. Such a model should be more appropriate than the ARMA model for radio wave DF or sound wave DF as well as signals-plus-noise processing problems.

Spectral analysis methods may be said to differ fundamentally depending on which noise-generating mechanism is included in the model. We will be discussing models which include output or measurement noise and will not be considering other models.

The discussion is arranged to include a short geometric description of the mathematical objects known as multivectors followed by examples of their use in two significant applications-oriented subdivisions of spectral analysis as follows.

(1) New Tools for Signal Processing

   (a) Vector Spaces

   (b) Multivector Spaces

   (c) exterior (or "wedge") product between multivectors

(2) Signal Processing Examples

   (a) The DF Problem

      - The Signal Subspace
      - The Array Manifold for scalar fields
      - The Array Manifold for vector fields

   (b) Time Series Analysis

      - Pole location, Frequency Estimation
      - Hilbert Transform, Multichannel Quadrature Generation
      - Spectral Analysis, Fitting Data to a Constraint Manifold

§2. Multilinear Algebra and Multivectors

Multilinear algebra (or exterior algebra) is the algebra of p-vectors whereas linear algebra applies to "ordinary" or 1-vectors. Actually, p-vectors are antisymmetric tensors of rank p so that a 2-vector could be represented as a square, antisymmetric matrix. Because of the antisymmetry, only $\binom{M}{2}$ of the M^2 matrix elements are independent—only those with two unequal subscripts are nonzero and, of those, half are the negatives of the other half. Perhaps more familiar to most readers, multivectors appear in line, surface, and volume integrals in a way which may suggest their importance. From a discussion in [5], the line integral

$$\int Adx + Bdy + Cdz$$

leads to the one-form

$$\alpha = Adx + Bdy + Cdz$$

[1]Many algorithms may already solve the problem where both noises are assumed zero even though the noise-free problem isn't necessarily trivial. But some algorithms will "work better" than others when data doesn't adhere to the assumed noise-free conditions (*i.e.*, they are more "robust").

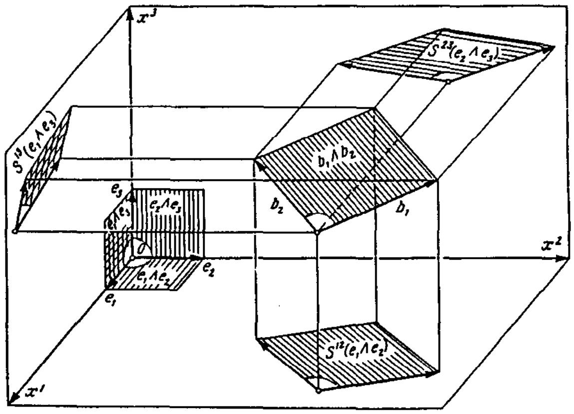

Figure 1. Pictorial Example of Bivectors

suggesting that dx, dy and dz be thought of as unit vectors. The surface integral

$$\int\int Pdydz + Qdzdx + Rdxdy$$

leads to the two-form

$$\beta = Pdydz + Qdzdx + Rdxdy$$

suggesting that $dydz$, $dzdx$ and $dxdy$ be thought of as unit bivectors. The volume integral

$$\int\int\int Hdxdydz$$

leads to the three-form

$$\alpha = Hdxdydz$$

suggesting that $dxdydz$ be thought of as a unit trivector. Since $dxdy$ appears but $dydx$ does not, there is a strong suggestion of symmetry—or antisymmetry. The absence of $dxdx$ favors antisymmetry which consists of the rules

$$dxdx = 0 \quad \text{and} \quad dydx = -dxdy$$

As it happens, the combinations of differentials do obey all of the properties of multivectors—line and surface integrals are "oriented", *etc.* —and we see that

$$dxdy \Leftrightarrow dx \wedge dy \quad \text{and} \quad dxdydz \Leftrightarrow dx \wedge dy \wedge dz \quad etc.$$

In Figure 1, the concept of unit bivectors is portrayed in the context of ordinary unit vectors. The example includes a 3-dimensional coordinate system with three unit vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$. In terms of these unit vectors, there are also three unit bivectors

$$\mathbf{e}_{12} = \mathbf{e}_1 \wedge \mathbf{e}_2, \qquad \mathbf{e}_{13} = \mathbf{e}_1 \wedge \mathbf{e}_3, \qquad \mathbf{e}_{23} = \mathbf{e}_2 \wedge \mathbf{e}_3$$

shown as unit areas in the principal planes of the coordinate axes. The components of a vector, *e.g.*, $\mathbf{b}_1$ or $\mathbf{b}_2$ as shown, are found by projecting the vectors onto the unit vectors in turn. The components of the bivector, *viz.*, $\mathbf{b}_1 \wedge \mathbf{b}_2$ are likewise found by projecting the bivector onto the principal planes to determine the amounts of each unit bivector required to express the bivector within this vector space.

Note that, in an M-dimensional vector space, there would be $\binom{M}{2}$ unit bivectors and $\binom{M}{3}$ unit trivectors, *etc.* . Therefore, although vectors, bivectors, trivectors, *etc.* can be viewed as geometric objects—lines, surfaces, volumes, *etc.* —the actual dimensionality (*i.e.*, the number of degrees of freedom required to specify) is not the same among them. Examples of vectors, bivectors, and trivectors are given in Figure 2. Their numerical coordinate representations can be obtained as follows.

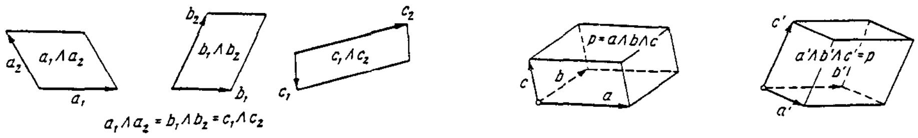

Figure 2. Examples of Vectors, Bivectors, and Trivectors

Starting in a 4 dimensional vector space with vectors **x**, **y** and bivector **z** where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} z_{12} \\ z_{13} \\ z_{14} \\ z_{23} \\ z_{24} \\ z_{34} \end{bmatrix}.$$

The bivector **z** has six components corresponding to the six unit bivectors arising as the four unit vectors are taken two at a time. We can write the components of a bivector α as

$$\alpha_{ij} = (\mathbf{x} \wedge \mathbf{y})_{ij} \quad \Rightarrow \quad \begin{bmatrix} \alpha_{12} \\ \alpha_{13} \\ \alpha_{14} \\ \alpha_{23} \\ \alpha_{24} \\ \alpha_{34} \end{bmatrix} = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} = \begin{bmatrix} x_1 y_2 - x_2 y_1 \\ x_1 y_3 - x_3 y_1 \\ x_1 y_4 - x_4 y_1 \\ x_2 y_3 - x_3 y_2 \\ x_2 y_4 - x_4 y_2 \\ x_3 y_4 - x_4 y_3 \end{bmatrix}$$

The bivector α is shown as having six components expressed as the minors of the 4 by 2 matrix with columns **x** and **y**. (For discussion and details, consult [6].) The subscripting for a wedge product between a bivector **z** and a vector **x** is shown in the calculation of the components of the trivector β.

$$\beta_{ijk} = (\mathbf{z} \wedge \mathbf{x})_{ijk} \quad \Rightarrow \quad \begin{bmatrix} \beta_{123} \\ \beta_{124} \\ \beta_{134} \\ \beta_{234} \end{bmatrix} = \begin{bmatrix} z_{12}x_3 + z_{23}x_1 + z_{31}x_2 \\ z_{12}x_4 + z_{24}x_1 + z_{41}x_2 \\ z_{13}x_4 + z_{34}x_1 + z_{41}x_3 \\ z_{23}x_4 + z_{34}x_2 + z_{42}x_3 \end{bmatrix}$$

Although the subscript *juggling* needed to carry out a wedge product is not necessarily obvious (and certainly was not derived herein), it is clearly something that can be done by a computer.

The geometric objects represented by multivectors have many interesting and geometrically interpretable properties. There are even eigen multi-vectors and -values which are closely tied to eigen vectors and values. But we shall now discuss examples of signal processing which can make use of these tools in description and perhaps algorithmic implementation.

§3. The DF Problem

The direction finding problem is portrayed in Figure 3. There are M sensors with arbitrary directional response characteristics (amplitude, phase, and polarization). The idea is to make use of any structure in the received vectors of voltages that is due to the sources being point sources. In time series analysis, this is equivalent to the structure arising because of the points located on the z plane (or s plane) called poles which characterize the noise-free data.

We shall see that the generality of the problem statement does not detract from its usefulness. On the contrary, restricting the problem to uniform collinear arrays, omnidirectional antenna elements, polarizationally non-diverse arrays (*e.g.*, parallel dipoles), *etc.* does not provide any conceptual or performance advantages—the general concepts are revealing and disarmingly simple. The advantages, if any, would be simplicity in implementation.

3.1 The Signal Subspace Approach to DF

The following description of a completely general DF algorithm is unique in that it is virtually free of equations yet is more than just believable—it is convincing. There are two geometric objects of interest involved and the description will take the form of a "thought experiment". The two objects of interest are the signal subspace and the array manifold. We shall see that the complete

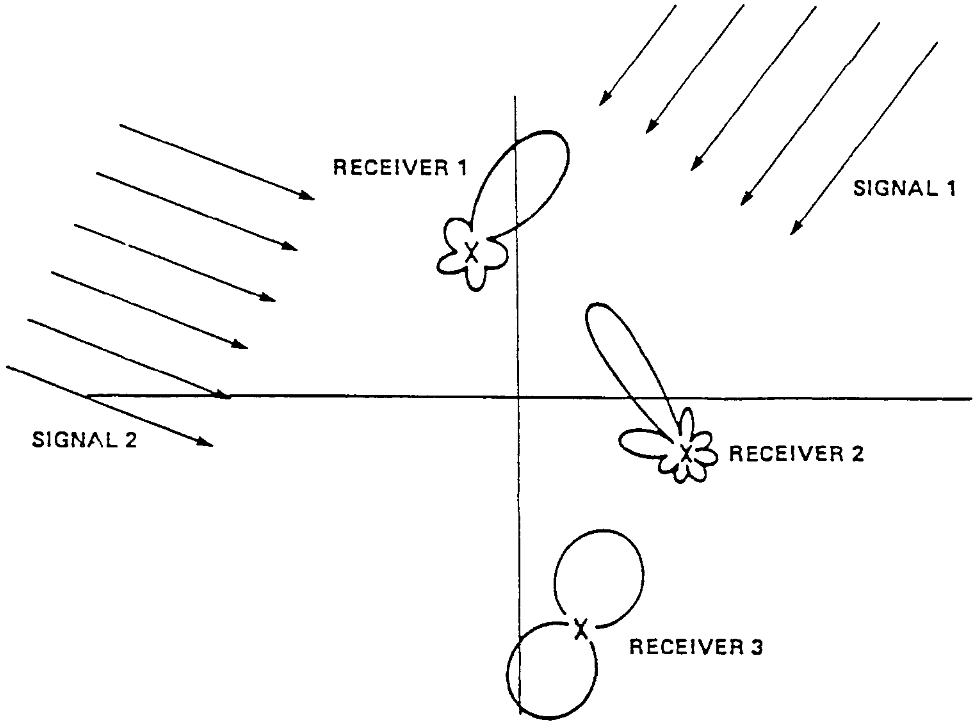

Figure 3. The General DF Problem

solution to the DF problem requires the intersection of these two and that estimates of all parameters of the multiple sources present are made available.

3.2 The Signal Subspace

Let us regard the (complex) voltage received on an array of M completely arbitrary antennas as a (complex) vector function of time $\mathbf{x}(t)$. In our minds eye, we may "watch" the vector in M space as in Figure 4 in an attempt to ferret out useful patterns and/or structure useful in the DF problem. As we shall discover, there is a great deal of structure to perceive even though the array is completely arbitrary in the positions, orientations, and responses of the individual antennas.

First, consider that if only one (narrowband) point source were present (or, equivalently, if it were radiating strongly enough to predominate over all other sources as well as noise), the received vector of voltages would, at any point in time, be a mere scaled version of what is known as the steering vector in common DF terminology. Corresponding to each particular direction of arrival (DOA) in physical space, there is a steering vector, sometimes referred to as a mode vector or a DOA vector. (In order that the steering vector $\mathbf{a}(\theta, \phi)$ corresponding to the DOA with elevation θ and azimuth ϕ be unique, let us normalize it so that its squared length is M and the phase of a selected element, say the first, is zero.)

For this case of a single source, the observed vector of voltages $\mathbf{x}(t)$ will change in time but it will always be expressible as an "amount" of $\mathbf{a}(\theta, \phi)$ —its direction in M space does not change as long as the source location does not change. The received vector $\mathbf{x}(t)$ is confined as in Figure 5 to the one dimensional subspace having, as a basis vector, the DOA vector $\mathbf{a}(\theta, \phi)$ corresponding to the DOA of the source.

Thus, the actual source signal can be directly identified with the (complex) magnitude of the DOA basis vector $\mathbf{a}(\theta, \phi)$ being observed. Let us refer to the one dimensional subspace with $\mathbf{a}(\theta, \phi)$ as its basis vector as the signal subspace reachable by the arbitrary