

# Linear Hopfield Networks, Inverse Kinematics and Constrained Optimization

Karl Mathia, Richard Saeks, George G. Lendaris\*  
Accurate Automation Corporation  
7001 Shallowford Road  
Chattanooga, TN 37421

*Abstract. Methods for the design of different types of linear Hopfield networks are presented. The resulting neural networks are guaranteed to converge to their stable equilibrium, i.e. to solutions of the linear equations implicitly represented by the network. The construction of a step size is introduced, which allows convergence of the dynamic process at or near maximum rate. This work is a continuation of [7], and as an application example a neural network solution to the inverse kinematics problem is described.*

## 1 Introduction

Linear, recurrent networks have been frequently used as the basis for the solution of optimization problems [5][7][8]. The networks cited are Hopfield networks with linear transfer functions, therefore we call this variant a Linear Hopfield Network. In the present paper we indicate the applicability of these networks to **constrained** optimization problems. As application examples we offer solutions to the inverse kinematics problem in robotics. To this end we demonstrate the hybrid design of a recurrent *linear Hopfield network*, augmented with an *additional feedforward layer*, which solves the equations for a constrained optimization problem in a least square sense. More generally, such a network can be configured to solve an arbitrary set of linear equations with either real or complex coefficients. A general setting of this approach, including proofs for different network types, is represented in [6].

The control objective for robot manipulators (exemplified in Figure 4) is usually to track a prescribed end-effector trajectory  $y(t)$ , typically given in Cartesian space, while the control system operates in joint angle space. The two spaces are related by a set of nonlinear equations  $y(t) = f(\theta(t))$ , where  $f(\theta(t))$  is defined by the forward kinematics of the robot manipulator, mapping joint angles  $\theta(t)$  to end-effector positions  $y(t)$  [1]. Given  $\theta(t)$ , the forward computation of  $y(t)$

is a straightforward task. The inverse process of determining  $\theta(t)$  from a given  $y(t)$ , however, is substantially more difficult. If, in addition, the robot has more than three degrees of freedom, the equations  $f(\cdot)$  are typically underdetermined and one must choose a trajectory  $\theta(t)$  which is, in some sense, optimal among all possible solutions. This is the common *inverse kinematics problem* in robotics. Possible solution approaches to this problem using constrained optimization and linear Hopfield networks (LHNs) are: 1) The problem is represented in the form of an appropriate criterion ('energy') function, which then is minimized by a standard LHN, as reported in [7]. 2) The problem is directly implemented (and solved) in the form of a hybrid neural network as stated above. The latter approach is described in this paper.

## 2 Linear Hopfield Networks and Constrained Optimization

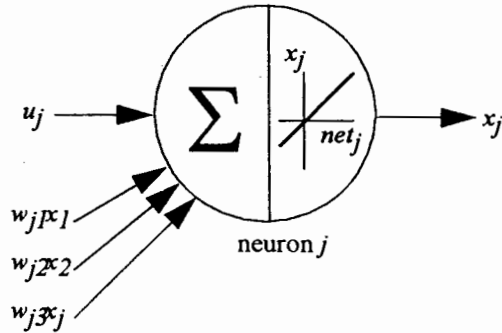
John Hopfield originally introduced the Hopfield network in 1982 [2]. Since then, this type of artificial neural network is possibly the best known example of a recurrent neural network, and is typically employed 1) as a 'problem solver' for solving optimization problems (e.g., the famous traveling salesman problem), or 2) as an 'autoassociator', more often in the latter mode. Although the Hopfield network was originally designed as a continuous-time/discrete-state system (implemented by an analog electronic circuit), Hopfield and his successors extended the concept to various combinations of continuous-/discrete-state and continuous-/discrete-time systems [3][4][9].

More recently it was observed that the linear Hopfield network, after replacing the classical 'squashing' transfer function with a linear transfer function (Figure 1), could be used to solve least squares optimization problems. The approach is straightforward: define a linear Hopfield network whose energy function is identical to the least squares

The authors gratefully acknowledge the continuing financial support for this research by the Office of Naval Research Small Business Innovative Research Program under contract N00014-91-C-0268 in the Department of Defense.

\* On sabbatical leave from Portland State University, 1993-94

performance measure one desires to minimize, as stated e.g. in [5]. In this reference, however, the values of neuron states are not guaranteed to lie in a compact set and, therefore, an alternative stability theory is required. An overview of all six Hopfield network variations is presented in [6].



**Figure 1: Neurons with linear transfer functions characterize the linear Hopfield network.**

For the purpose of this paper we use a linear continuous-state/discrete-time Hopfield network defined by the difference equation

$$\mathbf{x}(i+1) = \mathbf{W}\mathbf{x}(i) + \mathbf{u}$$

Here  $x_i$  is the  $i$ -th iterate of an  $n$ -dimensional vector of (real or complex) neuron states,  $\mathbf{W}$  is a symmetric  $n$ -by- $n$  (real or complex) weight matrix, and  $\mathbf{u}$  is an  $n$ -vector of constant (real or complex) inputs. Moreover, this network will be convergent if its spectral radius  $\rho(\mathbf{W})$  is less than 1, i.e., the eigenvalues of  $\mathbf{W}$  lie within the unit circle of the complex plane.

Besides the above classical result, we introduce two additional solution approaches, which provide design methods for stable LHNs as well as hybrid feedforward/LHNs, which solve various types of linear (constrained or unconstrained) problems.

### 2.1 The Linear Hopfield Network

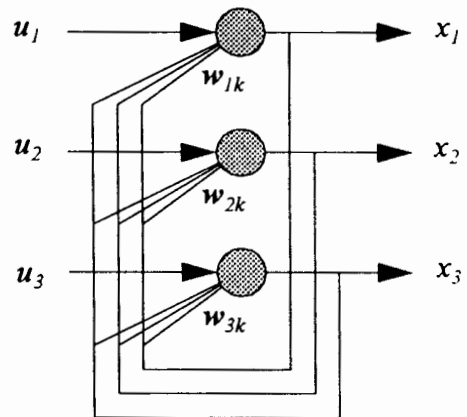
Let the first problem class be represented by a linear equation of the form  $\mathbf{A}_1 \mathbf{x} = \mathbf{b}_1$ , where  $\mathbf{A}_1$  is a positive definite Hermitian matrix,  $\mathbf{b}_1$  is an arbitrary constant vector, and  $\mathbf{x}$  is the state vector of the LHN. In [6] we prove that a linear Hopfield network converges to the solution vector from any initial condition, if weight matrix  $\mathbf{W}$  and input vector  $\mathbf{u}$  are given by (where  $\mathbf{I}$  is the identity matrix)

$$\begin{aligned} \mathbf{W} &= \mathbf{I} - \alpha \mathbf{A}_1 \\ \mathbf{u} &= \alpha \mathbf{b}_1 \end{aligned}$$

and the 'scaling factor'  $\alpha$  satisfies

$$0 < \alpha < \frac{2}{\rho(\mathbf{A}_1)}$$

where  $\rho(A)$  is the spectral radius of some matrix  $A$ , i.e. the absolute value of the maximum eigenvalue,  $\max |\lambda(A)|$ . Fast convergence is achieved if  $\alpha$  is chosen close to its upper limit, and consequently  $\alpha$  may be interpreted as step size or, more precisely, convergence rate. In [7] we applied this result to the inverse kinematics problem. Although expressed in terms of a linear equation rather than an optimization problem, the result is essentially that a linear Hopfield network can be used to solve least squares optimization problems, which take the form  $\mathbf{A}\mathbf{x} = \mathbf{b}$ , where  $\mathbf{A}$  is a positive definite Hermitian.



**Figure 2: The general topology of Hopfield networks (here with 3 inputs/states, according to the inverse kinematics example).**

### 2.2 Hybrid Feedforward/Linear Hopfield Networks

Let the second problem class be by a linear equation of the form  $\mathbf{A}_2 \mathbf{x} = \mathbf{b}_2$ , where  $\mathbf{A}_2$  is an arbitrary nonsingular (real or complex)  $n$ -by- $n$  matrix, and  $\mathbf{b}_2$  is an arbitrary constant (real or complex)  $n$ -vector. The design method described in Section 2.1 can be enhanced for hybrid linear dynamic networks, comprising a LHN and a preprocessing feedforward layer (Figure 3), which yield the solution to the

above arbitrary (nonsingular) linear equation. Let the weight matrix  $W$  and constant input vector  $u$  be given by

$$W = I - \alpha A_2^H A_2$$

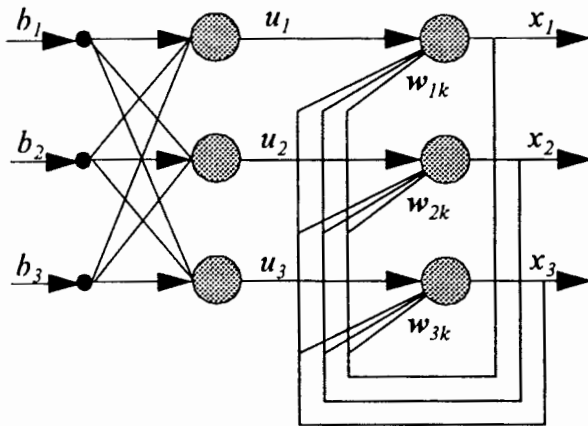
$$u = \alpha A_2^H b_2$$

where  $u$  is implemented by the preprocessing feedforward layer. The hybrid network then converges to the solution of the given linear equation, if the 'scaling factor'  $\alpha$  satisfies

$$0 < \alpha < \frac{2}{\rho(A_2^H A_2)}$$

( $A^H$  is the Hermitian of a complex matrix  $A$ , the equivalent to the transpose  $A^T$  in the real case.)

The above design method for a hybrid feedforward/LHN implies that an arbitrary set of linear equations in  $n$ -dimensional space (real or complex) can be solved by an artificial neural network with a topology shown in Figure 3. The formulation of the *dual* network with weight matrix  $W = I - \alpha A A^H$  and a postprocessing feedforward layer implementing  $y = \alpha A^H x$  is a straightforward task.



**Figure 3: The topology of a hybrid linear dynamic network, comprising a linear Hopfield network and a feedforward layer.**

As reported in [6], the two dual network solutions can be extended to solve the *generalized inverse* of an equation  $A_3 x = b_3$  for a (real or complex) rectangular matrix,  $A_3$ , whenever it has full column rank or row rank. Finally, in order to reduce computation time the spectral radius  $\rho(\cdot)$  of a

matrix may be replaced by  $trace(\cdot)$ . The trace of a matrix is an upper bound of  $\rho(\cdot)$ , and is simply the sum of all diagonal matrix elements.

### 3 Constrained Optimization and the Inverse Kinematics Problem

Manipulator kinematics describe the *geometry* of robot arm (manipulator) motions. The end-effector is moved to a target position  $y$  by controlling the manipulator's joint angles  $\theta$  (Figure 4). This process is described by the manipulator's forward kinematics, a set of nonlinear equations  $f(\cdot)$ ,

$$y(t) = f[\theta(t)]$$

where  $y \in \mathfrak{R}^m$ ,  $\theta \in \mathfrak{R}^n$  and  $f: \mathfrak{R}^n \rightarrow \mathfrak{R}^m$ . If the trajectory  $\theta(t)$  in joint angle space were given, the forward computation of  $x(t)$  would be a straightforward task. Unfortunately, the control objective for robot manipulators is usually to track a prescribed end-effector trajectory  $y(t)$ , typically given in 3-dimensional Cartesian space, while the control system operates in joint angle space. The *inverse kinematics problem* is to find a corresponding trajectory  $\theta(t)$  in joint-angle space, i.e. to find the inverse mapping

$$\theta(t) = f^{-1}[y(t)]$$

The inversion of  $f(\cdot)$  is a difficult task, due to the multi-dimensional, nonlinear properties of  $f(\cdot)$ . If, in addition, the robot has more than three degrees of freedom (i.e. joint angles), the equations are underdetermined ( $m < n$ ). Then a trajectory  $\theta(t)$  must be determined, which is, in some sense, optimal among all possible solutions.

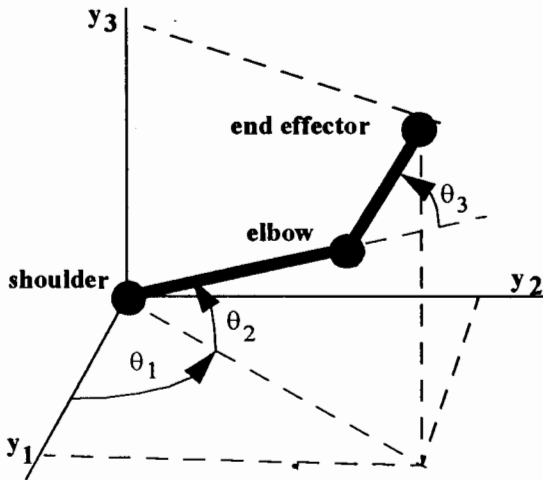
A common method to facilitate the solution of the inverse kinematics problem is to linearize the forward kinematics, i.e. to differentiate  $f(\cdot)$  with respect to time, yielding the velocity equation (or differential kinematics) [1]

$$\frac{dy}{dt} = \frac{d}{dt}[f(\theta)] = J(\theta) \frac{d\theta}{dt}$$

The prescribed trajectory  $y(t)$  is then tracked by a linear approximation via inversion of the linear velocity equation and integrating the obtained joint angle velocities. Using this approach, the inverse kinematics problem reduces to the inversion of the Jacobian matrix  $J[\theta(t)]$  at all time instants along  $y(t)$ , yielding

$$\frac{d\theta}{dt} = J^{-1}(\theta) \frac{dy}{dt}$$

Efficient algorithms for the computation and inversion of manipulator Jacobians have been intensively researched. We employ the hybrid feedforward/LHN in Figure 3 to solve the inverse velocity, which results in the implicit inversion of the Jacobian  $J(\theta)$ .



**Figure 4: The robot manipulator used for simulation.**

In an earlier work [7] we designed a criterion function, which represents a ‘compromise’ between 1) the error between desired and actual Cartesian velocities for trajectory tracking, and 2) the joint angle velocities for smooth motion. To minimize the criterion function, a linear Hopfield network as defined in Section 2.1 was specified and run at all discrete time instants along the prescribed trajectory to obtain the corresponding  $\Delta\theta$  for a given  $\Delta y$ . (In discrete time applications the velocities correspond to small changes in position,  $\Delta y$ , and joint angles,  $\Delta\theta$ .) The trajectory error obtained during simulations was small but not zero, depending on the arbitrarily weighted ‘importance’ of the two criteria.

In the work reported here we set up a constrained linear optimization problem and employed a hybrid feedforward/LHN as defined in Section 2.2 to solve it. The joint angle changes  $\Delta\theta(t)$ , resulting from constrained optimization processes at ‘fixed physical’ times  $t$ , are minimized in a least square sense. Again the manipulator model in Figure 4 was used for simulation, i.e.  $y(t) \in \mathcal{R}^3$  and  $\theta(t) \in \mathcal{R}^3$ . After

computing the manipulator’s Jacobian matrix  $J[\theta(t)]$  at time  $t$ , the hybrid feedforward/LHN was designed according to

$$W = I - \alpha J J^T$$

$$u = \alpha J^T \Delta y$$

where  $\Delta y$  is the desired change in Cartesian coordinates at time instant  $t$ . (Since the manipulator Jacobian is a real matrix, the Hermitian reduces to a transposed matrix.) The hybrid network converges to the ‘best’ joint angles changes  $\Delta\theta(t+1)$  in a least squares sense,

$$\Delta\theta(i+1) = W\Delta\theta(i) + u$$

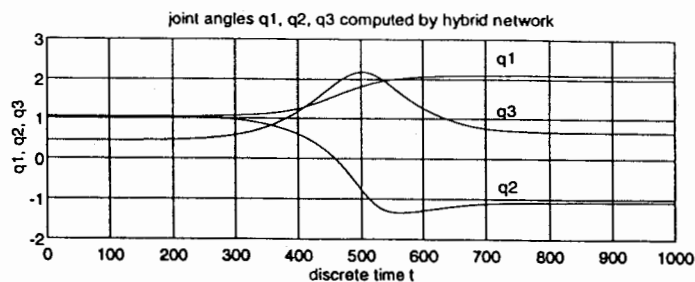
which allow a small movement of the manipulator. This process must be repeated at all discrete times along  $y(t)$ .

**Simulation.** The three distinct modes of the joint angle trajectory, computed during simulation by the hybrid feedforward/LHN, is shown in Figure 5. The corresponding desired Cartesian trajectory is a straight line in 3D Cartesian space, consisting of 1000 points. The desired velocity profile  $\Delta y(t)$ , given by the time delays between the desired 1000 points, required a slow start, maximum speed at the trajectory’s midpoint, and a slow approach to the final point. The actual end-effector trajectory in Cartesian space, given in terms of its three distinct modes, were obtained via the manipulator’s forward kinematics, and is shown in Figure 6. These curves demonstrate that the number of points used to prescribe the end-effector’s path, determines the accuracy achieved by the linear approximation. For example, the error of mode  $x_2(t)$  vanished if the desired trajectory consisted of 1500 data points, instead of 1000 points. Note that modes  $x_1(t)$  and  $x_3(t)$  are practically identical with their desired counterparts.

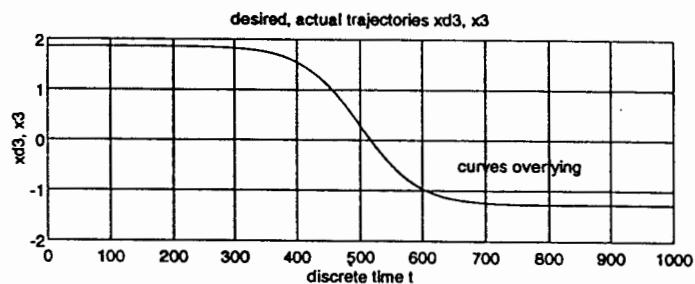
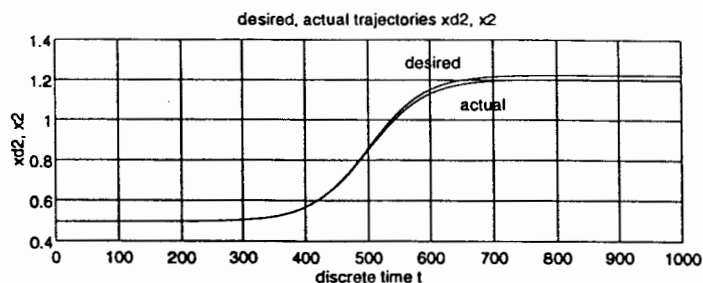
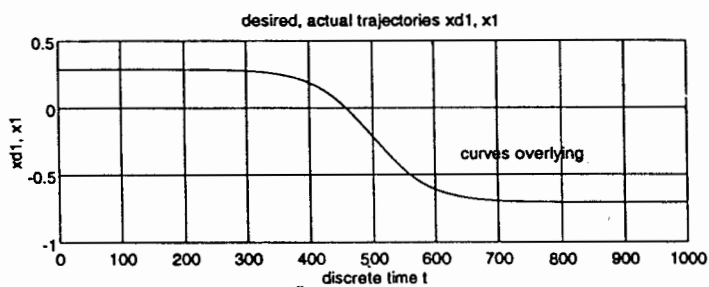
## 4 Conclusions

In this paper we presented techniques for the design of different types of linear Hopfield networks. All design methods guarantee the convergence of these networks to solutions of various linear equations. A hybrid neural network, consisting of a linear Hopfield network and a feedforward layer, computes the generalized inverse of a rectangular matrix, and therefore can be used to solve arbitrary linear equations. Furthermore, the construction of a step size is introduced, which allows convergence of the dynamic optimization process at or near maximum rate. As an application example a neural network solution to the inverse kinematics problem

was tested, employing the hybrid neural network suggested. This approach yielded very good results.



**Figure 5: Simulation results: joint angle trajectory (1000 points), computed by the hybrid feedforward/linear Hopfield network.**



**Figure 6: Desired and actual end-effector trajectory in Cartesian space (1000 points). The error of mode 2 vanished if the desired trajectory consisted of 1500 points.**

## 5 References

- [1] Craig, J.J., (1989). *Introduction to Robotics: Mechanics and Control*, Addison Wesley, Reading.
- [2] Hopfield, J.J., (1982). "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. of the Nat. Acad. of Sci. USA*, Vol. 79, pp. 2554-2558.
- [3] Hopfield, J.J., (1984). "Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons", *Proc. of the Nat. Acad. of Sci. USA*, Vol. 81, pp. 3088-3092.
- [4] Hopfield, J.J., and D.W. Tank, (1985). "Neural Computation of Decision Optimization Problems", *Biological Cybern.*, Vol. 52, pp. 141-152.
- [5] Kelly, M.F., Parker, P.A., and R.N. Scott, (1990). "Myoelectric Signal Analysis using Neural Networks", *IEEE Engineering in Medicine and Biology Magazine*, pp. 61-64.
- [6] Mathia, K., Lendaris, G.G., Sacks, R., (1994). "Linear Hopfield Networks and Constrained Optimization", (1994). (Submitted for publication).
- [7] Mathia, K., Sacks, R., (1994). "Inverse Kinematics via Linear Dynamic Networks", *Proc. of the World Congress on Neural Networks 1994*, San Diego, Earlbaum, 1994.
- [8] Pap, R.M., Sacks, R., Thomas, C.R., and Akita, R.M., (1992). "Neural Network Implementation of Stochastic Filters for Radar Tracking", *Proc. of the IEEE Workshop on Neural Networks and Signal Processing*, Elsinore, October 1992.
- [9] Tank, D.W., and J.J. Hopfield, (1986). "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", *IEEE Trans. on Circuits and Systems*, Vol. CAS-33, pp. 533-541.