ITERATIVE OPTICAL VECTOR-MATRIX PROCESSORS

(SURVEY OF SELECTED ACHIEVABLE OPERATIONS)

David Casasent and Charles Neuman
Carnegie-Mellon University
Department of Electrical Engineering
Pittsburgh, Pennsylvania 15213

ABSTRACT

An iterative optical vector-matrix multiplier with a microprocessor-controlled feedback loop is capable of performing a wealth of diverse operations. In this paper, we survey and describe many of these operations to demonstrate the versatility and flexibility of this class of optical processor and its use in diverse applications.

1. INTRODUCTION

The optical vector-matrix multiplier [1] is a general purpose optical processor. The addition of a microprocessor-controlled feedback loop results in an even more general purpose and far more powerful optical processor [2-6]. In this paper, we survey and describe a selected set of the operations achievable on such a processor.

In Section 2, a general description of the system is advanced. This is followed in Section 3 by a description of how bipolar and complex-valued data are handled on the system and how the convergence of the iterative algorithm is insured. We then address in Section 4 its use in the solution of linear difference and differential equations and linear algebraic equations. In Section 5, we consider application of this processor for the solution of the least-squares problem. In Section 6, we address its use for deconvolution and for the computation of the eigenvalues and eigenvectors of a matrix. In Section 7, our attention turns to the use of the system for matrix-matrix multiplication, the solution of linear matrix-matrix equations and matrix inversion. We then consider in Section 8 its use in the solution of nonlinear matrix equations with specific application to the linear-quadratic-regulator problem and algebraic Riccati equation of optimal control engineering.

2. ITERATIVE OPTICAL PROCESSOR SYSTEM

The general schematic of the iterative optical processor (IOP) is shown in Figure 1. This figure illustrates the use of the IOP both as a vector-matrix multiplier and as an iterative processor for the solution of linear vector-matrix equations. Since an understanding of this architecture is paramount to the remainder of this paper, we review its operations. To multiply a vector by a matrix, the elements of the vector are used to spatially modulate a linear array of light emitting diodes (LEDs) or laser diodes (LDs) at P_1. We denote the light distribution leaving P_1 by the vector $\underline{x}$. Plane P_1 is imaged vertically onto P_2 and the output from each LED is expanded to uniformly illuminate the corresponding row of a 2-D mask at P_2. The light

distribution leaving each column of P_2 is then summed onto a different photodetector at P_3. With the transmittance of P_2 specified by the matrix $\underline{H}$, the vector output from the linear photodetector array at P_3 is the matrix-vector product $\underline{H}\,\underline{x}$ as described in [1] and earlier by others [7,8].

In our system [2,3], we have included feedback of the photodetector outputs to the LED inputs through an electronic feedback system. In our original description of this system [2,3], the mask was $[\underline{I} - \underline{H}]$, where $\underline{I}$ is the identity matrix. The P_3 output is then $[\underline{I} - \underline{H}]\underline{x}(j)$, where $\underline{x}(j)$ denotes the vector $\underline{x}$ at the j-th iteration. An external vector $\underline{y}$ was added to this matrix-vector product in the electronic feedback system to form the new iterative input $\underline{x}(j + 1)$. Our system thus implemented the iterative algorithm

$$\underline{x}(j + 1) = [\underline{I} - \underline{H}]\underline{x}(j) + \underline{y} = \underline{x}(j) - \underline{H}\,\underline{x}(j) + \underline{y}. \tag{1}$$

In the steady-state when $\underline{x}(j + 1) \simeq \underline{x}(j)$, equation (1) reduces to $\underline{H}\,\underline{x} = \underline{y}$ and the output $\underline{x}$ is the solution

$$\underline{x} = \underline{H}^{-1}\underline{y} \tag{2}$$

of the vector-matrix equation

$$\underline{H}\,\underline{x} = \underline{y}. \tag{3}$$

In the newest [5,6] version of this IOP, we have: (1) used fiber optics to realize the required projection from P_1 onto P_2 (this greatly improves the system's accuracy as well as its mechanical and positional stability and reduces its size and weight); (2) placed the photodetector at P_3 in direct contact with the matrix mask at P_2 (this is possible when the 4 mm height of each photodetector is matched to the height of the matrix mask at P_2. This further reduces the size and weight of the system and makes it even more stable for airborne applications); (3) included a microprocessor system with an arithmetic logic unit (ALU), memory and hardwired multiplier in the electronic feedback loop (Figure 2), (this increases the system's flexibility and versatility); and (4) modified the feedback system and the iterative algorithm to incorporate an acceleration parameter to insure convergence of the iterative algorithm (Section 3).

When funding permits, we plan: (1) to increase the size of the P_1 input and the P_2 mask from its present 10×10 level; (2) to incorporate a real-time and reuseable spatial-light modulator electro-optical mask element (such as a CCD-addressed liquid crystal light valve [9]) into the system; and (3) to increase the repertoire of operations and applications for the system. In Sections 4-8, we advance the first description of a selected number of general operations and problems as well as applications for which this new optical processor can be used.

3. CONVERGENCE AND HANDLING BIPOLAR AND COMPLEX-VALUED DATA

In this section, we first detail how this non-coherent optical system can be used to operate on bipolar and complex-valued data [5,6]. Since the LED outputs at P_1, the transmittance of the mask elements at P_2 and the photodetector outputs at P_3 are all real and positive, and since the dynamic range of the mask at P_2 is finite, various scaling, biasing and data encoding techniques are required to process vectors

and matrices with bipolar and complex-valued data on the system. To handle complex-valued data, we partition the matrix $\underline{H}$ into its real and imaginary parts $\underline{H}_r$ and $\underline{H}_i$, respectively, as

$$\underline{H} = \begin{bmatrix} \underline{H}_r & -\underline{H}_i \\ \underline{H}_i & \underline{H}_r \end{bmatrix} , \tag{4}$$

where $\underline{H}_r$ and $\underline{H}_i$ are bipolar. This requires a P_2 mask with four times the space-bandwidth product of $\underline{H}$, an input LED array with twice the number of elements in $\underline{x}$ and a linear detector array with twice the number of elements present in $\underline{y}$.

To handle bipolar data, we decompose the input vector $\underline{x}$ into its positive and negative components. Let $\underline{a}^+$ and $\underline{a}^-$ denote the positive and negative components of the input vector $\underline{a}$ used in the actual system. The M elements a_{1m} and a_{2m} of $\underline{a}^+$ and $\underline{a}^-$ are generated according to

$$\left.\begin{aligned} a_{1m} &= 0.5\left(x_m + |x_m|\right) \\ a_{2m} &= 0.5\left(x_m - |x_m|\right) \end{aligned}\right\} , \tag{5}$$

where the x_m are the elements of the bipolar physical vector $\underline{x}$.

The system is then operated twice, first with $\underline{a}_1$ as the input and then with $\underline{a}_2$ as the input (with the same physical mask $\underline{B}$ used for all cycles). The outputs from the system on successive cycles are $\underline{B}\,\underline{a}_1$ and $\underline{B}\,\underline{a}_2$. These outputs are: (1) subtracted; (2) scaled by $(\overline{h} - \underline{h})$, where $\overline{h}$ and $\underline{h}$ are the maximum and minimum values of the elements of the matrix $\underline{H}$; and (3) biased by $\underline{h}\sum_{m=1}^{M} x_m$. The system's output $\underline{y}$ after two successive cycles is thus

$$\underline{y} = \underline{H}\,\underline{x} = \left(\overline{h} - \underline{h}\right)\left[\underline{B}\,\underline{a}_1 - \underline{B}\,\underline{a}_2\right] + \left(\underline{h}\sum_{m=1}^{M} x_m\right)[11\ldots1]^T . \tag{6}$$

To insure that the transmittance of the physical mask $\underline{B}$ has a transmittance for each element satisfying $0 \leq b_{mn} \leq 1$, we scale and bias $\underline{B}$ such that

$$b_{mn} = \frac{h_{mn} - \underline{h}}{\overline{h} - \underline{h}} , \tag{7}$$

where h_{mn} denotes the values of the elements of $\underline{H}$ and b_{mn} denotes the transmittances of the physical mask $\underline{B}$ placed at P_2.

The second issue to be detailed in this section is how rapid convergence of the iterative algorithm is insured. This is achieved by modifying our original iterative algorithm to include an acceleration parameter ω (as shown in Figure 1). The new

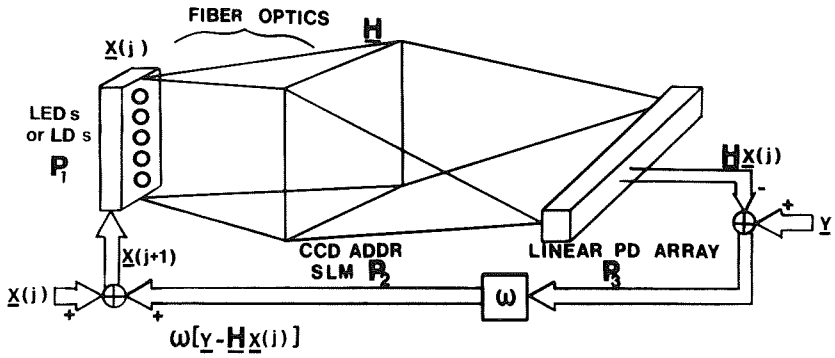

FIGURE 1 Schematic diagram of the iterative optical processor.

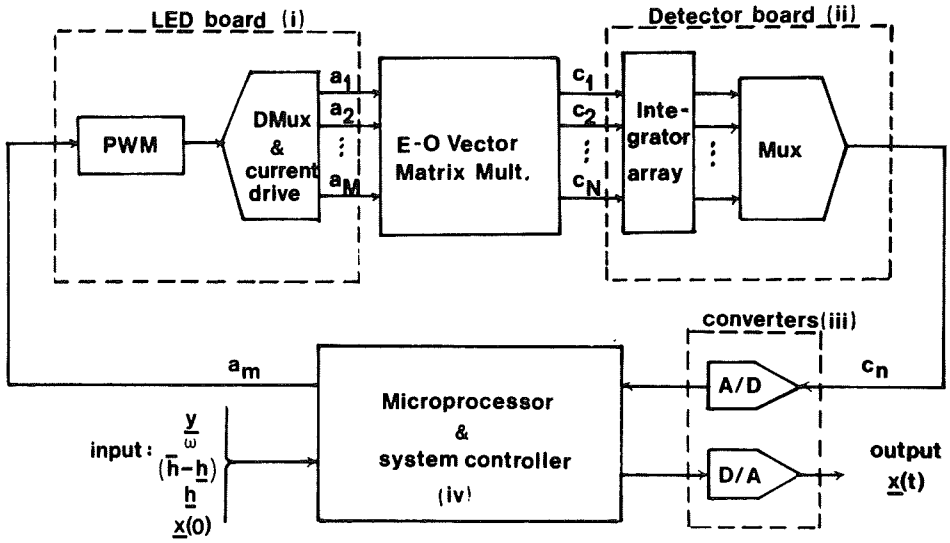

FIGURE 2 Schematic diagram of the microprocessor-based fiber-optic iterative optical processor.

system thus implements the iterative algorithm [4-6]

$$\underline{x}(j + 1) = \underline{x}(j) + \omega[\underline{y} - \underline{H}\,\underline{x}(j)]. \tag{8}$$

We select ω by one of the following criteria to insure rapid convergence of the algorithm. We can select

$$\omega = -1/\lambda_{max}, \tag{9}$$

where λ_{max} is the maximum eigenvalue, in absolute value, of the matrix $\underline{H}$, or we can obtain an approximation to (9) by [10]

$$\lambda_{max} \leq ||\underline{H}|| = \left[\sum_{m=1}^{M}\sum_{n=1}^{N} h_{mn}^2\right]^{1/2}, \tag{10}$$

where $||\underline{H}||$ is the Euclidean norm of the $(N \times N)$ matrix $\underline{H}$. The ω criterion resulting from the upper-bound in (10) can be modified by selecting $\omega = -K/\lambda_{max}$, where $K > 1$ is a constant selected empirically from analysis of a specific problem. (In several specific case studies that we have considered, $K \simeq 2$-3 was found to be adequate.) The microprocessor feedback system (Figure 2) performs the necessary scaling, biasing and preprocessing described by (5) and (7), the detector post-processing in (6), and the acceleration parameter selection noted in (10). In Section 6, we describe how the IOP itself can be used to calculate the acceleration parameter ω in (9).

4. SOLUTION OF SIMULTANEOUS LINEAR EQUATIONS

As our first general IOP application, we consider the use of the system of Figure 1 for the solution of simultaneous linear (difference, differential or algebraic) equations. The general iterative algorithm for the solution of linear difference equations is

$$\underline{x}(j + 1) = \underline{\Phi}\,\underline{x}(j) + \underline{f}\,, \tag{11}$$

where j is the discrete-time index, $\underline{f}$ is the forcing function equal to a constant vector (or, more generally, a vector of time-functions), $\underline{x}(j)$ is the state of the system, and $\underline{\Phi}$ is the open-loop system matrix.

The first application of the IOP of Figure 1 to the problem in (11) is as a dynamic system simulator. In this case, we model the physical system by the state-space differential equation model

$$\frac{d\underline{x}}{dt} = \underline{A}\,\underline{x} + \underline{b}\,, \tag{12}$$

where $\underline{A}$ is the open-loop system matrix and the vector $\underline{b}$ is a constant or function of

time. We then utilize a numerical analysis algorithm to discretize this system model. We illustrate the approach by the forward-Euler approximation [11]

$$\left.\frac{d\underline{x}(t)}{dt}\right|_{t=jT} \simeq \frac{\underline{x}(j+1) - \underline{x}(j)}{T} , \tag{13}$$

where T is the constant time-increment (or step-size) between discrete samples described by the index j. Substituting (13) into (12), we can simulate the physical system modeled by (12) on the IOP of Figure 1 by the linear iterative algorithm

$$\underline{x}(j+1) = [\underline{I} + T\,\underline{A}]\,\underline{x}(j) + T\,\underline{b} . \tag{14}$$

By analogy with (11), $[\underline{I} + T\,\underline{A}] = \underline{\Phi}$ and $T\,\underline{b} = \underline{f}$. Other numerical analysis algorithms, such as the trapezoidal rule, are possible. The most preferable ones appear to be the Runge-Kutta and predictor-corrector algorithms [12].

The second application of (11) on the IOP is the iterative solution of linear algebraic equations. In this case, the iterative system algorithm of (8) is used. When rearranged in the form in (1), we can identify the $\underline{\Phi}$ matrix and the $\underline{f}$ vector in (11) as $[\underline{I} - \omega\underline{H}] = \underline{\Phi}$ and $\omega\underline{y} = \underline{f}$. The algorithms in (8) and (1) have been used [2,13] to calculate the set of adaptive weights necessary in adaptive phased-array radar signal processing. While space does not allow us to elaborate on this application, we note that, in this case, the matrix $\underline{H}$ in (3) is the covariance matrix $\underline{M}$ of the far field input to the antenna, $\underline{y}$ is the steering vector $\underline{s}$ for the antenna and the unknown vector $\underline{x}$ to be determined is the set of adaptive weights $\underline{w}$ to be applied to the received signals to steer the antenna in the desired direction (defined by $\underline{s}$) and to null the noise field (defined in frequency, velocity, angle and range by $\underline{M}$).

Four iterative algorithms to solve (3) for $\underline{x}$ given by (2) can directly be identified. The first is the Richardson algorithm [14] of (1) implemented with the acceleration parameter ω as in (8). The remaining three algorithms can be described in a new and quite useful formulation by decomposing the matrix $\underline{H}$ into the sum of a diagonal matrix $\underline{D}$ and lower and upper triangular matrices $\underline{L}$ and $\underline{U}$ as

$$\underline{H} = \underline{D} - \underline{L} - \underline{U} . \tag{15}$$

In terms of the $\underline{D}$, $\underline{L}$ and $\underline{U}$, we can write the remaining three algorithms as [15]:

$$\text{Jacobi:} \quad \underline{x}(j+1) = \left[\underline{D}^{-1}(\underline{L} + \underline{U})\right]\,\underline{x}(j) + \underline{D}^{-1}\underline{y} \tag{16a}$$

$$\text{Gauss-Seidel:} \quad \underline{x}(j+1) = \left[(\underline{D} - \underline{L})^{-1}\underline{U}\right]\,\underline{x}(j) + (\underline{D} - \underline{L})^{-1}\underline{y} \tag{16b}$$

$$\text{Overrelaxation:} \quad \underline{x}(j+1) = \left\{(\underline{D} - \omega\underline{L})^{-1}[(1-\omega)\underline{D} + \omega\underline{U}]\right\}\,\underline{x}(j) + \omega(\underline{D} - \omega\underline{L})^{-1}\underline{y}. \tag{16c}$$

The choice of one of the four algorithms in (8) or (16) depends on many factors that are highly application and problem dependent (e.g., convergence of the algorithm, dy-

dynamic range of the mask, number of iterations required and the need for an adaptive mask).

5. LEAST-SQUARES PROBLEMS

The solution to many physical and statistical problems can be formulated in the general context of a least mean-squares problem. A model is postulated to approximate measured data and the parameters of the model are selected to minimize the mean-square error between the measured data and the model. Curve-fitting and linear and polynomial regression are classical examples of least mean-squares problems. We formulate and address below the solution on our IOP of this class of important problems.

We begin by reformulating our simultaneous linear algebraic equation problem solution in Section 4 as a least-squares problem. In the context of Figure 3, the input to the operator $\underline{H}$ is $\underline{x}$ and noise or an error source $\underline{\varepsilon}$ is present at the output such that the observable output $\underline{y}$ differs from $\underline{H}\,\underline{x}$. The problem is thus to find the $\underline{x}$ that minimizes the square of the difference

$$J = ||\underline{y} - \underline{H}\,\underline{x}||^2 \tag{17}$$

between $\underline{y}$ and $\underline{H}\,\underline{x}$.

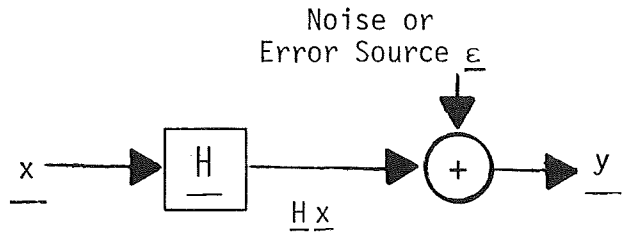

FIGURE 3 Schematic diagram of a least-mean squares problem formulation.

Upon expanding (17), we obtain

$$J = [\underline{y} - \underline{H}\,\underline{x}]^T [\underline{y} - \underline{H}\,\underline{x}] = \underline{y}^T\underline{y} - \underline{y}^T\underline{H}\,\underline{x} - \underline{x}^T\underline{H}^T\underline{y} + \underline{x}^T\underline{H}^T\underline{H}\,\underline{x}. \tag{18}$$

To minimize (18), we set the partial derivative or gradient of the scalar performance index in (17) with respect to the unknown vector $\underline{x}$ equal to zero. The resultant column vector is then the solution of

$$\left.\begin{aligned} -2\underline{H}^T\underline{y} + 2\underline{H}^T\underline{H}\,\underline{x} &= \underline{0} \\ \text{or} \\ \left[\underline{H}^T\underline{H}\right]\underline{x} &= \underline{H}^T\underline{y} \end{aligned}\right\} \tag{19}$$

In least-squares data processing, $\underline{y}$ is an M-vector, $\underline{H}$ is an (M × N) matrix and $\underline{x}$ is an N-vector, where $M \geq N$. We consider first the case when the number of unknowns N equals the number of equations M. In this case, $\underline{H}$ is a square matrix and the solution to (19) becomes

$$\underline{x} = \left[\underline{H}^T\underline{H}\right]^{-1}\underline{H}^T\underline{y} = \underline{H}^{-1}\underline{H}^{-T}\underline{H}^T\underline{y} = \underline{H}^{-1}\underline{y}\ , \tag{20}$$

where $\underline{H}$ is assumed by implication to be a non-singular matrix and therefore to be invertible. This occurs when the $\underline{y} = \underline{H}\,\underline{x}$ problem being solved is well-formulated as occurs in all practical engineering problems. Thus, in this case, the solution $\underline{x} = \underline{H}^{-1}\underline{y}$ minimizes the quadratic performance index in (17). For this case, we solve the least mean-square problem by our Richardson algorithm of (8).

A more interesting problem (corresponding to the practical case of curve-fitting) occurs when there are more equations M than unknowns N (i.e., when $M > N$). In this case, $\underline{H}$ is non-square and therefore non-invertible. We thus solve (19) by the new iterative algorithm

$$\underline{x}(j+1) = \underline{x}(j) + \omega\left\{\left[\underline{H}^T\underline{y}\right] - \left[\underline{H}^T\underline{H}\right]\underline{x}(j)\right\}\ . \tag{21}$$

This is equivalent to pre-multiplying (3) by $\underline{H}^T$ and applying our iterative Richardson algorithm in (8) to the solution of $[\underline{H}^T\underline{H}]\underline{x} = \underline{H}^T\underline{y}$ rather than $\underline{H}\,\underline{x} = \underline{y}$.

The least-squares fitting of experimental observations leads to a more general iterative vector-matrix solution. Suppose that we have L data points $\underline{p} = \{p_\ell\}$ and wish to approximate the set of observations $z(\underline{p})$, in the least-squares sense, by the finite linear combination

$$z(\underline{p}) \simeq \underline{c}^T\underline{\phi}(\underline{p}) \tag{22}$$

of basis functions $\underline{\phi} = \{\phi_n\}$ with the associated weighting coefficients $\underline{c} = \{c_n\}$. In (22), each of the N basis functions $\underline{\phi}$ are evaluated at the L data points $\{p_\ell\}$ and the relationship in (22) is approximate if there are more data points L than basis functions N. The errors or residuals $[z(\underline{p}) - \underline{c}^T\underline{\phi}(\underline{p})]$ of the curve-fit are the differences between the observed data $z(\underline{p})$ and the approximation $\underline{c}^T\underline{\phi}(\underline{p})$. According to the principle of least-squares, we select the weighting coefficients in (22) to minimize the sum-of-squares of the residuals. To find the coefficients $\underline{c}$, therefore, we minimize the mean-square difference

$$J = \sum_{\substack{\text{Data} \\ \{p_\ell\}}}\left[z(p_\ell) - \underline{c}^T\underline{\phi}(p_\ell)\right]^2\ . \tag{23}$$

We proceed as before. Upon setting $\partial J/\partial \underline{c} = \underline{0}$, we find

$$\underbrace{\left[\sum_{\substack{\text{Data}\\ \{p_\ell\}}} \underline{\phi}(p_\ell)\underline{\phi}^T(p_\ell)\right]}_{\underline{H}} \underbrace{\underline{c}}_{\underline{x}} = \underbrace{\left[\sum_{\substack{\text{Data}\\ \{p_\ell\}}} \underline{\phi}(p_\ell) z(p_\ell)\right]}_{\underline{y}} \tag{24}$$

which is again of the general form $\underline{H}\,\underline{x} = \underline{y}$, with the summation over the $\{p_\ell\}$ data points incorporated into the matrix and vectors in the algorithm in (21). When the basis functions $\underline{\phi} = \{\phi_n\}$ are specified, the matrix $\underline{H}$ can be precomputed and the problem solved by the iterative algorithm in (8) or (21). A simple discrete example of $\{\phi_n\}$ arises in the calculation of the best straight-line fit through a set of data points. In this case, $\phi_1 = 1$ and $\phi_2 = p$ and $z(p) = c_1 + c_2 p$. A simple continuous example case is the Fourier series expansion of $z(p)$, in which case the $\{\phi_n\}$ are the complex exponentials $\exp(\pm jn\omega_0 p)$.

In this section we have illustrated and formulated the least mean-squares problem as the iterative solution of the system of linear algebraic equations $\underline{H}\,\underline{x} = \underline{y}$ on the IOP. We recognize that the range of applications of our formulation includes curve-fitting, linear and nonlinear regression, state and parameter estimation, orbit determiniation and signal processing in control and communication engineering. We will address these applications in our future work.

6. DECONVOLUTION AND EIGENVALUE PROBLEMS

The need to implement a deconvolution frequently arises. In such applications (Figure 4), the measured output data $\underline{y} = \{y_m\}$ from a system characterized by the impulse response $\{h_m\}$ will be a modified version of the original input vector $\underline{x} = \{x_n\}$; i.e.,

$$y_m = \sum_{n=0}^{m} h_{(m-n)}\, x_n \,. \tag{25}$$

In (25), we assume that the system is causal, otherwise we appropriately shift the impulse response so that $h_m = 0$ for $m < 0$. The solution of (25) for the unknown input $\underline{x}$ is directly described by the vector-matrix equation $\underline{y} = \underline{H}\,\underline{x}$, where (for the case of $M = N = 3$) the matrix $\underline{H}$ is shown below:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 \\ h_1 & h_0 & 0 \\ h_2 & h_1 & h_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} . \tag{26}$$

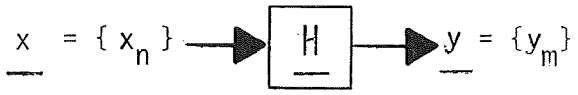

FIGURE 4 Simplified schematic diagram of a deconvolution processing problem.

In the general case of $M > N = 3$, the size of $\underline{H}$ grows, but $\underline{H}$ still has only three non-zero diagonals (with equal values h_0, h_1 and h_2, respectively, along each diagonal), and there are an additional $(M - N)$ elements with zero value added to $\underline{x}$. The solution of the deconvolution problem in (25) is thus directly realizable on the IOP by the iterative algorithm in (8) with the matrix mask in the form shown in (26). Once again, we see how different problems can be solved on the same IOP by different choices of the matrix mask and the iterative algorithm used.

Another quite general and useful matrix operation is the calculation of the eigenvalues λ_n and corresponding eigenvectors $\underline{\phi}_n$ of the $(N \times N)$ matrix $\underline{H}$. This operation can be performed by the iterative algorithm [16]

$$\underline{x}(j + 1) = \underline{H}\,\underline{x}(j) \ , \tag{27}$$

which is equivalent to (8) with the exogenous vector $\underline{y}$ set equal to zero. To see how (27) allows calculation of the λ_n and $\underline{\phi}_n$, we assume that $|\lambda_1|$ is the largest eigenvalue and that the eigenvalues are ordered, in absolute value, according to $|\lambda_1| > |\lambda_2| > \cdots\cdots > |\lambda_N|$. By singular value decomposition, we write $\underline{H}$ and the original initialization vector $\underline{x}_1(0)$ as

$$\left.\begin{aligned} \underline{H} &= \sum_{i=1}^{N} \underline{\phi}_i \lambda_i \underline{\phi}_i^T \\ \text{and} \\ \underline{x}_1(0) &= \sum_{\ell=1}^{N} a_\ell \underline{\phi}_\ell \end{aligned}\right\} \ . \tag{28}$$

After j iterations, $\underline{x}(j + 1) = \underline{H}^j \underline{x}_1(0)$ is obtained. Substituting (28) into this expression, we find (for large j)

$$\underline{x}(j + 1) = \underline{H}^j \underline{x}_1(0) \underset{\text{Large } j}{\sim} a_1 \lambda_1^j \underline{\phi}_1 \ . \tag{29}$$

We form the component-wise ratio $x_i(j + 1)/x_i(j)$ and from it find λ_1 (the largest eigenvalue of $\underline{H}$). We divide the denominator of the output vector $\underline{x}$ by the sum of the squares of its elements and thus obtain the principal eigenvector $\underline{\phi}_1$. We then use the new initial vector $\underline{x}_2(0) = \underline{x}_1(0) - a_1 \underline{\phi}_1$. Repeating the same iterative procedure in

(27), we then find λ_2 and $\underline{\phi}_2$. By continuing this process, all of the eigenvalues and eigenvectors of $\underline{H}$ can be determined in decreasing order. Modifications to this procedure allow us to find the eigenvector whose eigenvalue lies closest to a prescribed value [16]. Many other extensions of these techniques and applications of the above results are possible and will be the subject of future work.

7. SOLUTION OF MATRIX EQUATIONS

Many problems involve matrix-matrix multiplication. There are two ways to realize a matrix-matrix multiplication on our vector-matrix multiplier. In the case of large matrices, the preferable technique is to form the product $\underline{Y} = \underline{M} \cdot \underline{N}$ of the matrices $\underline{M}$ and $\underline{N}$ by feeding sequentially the columns $\underline{n}_n$ of $\underline{N}$ as successive input vectors. We thus realize the matrix-matrix product by performing N vector-matrix multiplications; i.e.,

$$\underline{Y} = \underline{M} \cdot \underline{N} = \underline{M}[\underline{n}_1\, \underline{n}_2 \cdot \cdot \cdot \underline{n}_N] \ , \tag{30}$$

where the matrix output $\underline{Y}$ appears sequentially as N column vectors; i.e., $\underline{Y} = [\underline{y}_1\, \underline{y}_2 \cdot \cdot \cdot \underline{y}_N]$. A second technique is to vectorize the matrix $\underline{N}$ into an N^2 element vector whose first N elements are the elements of $\underline{n}_1$ and the next N elements are the elements of $\underline{n}_2$,... . The matrix $\underline{M}$ is formatted as an $(N^2 \times N^2)$ matrix whose diagonal blocks are replications of the matrix $\underline{M}$; i.e.,

$$\underline{M} \cdot \underline{N} = \begin{bmatrix} \underline{M} & & & \\ & \underline{M} & & \\ & & \ddots & \\ & & & \underline{M} \end{bmatrix} \begin{bmatrix} \underline{n}_1 \\ \underline{n}_2 \\ \vdots \\ \underline{n}_N \end{bmatrix} = \begin{bmatrix} \underline{y}_1 \\ \underline{y}_2 \\ \vdots \\ \underline{y}_N \end{bmatrix} . \tag{31}$$

With a matrix-matrix multiplier realized by (30) or (31), we can thus use the IOP to solve matrix-matrix equations such as

$$\underline{H}\,\underline{X} = \underline{Y} \tag{32}$$

by vectorizing the matrix $\underline{Y}$ or by operating the system on sequential cycles with the column vectors of $\underline{Y}$ as successive inputs. A particularly useful operation that is now possible is matrix inversion. In this case, we solve (32) using (8), but with $\underline{Y} = \underline{I}$ (the identity matrix). A final useful matrix equation that frequently arises is the solution for the embedded matrix $\underline{X}$ in an equation of the form

$$\underline{A}\,\underline{X}\,\underline{B} = \underline{Y} \ . \tag{33}$$

In this case, we write the matrices $\underline{Y}$ and $\underline{X}$ as the column vectors $\underline{C}[\underline{Y}]$ and $\underline{C}[\underline{X}]$, whose elements are the lexigraphically-ordered elements of the matrices $\underline{Y}$ and $\underline{X}$, respectively. We solve (33) for $\underline{X}$ by writing (33) in the form

$$[\underline{A} \otimes \underline{B}^T]\underline{C}[\underline{X}] = \underline{C}[\underline{Y}] \ , \tag{34}$$

where $\otimes$ denotes the outer or Kronecker product. For the case of the (2 × 2) matrix $\underline{A}$,

$$\underline{A} \otimes \underline{B}^T = \begin{bmatrix} a_{11}\,\underline{B}^T & a_{12}\,\underline{B}^T \\ a_{21}\,\underline{B}^T & a_{22}\,\underline{B}^T \end{bmatrix} \tag{35}$$

Solution of the Lyapunov matrix equation $\underline{X}\,\underline{A} + \underline{A}^T\underline{X} = \underline{Y}$ is now possible on the vector-matrix IOP.

8. SOLUTION OF NONLINEAR MATRIX EQUATIONS

In Section 7, we described how we have broadened the repertoire of operations achievable on the IOP to include the full-class of linear and embedded matrix equations. As our final general iterative vector matrix operation, we consider its use in the solution of nonlinear matrix equations. The need for the solution of such equations arose in our original use of the system for adaptive phased-array radar processing [2] and in our present optimal control linear-quadratic-regulator application [4].

The general problem of two quadratic equations in the two unknowns p and q can be written as

$$f_n(p,q) = a_n p^2 + b_n pq + c_n q^2 + d_n p + e_n q + r_n = 0 \tag{36}$$

for n = 1 and 2. We rewrite this pair of two nonlinear equations in vector form as

$$\underline{f}[\underline{x}] = \underline{0} \tag{37}$$

where $\underline{f}[\underline{x}] = [f_1(\underline{x})\ f_2(\underline{x})]^T$ and $\underline{x} = [p\ q]^T$. We then solve (37) by the Newton-Raphson iterative algorithm

$$\underline{x}(j+1) = \underline{x}(j) - \left[\frac{\partial \underline{f}}{\partial \underline{x}}\right]^{-1}_{\underline{x}(j)} \underline{f}[\underline{x}(j)]. \tag{38}$$

The solution of (38) requires <u>two</u> iterative loops. The Jacobian matrix $\underline{J} = [\partial \underline{f}/\partial \underline{x}]$ is stored algebraically. At each iteration, $\underline{J}$ is evaluated numerically with dedicated electronics at the last iterate $\underline{x}(j)$. The inverse matrix $[\underline{J}[\underline{x}(j)]]^{-1}$ is evaluated on the optical vector-matrix system in an <u>inner</u> iterative loop. The new $\underline{x}(j+1)$ iterate is then evaluated optically in an <u>outer</u> iterative loop.

Three immediate applications for such a nested two-loop iterative algorithm

arise. The first occurs in the implementation of the overrelaxation linear algebraic equation solution of (16c) with $[\underline{D} - \omega\underline{L}]$ on the left-hand side of the equation. In this case, a matrix-vector multiplication is required to obtain the external vector to be added. The second case arises in the use of (8) to solve for the adaptive weights $\underline{w}$ for a phased-array radar whose noise field is described by the covariance matrix $\underline{M}$ and the direction of its main lobe is defined by the steering vector $\underline{s}$. In this case, we solve the vector-matrix equation $\underline{M} \cdot \underline{w} = \underline{s}$. When the noise distribution varies, the matrix mask $\underline{M}$ must be updated.

The IOP application we are presently considering under support from the NASA-Lewis Research Center is the use of the IOP in the solution of the linear-quadratic-regulator (LQR) problem of optimal control. In this application, the IOP is used to calculate the optimal controls to be applied to an F100 aircraft engine. To determine these control signals, we must solve the algebraic Riccati equation and calculate the LQR feedback gains. In this application, we use: (1) the vectorization of a matrix; (2) the Kronecker product technique; and (3) the nested inner and outer loop system to solve the nonlinear algebraic Riccati equation [4].

9. SUMMARY

In this paper, the general-purpose nature of an iterative vector-matrix processor has been emphasized. This system is capable of solving a wealth of general purpose applications. General operations described included: linear difference and differential equations, linear algebraic equations, matrix equations, matrix inversion, nonlinear matrix equations, deconvolution and eigenvalue and eigenvector computations. Engineering applications being addressed for these different operations and for the IOP are: adaptive phased-array radar, time-dependent system modeling, deconvolution and optimal control.

ACKNOWLEDGEMENTS

The authors thank Dr. Mark Carlotto for fabrication of the laboratory IOP and for the demonstration of its use in many of the applications described in this paper. We also thank the Air Force Office of Scientific Research for intermediate support of the general applications of this architecture and the NASA-Lewis Research Center for present support of this data processing concept for linear-quadratic-regulator optimal control applications.

REFERENCES

1. J. Goodman et al, Optics Letters, 2, 1 (1978).

2. D. Psaltis et al, SPIE, 180, 114 (1979).

3. D. Psaltis et al, Optics Letters, 4, 348 (1979).

4. D. Casasent et al, SPIE, 295, (1981).

5. M. Carlotto and D. Casasent, A Fiber Optic Microprocessor-Based Vector Matrix Optical Processor. Applied Optics, [1982].

6. M. Carlotto and D. Casasent, An Iterative Optical Vector-Matrix Processor. Transformations in Optical Signal Processing, W.T. Rhodes, J.R. Fienup, and B.E.A. Saleh, editors, Soc. of Photo-Optical Instrumentation Engineers, Dellingham, Washington, [1982].

7. P. Mengert et al, U.S. Patent #3,525,856, filed Oct, 6, 1966.

8. A. Edison and M. Noble, Optical Analog Matrix Processors, AD646060 (Nov. 1966).

9. J. Grinberg et al, Proc. Soc. Photo. Instr. Engrs., 128, 253 (1977).

10. E. Kreyszig, Advanced Engineering Mathematics, John Wiley & Sons, Inc., New York, 1972, p. 686.

11. L.O. Chua and P.M. Lin, Computer-Aided Analysis of Electronic Circuits, Prentice-Hall, Englewood Cliffs, New Jersey, 1975, Chapter 1.

12. L.O. Chua and P.M. Lin, Computer-Aided Analysis of Electronic Circuits, Prentice-Hall, Englewood Cliffs, New Jersey, 1975, Chapter 11.

13. D. Casasent, Optical Processing for Adaptive Phased-Array Radar - Final Technical Report. RADC-TR-81-152, June 1981.

14. L. Richardson, Philos. Trans. Roy. Soc. London, Ser A210, 307 (1910).

15. R.S. Varga, Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs, New Jersey, 1962, pp. 56-59.

16. B. Kumar and D. Casasent, Eigenvector Determination by Iterative Optical Methods. Applied Optics, [Nov. 1981].