

Correspondence

Linear Hopfield Networks and Constrained Optimization

G. G. Lendaris, K. Mathia, and R. Sacks

Abstract—It is shown that a Hopfield neural network (with linear transfer functions) augmented by an additional feedforward layer can be used to compute the Moore–Penrose generalized inverse of a matrix. The resultant augmented linear Hopfield network can be used to solve an arbitrary set of linear equations or, alternatively, to solve a constrained least squares optimization problem. Applications in signal processing and robotics are considered. In the former case the augmented linear Hopfield network is used to estimate the “structured noise” component of a signal and adjust the parameters of an appropriate filter on-line, whereas in the latter case it is used to implement an on-line solution to the inverse kinematics problem.

Index Terms—Author, please supply index terms. E-mail keywords@ieee.org for info.

I. INTRODUCTION

Over the past few years a number of authors have observed that a linear Hopfield neural network [6], [7], augmented by a feedforward layer can be used to solve a set of simultaneous equations [17]. Unfortunately, since these techniques solve the equation via a convergent Neumann series, they do not compare favorably with modern numerical methods in a classical computing environment. They are, however, well suited for implementation on the new generation of special purpose analog [17] and digital [15] neural network hardware which has been optimized for neural computations—machines which implement the required dot products and transfer functions at high speed and are designed to exploit the intrinsic massive parallelism inherent in a neural network [12]. The purpose of this note is to verify this observation and describe applications in signal processing and robotics where the augmented linear Hopfield network implemented on a special purpose neural network processor has proven to be useful.

Although the required network can be derived from first principles in a number of special cases, the general case follows from a classical theorem on Moore–Penrose generalized inverses [14].

Let A be an $n \times m$ matrix, A^* its Hermitian conjugate, and $0 < \alpha < 2/c$, where $c = \max \text{eigenvalue of } A^*A$. Then the following equalities hold, with convergence assured:

$$A^+ = \alpha \sum_{k=0}^{\infty} (I_m - \alpha A^*A)^k A^* \quad (1)$$

$$A^+ = \alpha \sum_{k=0}^{\infty} A^*(I_n - \alpha AA^*)^k \quad (2)$$

Manuscript received September 4, 1995; revised December 2, 1996, March 15, 1997, and March 9, 1998. This work was supported by the Naval Command, Control, and Ocean Surveillance Center, NR&D Division, under Contract N60001-90-C-7021 and the Office of Naval Research under Contract N00014-91-C-0268.

G. G. Lendaris is on sabbatical from Portland State University, Portland, OR 97207 USA.

K. Mathia is with Equipe Technologies, Sunnyvale, CA 94086 USA.

R. Sacks is with the Accurate Automation Corp., Chattanooga, TN 37421 USA (e-mail: rsacks@gte.net).

Publisher Item Identifier S 1083-4419(99)00902-4.

Here, A^+ is the Moore–Penrose generalized inverse of A , I_m , is an identity matrix of dimension m , and I_n is an identity matrix of dimension n .

Note, in practice one takes $0 < \alpha < 2/\text{tr}(A^*A) \leq 2/c$ to eliminate the need for computing c in the above theorem.

Since a linear (continuous state, discrete time) Hopfield network with $p \times p$ weight matrix, W , realizes the infinite series $\sum_{j=0}^{\infty} W^j$ when it is convergent, the generalized inverses of (1) and (2) may be realized by the augmented linear Hopfield networks of Fig. 1(a) and (b), respectively.

To evaluate the performance of the augmented linear Hopfield network in this application, it was implemented on a single accurate automation neural network processor (NNP), a special purpose coprocessor optimized for neural network computations [15], and compared with the CLAPACK function *agevsu*(.) [1] running on a 2 (R4400) processor SGI Onyx and a 4 (R3000) processor SGI 340VGX, for matrices of dimension 2 through 128. This yielded the relative performance curves of Fig. 2. Although one must be careful to take cognizance of the differing processor speeds of the three machines (35 MHz for the NNP and the 340VGX versus 200 MHz for the Onyx) and the numerical accuracy of the SGI's versus the 16 bit precision of the NNP, these results are indicative of the potential of the augmented linear Hopfield algorithm, when implemented on a machine which has been optimized for neural computations.

Although we did not run any benchmarks for this problem with multiple neural network processors, the NNP is designed to operate with up to 10 processors running in an MIMD parallel processing mode. To a first order approximation (which has been verified in other applications [13]), a parallel NNP system yields “near linear” performance improvements for up to $p = (f+1)/4$ processors [15] at which point bus contention takes over and throughput saturates. Here, f is the average fan-in of the neural network (equal to the dimension of the matrix in the present application). As such, a further improvement in the performance of the augmented linear Hopfield algorithm can be anticipated for large matrices, when it is implemented on hardware which takes full advantage of the massively parallel nature of the neural network.

II. OPTIMAL FILTERING

Our first example for application of the above ideas is a filtering problem, illustrated in Fig. 3. Here, X_k represents the input to the filter at time k , comprising an information-bearing component (the “signal”), X_{sk} , and a structured noise component, X_{nk} . The latter is assumed to represent contaminating phenomena which can be modeled by an autoregressive formula [5]

$$\hat{X}_{nk} = \sum_{i=1}^m a_i X_{n(k-i)}, \quad k \geq m+1. \quad (3)$$

Depending on the application, the signal and autoregressive coefficients may be real or complex.

An adaptive filter of this form has been developed for a tracking context [10] to distinguish targets from background and noise. In this context the sequence X_k is complex, containing both in-phase and quadrature components, $X_k = I_k + i \cdot Q_k$, and the a_i of (3) are arbitrary complex coefficients. An implementation using a standard linear Hopfield network to perform this task was reported in [10].

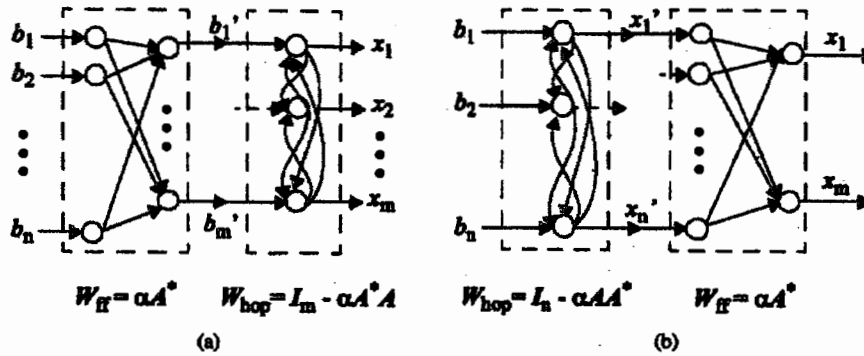


Fig. 1. Augmented linear Hopfield networks.

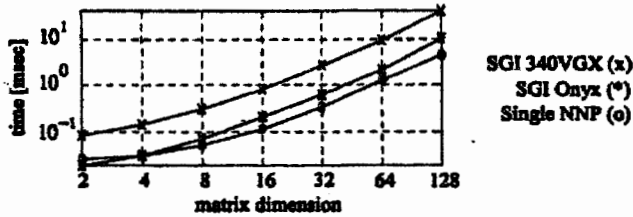
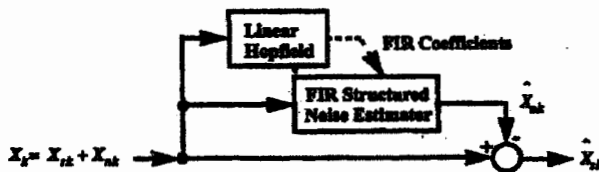
Fig. 2. Speed of matrix inversion: augmented linear Hopfield network on a single NNP versus the CLAPACK function *sgesv*(.) on a two-processor SGI onyx and a four-processor SGI 340VGX.

Fig. 3. Neural-network-based adaptive filter to extract signal in context of structured noise.

Unfortunately, while this approach yielded an excellent predictor of the structured-noise component, it also contained a good prediction of the *signal* as well, and, as such, failed to yield a significant improvement in the signal-to-noise ratio. In the present context an improved signal-to-noise ratio is obtained with the aid of an augmented linear Hopfield network. To this end, we formulate the problem to be solved with a *constraint* that the filter “ignore” certain known characteristics of the signal [10]. This formulation leads to a nonpositive-definite matrix to be inverted, hence the need for an augmented linear Hopfield network, and a filter with improved signal-to-noise ratio and real-time adaptation to changing background and noise conditions.

Our goal is to perform an on-line estimate of the noise-process coefficients $a = [a_1, a_2, \dots, a_m]^T$ that is optimal in some sense, and provide them to the FIR filter which uses these coefficients to estimate the structured noise at time k . The resulting estimate, \hat{X}_{nk} , is then subtracted from the input, X_k , to (hopefully) yield a good estimate of the signal \hat{X}_{sk} at time k . Without the “ignore the signal” constraint the coefficients a that optimally model background and noise data may be selected by minimizing

$$J(a) = \frac{1}{2} \sum_{k=m+1}^M \|X_k - \hat{X}_k\|^2$$

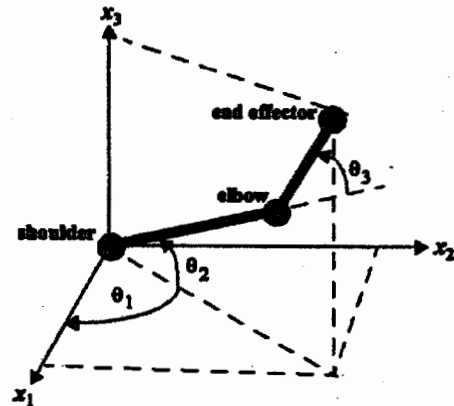


Fig. 4. Coordinate systems for a robot manipulator with three joint angles.

$$= \frac{1}{2} \sum_{k=m+1}^M \left\| X_k - \sum_{i=1}^m a_i X_{k-1} \right\|^2 \quad (4)$$

over all possible complex coefficient vectors $a = [a_1, a_2, \dots, a_m]^T$. Of course, this is a standard least-squares problem which, after taking into account the complex nature of the coefficients reduces to the solution of the linear equation

$$U^* + T a = 0 \quad (5)$$

where the complex m -vector U and the complex $m \times m$ matrix T are given by

$$U_i = \sum_{k=m+1}^M (X_k X_{k-1}^*) \text{ and } T_{ij} = \sum_{k=m+1}^M (X_{k-1} X_{k-j}^*). \quad (6)$$

Since T is Hermitian and positive definite this equation can be solved with a standard linear Hopfield network [8], [9], [16] without the augmenting layer of Fig. 1.

While the above approach yielded an excellent predictor for the structured noise when applied in the tracking context [10], it often also contained a good prediction of the *signal* as well, and, as such, the filter did not significantly improve the signal-to-noise ratio. To address this problem, a constraint is added to the optimization problem which, in effect, says “predict the structured noise as well as possible, being mindful of what is known about the expected signal(s).” To this end, we assume that the signal from a target has *constant magnitude*, and a *phase that has a constant rate of change* over the extent of the target, i.e., $\hat{X}_{sk} = c e^{a+ibk}$. Although not precisely true, the constant magnitude and the linear (afine) component of the phase typically dominate the return from a target.

Moreover, as a first approximation, it is reasonable to assume that the phase is constant over the relatively short m -sample interval seen by the filter, in which case the signal representing the target may be approximated by $\hat{X}_{sk} = ce^{\alpha} = C$. Accordingly, a pure signal from a target would yield

$$\sum_{i=1}^m a_i X_{s(k-i)} = \sum_{i=1}^m a_i C = C \sum_{i=1}^m a_i. \quad (7)$$

As such, to force the filter to ignore the signal we minimize (4) subject to the constraint that

$$\sum_{i=1}^m a_i = 0. \quad (8)$$

With the aid of the Lagrange multiplier theorem, this constrained optimization problem can be converted into an equivalent unconstrained optimization that minimizes

$$J(\mathbf{a}) + \lambda \sum_{j=1}^m a_j \quad (9)$$

over all \mathbf{a} and λ . As before, the solution is obtained by standard least squares techniques resulting in the $m+1$ dimensional complex equation

$$T_* \mathbf{a}_* + U_* = 0 \quad (10)$$

where

$$U_* = \begin{bmatrix} U \\ 0 \end{bmatrix}, \quad T_* = \begin{bmatrix} T & \mathbf{1} \\ \mathbf{1}^T & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{a}_* = \begin{bmatrix} \mathbf{a} \\ -\lambda \end{bmatrix}.$$

Unlike the solution of the unconstrained problem of (5), where T is Hermitian positive definite, T_* is Hermitian but not positive definite. As such, the augmented linear Hopfield network of Fig. 1 is required for its solution. Indeed, the authors have extensively tested this algorithm on actual target tracking data with excellent results. Moreover, when implemented on the special purpose neural network processor described above the algorithm can be implemented in real time, providing an improved signal-to-noise ratio and real-time adaptation to changing background and noise conditions.

III. INVERSE KINEMATICS

Our second application is a tracking control system for a robot manipulator illustrated in Fig. 4. For such manipulators, the control objective is usually to track a prescribed end-effector trajectory, where the trajectory is typically given in Cartesian coordinates $\mathbf{x} = [x_1, x_2, x_3]^T$. Unfortunately, the control system operates on the joint angles $\Theta = [\theta_1, \theta_2, \dots, \theta_n]^T$, and therefore, we are obliged to compute a corresponding trajectory in Θ -space to provide to the controller. Fig. 4 illustrates the relationships between the two coordinate systems for a manipulator with three adjustable joint angles.

Manipulator *kinematics* take into account the geometry of robot arm motions (as opposed to their dynamics). In mathematical terms, the manipulator's *forward* kinematics define the mapping from joint-angle space to Cartesian space, i.e., from control inputs to end-effector position. The trajectories in the two reference frames are related by

$$\mathbf{x}(t) = f(\Theta(t)) \quad (11)$$

where $f(\cdot)$ is a set of nonlinear equations (defined by the forward kinematics), \mathbf{x} is typically a three-dimensional vector and $\Theta(t)$ is an n -dimensional vector, where n is the number of adjustable joint angles [4]. If the $\Theta(t)$ trajectory were given, the forward computation of $\mathbf{x}(t)$ would be a straightforward task; however, the *inverse* process of determining $\Theta(t)$ from a given $\mathbf{x}(t)$, is substantially more difficult,

both because of the numerical complexity of the inversion process and because the equations may be under determined.

A common method to facilitate the solution of a nonlinear system of equations along a trajectory is to linearize them about selected operating points, and solve the resulting linear equations for small excursions from the operating points [4]. Solutions of the inverse kinematics problem typically use this approach, where a linear mapping from velocities in joint angle space to velocities in Cartesian space is used as basis for controlling robot manipulators. Thus, the problem is treated via time derivatives, and we differentiate (11) yielding

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = \frac{d}{dt} [f(\Theta)] = J(\Theta) \frac{d\Theta}{dt} = J(\Theta) \dot{\Theta} \quad (12)$$

where $J(\Theta)$ is the Jacobian matrix of f evaluated at Θ which may be solved for $\dot{\Theta}$ if the dimensionality of the joint space coincides with that of the Cartesian space.

If, however, the joint space of a redundant robot manipulator has a higher dimensionality than the Cartesian target space the Jacobian matrix is not square, and the solution to (12) is not unique. In this case one may define a criterion function and select the "best" $\dot{\Theta}$ which satisfies the constraint of (12) [11]. A typical criterion is to select $\dot{\Theta}$ to minimize some measure of the position error. In addition, the designer may also desire to keep the joint-angle velocities as small as possible.

A criterion function that embodies these two considerations is

$$E = \frac{1}{2} (\|\dot{\mathbf{x}}_*\|^2 + \epsilon \|\dot{\Theta}\|^2) \quad (13)$$

where $\mathbf{x}_*(t) = \mathbf{x}(t) - \mathbf{x}_d(t)$ is the position error of the end effector at time t and $\mathbf{x}_d(t)$ is the desired position. Using (12), this criterion function becomes

$$2E = (J(\Theta) \dot{\Theta} - \dot{\mathbf{x}}_d)^T (J(\Theta) \dot{\Theta} - \dot{\mathbf{x}}_d) + \epsilon \dot{\Theta}^T \dot{\Theta} \\ = \dot{\Theta}^T A(\Theta) \dot{\Theta} - 2b^T(\Theta) \dot{\Theta} + \mathbf{x}_d^T \mathbf{x}_d \quad (14)$$

where $A = \epsilon I + J^T J$, $b = J^T \dot{\mathbf{x}}_d$, and I is the $(n \times n)$ identity matrix. Upon differentiation, we conclude that the solution of this problem reduces to the solution of the set of linear equations

$$-A(\Theta) \dot{\Theta} + b(\Theta) = 0 \quad (15)$$

at each sample point along the prescribed (differential) trajectory, $\dot{\mathbf{x}}_d$. Moreover, since A is positive definite a real-time solution of (15) can be generated via a standard linear Hopfield network without augmentation. Simulations using this approach are shown in Fig. 5. Note the offset of the actual end-effector trajectory from the desired trajectory.

An alternative to the above formulation which eliminates the offset between the actual and desired trajectories is to compute $\dot{\Theta}$ via

$$\dot{\Theta} = J(\Theta)^+ \dot{\mathbf{x}} \quad (16)$$

where $J(\Theta)^+$ is the Moore-Penrose generalized inverse of $J(\Theta)$. Since the inverse kinematics problem is typically under determined, the Moore-Penrose generalized inverse yields an exact solution of (12) minimizing $\|\dot{\Theta}\|$ over all such solutions. As such, it eliminates the offset between the actual and desired trajectories, while still minimizing joint velocity to the maximum extent possible without introducing additional trajectory error.

Given the results of Section I, the required generalized inverse can be computed in real-time with the aid of an augmented linear Hopfield network, where, one defines the weights of the Hopfield and feedforward layers of the network by $W_{hop} = I - \alpha J^T J$ and $W_{ff} = \alpha J^T$, respectively. The performance of this alternative inverse kinematics algorithm is illustrated in Fig. 6.

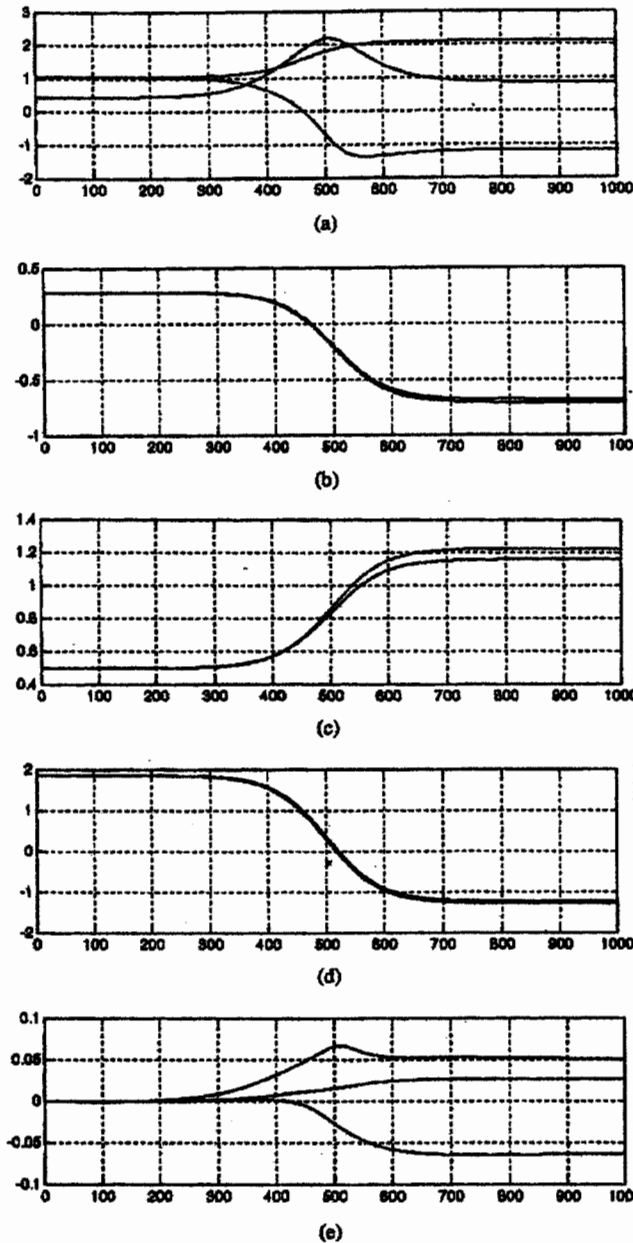


Fig. 5. (a) Joint angles generated by error criterion minimization algorithm; (b), (c), and (d) end effector tracking in the three Cartesian coordinates; and (e) end effector errors.

An inverse kinematics system employing an augmented linear Hopfield network was implemented on NASA MSFC's extendable stiff arm manipulator (ESAM). Our neurocontrol system for the ESAM robot included two neural network based control loops as shown in Fig. 7: the first loop implements the augmented linear Hopfield based inverse kinematics system, while the second implements a neural adaptive joint controller described in [2] and [3].

IV. CONCLUSIONS

An augmented linear Hopfield network, capable of implementing a full Moore-Penrose generalized inverse for an arbitrary matrix, has been developed and applied to two real-time computational problems. In the case of the signal processing problem structured noise is

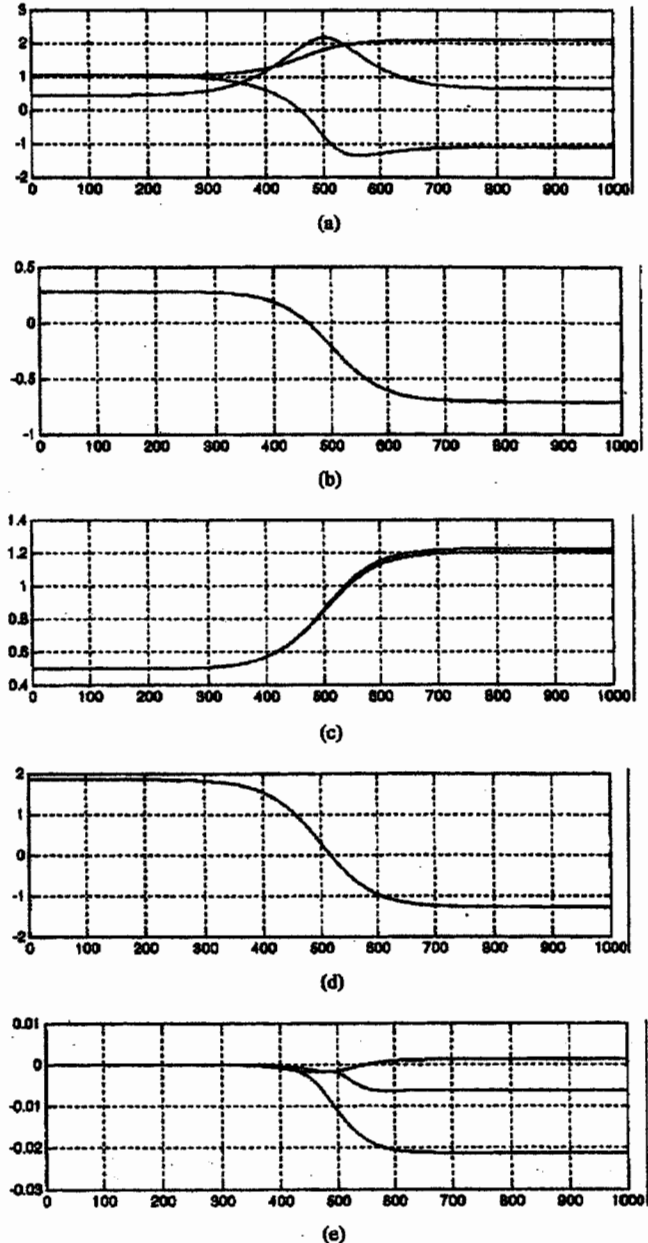


Fig. 6. (a) Joint angles generated by the generalized inverse algorithm; (b), (c), and (d) end effector tracking in the three Cartesian coordinates; and (e) end effector errors.

observed on-line and used to define the weights of an augmented linear Hopfield network which adaptively produces the coefficients of an appropriate filter. In the second example an augmented linear Hopfield network is used to solve the inverse kinematics problem for a robot arm at each point along a specified end-effector trajectory. Note, in both examples the power of the linear Hopfield network lies with the fact that the data which defines the problem is received in real-time and the weights of the Hopfield network are adapted on-line. Indeed, if the required data was available *a priori* one could compute the generalized inverse off-line and implement it with a simple feedforward network. For an on-line application, however, the weights of the linear Hopfield network can be readily updated at each time step and the Hopfield iteration can be efficiently implemented

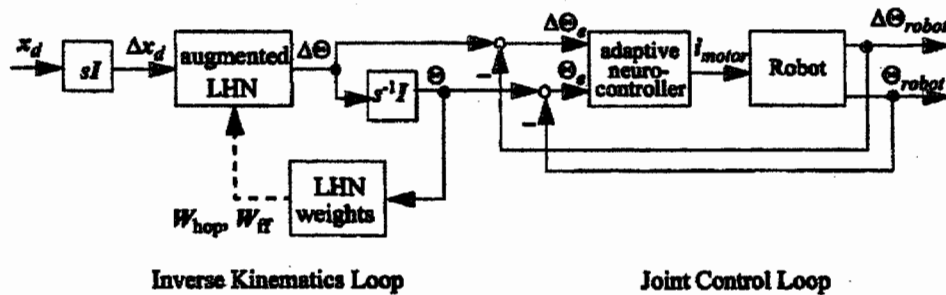


Fig. 7. Neurocontrol system for tracking an end-effector trajectory.

on an appropriate parallel processor if necessary.

REFERENCES

- [1] B. Anderson *et al.*, *An LAPACK User's Guide*, Soc. Industrial Applied Math., Philadelphia, PA, 1995.
- [2] C. Cox, J. Edwards, R. Saecks, R. Pap, and K. Mathia "Adaptive semi-autonomous robotic neurocontroller," in *Proc. 1994 SPIE Conf.: Applications Artificial Neural Networks V*, Orlando, FL, SPIE, Apr. 1994, vol. 2243, pp. 440-449.
- [3] C. Cox, M. Lothers, R. Saecks, and R. Pap, "A neurocontroller for robotic applications," in *Proc. 1992 IEEE Int. Conf. Systems, Man, Cybernetics*, Chicago, IL, Oct. 1992, pp. 712-716.
- [4] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Reading, MA: Addison-Wesley, 1989.
- [5] S. Haykin, W. Stehwiien, C. Deng, P. Weber, and R. Mann, "Classification of radar clutter in an air traffic control environment," *Proc. IEEE*, vol. 79, no. 6, pp. 742-772, 1991.
- [6] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Acad. Sci.*, 1982, vol. 79, pp. 2554-2558.
- [7] ———, "Neurons with graded response have collective computational properties like those of two-state neurons," in *Proc. Nat. Acad. Sci.*, 1984, vol. 81, pp. 3088-3092.
- [8] J. J. Hopfield and D. W. Tank, "Neural computation of decision optimization problems," *Biol. Cybern.*, vol. 52, pp. 141-152, 1985.
- [9] M. F. Kelly, P. A. Parker, and R. N. Scott, "Myoelectric signal analysis using neural networks," *IEEE Eng. Med. Biol. Mag.*, pp. 61-64, Mar. 1990.
- [10] G. G. Lendaris, R. M. Pap, R. E. Saecks, C. R. Thomas, and R. M. Akita, "Hardware neural network implementation of tracking system," in *Neural Networks for Signal Processing IV: Proc. of the 1994 IEEE Workshop*, J. Viontzos, J. N. Hwang, and E. Wilson, Eds., Ermioni, Greece, Sept. 1994, pp. 451-460.
- [11] K. Mathia and R. E. Saecks, "Inverse kinematics via linear dynamic networks," in *Proc. 1994 World Congr. Neural Networks*, San Diego, CA, June 1994, vol. II, pp. 47-53.
- [12] K. Mathia, "Solution of linear equations and a class of nonlinear equations using recurrent neural networks," Ph.D. dissertation, Portland State Univ., Portland, OR, 1996.
- [13] K. Mathia and R. Saecks, "Benchmarking an MIMD neural network processor," in *Proc. 1996 World Congr. Neural Networks*, San Diego, CA, Jan. 1996, pp. 1321-1326.
- [14] C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and Its Applications*. New York: Wiley, 1971.
- [15] R. E. Saecks, K. Priddy, R. M. Pap, and S. Stowell, "On the design of the MIMD neural network processor," in *Proc. 1994 SPIE Conf.: Applications Artificial Neural Networks V*, Orlando, FL, SPIE, Apr. 1994, vol. 2243, pp. 318-323.
- [16] D. W. Tank and J. J. Hopfield, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 533-541, 1986.
- [17] J. Wang and H. Li, "Solving simultaneous linear equations using recurrent neural networks," *Inf. Sci.*, vol. 76, no. 3/4, pp. 255-277, 1994.