

Inverse Kinematics via Linear Dynamic Networks

Dr. Karl Mathia* and Dr. Richard Saeks

Accurate Automation Corporation
7001 Shallowford Road
Chattanooga, Tennessee 37421

Abstract

A solution to the inverse kinematics problem in robotics is presented. The usual difficulty of inverting robot kinematics is circumvented by minimizing an appropriate cost function using linear dynamic networks. The results of the optimization process are the 'best' joint angle rates which minimize the manipulator's position error. We call the class of networks used *Linear Hopfield Networks* (LHN), due to its similarities with the original Hopfield Network. A convergence proof for the LHN is given. An explicit upper bound for the step size which guarantees convergence and fast convergence is presented. The simulated tracking control for a three-joint robot manipulator demonstrates the performance of our approach.

1 Introduction

The *forward kinematics* of a robot arm (manipulator) describe the geometry of manipulator motions, i.e. the mapping from given joint angles to the associated position of the manipulator in Cartesian space (or some other space). The *inverse kinematics* problem is to find the joint angles needed to move the end-effector to a desired position, i.e. an inverse mapping from a point in Cartesian space to an associated point in joint angle space must be determined. A closed form solution of the inverse kinematics can be non-trivial to determine or may not even exist, depending on the complexity or redundancy of the robot's forward kinematics.

Initial ideas of circumventing a closed form solution by linearizing the direct kinematics and then defining and minimizing a cost function were presented earlier [2]. The present paper extends the theory and provides simulation results. To find the required joint angles for a desired manipulator position, the inverse kinematics problem is reduced to the numerical solution of a linear system. The linear system is specified and solved for each manipulator position, i.e. for each point on the manipulator's trajectory in Cartesian space, the required associated point in joint angle space is determined via an optimization process. An appropriate cost function is represented in a form which can be minimized by a linear, fully connected, recurrent network. Due to its similarities with the original Continuous Hopfield Network (CHN), we call this class of linear dynamic networks *Linear Hopfield Network* (LHN). As opposed to the CHN, which has nonlinear transfer functions and is used as an 'autoassociator' [4], the LHN has linear transfer functions and solves systems of linear equations. A design method for the LHN is presented which guarantees convergence to the optimal solution, here the 'best' joint angle velocity to minimize the manipulator's velocity error. The resulting discrete velocities are integrated to track the prescribed end-effector path. As an example, the simulated control of a three-joint manipulator in 3-D Cartesian space is presented.

2 The Inverse Kinematics Problem

Manipulator kinematics describe the *geometry* of robot arm (manipulator) motions. The end-effector of a manipulator is moved to a Cartesian target position by controlling the manipulator's joint angles. The manipulator's forward kinematics describe this mapping from joint angle space to Cartesian space (Figure 1a). Let $\mathbf{x}(t)$ be the m -dimensional Cartesian position vector and $\theta(t)$ the n -dimensional joint angle vector, both functions of time. Then the forward kinematics are:

$$\mathbf{x}(t) = \mathbf{f}(\theta(t)), \quad (1)$$

$$\mathbf{f}: \mathcal{R}^n \rightarrow \mathcal{R}^m, \theta \in \mathcal{R}^n, \mathbf{x} \in \mathcal{R}^m.$$

*Corresponding author. E-mail: karl@mathia.com. Homepage: <http://www.mathia.com/karl/index.htm>.

The end-effector trajectories for the robot manipulator are usually planned in the Cartesian work space. The inverse kinematics problem is to find joint angles $\theta(t)$ (in the robot's control space) such that the manipulator's end-effector is placed at a desired position $\mathbf{x}(t)$ in Cartesian space, thus

$$\theta(t) = \mathbf{f}^{-1}(\mathbf{x}(t)) \quad . \quad (2)$$

Unique closed form solutions for the inverse mapping in Equation 2 are difficult to find for non-redundant manipulators ($m \geq n$), due to the complexity and nonlinearities of \mathbf{f} , and do not exist for redundant manipulators ($m < n$, fewer equations than unknowns). Extending initial ideas expressed in [4], the present paper proposes a solution of the inverse kinematics problem by 1) linearizing the forward kinematics \mathbf{f} , 2) specifying a cost functions E , and 3) numerically solving the linear system by optimizing E . The cost functions E is implemented as a Linear Hopfield Network (LHN), which carries out the optimization processes at each point \mathbf{x}_d along the desired trajectory $\mathbf{x}_d(k)$, where k represents discrete time.

3 Linearization and Velocities

Difficulties in computing the inverse kinematics in Equation 2, introduced by the complexity of \mathbf{f} or the redundancy of the forward kinematics in Equation 1, can be overcome by linearize the forward kinematics with respect to time and use the *velocity equation*, with joint angle velocities $\dot{\theta} = d\theta/dt$ and Cartesian velocities $\dot{\mathbf{x}} = d\mathbf{x}/dt$ as input and output for the manipulator control system. The inverse is then obtained by

$$\dot{\mathbf{x}} = \mathbf{J}(\theta) \dot{\theta} \quad , \quad (3)$$

$$\dot{\theta} = \mathbf{J}^{-1}(\dot{\mathbf{x}}) \quad , \quad (4)$$

where the Jacobian matrix \mathbf{J} is defined by

$$\mathbf{J}(\theta) = \begin{bmatrix} \frac{df_1}{d\theta_1} & \cdots & \frac{df_n}{d\theta_1} \\ \vdots & & \vdots \\ \frac{df_1}{d\theta_m} & \cdots & \frac{df_n}{d\theta_m} \end{bmatrix} .$$

Given a desired Cartesian velocity profile $\dot{\mathbf{x}}_d$, the manipulator joint angles are obtained by integrating the joint angle velocities. For solving Equation 4, numerical optimization of a cost function is often preferred over the matrix inversion approach for two reasons: 1) the inversion of \mathbf{J} can be numerically unstable (assuming the inverse exists), and 2) \mathbf{J} may be singular and therefore a unique solution of Equation 4 does not exist. If \mathbf{J} is not invertible, additional constraints can be added. A common way is to apply the Moore-Penrose pseudo-inverse, where the additional constraint is given by finding the 'best' solution in a least squares sense. In the present paper also an approach is used, which gives a numerical solution without using the inverse Jacobian.

4 Cost Function and Optimization Process

The inverse kinematics problem for a robot manipulator is converted into an *optimization problem*. First a cost function E (error function, performance measure) is designed, then E is optimized using linear dynamic networks at every point along the desired end-effector trajectory. The primary control objective is to minimize the position error $\mathbf{e}(k) = \mathbf{x}(k) - \mathbf{x}_d(k)$ of the manipulator's end-effector. According to Equation 4, E is expressed in terms of the velocity error $\mathbf{e}(k) = \dot{\mathbf{x}}(k) - \dot{\mathbf{x}}_d(k)$. An additional constraint is the minimization of the joint angle velocity $\dot{\theta}$. This constraint results in a smoother trajectory in joint angle space (and therefore Cartesian space). E is defined as

$$2E = \|\mathbf{e}\| + \varepsilon \|\dot{\theta}\| \quad , \quad (5)$$

$$\begin{aligned} &= \|\mathbf{x} - \mathbf{x}_d\|^2 + \varepsilon \|\boldsymbol{\theta}\|^2, \\ &= (\mathbf{x} - \mathbf{x}_d)^T (\mathbf{x} - \mathbf{x}_d) + \varepsilon \boldsymbol{\theta}^T \boldsymbol{\theta}, \end{aligned}$$

where ε is an arbitrary non-zero constant (discussed below). The velocity error is expected to increase with ε , which will be demonstrated using simulation. The factor 2 in Equation 5 is used for later convenience when differentiating E , which can now be represented in terms of the joint angle velocity:

$$\begin{aligned} 2E(\boldsymbol{\theta}) &= (\mathbf{J}\boldsymbol{\theta} - \mathbf{x}_d)^T (\mathbf{J}\boldsymbol{\theta} - \mathbf{x}_d) + \varepsilon \boldsymbol{\theta}^T \boldsymbol{\theta}, \tag{6} \\ &= \boldsymbol{\theta}^T (\varepsilon \mathbf{I} - \mathbf{J}^T \mathbf{J})^T \boldsymbol{\theta} - (\mathbf{J}\boldsymbol{\theta} - \mathbf{x}_d) + \varepsilon \boldsymbol{\theta}^T \boldsymbol{\theta}, \\ &= \boldsymbol{\theta}^T (\varepsilon \mathbf{I} - \mathbf{J}^T \mathbf{J})^T \boldsymbol{\theta} - (\mathbf{J}\boldsymbol{\theta} - \mathbf{x}_d) + \varepsilon \boldsymbol{\theta}^T \boldsymbol{\theta}, \quad \mathbf{A} \in \mathfrak{R}^{n \times n}, \mathbf{b} \in \mathfrak{R}^n, \end{aligned}$$

where $\mathbf{A} = \varepsilon \mathbf{I} + \mathbf{J}^T \mathbf{J}$ and $\mathbf{b} = \mathbf{J}^T \mathbf{x}_d$. The constant ε weights the importance of the angle velocity (this strategy is common in optimal control methods), and also provides invertibility of \mathbf{A} (with $\varepsilon \mathbf{I}$) for all forms of \mathbf{J} , and therefore a unique solution of the equation system Equation 6. E must be minimized at every point along \mathbf{x}_d for the 'best' joint angle velocity, in order to obtain the 'best' desired Cartesian velocity. The derivative of E in Equation 6 with respect to 'iteration time τ ' is (the manipulator moves in physical time t , which 'stays still' during optimization):

$$\begin{aligned} \frac{dE(\boldsymbol{\theta})}{d\tau} &= \frac{1}{2} \frac{dE(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \frac{d\boldsymbol{\theta}}{d\tau}, \\ \mathbf{E} &= \mathbf{A}\boldsymbol{\theta} - \mathbf{b}, \tag{7} \end{aligned}$$

The optimization process will settle at $\mathbf{E} = \mathbf{0}^T$ (a convergence proof is given below). Minimizing E therefore solves the linear system

$$\mathbf{A}\boldsymbol{\theta} = \mathbf{b}. \tag{8}$$

5 The Linear Hopfield Network

The optimization process for solving Equation 7 is mapped onto a linear dynamic network, the Linear Hopfield Networks (LHN), which is a natural outgrowth of the Continuous Hopfield Network (CHN).

Continuous Hopfield Network. In the original paper by Hopfield and Tank [4], the cost function of CHNs (recurrent, with a single layer) is defined by $2E = -\mathbf{V}^T \mathbf{T} \mathbf{V} - 2\mathbf{I}^T \mathbf{V}$, with weight matrix \mathbf{T} , optimization variable \mathbf{V} , and constant input vector \mathbf{I} , and is similar to the cost function in Equation 6. The derivative of Hopfield's cost function is $\mathbf{E} = -\mathbf{T} \mathbf{V} - 2\mathbf{I}^T$, thus similar to Equation 7. The CHN is primarily used as an 'autoassociator' and requires sigmoidal transfer functions to guarantee convergence to the nearest minimum of E [3]. If instead a linear transfer function is used, the CHN in [4] becomes $\mathbf{V} = \mathbf{T}^* \mathbf{V} + \mathbf{I}^*$, with $\mathbf{T}^* = -\text{diag}(k_i) + \mathbf{T}$ and $\mathbf{I}^* = -\mathbf{I}$. When converging to the equilibrium ($\mathbf{V} = \mathbf{0}$), the linear system $\mathbf{T}^* \mathbf{V} = \mathbf{I}^*$ is solved, which is a linear system like Equation 8.

Linear Hopfield Network. As shown above, the linear dynamic network defined by Equation 6 and Equation 7 is very similar to the CHN, which also is characterized by a cost function and a differential equation. The main difference between CHN and LHN is the sigmoidal transfer function, which is missing in Equation 6 and Equation 7. This fact leads to our notion of the LHN (single layer, recurrent), which solves the linear system Equation 7, provided convergence to the solution $E(\boldsymbol{\theta}) = 0$. A design method for the LHN, which satisfies the convergence requirements, is given below in Equation 9 and Equation 10.

Definition. The Linear Hopfield Network is a single-layer, recurrent network with weight matrix \mathbf{W} , constant input vector \mathbf{u} and linear transfer functions. The dynamics of the continuous-time LHN and the discrete-time LHN are defined by Equation 9:

$$\frac{d\theta}{d\tau} = -\mathbf{W}\theta + \mathbf{u}, \quad (9a)$$

$$\mathbf{W} = \alpha\mathbf{A} = \alpha(\varepsilon\mathbf{I} + \mathbf{J}^T\mathbf{J}), \mathbf{W} \in \mathfrak{R}^{n \times n}, \quad (9b)$$

$$\mathbf{u} = \alpha\mathbf{b} = \alpha\mathbf{J}^T\mathbf{x}_d, \mathbf{u} \in \mathfrak{R}^n, \quad (9c)$$

and for numerical solutions using digital computers, the discrete time representation of Equation 9a is

$$\Delta\theta(n) = \theta(n+1) - \theta(n) = -\mathbf{W}\theta(n) + \mathbf{u}, \quad (9d)$$

In the following we prove that this linear system converges for

$$0 \leq \alpha < \frac{1}{\|\mathbf{A}\|}, \quad (10)$$

Convergence of the discrete-time LHN in Equation 9d is guaranteed for any initial condition if the step size α satisfies Equation 10 for some matrix norm. The step size should be near its upper limit for fast convergence. Suitable norms are e.g. the spectral radius or the maximum singular value (MSV) of \mathbf{A} .

Convergence Proof. While the sigmoidal transfer function plays a central role in the convergence proof for the CHN, this does not apply to LHNs [3]. Here Equation 9 is 'scaled' with a constant factor α to provide convergence. This is possible since the unscaled system (Equation 11a) and the scaled system in Equation 11b have the same solution:

$$\mathbf{A}\theta = \mathbf{b}, \quad (11a)$$

$$\alpha\mathbf{A}\theta = \alpha\mathbf{b}, \quad (11b)$$

$$\theta = \mathbf{A}^{-1}\mathbf{b} = \alpha\mathbf{A}^{-1}\frac{1}{\alpha}\mathbf{b} = \mathbf{W}^{-1}\mathbf{u}, \quad (12)$$

A sufficient condition for the iterative solution of Equation 11a and Equation 11b is (see e.g. [1])

$$\|\mathbf{W}\| = \|\alpha\mathbf{A}\| = \alpha\|\mathbf{A}\| < 1,$$

which results in the inequality in Equation 10 for some matrix norm. (q.e.d.)

With a suitable α and $\Delta\theta(n) = \theta(n+1) - \theta(n)$, the discrete-time LHN in Equation 9d becomes

$$\theta(n+1) = \theta(n) - \mathbf{W}\theta(n) + \mathbf{u} \quad (13)$$

$$= \theta(n) + \alpha(-\mathbf{A}\theta + \mathbf{b}) \quad (14)$$

$$= \theta(n) + \alpha\Delta\theta(n). \quad (15)$$

Equation 15 suggests the interpretation of α as a step size, or 'convergence rate'. It is well known that a greater step size provides faster convergence, and Equation 10 gives a guideline to choose an upper bound to achieve a fast convergence to the solution.

6 Practical Considerations and Simulation

Computational Cost. It was intended to keep the required computational cost minimal for LHN implementations. This is of particular interest for the inverse kinematics problem considered here, because the LHN must be specified at each point along the desired trajectory. Possible matrix norm for specifying

the step size is the maximum singular value (MSV), which is, unfortunately, computationally expensive. Here the trace, $trace\{\mathbf{A}\}$, which is easy to compute, was used as an upper bound for the MSV:

$$\sigma_{max}\{\mathbf{A}\} \leq trace\{\mathbf{A}\} = \sum_{i=1}^n A_{ii}.$$

It is important to choose appropriate initial conditions $\mathbf{x}(k=0)$: if the initial Jacobian matrix may be close to singularity, so the algorithm may be unstable or may converge very slowly.

Simulation. For the simulation experiment a 3-joint robot manipulator model was used (Figure 1a). The desired Cartesian end-effector trajectory was a straight line. The manipulator has two links of size one, and three controllable joint angles (two shoulder joints, one elbow joint). The shoulder is positioned at the origin of the Cartesian work space. The inverse kinematics control system therefore has 3-dimensional reference inputs \mathbf{x}_d , control signals θ , and control variable \mathbf{x} (Figure 1b).

The desired end-effector trajectory from time $t=0$ to $t=T$ was constructed as follows: Starting and ending point were obtained by Equation 1 for $\theta(0) = [\pi/3, \pi/3, \pi/7]^T$, $\theta(T) = [2\pi/3, -\pi/3, \pi/5]^T$ to $\mathbf{x}_d(0) = [0.28, 0.50, 1.87]^T$, $\mathbf{x}_d(T) = [-0.70, 1.22, -1.27]^T$. Between the Cartesian start and end point a straight line of 1000 points was constructed, where the distances between the points represented the desired end-effector velocity. A bell-shaped velocity curve was chosen. The joint angle velocities computed by the LHN were integrated to obtain the joint angles (Figure 2a). The position tracking error e increases with increasing ε . Here the results for $\varepsilon = 0.01$ are shown (Figure 2b). The three elements of the desired and actual Cartesian trajectories are shown in Figure 3 and Figure 4. Because the cost function implements a 'compromise' between accurate position and smooth angle velocity curve, thus the error is non-zero. The linearization error can be further reduced by using more points along the desired trajectory. With increasing ε the LHN gave towards smoother curves, but with greater position error.

7 Conclusion

A solution to the inverse kinematics problem in robotics is presented, which utilizes a cost function and linear dynamic networks (Linear Hopfield Network, LHNs) for minimization. The method provides a variety of control strategies for a large number of manipulator kinematics, but requires a prescribed desired trajectory with a fixed number of points. An upper bound for the LHN step size is presented, which provides a high LHN convergence speed.

8 Acknowledgment

We thank Chadwick Cox and Professor George G. Lendaris for helpful discussions and suggestions. This research is funded by the Office of Naval Research (ONR) under contract N00014-91-C-0268.

9 References

- [1] Golub G.H., and van Loan C.F. (1989), *Matrix Computations*, Johns Hopkins University Press, Baltimore.
- [2] Guo J., Cherkassky V. (1989), "A solution to the Inverse Kinematics Problem in Robotics using Neural Network Processing," *Proc. Int. Joint Conference on Neural Networks (IJCNN)* 1989, Vol. II, pp. 299-304.
- [3] Hertz J., Krogh A., Palmer R.G. (1991), *Introduction to the Theory of Neural Computation*, Edison-Wesley Publishing Company, New York.
- [4] Hopfield J.J., Tank D.W. (1986), "Computing with Neural Circuits: A Model," *Science*, Vol. 233, August 1986, pp. 625-633.

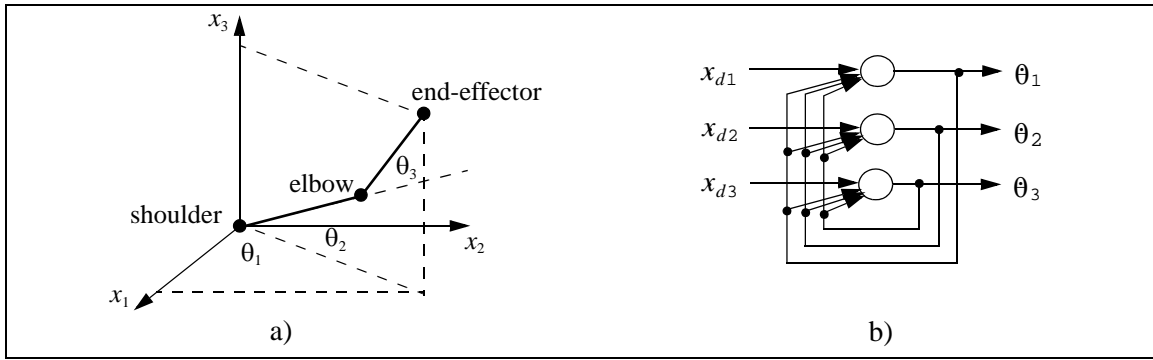


Figure 1 a) The example manipulator in joint and work space. b) Linear Hopfield Network.

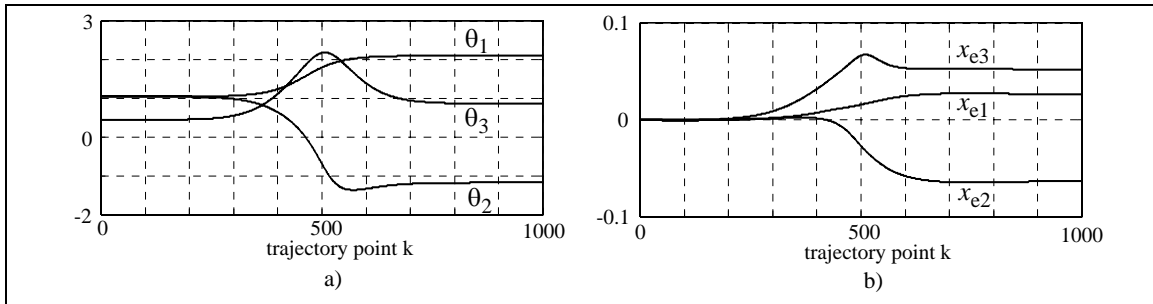


Figure 2 a) Joint angles $\theta(k)=[\theta_1, \theta_2, \theta_3]^T$, b) Cartesian errors $\mathbf{x}_e(k)=[x_{e1}, x_{e2}, x_{e3}]^T$.

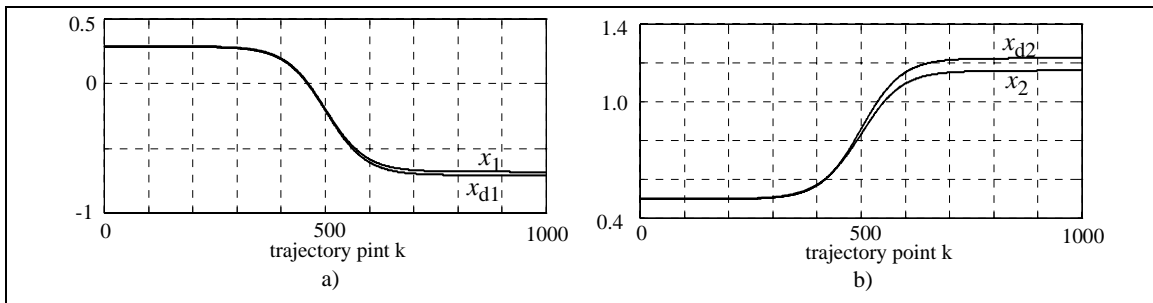


Figure 3 Desired and actual end-effector trajectories: a) $x_{d1}(k), x_1(k)$, and b) $x_{d2}(k), x_2(k)$.

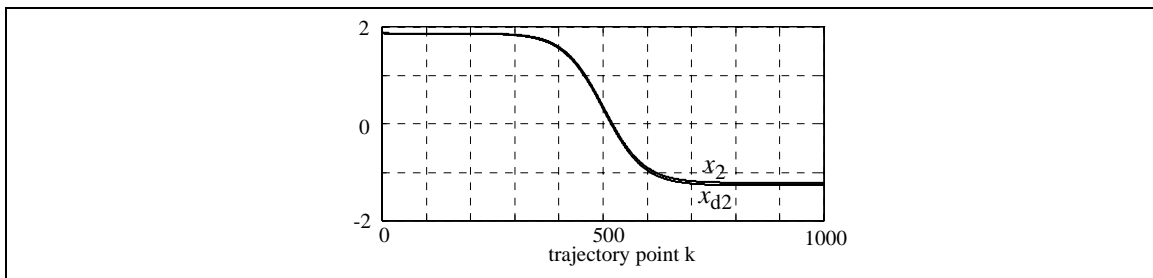


Figure 4 Desired and actual end-effector trajectories $x_{d3}(k), x_3(k)$.