A Neural Network Model of an Anthropomorphic Arm

Lina L.E. Massone and **Jennifer D. Myers**

Technical Report 1/94
Neural Information Processing Laboratory
Northwestern University

Lina L.E. Massone (massone@eecs.nwu.edu) is with the Department of Electrical Engineering & Computer Science and the Biomedical Engineering Department of Northwestern University, 2145 Sheridan Road, Evanston, IL 60208-3118.

Jennifer D. Myers (jmyers@eecs.nwu.edu) is with the Institute for Neuroscience of Northwestern University, 2153 North Campus Drive, Evanston, IL 60208-3520.

Submitted July 1994 to *IEEE Transactions on Systems, Man, and Cybernetics.*

Abstract

This paper introduces a neural network model of a planar redundant arm whose structure and operation principles were inspired by those of the human arm. We developed the model for two purposes. One purpose was to study the relative role of control strategies and plant properties in trajectory formation, namely which features of simple arm movements can be attributed to the properties of the plant alone. We address this matter in a companion paper [1]. The second purpose was a motor-learning one: to design an arm model that, because of its neural-network quality, can be eventually incorporated in a parallel distributed learning scheme for the arm controller. We modeled the arm with two joints (shoulder and elbow) and six muscle-like actuators: a pair of antagonist shoulder muscles, a pair of antagonist elbow muscles and a pair of antagonist double-joint muscles. The arm was allowed to move in the horizontal plane subject to the action of gravity. The model computes the transformation between the control signals that activate the muscle-like actuators and the coordinates of the arm endpoint. This transformation comprises four interacting stages (muscle dynamics, joint geometry, forward arm dynamics, forward arm kinematics) that we modeled with a number of feedforward and recurrent neural networks. In this paper we introduce and describe in detail the modeling methods, that are efficient, highly flexible (some of the resulting networks can be easily modified to accommodate different parametric choices and temporal scales), and quite general and hence applicable to a number of different scientific domains.

1 Introduction

A major role in the outcome of a motor act is played by the properties of the plant that executes the act. In biological systems, the cut-off line between which features of a movement are the result of processing that is carried out at the level of the central nervous system and which features are instead due to biomechanics is not yet clearly defined. In artificial systems, controllers should be designed taking into account the intrinsic properties of the plants to be controlled. A clear understanding of the properties of a plant is then an important step towards the study and modeling of the so-called sensorimotor transformations, i.e., those transformations that map signals from various sensory modalities into an appropriate set of efferent motor commands to the skeletal muscles or to artificial actuators. Understanding the plant properties becomes crucial when one wants to study the sensorimotor transformations that guide the human arm motor system – a mystery yet to be unveiled, or design a controller for a biologically-inspired artificial arm: The arm is, in fact, an extremely complex device and its complexity alone can underlie a rich repertoire of motor skills. The relationship of the plant properties to the study of sensorimotor transformations is particularly evident in the context of motor learning [2], [3]. Motor learning can be described in general as a process that involves two primary entities (see Figure 1), a controller and a controlled object (the plant), and that consists of tuning the parameters of the controller based on some measure of performance of the controlled object. One important aspect of motor learning is that of coordinate

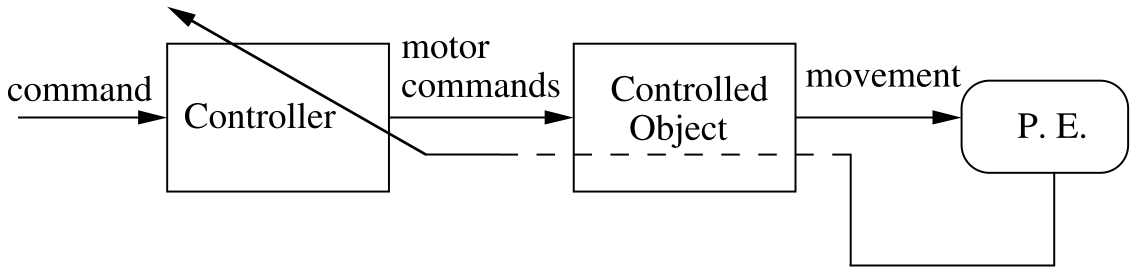

Figure 1: A motor-learning scheme, in which the parameters of the controller are tuned based on some measure of performance of the controlled object. P.E.: performance evaluation.

transformations. Often in fact the performance of the controlled object is evaluated in the coordinate frame of the object's endpoint, which is remote and different from the coordinate frame in which the controller operates. Hence coordinate transformations are necessary in order to provide the controller with useful and consistent information [4], [5], [6]. In order to better illustrate this concept we provide here two examples of coordinate transformations that occur, in biological systems, within motor-learning schemes. One example related to arm movements is that of an error expressed in the coordinates of the hand (measured for example by the visual system) that needs to be translated into an error expressed in the coordinates of the arm muscles [7], [8]. Another example, involving a different system, the vestibulo-ocular reflex, is the translation of the retinal slip signal measured by the accessory optic system into gain and phase errors [9]. It is clear from these examples that the coordinate-transformation process that builds the training information for the controller is a process that involves the plant and its outcome is heavily dependent upon the plant's properties. In other words, in the motor-learning scheme of Figure 1, the information that is made available to the controller for learning purposes is plant-dependent and hence the plant properties affect what the controller learns and what its resulting functions will be.

From a computational standpoint, neural networks and parallel distributed learning algorithms constitute a very effective framework for computing coordinate transformations. The transformation f that occurs between an n-dimensional input layer and an m-dimensional output layer of a neural network

$$\underline{y} = f(\underline{x}); f : \Re^n \longrightarrow \Re^m \tag{1}$$

can, in fact, be viewed as a very general, nonlinear coordinate transformation process. Furthermore, the ability of connectionist learning algorithms to solve the credit-assignment problem constitutes an efficient mechanism for computing the inverse mapping, i.e., for translating errors expressed in output coordinates ($\Re^m$) into errors expressed in input coordinates ($\Re^n$). In the motor-learning framework of Figure 1, these two spaces would correspond respectively to the movement space and the motor-commands space, with the function f representing the properties of the plant. It should be hence clear what the advantage of expressing the plant's properties in a neural-network format is: that of being able to directly incorporate the plant in a parallel, distributed motor-learning scheme that can provide an important and fundamental link: the link between behavior and control.

In this paper we present a neural-network model of a planar redundant arm whose structure and operation principles were inspired by those of the human arm. The motivation for choosing the human arm as a reference is that the properties of the human arm, particularly the elastic properties of muscles, are likely to be an important component of the human ability to exhibit dexterous, skilled behavior – a kind of behavior yet to be achieved with artificial arms. We developed the model with a motor-learning framework in mind, i.e., with the long-term goal of exploring various control strategies for the actuators of the arm using an approach based on learning and with the shorter-term goal, that we address in a companion paper [1], of investigating which features of movement can be accounted for by the plant's properties alone – a study that can potentially provide some guidelines for the design of artificial arms. This paper addresses the neural-network modeling of the anthropomorphic arm and describes the model's various component networks. Hence the primary goal of this paper is to introduce and describe in detail the modeling methods. We also demonstrate that these methods are highly flexible because some of the resulting networks can be easily modified to accommodate different parametric choices and temporal scales, and quite general and hence applicable to a number of different scientific domains.

2 The Arm Model

2.1 General Structure

We modeled the arm with two joints (shoulder and elbow) and six muscle-like actuators: two antagonist actuators that act across the shoulder joint (shoulder muscles), two antagonist actuators that act across the elbow joint (elbow muscles), two antagonist actuators that act across both the shoulder joint and the elbow joint (double-joint muscles.) The arm moved in the horizontal plane subject to the action of gravity. The overall transformation that the neural-network model computes is the transformation between the control signals that activate the six muscles (that we will call "neural inputs" in analogy with the biological system) and the coordinates of the arm endpoint. The various stages of this transformation are depicted in Figure 2. The muscles are activated by their neural inputs, $\underline{ni}$, and they produce torques, $\underline{\tau_m}$,

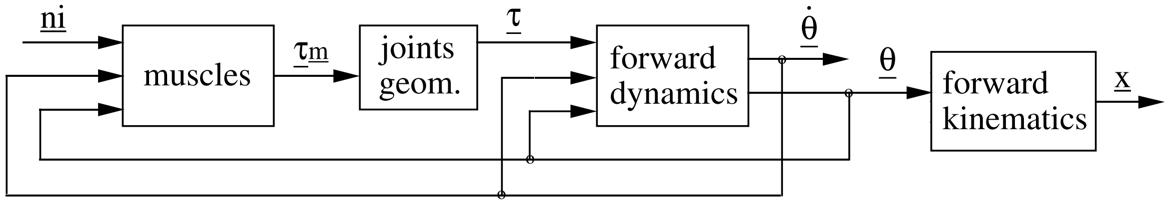

Figure 2: Architecture of the arm model.

as functions of the neural inputs, the angular positions, $\underline{\theta}$, and the angular velocities, $\dot{\underline{\theta}}$. The muscles torques are then combined into joint torques, $\underline{\tau}$, according to geometric principles. The forward-dynamic block computes the angular accelerations, $\ddot{\underline{\theta}}$, as functions of the current values of $\underline{\tau}$, $\underline{\theta}$ and $\dot{\underline{\theta}}$. The resulting accelerations are then integrated to yield the new values of $\underline{\theta}$ and $\dot{\underline{\theta}}$. The forward-kinematic block transforms the angular coordinates into coordinates of the endpoint ($\underline{x}$). The remainder of this section describes each stage in detail.

2.2 The Muscles

Figure 3 shows all the signals involved in the transformation between neural inputs and joint torques, in particular the input signals to each muscle[1] and the output signals that they produce, i.e., their torques. The muscle torques are then combined to produce the joint torques, τ_1 (shoulder) and τ_2 (elbow). Figure 4 shows the general model that we utilized for the muscle-like elastic actuators. We modeled each actuator with a contractile element, a series element and a parallel element [10], [12]. The contractile element was composed of a force generator, F_0, in parallel with a damper of viscosity B. The force generator was modulated by the neural input. The series element was an elastic element with stiffness K_s. The parallel element was a viscoelastic element with stiffness K_p and viscosity B_p. The approach that we used to describe the response of a generic muscle-like actuator is a simplification (specifically a linear approximation) of the approach described in [12]. In biological systems the neural input, ni, is transformed into EMG activity by a process called neural excitation (de-excitation) dynamics which describes the relationship between an idealized neurocontroller and the motoneuronal system. It represents processes such as neural-pool and recruitment dynamics and electrical filtering between the motorneuron and the muscle fiber. In our model we described this process as a low-pass filtering operation regulated by the following differential equation:

$$e\dot{m}g = \frac{1}{\tau_{ne}}(ni - emg) \tag{2}$$

where τ_{ne} is the time constant of the filter. Another process, called activation (de-activation) dynamics, represents the temporal dynamics between the EMG signals and the contractile process, which in biological systems is regulated by the fraction of troponin bound with calcium. In our model this process transformed EMG activity into hypothetical tension F_0 as follows:

$$\dot{F}_0 = \frac{1}{\tau_{ac}}(emg - F_0) \tag{3}$$

[1]In this paper we use the terms flexor and extensor to indicate muscles that move the arm towards the body and away from the body respectively.

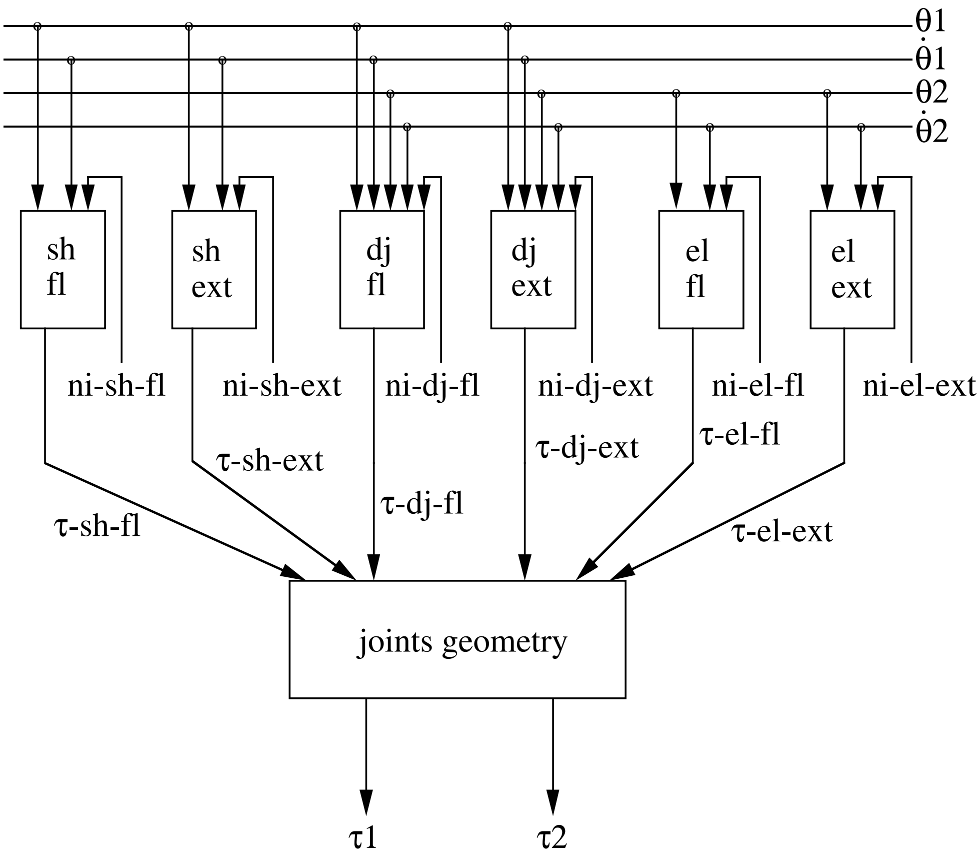

Figure 3: The transformation computed by the muscles model. θ_1: shoulder angle; θ_2: elbow angle; $\dot{\theta}_1$: shoulder velocity; $\dot{\theta}_2$: elbow velocity; ni: neural input; sh: shoulder; el: elbow; dj: double-joint; fl: flexor; ext: extensor; τ_1: shoulder torque; τ_2: elbow torque.

where τ_{ac} is the time constant of this process. We then translated the value of F_0 to force units by multiplying it by a constant, F_{max}. The result was then multiplied by a constant average moment arm, m, to yield the hypothetical torque T_0.

$$T_0 = F_0 * F_{max} * m \tag{4}$$

We then used the following differential equation to relate the hypothetical torque to the length of the contractile element:

$$T_0 + B\dot{x_0} = K_s(x - x_0) \tag{5}$$

Finally, the torque τ_m produced by the muscle was given by:

$$\tau_m = B_p\dot{x} + K_p x + K_s(x - x_0) \tag{6}$$

We computed the length of the muscle x and its time derivative as follows:

$$x = qm \tag{7}$$

$$\dot{x} = \dot{q}m \tag{8}$$

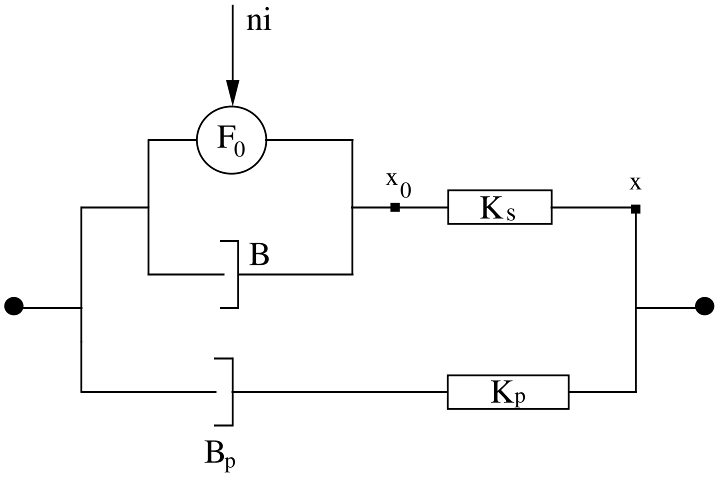

Figure 4: General structure of a muscle. ni: neural input; F_0: hypothetical tension; B: damper; K_S: series elasticity; K_P: parallel elasticity; B_P: parallel damper; x: muscle length; x_0: length of contractile element.

where m is the average moment arm, q is the angle subtended by the muscle (see Appendix A) and $\dot{q}$ its time derivative.

In order to express the overall relationship between neural input and muscle torque in a neural-network form we wrote equations (2), (3) and (5) as three difference equations according to the Euler integration method:

$$emg(t+\Delta t) = \frac{\Delta t}{\tau_{ne}} ni(t) + (1 - \frac{\Delta t}{\tau_{ne}}) emg(t) \tag{9}$$

$$F_0(t+\Delta t) = \frac{\Delta t}{\tau_{ac}} emg(t) + (1 - \frac{\Delta t}{\tau_{ac}}) F_0(t) \tag{10}$$

$$x_0(t+\Delta t) = \frac{K_s \Delta t}{B} x(t) - \frac{\Delta t}{B} F_0(t) + (1 - \frac{K_s \Delta t}{B}) x_0(t) \tag{11}$$

where Δt is the integration step. Equations (9) through (11) together with equations (6), (7) and (8) dictated the architecture and the strength of the connections for the recurrent network shown in Figure 5. This network implements the relationship between neural input and torque for a generic muscle. We employed six such networks, one for each of the six muscles in our model, with the appropriate values of q as indicated in Appendix A.

We chose the values for the muscles' parameters, based on the literature [see, for example, 12], as indicated in Table 1. In the present implementation of the model all the parameters were equal across the six muscles. Hence the muscles differed only in the angle that they subtended and in the value of F_{max} across different antagonist pairs. However the model can be easily modified to accommodate different choices for the muscles parameters as well as parametric differences between muscles. Such modifications simply involve changing the strength of some connections in the recurrent network of Figure 5.

2.3 Joint Geometry

Figure 6 shows how the muscle torques were transformed into joint torques. We computed

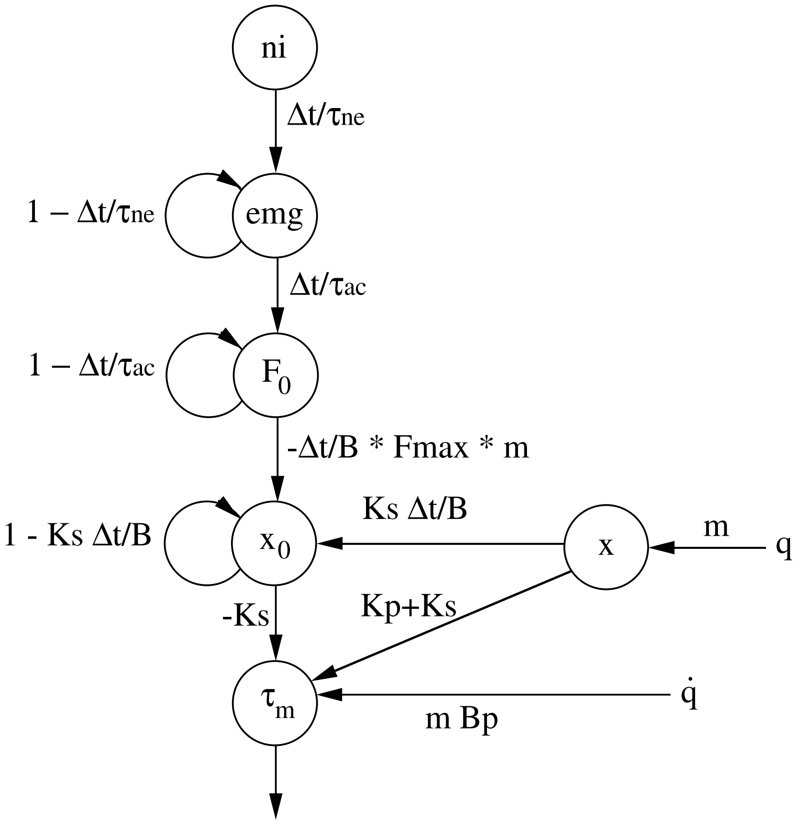

Figure 5: The recurrent network that implements the relationship between neural input and torque for a generic muscle. The lettering inside the network's units indicates the variable output by each unit. The lettering outside the network's units indicates the values of the weights. All units are linear.

the joint torques as the difference between the contribution of the agonist and antagonist muscles impinging on each joint because each agonist/antagonist pair produces torques that pull the arm in opposite directions. We assumed that the double-joint muscles contributed to the shoulder and elbow torques with fixed portion of their torques (a and b in Figure 6, with $a + b = 1$). The following simple equations describe the transformation between muscle torques and joint torques and dictated the network structure shown in Figure 6.

$$\tau_1 = (\tau_{sh-ag}) - (\tau_{sh-ant}) + a \cdot (\tau_{dj-ag}) - a \cdot (\tau_{dj-ant}) \tag{12}$$

$$\tau_2 = (\tau_{el-ag}) - (\tau_{el-ant}) + b \cdot (\tau_{dj-ag}) - b \cdot (\tau_{dj-ant}) \tag{13}$$

The connection strengths a and b can be either arbitrarily set or learned at the time that the arm controller is learned. For all the simulations presented in [1] we chose a equal to 0.6 and b equal to 0.4.

2.4 The Forward Dynamics

The forward dynamics of the arm is defined as a mapping from joint torques, angular positions and angular velocities to angular accelerations. Such a mapping can be represented as shown

Parameter	Value	Units
B_p	0.2	$N \cdot sec/rad$
B	4.5	$N \cdot sec/rad$
K_p	1.5	N/rad
K_s	30	N/rad
τ_{ne}	0.035	sec
τ_{ac}	0.005	sec
m	0.03	m
F_{max}(shoulder)	800	N
F_{max}(elbow)	700	N
F_{max}(double joint)	1000	N

Table 1: The chosen values for the muscles' parameters.

in Figure 7a, where the new values of $\dot{\underline{\theta}}$ and $\underline{\theta}$ are derived from the computed accelerations with two cascaded integrations. One could also view the forward dynamics as a black box that computes directly angular velocities and positions as indicated in Figure 7b. If one adopts the scheme of Figure 7a, the forward dynamics of a two-joint arm is a mapping from $\Re^6$ to $\Re^2$. If one adopts the scheme of Figure 7b, the forward dynamics of a two-joint arm becomes instead a mapping from $\Re^6$ to $\Re^4$. In any case, the forward-dynamic transformation is complicated and highly nonlinear. In order to cope with its complexity and nonlinearity we followed an approach based on learning.

No matter which scheme one adopts (Figure 7a or Figure 7b), learning the forward dynamic transformation can be a prohibitive task, because it requires the exploration of a six-dimensional input space. Such an exploration leads to a number of items in the training set equal to 6^N, where N is the number of samples along each dimension. Even in a situation in which each of the input dimensions is coarsely sampled the training set can become easily unmanageable and the training procedure extremely lengthy. One possibility for overcoming these problems is to construct the training set in a goal-oriented fashion, i.e., so as to cover a subset of the six-dimensional input space related to some specific motor task. We chose to construct the training set for the forward dynamic transformation of a reaching task in which the endpoint was initially positioned in a fixed location of the workspace and then moved towards a number of predefined targets along straight paths. The eight reaching movements that we chose to construct the training set for the forward-dynamic transformation are depicted in Figure 8. The procedure that we followed to actually construct the training set will be described shortly. The network that we employed to learn the transformation is shown in Figure 9, and it is a network realization of the scheme of Figure 7a. The advantage of this scheme over the scheme of Figure 7b is that it contains a causal relationship between velocity and acceleration and between position and velocity: the resulting velocity is an integrated acceleration and the resulting position is an integrated velocity. Another advantage is the possibility of varying the sampling step of the system by simply modifying the strength of some connections, namely the connections that contain Δt (see Figure 9.) This makes it possible to observe the behavior of the arm at variable temporal scales. The disadvantage, however, is that when the network operates after learning, the residual training errors are amplified by the two cascaded integrators and by the recurrent connections to the input layer, and can

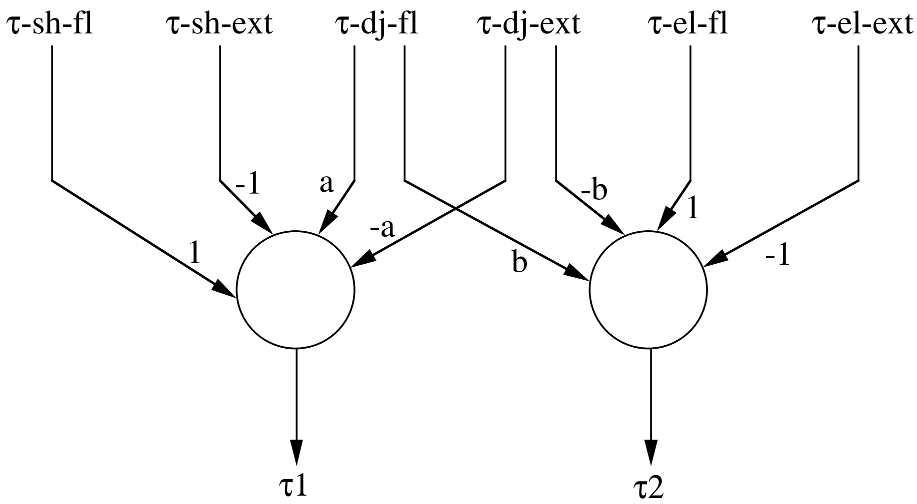

Figure 6: The network that transforms muscle torques into joint torques.

hence lead to undesired behaviors, namely instability. The latter problem does not emerge as strongly in the scheme of Figure 7b because the two cascaded integrations are incorporated in the input-output mapping that the network learns. However this scheme does not contain a causal relationship between position and velocity: the actual relationship between position and velocity depends upon how the residual training errors are distributed over the six-dimensional input space. This is a major drawback, that can result, for example, in the arm settling in an equilibrium configuration with non-zero velocity. Furthermore the fact that this method incorporates the cascaded integrations in the training set implies that the sampling step Δt must be chosen at the time that the training set is built and kept fixed. Hence this scheme is also less flexible than the scheme of Figure 7a. Because flexibility is, in general, a desirable quality in a model, we decided not to consider a hybrid scheme in which the forward dynamic block computes the angular velocities that are then integrated to yield angular positions. Although this method exhibits causality between position and velocity, it requires, like the scheme of Figure 7b, that the time step Δt be hard-wired in the training set. These observations point out that although the three schemes are similar in their outcome (they all compute velocity and position signals), their network realization can lead to quite different performances. As pointed out previously, we chose the scheme of Figure 7a for our model, and we coped with the problem of the amplified errors due to the cascaded integrators in two ways: by using small sampling steps (in the range 1 to 5 *msec.*), which is also a requirement of the Euler method utilized for the integrations in this network and in the muscles networks, and by oversampling the training set in the proximity of the starting posture of the arm in order to enforce stability.

We hence trained the network shown in Figure 9 in a supervised fashion to compute the forward dynamic transformation. The weights that were learned are the weights from the input layer to the hidden layer and the weights from the hidden layer to the two units that output the two angular accelerations. All the other weights were held fixed.

The remainder of this section describes the procedure that we followed in order to construct the training set.

We sampled the endpoint coordinates of each of the eight movements shown in Figure 8

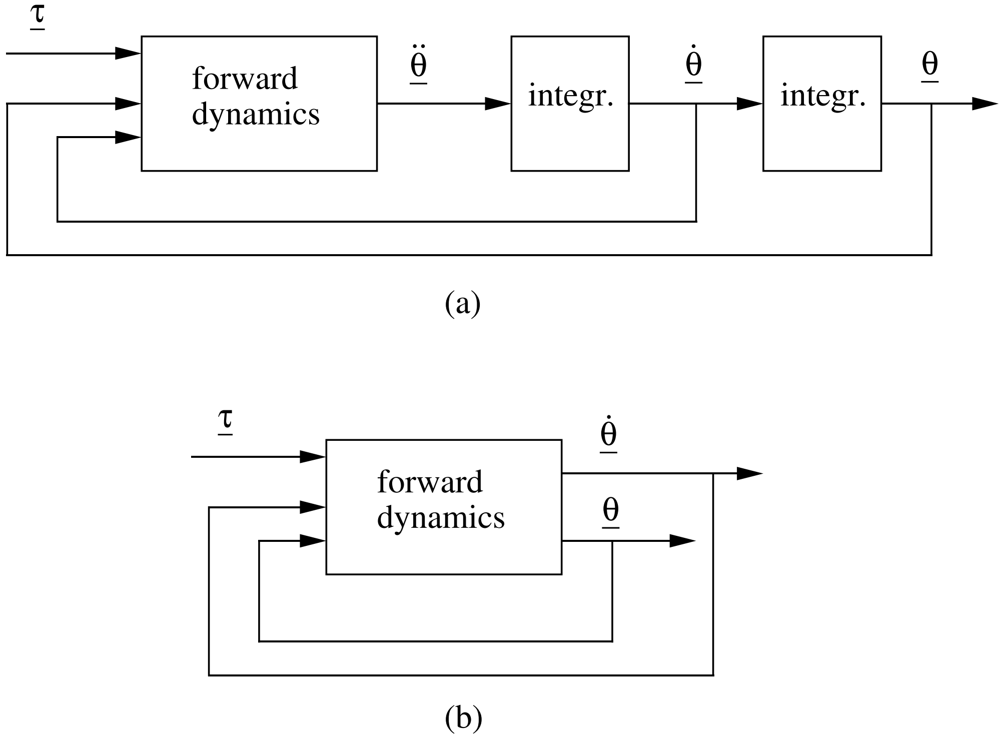

Figure 7: (a) A representation of the forward dynamics as a mapping from joint torques, angular positions and angular velocities to angular accelerations. (b) A black-box representation of the forward dynamics, in which the output signals are angular positions and angular velocities.

with 20 samples in each direction (from the initial position to the target and from the target back to the initial position) assuming a bell-shaped velocity profile and that the movement speed would scale with the movement amplitude (constant duration of 400 *msec.*) We chose a bell-shaped velocity profile as a sampling strategy in order to oversample the space in the proximity of the initial position which, as previously pointed out, is important for the stability of the system. For each of the resulting 320 samples we computed the corresponding values of the joint angles using the following standard inverse kinematic equations:

$$\theta_1 = \arctan\left(\frac{y}{x}\right) - \arctan\left(\frac{\ell_2 \sin(\theta_2)}{\ell_1 + \ell_2 \cos(\theta_2)}\right) \tag{14}$$

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - \ell_1^2 - \ell_2^2}{2\ell_1\ell_2}\right) \tag{15}$$

where x and y are the coordinates of the endpoint. We then computed the first and second time derivatives of equations (14) and (15) in order to obtain angular velocities and angular accelerations as functions of the endpoint positions, velocities and accelerations. Because the resulting equations are extremely lengthy, we present them in Appendix B. We then substituted the resulting values of angular positions, velocities and accelerations for each of the 320 samples in the equations that describe the inverse dynamics of a two-joint arm with

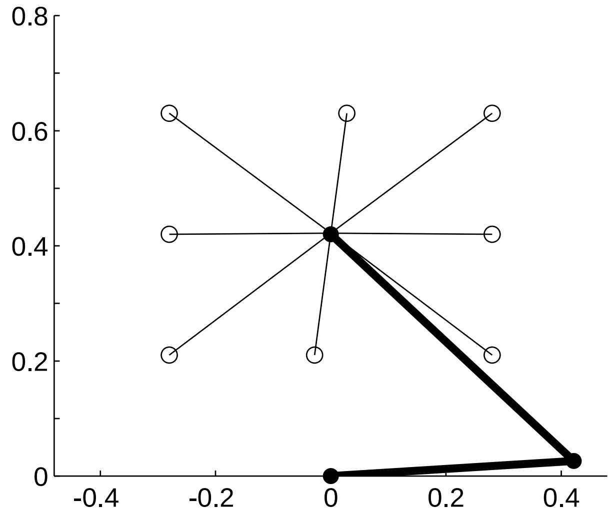

Figure 8: The reaching movements that were chosen to construct the training set for the forward dynamic network. The zero of the horizontal axis indicates the location of the shoulder. The scales on both axes are normalized to the arm length.

gravity [13]:

$$\begin{aligned}
\tau_1 \quad = \quad & m_1 c_{x_1} g \cos(\theta_1) - m_1 c_{y_1} g \sin(\theta_1) + I_1 \ddot{\theta}_1 + m_2[g \ell_1 \cos(\theta_1) + \ell_1^2 \ddot{\theta}_1] + I_2(\ddot{\theta}_1 + \ddot{\theta}_2) + \\
& m_2 c_{x_2}[g \cos(\theta_1 + \theta_2) + 2\ell_1 \ddot{\theta}_1 \cos(\theta_2) + \ell_1 \ddot{\theta}_2 \cos(\theta_2) - 2\ell_1 \dot{\theta}_1 \dot{\theta}_2 \sin(\theta_2) - \ell_1 \dot{\theta}_2^2 \sin(\theta_2)] + \\
& m_2 c_{y_2}[-g \sin(\theta_1 + \theta_2) - 2\ell_1 \ddot{\theta}_1 \sin(\theta_2) - \ell_1 \ddot{\theta}_2 \sin(\theta_2) - 2\ell_1 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_2) - \\
& \ell_1 \dot{\theta}_2^2 \cos(\theta_2)] \qquad (16)
\end{aligned}$$

$$\begin{aligned}
\tau_2 \quad = \quad & m_2 c_{x_2}[g \cos(\theta_1 + \theta_2) + \ell_1 \ddot{\theta}_1 \cos(\theta_2) + \ell_1 \dot{\theta}_1^2 \sin(\theta_2)] + \\
& m_2 c_{y_2}[-g \sin(\theta_1 + \theta_2) - \ell_1 \ddot{\theta}_1 \sin(\theta_2) + \ell_1 \dot{\theta}_1^2 \cos(\theta_2)] + I_2(\ddot{\theta}_1 + \ddot{\theta}_2) \qquad (17)
\end{aligned}$$

where m_1, m_2 are the arm links masses, $c_{x_1}, c_{y_1}, c_{x_2}, c_{y_2}$ are the coordinates of the centers of mass of the two links, I_1, I_2 are the inertias of the two links. We chose these parameters as follows [13]:

$$m_1 = 0.028 \cdot W \qquad (18)$$

$$m_2 = 0.022 \cdot W \qquad (19)$$

$$c_{x_1} = 0.436 \cdot \ell_1 \cdot \cos(\theta_1) \qquad (20)$$

$$c_{y_1} = 0.436 \cdot \ell_1 \cdot \sin(\theta_1) \qquad (21)$$

$$c_{x_2} = 0.682 \cdot \ell_2 \cdot \cos(\theta_2) \qquad (22)$$

$$c_{y_2} = 0.682 \cdot \ell_2 \cdot \sin(\theta_2) \qquad (23)$$

$$I_1 = m_1 \cdot 0.104 \cdot \ell_1^2 \qquad (24)$$

$$I_2 = m_2 \cdot 0.220 \cdot \ell_2^2 \qquad (25)$$

where W is the body weight and ℓ_1, ℓ_2 are the lengths of the two links, with ℓ_1 representing the length of the upper arm and ℓ_2 representing the length of forearm plus wrist. We computed

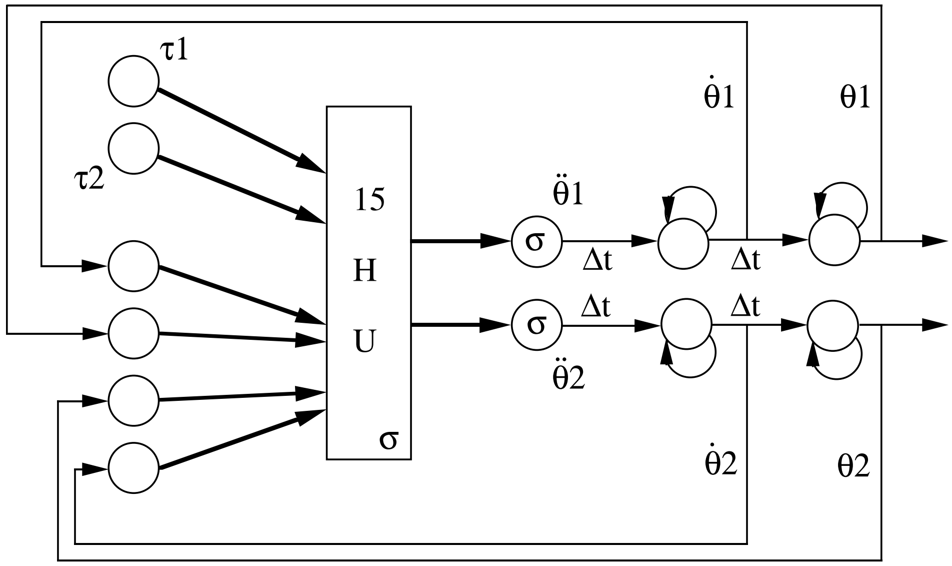

Figure 9: The recurrent network that was trained to implement the arm forward dynamics. Sigmoidal units are indicated with σ. All other units are linear. HU: hidden units.

the values of ℓ_1 and ℓ_2 according to anthropometric tables [13] as follows:

$$\ell_1 = 0.186 \cdot H \tag{26}$$

$$\ell_2 = 0.254 \cdot H \tag{27}$$

where H represents the body height. We chose H and W so as to fit the body of one of the authors (JM).

The substitution of angular positions, velocities and accelerations in equations (16) and (17) allowed us to compute the values of τ_1 and τ_2 for each of the 320 samples. We then used the values of torques, angular positions and angular velocities as the input values of the training set and the values of angular accelerations as the output values of the training set. The fact that we used an analytical method to compute the derivatives of the angular positions guarantees that our training set is free from numerical errors. We then trained the three-layer network on the resulting training set using a standard backpropagation algorithm [14] and a teacher-forcing technique[2] [15], [16] for the recurrent connections to the input layer. The network was able to learn well the training set and to generalize acceptably to values of torques, angular positions and angular velocities other than those contained in the training set. In [1] we further discuss the outcome of the learning as well as the generalization properties of this network. It is worth pointing out here that the recurrent connections to the input layer play no role during the training phase and that they are utilized only when the trained network operates. When teacher forcing is in place, the input units are, in fact, always presented with the correct values of angular positions and velocities rather than the values computed by the network. Another positive effect of teacher forcing is that the sampling step Δt that appears as a parameter of the weights that integrate the values of acceleration

[2]Teacher forcing is a technique that consists of presenting a recurrent network with the correct input values rather than the values computed by the network at every learning iteration. This technique greatly enhances the learning abilities of a recurrent network because wrong input values are never encountered. In our case teacher forcing was used for the two angular positions and the two angular velocites.

and velocity also does not play a role during training and can hence be modified at any time without the need for retraining the network.

2.5 The Forward Kinematics

Figure 10 shows the network that implements the forward kinematics transformation. We

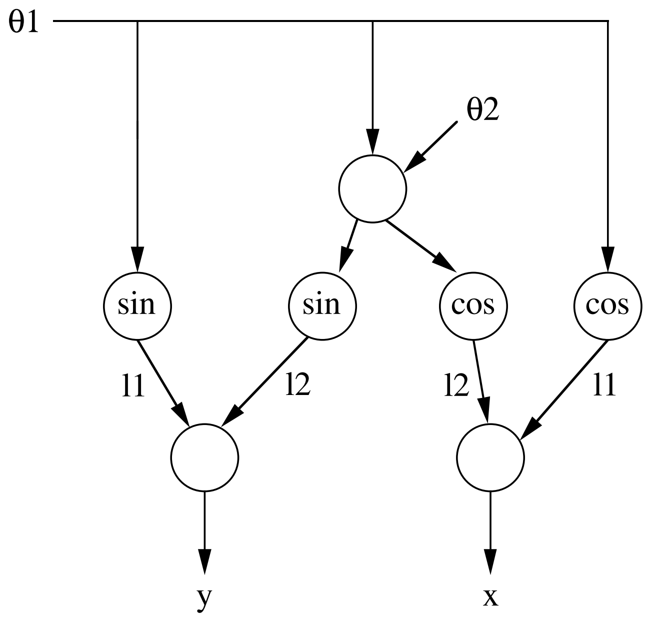

Figure 10: The network that implements the forward kinematics. ℓ_1, ℓ_2: link lengths; x, y: endpoint coordinates. The empty circles indicate linear units. The other units have a sine or a cosine as their activation function.

chose the architecture and the values of the weights for this network so as to implement the standard forward kinematics equations:

$$x = \ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1 + \theta_2) \tag{28}$$

$$y = \ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2) \tag{29}$$

2.6 Networks Initialization

In order for the model to be operative, the various networks must be properly initialized. This section describes a possible initialization procedure. We assumed that the endpoint was positioned as shown in Figure 8 with the arm in an equilibrium configuration (all time derivatives equal to zero). We then initialized the angular coordinates $(\theta_1(0), \theta_2(0))$ according to equations (14) and (15) and the angular velocities to zero. We computed the values of the joint torques for the initial posture $(\tau_1(0), \tau_2(0))$ from equations (16) and (17) and we distributed them over the six muscles taking into account the constraints dictated by equations (12) and (13). Among the many possible configurations, we picked one of the configurations for the initial muscle torques that would cause all the neural inputs to be in the range $[0, 1]$. Let's call the initial value of the torque for a generic muscle $\tau_m(0)$. Then for each muscle we computed the initial values of T_0, F_0, emg and ni as follows. Equations (7) and (8) gave us the initial values for x and $\dot{x}$:

$$x(0) = q(0) \cdot m \tag{30}$$

$$\dot{x}(0) = 0 \tag{31}$$

We then used equation (6) to compute the initial value of x_0

$$x_0(0) = \frac{K_p + K_s}{K_s} x(0) - \frac{1}{K_s} \tau_m(0) \tag{32}$$

and equation (5) with $\dot{x_0}$ equal to zero to compute the initial value of T_0:

$$T_0(0) = K_s(x(0) - x_0(0)) \tag{33}$$

The initial value for F_0 was hence given by equation (4):

$$F_0(0) = T_0(0)/(F_{max} * m) \tag{34}$$

Finally we computed the initial values for ni and emg, respectively, by setting the left sides of equations (2) and (3) equal to zero:

$$ni(0) = emg(0) = F_0(0) \tag{35}$$

Figure 11 shows the resulting initial configuration of the neural inputs. This activity represents

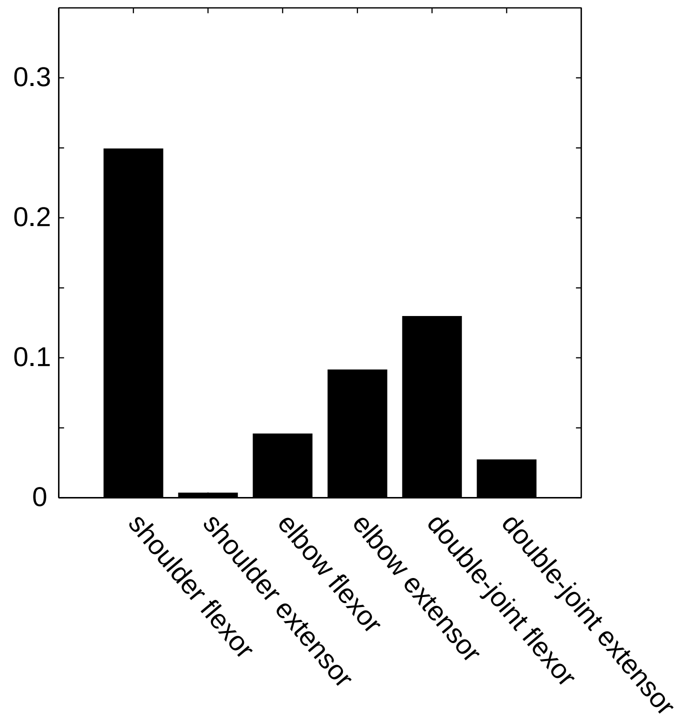

Figure 11: The initial configuration of the neural inputs.

the tonic component necessary to counteract the action of gravity. Some of the other resulting values that we used to initialize the networks are given in Appendix C. The configuration shown in Figure 11 is by no means the only possible one, because there are many configurations of the muscle torques that can produce the same amount of joint torques. It is, however, the configuration that we chose to experiment with our model [1].

3 Discussion

In this paper we presented a neural-network model of a planar redundant arm whose structure and operation principles were inspired by those of the human arm. The motivation for choosing the human arm as a reference is that the properties of the human arm, particularly the elastic properties of muscles, are likely to be an important component of the human ability to exhibit dexterous skilled behavior – a kind of behavior yet to be achieved with artificial arms. We modeled the arm with two joints (shoulder and elbow) and six muscle-like actuators: a pair of antagonist shoulder muscles, a pair of antagonist elbow muscles, a pair of antagonist double-joint muscles. The arm was allowed to move in the horizontal plane subject to the action of gravity. The model computes the transformation between the control signals that activate the muscle-like actuators and the arm endpoint. This transformation occurs in four stages:

(1) The muscles are activated by their neural inputs $\underline{ni}$ and they produce torques $\underline{\tau_m}$ as functions of the neural inputs, the angular positions $\underline{\theta}$ and the angular velocities $\underline{\dot{\theta}}$.

(2) The muscles torques are combined into joint torques $\underline{\tau}$ according to geometric principles.

(3) The forward dynamic block computes the angular accelerations $\underline{\ddot{\theta}}$ as functions of the current values of $\underline{\tau}, \underline{\theta}, \underline{\dot{\theta}}$. The resulting accelerations are then integrated to yield the new values of $\underline{\theta}$ and $\underline{\dot{\theta}}$.

(4) The forward kinematic block transforms the angular coordinates into coordinates of the endpoint ($\underline{x}$).

The various stages interact with each other through a number of recurrent connections.

We modeled each stage of the overall transformation with feedforward and recurrent networks. For stage (1) we utilized six recurrent networks, one for each muscle. We derived the architecture and weights for such networks from the equations that describe the action of an elastic actuator according to a linearized version of the Hill's model. For stage (2) we utilized a linear feedforward network that implemented simple geometrical constraints at the joints. Stage (3) consisted of training a nonlinear recurrent network to compute the forward dynamics of the arm. We built the training set for this network in a goal-oriented fashion in order to contain its size and the length of the training procedure. We computed its elements analytically in order to eliminate all possible numerical errors. The training procedure consisted of a standard backpropagation algorithm plus teacher forcing – a technique that consists of always presenting the network with the correct input values rather than the computed ones. In Section 2.4 we discussed the pros and cons of different neural architectures for the forward dynamic transformation and we laid out the arguments for our choice: (i) causality between velocity and acceleration and between position and velocity, (ii) flexibility in the choice of the integration step Δt because with our learning scheme Δt plays no role in the training procedure. The step, however, must be kept reasonably small in order to avoid amplification of the residual learning errors, and in order to also satisfy a requirement of the Euler integration method. We must point out that the training of the forward dynamic network is crucial to the functioning of the overall model because the latter relies on the generalization abilities of the former in order to operate. Finally we implemented stage (4) with a feedforward nonlinear network, whose architecture and weights were derived from the equations of the forward kinematics of the arm. The resulting overall model is highly nonlinear due to both the inherent nonlinearities of some of the stages (e.g., stages 3 and 4) and the interactions among stages via their recurrent connections. In [1] we demonstrate the response of this model to various patterns of neural input and we analyze the resulting movements. It is important to point out here that three of the four stages of the transformation that the model computes – stages (1), (2) and (4) – did not require learning. Consequently their corresponding networks can be easily modified to accommodate different choices for the muscle parameters, the joint geometry and the link lengths because all this information was expressed explicitly in the strength of the connections. This feature is very important for eventual analyses of the model's robustness to parametric changes. Furthermore, it is possible, with our approach, to select the sampling step and hence observe the response of the system at different temporal scales. However, modification of the arm parameters requires retraining of the forward dynamic network.

We would like now to compare our model to another neural-network-based model of a two-joint arm – the model by Jordan et al. [8]. Although the main components of Jordan et al.'s model are similar to ours, the two models differ in a number of aspects. For example, Jordan et al.'s model does not represent the effects of gravity, does not include double-joint muscles and utilizes a different approach [17] for the muscle networks. Furthermore, the two models differ in the procedure followed to train the forward dynamic network. In Jordan et al.'s model the items that constitute the training set were derived by providing random set points to a PD controller for a simulated two-joint arm. In our learning scheme we chose the items for the training set in a goal-oriented fashion rather than randomly and we demonstrated an analytical method to compute the angular accelerations based on derivatives of the inverse kinematic equations of the arm and on the inverse dynamics of the arm. Such method is exact and naturally free from numerical errors. We must however point out that our analytical method is applicable only if the arm is not redundant in joint space (as is the case in our model and in Jordan et al.'s model) because it relies on the explicit computation of the inverse kinematics. Additional differences between the two models are the modifiable sampling step (a feature not present in Jordan et al.'s model) and our use of an equation-based network for the forward kinematics of the arm vs. a learning-based approach. Because Jordan et al. addressed in their paper some interesting control issues, in [1] we further elaborate on their findings as well as on findings by other investigators.

Finally, we would like to emphasize the generality of the modeling methods presented in this paper, particularly the methods that we utilized to model the muscle-like actuators and the forward dynamics of the arm. The procedure that we followed to design the muscles' recurrent networks can, in principle, be followed to model in a neural fashion any phenomenon described by linear differential equations. Furthermore, our solution to the forward-dynamics problem is a demonstration of how to exploit the natural ability of multilayer networks to handle nonlinearities and of how to construct an exact, error-free training set for a complicated and highly nonlinear transformation. These two aspects, nonlinear transformations and construction of their training sets, can be found in a number of different scientific domains.

Acknowledgements. This research was supported by the National Science Foundation, grant BCS-9113455 to Lina L.E. Massone. Thanks to Wynne Lee for her comments on the manuscript.

References

[1] L.L.E. Massone and J.D. Myers, "A study of the role of the plant properties in arm trajectory formation," *IEEE Trans. SMC*, submitted.

[2] M.I. Jordan, "Motor learning and the degrees of freedom problem," in *Attention and Performance XIII*, M. Jeannerod, Ed., pp. 796-836. Hillsdale, NJ: Lawrence Erlbaum, 1990.

[3] L. Massone, "Sensorimotor learning," in *The Handbook of Brain Theory and Neural Networks*, M.A. Arbib, Ed., Cambridge, MA: MIT Press, to appear.

[4] M.I. Jordan and D.E. Rumelhart, "Forward models: supervised learning with a distal teacher," *Cognitive Science*, vol. 16, pp. 307-354, 1992.

[5] A.G. Barto, "Connectionist learning for control: an overview," in *Neural Networks for Control*, W.T. Miller, R.S. Sutton and P. Werbos, Eds., Cambridge, MA: MIT Press, pp. 5-58, 1990.

[6] M. Kawato, M. Isobe, Y. Maeda and R. Suzuki, "Coordinate transformation and learning control for visually guided voluntary movements with iteration: a Newton-like method in function space," *Biological Cybernetics*, vol. 59, pp. 161-177, 1988.

[7] L. Massone and E. Bizzi, "A neural network model for limb trajectory formation," *Biological Cybernetics*, vol. 61, pp. 417-425, 1989.

[8] M.I. Jordan, T. Flash and Y. Arnon, "A Model of the learning of arm trajectories from spatial deviations," *Journal of Cognitive Neuroscience*, in press.

[9] K.J. Quinn, N. Schmajuk, J.F. Baker and B.W. Peterson, "Simulation of adaptive mechanisms in the vestibulo-ocular reflex," *Biological Cybernetics*, vol. 67, pp. 103-112, 1992.

[10] A.V. Hill, "The heat of shortening and the dynamic constants of muscle," *Proc. Royal Society of London*, vol. B126, pp. 136-195, 1938.

[11] J.M. Winters and L. Stark, "Muscle models: what is gained and what is lost by varying model complexity," *Biological Cybernetics*, vol. 55, pp. 403-420, 1987.

[12] J.M. Winters and L. Stark, "Analysis of fundamental human movement patterns through the use of in-depth antagonistic muscle models," *IEEE Trans. Biomed. Engng.*, vol. BME-12, pp. 826-839, 1985.

[13] C.G. Atkeson, "Learning arm kinematics and dynamics," *Annual Review of Neuroscience*, vol. 12, pp. 157-183, 1989.

[14] D.A. Winter, *Biomechanics of Human Movement*, New York: John Wiley and Sons, 1991.

[15] M.I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," *Proc. 8th Annual Conference of the Cognitive Science Society*, 531-546, 1986.

[16] F.J. Pineda, "Dynamics and architecture for neural computation," *Journal of Complexity*, vol. 4, pp. 216-245, 1988.

[17] W. Zangemeister, S. Lehman and L. Stark, "Simulation of head movement trajectories: model and fit to the main sequence," *Biological Cybernetics*, vol. 41, pp. 19-32, 1981.

A Model geometry

Angles subtended by the six muscles expressed as a function of θ_1 and θ_2, the angles of the forward kinematic transformation (see Figure 12.)

Shoulder Flexor: $q = \frac{3\pi}{2} - \theta_1, \dot{q} = -\dot{\theta}_1$

Shoulder Extensor: $q = \frac{\pi}{2} + \theta_1, \dot{q} = \dot{\theta}_1$

Elbow Flexor: $q = \pi - \theta_2, \dot{q} = -\dot{\theta}_2$

Elbow Extensor: $q = \pi + \theta_2, \dot{q} = \dot{\theta}_2$

Double-Joint Flexor: $q = \frac{3\pi}{2} - \theta_1 - \theta_2, \dot{q} = -\dot{\theta}_1 - \dot{\theta}_2$

Double-Joint Extensor: $q = \frac{\pi}{2} + \theta_1 + \theta_2, \dot{q} = \dot{\theta}_1 + \dot{\theta}_2$

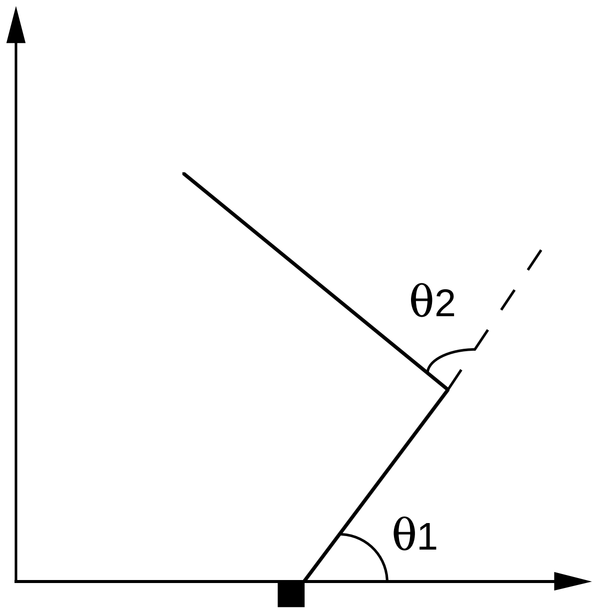

Figure 12: The two-joint arm.

B Inverse kinematics

We present here the equations that express angular velocities and accelerations as functions of the endpoint coordinates, velocities and accelerations. We derived these equations by computing the first and second derivatives of equations (14) and (15).

$$
\begin{aligned}
\dot{\theta}_1 &= \frac{-(y\dot{x}) + x\dot{y}}{x^2+y^2} - \\
&\quad \frac{\ell_1^4 x\dot{x} - \ell_1^2\ell_2^2 x\dot{x} - \ell_1^2 x^3\dot{x} - \ell_1^2 x y^2\dot{x} + \ell_1^4 y\dot{y} - \ell_1^2\ell_2^2 y\dot{y} - \ell_1^2 x^2 y\dot{y} - \ell_1^2 y^3\dot{y}}{2\ell_1^3\ell_2\left(x^2+y^2\right)\sqrt{1-\frac{\left(-\ell_1^2-\ell_2^2+x^2+y^2\right)^2}{4\ell_1^2\ell_2^2}}} \qquad (36)
\end{aligned}
$$

$$
\dot{\theta}_2 = \frac{-x\dot{x} - y\dot{y}}{\ell_1\ell_2\sqrt{1-\frac{\left(-\ell_1^2-\ell_2^2+x^2+y^2\right)^2}{4\ell_1^2\ell_2^2}}} \qquad (37)
$$

$$
\begin{aligned}
\ddot{\theta}_1 &= +\frac{\left(\ell_1^4 x\dot{x} - \ell_1^2\ell_2^2 x\dot{x} - \ell_1^2 x^3\dot{x} - \ell_1^2 x y^2\dot{x} + \ell_1^4 y\dot{y} - \ell_1^2\ell_2^2 y\dot{y} - \ell_1^2 x^2 y\dot{y} - \ell_1^2 y^3\dot{y}\right)^2}{\ell_1^5\ell_2(x^2+y^2)^2\left(\ell_1^2-\ell_2^2+x^2+y^2\right)\sqrt{1-\frac{\left(-\ell_1^2-\ell_2^2+x^2+y^2\right)^2}{4\ell_1^2\ell_2^2}}} \\
&\quad + \frac{2y\dot{x}^2 - 2x\dot{x}\dot{y} - xy\ddot{x} + x^2\ddot{y}}{x\left(x^2+y^2\right)} - \frac{\left(-(y\dot{x}) + x\dot{y}\right)\left(-2y^2\dot{x} + 2xy\dot{y}\right)}{x(x^2+y^2)^2} \\
&\quad - \frac{1}{8\ell_1^5\ell_2^3\left(x^2+y^2\right)\left(\ell_1^2-\ell_2^2+x^2+y^2\right)\left(1-\frac{\left(-\ell_1^2-\ell_2^2+x^2+y^2\right)^2}{4\ell_1^2\ell_2^2}\right)^{\frac{3}{2}}} \cdot \\
&\quad [-\left(\ell_1^{10}\dot{x}^2\right) + 4\ell_1^8\ell_2^2\dot{x}^2 - 6\ell_1^6\ell_2^4\dot{x}^2 + 4\ell_1^4\ell_2^6\dot{x}^2 - \ell_1^2\ell_2^8\dot{x}^2 + 6\ell_1^8 x^2\dot{x}^2 - 18\ell_1^6\ell_2^2 x^2\dot{x}^2 \\
&\quad + 18\ell_1^4\ell_2^4 x^2\dot{x}^2 - 6\ell_1^2\ell_2^6 x^2\dot{x}^2 - 12\ell_1^6 x^4\dot{x}^2 + 12\ell_1^2\ell_2^4 x^4\dot{x}^2 + 10\ell_1^4 x^6\dot{x}^2 - 2\ell_1^2\ell_2^2 x^6\dot{x}^2 - 3\ell_1^2 x^8\dot{x}^2 \\
&\quad + 2\ell_1^8 y^2\dot{x}^2 - 2\ell_1^6\ell_2^2 y^2\dot{x}^2 - 2\ell_1^4\ell_2^4 y^2\dot{x}^2 + 2\ell_1^2\ell_2^6 y^2\dot{x}^2 - 12\ell_1^6 x^2 y^2\dot{x}^2 + 12\ell_1^2\ell_2^4 x^2 y^2\dot{x}^2 \\
&\quad + 18\ell_1^4 x^4 y^2\dot{x}^2 - 6\ell_1^2\ell_2^2 x^4 y^2\dot{x}^2 - 8\ell_1^2 x^6 y^2\dot{x}^2 + 6\ell_1^4 x^2 y^4\dot{x}^2 - 6\ell_1^2\ell_2^2 x^2 y^4\dot{x}^2 - 6\ell_1^2 x^4 y^4\dot{x}^2 \\
&\quad - 2\ell_1^4 y^6\dot{x}^2 - 2\ell_1^2\ell_2^2 y^6\dot{x}^2 + \ell_1^2 y^8\dot{x}^2 + 8\ell_1^8 xy\dot{x}\dot{y} - 32\ell_1^6\ell_2^2 xy\dot{x}\dot{y} + 40\ell_1^4\ell_2^4 xy\dot{x}\dot{y} \\
&\quad - 16\ell_1^2\ell_2^6 xy\dot{x}\dot{y} - 24\ell_1^6 x^3 y\dot{x}\dot{y} + 24\ell_1^2\ell_2^4 x^3 y\dot{x}\dot{y} + 24\ell_1^4 x^5 y\dot{x}\dot{y} - 8\ell_1^2 x^7 y\dot{x}\dot{y} - 24\ell_1^6 xy^3\dot{x}\dot{y} \\
&\quad + 24\ell_1^2\ell_2^4 xy^3\dot{x}\dot{y} + 48\ell_1^4 x^3 y^3\dot{x}\dot{y} - 24\ell_1^2 x^5 y^3\dot{x}\dot{y} + 24\ell_1^4 xy^5\dot{x}\dot{y} - 24\ell_1^2 x^3 y^5\dot{x}\dot{y} - 8\ell_1^2 xy^7\dot{x}\dot{y} \\
&\quad - \ell_1^{10}\dot{y}^2 + 4\ell_1^8\ell_2^2\dot{y}^2 - 6\ell_1^6\ell_2^4\dot{y}^2 + 4\ell_1^4\ell_2^6\dot{y}^2 - \ell_1^2\ell_2^8\dot{y}^2 + 2\ell_1^8 x^2\dot{y}^2 - 2\ell_1^6\ell_2^2 x^2\dot{y}^2 - 2\ell_1^4\ell_2^4 x^2\dot{y}^2 \\
&\quad + 2\ell_1^2\ell_2^6 x^2\dot{y}^2 - 2\ell_1^4 x^6\dot{y}^2 - 2\ell_1^2\ell_2^2 x^6\dot{y}^2 + \ell_1^2 x^8\dot{y}^2 + 6\ell_1^8 y^2\dot{y}^2 - 18\ell_1^6\ell_2^2 y^2\dot{y}^2 + 18\ell_1^4\ell_2^4 y^2\dot{y}^2 \\
&\quad - 6\ell_1^2\ell_2^6 y^2\dot{y}^2 - 12\ell_1^6 x^2 y^2\dot{y}^2 + 12\ell_1^2\ell_2^4 x^2 y^2\dot{y}^2 + 6\ell_1^4 x^4 y^2\dot{y}^2 - 6\ell_1^2\ell_2^2 x^4 y^2\dot{y}^2 - 12\ell_1^6 y^4\dot{y}^2 \\
&\quad + 12\ell_1^2\ell_2^4 y^4\dot{y}^2 + 18\ell_1^4 x^2 y^4\dot{y}^2 - 6\ell_1^2\ell_2^2 x^2 y^4\dot{y}^2 - 6\ell_1^2 x^4 y^4\dot{y}^2 + 10\ell_1^4 y^6\dot{y}^2 - 2\ell_1^2\ell_2^2 y^6\dot{y}^2 \\
&\quad - 8\ell_1^2 x^2 y^6\dot{y}^2 - 3\ell_1^2 y^8\dot{y}^2 - \ell_1^{10} x\ddot{x} + 4\ell_1^8\ell_2^2 x\ddot{x} - 6\ell_1^6\ell_2^4 x\ddot{x} + 4\ell_1^4\ell_2^6 x\ddot{x} - \ell_1^2\ell_2^8 x\ddot{x} + 2\ell_1^8 x^3\ddot{x} \\
&\quad - 2\ell_1^6\ell_2^2 x^3\ddot{x} - 2\ell_1^4\ell_2^4 x^3\ddot{x} + 2\ell_1^2\ell_2^6 x^3\ddot{x} - 2\ell_1^4 x^7\ddot{x} - 2\ell_1^2\ell_2^2 x^7\ddot{x} + \ell_1^2 x^9\ddot{x} + 2\ell_1^8 xy^2\ddot{x} \\
&\quad - 2\ell_1^6\ell_2^2 xy^2\ddot{x} - 2\ell_1^4\ell_2^4 xy^2\ddot{x} + 2\ell_1^2\ell_2^6 xy^2\ddot{x} - 6\ell_1^4 x^5 y^2\ddot{x} - 6\ell_1^2\ell_2^2 x^5 y^2\ddot{x} + 4\ell_1^2 x^7 y^2\ddot{x} - 6\ell_1^4 x^3 y^4\ddot{x} \\
&\quad - 6\ell_1^2\ell_2^2 x^3 y^4\ddot{x} + 6\ell_1^2 x^5 y^4\ddot{x} - 2\ell_1^4 xy^6\ddot{x} - 2\ell_1^2\ell_2^2 xy^6\ddot{x} + 4\ell_1^2 x^3 y^6\ddot{x} + \ell_1^2 xy^8\ddot{x} - \ell_1^{10} y\ddot{y} \\
&\quad + 4\ell_1^8\ell_2^2 y\ddot{y} - 6\ell_1^6\ell_2^4 y\ddot{y} + 4\ell_1^4\ell_2^6 y\ddot{y} - \ell_1^2\ell_2^8 y\ddot{y} + 2\ell_1^8 x^2 y\ddot{y} - 2\ell_1^6\ell_2^2 x^2 y\ddot{y} - 2\ell_1^4\ell_2^4 x^2 y\ddot{y} \\
&\quad + 2\ell_1^2\ell_2^6 x^2 y\ddot{y} - 2\ell_1^4 x^6 y\ddot{y} - 2\ell_1^2\ell_2^2 x^6 y\ddot{y} + \ell_1^2 x^8 y\ddot{y} + 2\ell_1^8 y^3\ddot{y} - 2\ell_1^6\ell_2^2 y^3\ddot{y} - 2\ell_1^4\ell_2^4 y^3\ddot{y} \\
&\quad + 2\ell_1^2\ell_2^6 y^3\ddot{y} - 6\ell_1^4 x^4 y^3\ddot{y} - 6\ell_1^2\ell_2^2 x^4 y^3\ddot{y} + 4\ell_1^2 x^6 y^3\ddot{y} - 6\ell_1^4 x^2 y^5\ddot{y} - 6\ell_1^2\ell_2^2 x^2 y^5\ddot{y} + 6\ell_1^2 x^4 y^5\ddot{y} \\
&\quad - 2\ell_1^4 y^7\ddot{y} - 2\ell_1^2\ell_2^2 y^7\ddot{y} + 4\ell_1^2 x^2 y^7\ddot{y} + \ell_1^2 y^9\ddot{y} \,] \qquad (38)
\end{aligned}
$$

$$\ddot{\theta}_2 = \frac{-\left(\left(-\ell_1^2 - \ell_2^2 + x^2 + y^2\right)\left(x\dot{x} + y\dot{y}\right)^2\right)}{2\ell_1^3\ell_2^3\left(1 - \frac{\left(-\ell_1^2 - \ell_2^2 + x^2 + y^2\right)^2}{4\ell_1^2\ell_2^2}\right)^{\frac{3}{2}}} - \frac{\dot{x}^2 + \dot{y}^2 + x\ddot{x} + y\ddot{y}}{\ell_1\ell_2\sqrt{1 - \frac{\left(-\ell_1^2 - \ell_2^2 + x^2 + y^2\right)^2}{4\ell_1^2\ell_2^2}}} \tag{39}$$

C Initialization values

The neural networks were initialized with the following values.

Joint Angles (rad)
$\theta_1(0) = 0.0619589$
$\theta_2(0) = 2.32845$

Shoulder Flexor
$\tau_m(0) = 6.2\ N \cdot m$
$T_0(0) = 5.990736\ N \cdot m$
$ni(0) = emg(0) = F_0(0) = 0.249614$

Elbow Flexor
$\tau_m(0) = 1.0\ N \cdot m$
$T_0(0) = 0.963417\ N \cdot m$
$ni(0) = emg(0) = F_0(0) = 0.045877$

Double-Joint Flexor
$\tau_m(0) = 4.0\ N \cdot m$
$T_0(0) = 3.895500\ N \cdot m$
$ni(0) = emg(0) = F_0(0) = 0.129850$

Joint Torques $(N \cdot m)$
$\tau_1(0) = 7.839807$
$\tau_2(0) = 0.030324$

Shoulder Extensor
$\tau_m(0) = 0.160193\ N \cdot m$
$T_0(0) = 0.086712\ N \cdot m$
$ni(0) = emg(0) = F_0(0) = 0.003613$

Elbow Extensor
$\tau_m(0) = 2.169676\ N \cdot m$
$T_0(0) = 1.923516\ N \cdot m$
$ni(0) = emg(0) = F_0(0) = 0.091596$

Double-Joint Extensor
$\tau_m(0) = 1.0\ N \cdot m$
$T_0(0) = 0.821760\ N \cdot m$
$ni(0) = emg(0) = F_0(0) = 0.027392$