Actuator and Kinematic Redundancy in Biological Motor Control*

Reza Shadmehr

Center for Neural Engineering
University of Southern California
Los Angeles, CA 90089-2520 USA

Abstract

A task must be specified in terms of both the position and stiffness of the limb before muscle forces and activations are unambiguously assigned. To illustrate this, we begin with the problem of how to control an inverted pendulum with a pair of muscles. An active state model of the frog's gastrocnemius is used to derive three criteria for the stiffness characteristics of the system during posture and movement. The differential equation representing this model is solved to indicate the relationship between force and stimulation frequency. This result leads to an interesting prediction of muscle forces in a minimum stiffness equilibrium point control scheme: neural activity in the agonist muscle should decrease as the joint rotates the limb against gravity. For the case where the number of joints exceeds the task's degrees of freedom, an algorithm for mapping end-effector position and stiffness to the lengths of the muscles is considered. We show that a previously proposed algorithm for control of multi-joint limbs (Berkinblit et al. 1986a, Hinton 1984) is in fact a special case of this mapping. We contend that these kinematic maps must be augmented by a mechanism that takes into account the dynamics of the muscle-load-feedback system. We suggest an adaptive control scheme where the derived kinematic relationships are used to set the bias of the stretch reflex feedback loop, while a learning mechanism produces a virtual equilibrium trajectory that compensates for the second order dynamics of the load, as well as the dynamics of the muscles.

1 Introduction

The process of generating a movement may be viewed as a series of transformations from an overall movement objective (e.g., hitting a target with a baseball) to a plan specifying the desired behavior of the end-effector (e.g., the path that the hand should follow before the ball is released), and finally to a pattern of muscle activations. There is a wealth of data that describes the patterns of muscle activity and behavior of proprioceptive feedbacks during execution of

*This research was supported in part by NIH grant R01-NS24926-05 to Prof. Michael Arbib. The author is recipient of an IBM Graduate Fellowship in Computer Science. He can be reached by E-Mail at reza@rana.usc.edu.

particular trajectories, yet there has been relatively little progress made in describing algorithms that the CNS might be using to learn the dynamics of the muscle-load-feedback structures, and to actually perform the planned trajectory. There has been some progress in understanding the movement planning process in the CNS: By looking at the kinematics of reaching, Hogan (1984) has suggested that it is the position of the hand, rather than the joint angles, that is being planned. Yet it is not apparent how a task planned in terms of hand coordinates can be executed by muscles that are inherently joint-based: There are generally more degrees of freedom in a limb than there are degrees of freedom in the movement, so producing a trajectory of muscle lengths for a desired hand trajectory is an ill-posed problem. This is the issue of kinematic redundancy.

The situation is further complicated by the fact that at least two muscles act on a single joint, and in many cases, a muscle spans more than one joint. The problem is that the map from joint torques to muscle forces is not one-to-one, i.e., many levels of co-contraction can lead to the same effective joint torque. How does one decide on the amount of co-contraction? This is the issue of actuator redundancy.

Aside from the fact that kinematic redundancies in the system make the transformation from a planned hand trajectory to muscle activations an ill-posed problem, there is also the issue that the skeleton has non-linear and coupled dynamics: Forces produced by a muscle at a given joint affect the position of all the joints in the limb. Therefore, to execute a task, one must take into account the dynamics of the muscles, the skeleton, and the load. The controller is further restricted by a relatively slow afferent system, prohibiting implementation of a host of techniques that are commonly used in robotics in order to avoid compensation for dynamics of the limb.

In this paper we begin with the problem of how to control a single joint (an inverted pendulum) with a pair of antagonistic muscles. The muscles are modeled to faithfully reproduce the mechanical behavior of a frog's gastrocnemius, and are based on the active state theory (Gasser and Hill 1924) as parameterized by Inbar and Adam (1976). Our objective is to be able to activate these muscles so that the joint moves along a desired trajectory, with a particular stiffness profile. Based on stability requirements of the limb and the intrinsic mechanical limitations of muscles, we will derive three criteria for joint stiffness during posture and movement. A minimum stiffness strategy is used to calculate forces that should be produced by the muscles if an equilibrium point model (Feldman 1966) is used for control of the limb. This approach is contrasted with a model that takes into account the dynamics of the task. We will then explore algorithms for the problem of learning trajectory control in multi-joint biological limbs, with particular emphasis on representing and regularizing motor redundancy. We will show that the mechanical characteristics of the muscles and the feedback systems can be used by the CNS to not only solve the redundancy issues (as has been suggested by Mussa Ivaldi et al. 1988), but also to provide a means for building an adaptive internal model of the muscle-load-feedback system for precise execution of desired tasks.

2 Actuator Redundancy

One of the fundamental differences between the way one programs robots to perform tasks and the way biological systems move is in the information that is needed before a command can be sent to the actuators. In robots, once appropriate joint torques are determined, the robot will move more-or-less along the desired trajectory. In this section we will see that this information alone will not be sufficient for assignment of muscle activation rates.

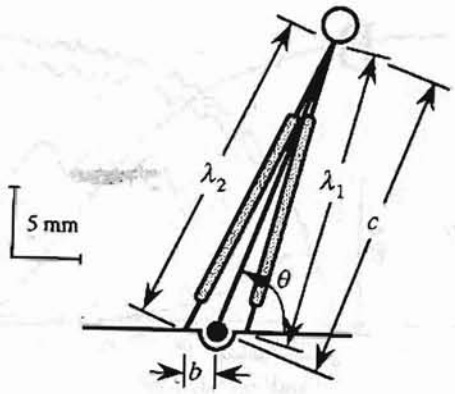

Figure 1: A ball-and-socket joint with two muscle-like actuators. Length of each muscle is denoted by λ_i.

Consider the ball-and-socket joint of Fig. 1, where the dynamics of the skeleton are described by the following:

$$\tau = mc^2\ddot{\theta} + \nu\dot{\theta} + mcg\cos(\theta) \tag{1}$$

where m is the point mass at the end of the limb, c is the length of the limb, ν is the joint's viscous parameter, and g is the gravitational constant. The lengths of the muscles, λ_1 and λ_2, are related to the joint angle θ by the functions:

$$\lambda_1 = \sqrt{b^2 + c^2 - 2bc\cos(\theta)} \tag{2}$$

$$\lambda_2 = \sqrt{b^2 + c^2 + 2bc\cos(\theta)} \tag{3}$$

Differentiating the muscle lengths with respect to the joint angle yields the following:

$$\frac{d\lambda_1}{d\theta} = \frac{bc\sin(\theta)}{\lambda_1} \tag{4}$$

$$\frac{d\lambda_2}{d\theta} = \frac{-bc\sin(\theta)}{\lambda_2} \tag{5}$$

The relationship between torque τ at the joint, and muscle forces ϕ_1 and ϕ_2 is:

$$\tau = -\frac{d\lambda_1}{d\theta}\phi_1 - \frac{d\lambda_2}{d\theta}\phi_2 = -\frac{bc\sin(\theta)}{\lambda_1}\phi_1 + \frac{bc\sin(\theta)}{\lambda_2}\phi_2 \tag{6}$$

Note that in Eq. (6), for a given joint torque trajectory $\tau(t)$, a trajectory in terms of muscle forces $\phi(t)$ cannot be calculated because there is an infinite set of antagonistic muscle forces that can lead to generation of the same joint torque. A simple example of this is evident in postural control: one can hold a limb at the same position with varying degrees of stiffness. For the limb in Fig. 1, joint stiffness, K_J, is defined as:

$$K_J = \frac{d\tau}{d\theta} = -\frac{d^2\lambda_1}{d\theta^2}\phi_1 - \left(\frac{d\lambda_1}{d\theta}\right)^2\frac{d\phi_1}{d\lambda_1} - \frac{d^2\lambda_2}{d\theta^2}\phi_2 - \left(\frac{d\lambda_2}{d\theta}\right)^2\frac{d\phi_2}{d\lambda_2} \tag{7}$$

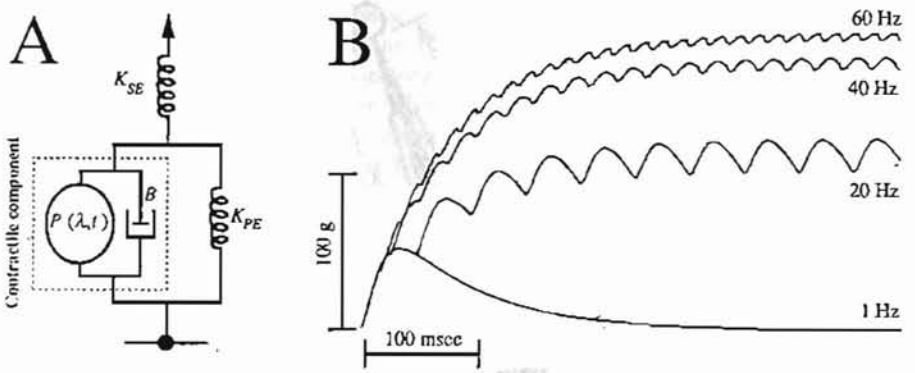

Figure 2: An active state muscle model. A) The active state $P(\lambda,t)$ is the tension developed by the force generator. B) Isometric tension produced by the muscle model at various stimulation rates.

By specifying joint stiffness, one is describing, for example, how the limb should react in case any part of it comes in contact with the environment. In the next section we will describe a muscle model and show that a unique set of muscle forces will result when joint torque and stiffness are specified.

From Fig. 1, we can see that K_J should always be negative: if a disturbance displaces the limb, the resulting change in torque should be in the opposite direction. Setting K_J depends on the stability requirements of the system: The equilibrium of the system is at $\theta_E = \cos^{-1}(\tau/mcg)$, and in order to guarantee stability, the stiffness must be:

$$K_J < -mcg\sin(\theta_E) \tag{8}$$

The fact that muscles cannot push against the skeleton will also limit the possible range of K_J during motion. In order to explain this, we need to fully specify (6) and (7) by formulating a muscle model.

2.1 A Muscle Model

Here we describe a two component active state muscle model: an elastic component in series with a contractile component (Fig. 2A). This is the model proposed by Gasser and Hill (1922) to explain the tension dynamics of various frog muscles. The following differential equation describes force development in this model:

$$\dot{\phi} = \frac{K_{SE}}{B}(K_{PE}\Delta\lambda + B\dot{\lambda} - (1+\frac{K_{PE}}{K_{SE}})\phi + P(\lambda,t)) \tag{9}$$

where K_{SE} is the series elastic component in the tendons, K_{PE} is the parallel elastic component which with K_{SE} accounts for the passive tension properties of the muscle, B represents the viscous resistance opposing force development during shortening, λ_0 is the rest-length of the muscle beyond which passive force is developed (Aubert et al. 1951), and $\Delta\lambda = \lambda - \lambda_0$ when

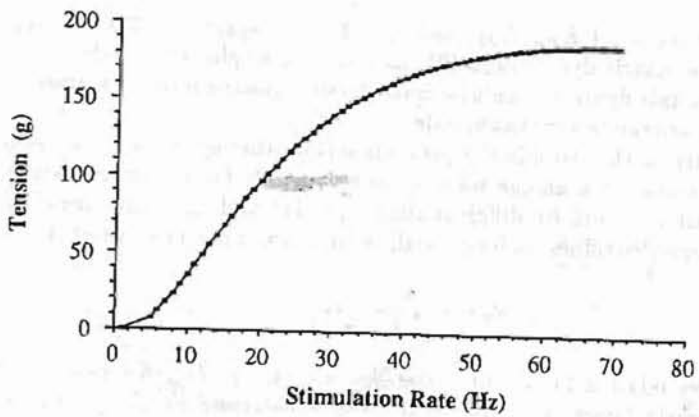

Figure 3: Tension output of the muscle model during isometric conditions at resting length after $1/\Delta t$ impulses, and just before the arrival of the next impulse.

$\lambda_0 < \lambda$, else $\Delta\lambda = 0$. $P(\lambda,t)$ is the active force produced by the contractile mechanism in the muscle. It can be represented by the product of a length dependent function $S(\lambda)$, and a function $Q(f)$ which depends on the history of muscle stimulation $f(t')$ for $t' \leq t$. Using a linear systems approach to $Q(f)$, the input to the contractile mechanism, $f(t)$, is a series of impulses, while the output is force:

$$P(\lambda,t) = S(\lambda)\int_{-\infty}^{t} f(t')h(t-t')dt'$$

Using a parameter estimation technique, Inbar and Adam (1976) have calculated $h(t)$, i.e., the impulse response of Q. We approximated their results in Fig. 5A by a sum of two exponentials, and derived the active tension (in grams) when the input to the system is a series of impulses of frequency $1/\Delta t$:

$$P = S(\lambda)\sum_{n=0}^{\infty}(\exp(-70(t-n\Delta t)) - \exp(-210(t-n\Delta t))(u(t-n\Delta t) - u(t-(n+1)\Delta t)) \tag{10}$$

where $u(t)$ is a unit step function at $t = 0$, and $S(\lambda) = 1200(2\lambda/\lambda_0 - 1)$ when $\lambda \leq \lambda_0$, else $S(\lambda) = 0$. Fig. 2B is the simulation results of (9) and (10) during isometric conditions at $\lambda = \lambda_0$.

By formally defining a muscle model we have specified the dynamics of the system in Fig. 1. However, control of this system requires understanding how much activation the muscles should receive in order to produce a particular force profile (the inverse dynamic model of the muscles). Our approach is as follows: We assumed that λ_0 for the muscles in Fig. 1 is at the point of maximal extension for each muscle in the physiological workspace (this means that the force produced by each muscle is at a maximum when the muscle has its greatest length in the workspace). We then solved the differential equation in (9) for an isometric muscle preparation in order to indicate the amount of tension in the entire muscle after n impulses, and just before the $(n+1)$st impulse:

$$\phi = S(\lambda)a_1\left(\frac{\exp(-210\Delta t)}{210-a_1a_2} - \frac{\exp(-70\Delta t)}{70-a_1a_2} + a_3\exp(-a_1a_2\Delta t)\right)\frac{\exp(-a_1a_2n\Delta t)-1}{\exp(-a_1a_2\Delta t)-1} \tag{11}$$

where $a_1 = K_{SE}/B$, $a_2 = 1 + K_{PE}/K_{SE}$, and $a_3 = 1/(70 - a_1 a_2) - 1/(210 - a_1 a_2)$. Eq. (11) is an approximation of muscle dynamics in (9). In Fig. 3 we've plotted muscle tension at $\lambda = \lambda_0$, and $n\Delta t = 1$. Using this figure we can now map a desired muscle force at a given muscle length into a frequency of activation for that muscle.

Let us now return to the claim in the previous section that specification of joint torque and joint stiffness will allow for a unique solution to the muscle forces. In analogy with eq. (7), $d\phi/d\lambda$ is muscle stiffness, and by differentiating eq. (11) and using the linear expression for $S(\lambda)$, we see that muscle stiffness is linear with respect to muscle force when $\lambda \leq \lambda_0$:

$$d\phi/d\lambda = \frac{\phi}{\lambda - \lambda_0/2}$$

By using the above relation for $d\phi_1/d\lambda_1$ and $d\phi_2/d\lambda_2$ in eq. (7), K_J now becomes a linear function of the muscle forces, and since it is linearly independent of eq. (6), we can find a unqiue set of muscle forces for a given set of joint torque and stiffness.

2.2 Experiments

Let us begin with the question of how to maintain posture with the muscle-skeleton system of Fig. 1, i.e., how to assign muscle forces so the limb stays at a desired position θ_d. The procedure is to find the set of muscle forces which position the equilibrium of the system at θ_d, and to ensure that this position is stable. By using the relationship between joint equilibrium and torque: $\tau = mcg\cos(\theta_E)$, and by setting $\theta_E = \theta_d$ and using eq. (8), we can solve (6) and (7) for the muscle forces. For the minimum joint stiffness that satisfies (8), we've plotted the muscle forces as a function of joint equilibrium position in Fig. 4A. Increasing joint stiffness (i.e., making it more negative) simply scales this approximately parabolic relationship between muscle forces and equilibrium joint angle, i.e., the ratio of muscle forces is independent of joint stiffness at equilibrium.

An elegant model of motor control (Feldman 1966, Flash 1987) suggests that movement may be thought of as a shift in equilibrium position of the system. Assume that we wish the limb in Fig. 1 to rotate from θ_1 to θ_2 in Δt seconds and follow a desired trajectory $\theta_d(t)$ which minimizes the time derivative of acceleration, i.e., a minimum jerk trajectory (Hogan 1984):

$$\theta_d(t) = \theta_1 + (\theta_1 - \theta_2)\left(15(t/\Delta t)^4 - 6(t/\Delta t)^5 - 10(t/\Delta t)^3\right) \qquad (12)$$

As an example, we considered a movement from 45 to 135 degrees in 0.5 seconds. By shifting the equilibrium position of the system along $\theta_d(t)$ and using, for example, a minimum stiffness protocol, we can solve for the muscle forces $\phi_1(t)$, and $\phi_2(t)$. The resulting force trajectory for each muscle is plotted in the "Equilibrium model" of Fig. 4B.

A second approach to programming muscle activation is to consider the dynamics of the moving limb when we assign muscle forces. This means that the torque trajectory should include the influence of joint velocity and acceleration along the desired trajectory θ_d:

$$\tau(t) = mc^2\ddot{\theta}_d + \nu\dot{\theta}_d + mcg\cos(\theta_d) \qquad (13)$$

Since this means that the torques experienced by the system will be much higher than when the equilibrium position of the system is changed, it may not be possible for the muscles to produce a high joint torque while maintaining the stiffness requirements of (8). Stiffness of a joint in motion is constrained by the fact that muscles cannot push against a load. We implemented

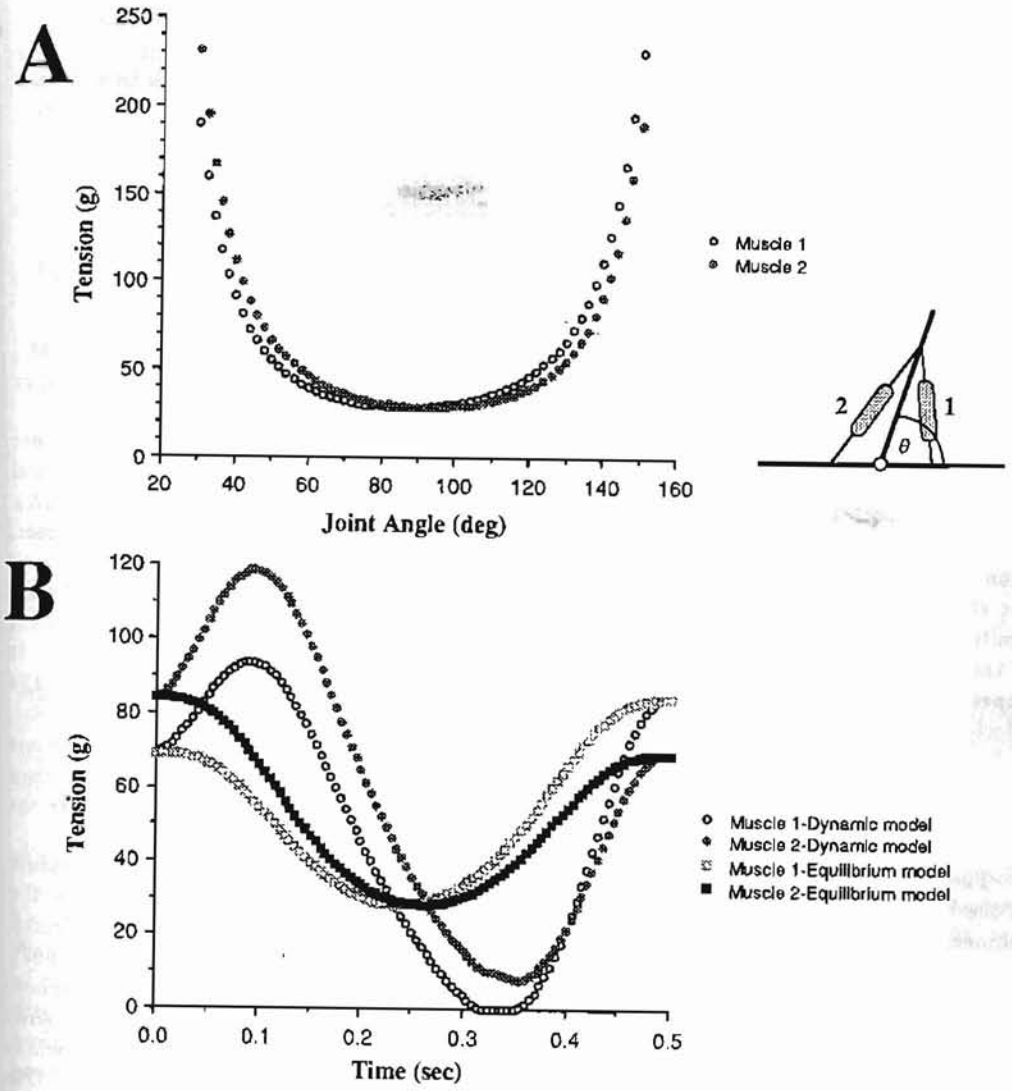

Figure 4: A) Muscle tension required to keep the joint at stable equilibrium, and at minimum joint stiffness. B) Muscle tension trajectory for a minimum stiffness movement from 45 to 135 degrees. The results for a Dynamic Model and an Equilibrium Model strategy are illustrated.

this constraint on the system by limiting the range of K_J so that a muscle is never asked to produce a negative active force (this criterion is a weaker condition than (8) when the limb is at rest, however during motion, it becomes the limiting factor). The procedure is to solve for ϕ_1 and ϕ_2 in terms of K_J and τ, and then find the conditions for which neither muscle force can become negative:

$$K_J < -\frac{\tau(\lambda_0 b^2 c^2 \sin^2(\theta) + \lambda_0 \lambda_2^2 bc\cos(\theta) - 2\lambda_2^3 bc\cos(\theta))}{\lambda_2^2 bc\sin(\theta)(2\lambda_2 - \lambda_0)} \quad (14)$$

$$K_J < -\frac{\tau(\lambda_0 \lambda_1^2 bc\cos(\theta) - \lambda_0 b^2 c^2 \sin^2(\theta) - 2\lambda_1^3 bc\cos(\theta))}{\lambda_1^2 bc\sin(\theta)(2\lambda_1 - \lambda_0)} \quad (15)$$

We repeated the original movement with the minimum stiffness that met the criteria of (8), (14), and (15), using the torque trajectory of (13). The force trajectory for each muscle is plotted in Fig. 4B (labeled Dynamic model).

Fig. 4B illustrates an essential property of the equilibrium model: It predicts that in order to move the limb in Fig. 1 from a small joint angle to a larger one, the force (and neural activity) for both the agonist and antagonist should *decrease* (this is also observed in Fig. 4A). The reason for this is that it takes less torque to hold the mass at, for example, 80 degrees, than 45 degrees. Never the less, the equilibrium model accomplishes the joint rotation due to the stiffness requirement of (8): when the current position is not at the desired equilibrium position, the joint stiffness is large enough to produce a correcting torque that exceeds the effect of the gravitational pull on the load and moves the limb toward the equilibrium position. It appears to us that this property of the equilibrium hypothesis can be directly tested if the effect of the spinal reflexes can be eliminated or explicitly accounted for in a separate model. Referring to Fig. 4B, in contrast to the results of the equilibrium strategy, using the dynamic strategy suggests muscle forces that require an increasing burst to begin moving the limb, then both muscles nearly become quiet, and finally the antagonist is strongly activated to brake the movement.

The point of this example was to show that even for a single joint movement, the nature of the biological actuators is such that a task must be specified in terms of both position and stiffness before muscle activations can be programmed. The notion of an equilibrium point, as introduced by Feldman (1966), and demonstrated as a control algorithm by Flash (1987), suggests that we can begin with the spring-like characteristics of the antagonistic muscles, ignore the dynamics of the skeleton, shift the minimum of a potential energy surface along the desired path, and the limb will more-or-less follow. We have specified the stiffness conditions for which this hypothesis holds true, as well as the expected muscle forces for a typical movement. Our results provide an easy test of the equilibrium hypothesis as a control paradigm for movement generation: the force and neural activity in the agonist should decrease as a joint rotates a load against gravity.

3 Kinematic Redundancy

The issue of kinematic redundancy arises when the degrees of freedom in a limb exceed the degrees of freedom in the movement, e.g., a planar three-joint arm. Formally, the definition is as follows: Consider a multi-joint limb with an end-effector attached to the distal link. If the position of the end-effector is denoted by vector $\mathbf{x} = [x_1, x_2, \ldots, x_m]^T$, and $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]^T$ specifies the joint angles, then there exists some continuous non-linear function $h(\boldsymbol{\theta})$, such that

$\mathbf{x} = h(\boldsymbol{\theta})$, i.e., the forward kinematic mapping. If $m < n$, then the arm is kinematically redundant, meaning that there is a unique $\mathbf{x}$ associated with each $\boldsymbol{\theta}$, but there may be many $\boldsymbol{\theta}$ associated with each $\mathbf{x}$. When there are muscles attached to the joints, the issue becomes how to assign muscle lengths for a given end-effector position: Assume that $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_q]^T$ is a vector of muscle lengths where λ_i is the length of muscle i, and $n < q$. Then there exists some continuous non-linear function $g(\boldsymbol{\theta})$, where $\boldsymbol{\lambda} = g(\boldsymbol{\theta})$. Given a planned end-effector trajectory $\mathbf{x}(t)$, describing an appropriate trajectory in terms of muscle lengths $\boldsymbol{\lambda}(t)$ requires finding $g(\ h^{-1}(\mathbf{x})\ )$. A trajectory in terms of the muscle lengths may be required for setting the bias of the stretch reflex loop (activation of the γ-motoneurons), and for assigning muscle activations as in the discussion in the previous section.

To find $h^{-1}(\mathbf{x})$, the most direct approach is certainly to find an analytical expression, an approach that has proved to be very difficult in robotics, due to the complexity of $h(\boldsymbol{\theta})$ (Sciavicco and Siciliano 1988). A novel approach has recently been suggested by Mussa Ivaldi et al. (1988): taking advantage of the elastic properties of the neuromuscular system, they have proposed an algorithm that allows one to map small changes in the position of the end-effector into changes in joint angles. We have sketched the derivation of the algorithm below:

$$d\boldsymbol{\theta} = \mathbf{C}_J\, d\boldsymbol{\tau} \quad (16)$$

$$d\boldsymbol{\tau} = \mathbf{J}_S^T\, d\mathbf{f} \quad (17)$$

$$d\mathbf{f} = \mathbf{K}_S\, d\mathbf{x} \quad (18)$$

$$\text{by substitution:} \quad d\boldsymbol{\theta} = \mathbf{C}_J\, \mathbf{J}_S^T\, \mathbf{K}_S\, d\mathbf{x} \quad (19)$$

where $\mathbf{C}_J$ is the limb's joint compliance, $\mathbf{J}_S$ is the Jacobian at the end-effector: $\mathbf{J}_S = \partial\mathbf{x}/\partial\boldsymbol{\theta}$, $\mathbf{f}$ is a force vector at the end-effector, and $\mathbf{K}_S$ is the limb's stiffness at the end-effector. Given a limb's joint compliance $\mathbf{C}_J$, by using the principle of virtual work one can calculate the end-point stiffness $\mathbf{K}_S$:

$$\mathbf{K}_S = (\mathbf{J}_S\, \mathbf{C}_J\, \mathbf{J}_S^T)^{-1} \quad (20)$$

which basically shows how to go from impedance at joint coordinates to impedance at end-point coordinates. The inverse in (20) always exists because from the change in potential energy of the system, it can be shown that $\mathbf{C}_J$ is a positive definite matrix (Mussa Ivaldi et al. 1988), and $\mathbf{J}_S$ is full rank by construction. The idea of the algorithm in (19) is that a particular pattern of changes in joint angles will occur if an external agent forced the end-effector to make small displacements $d\mathbf{x}$. The resulting changes in the limb's configuration can be fully specified if the limb's impedance is known. So to perform a movement with the algorithm in (19), one would need *a priori* knowledge regarding the impedance of the limb during the movement.

Assume that a task is specified in terms of an end-effector trajectory $\mathbf{x}(t)$ for a kinematically redundant biological limb. What (19) implies is that the task must also specify joint stiffness during motion (or that the CNS can assume a minimum stiffness value that is appropriate for the task). Since the task needs to be performed by muscles, we should rewrite the algorithm in (19) in terms of muscle lengths:

$$d\boldsymbol{\lambda} = \mathbf{J}_M\, d\boldsymbol{\theta} \quad (21)$$

$$d\boldsymbol{\lambda} = \mathbf{J}_M\, \mathbf{C}_J\, \mathbf{J}_S^T\, \mathbf{K}_S\, d\mathbf{x} \quad (22)$$

where $\mathbf{J}_M = \partial\boldsymbol{\lambda}/\partial\boldsymbol{\theta}$ (for example, see (4) and (5)). The algorithm in (22) suggests a method by which the CNS can set the bias of the stretch reflex circuitry for each muscle, given an arbitrary end-effector trajectory and joint impedance. The contribution of Mussa Ivaldi et al.'s

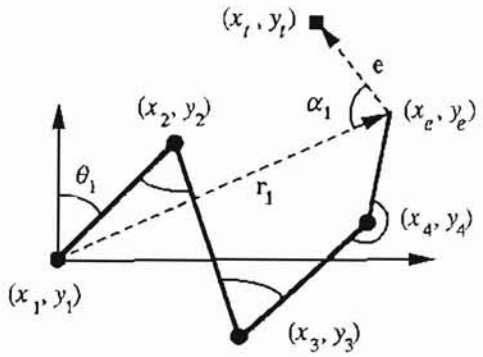

Figure 5: Model robot nomenclature and vector definition for the Error Vector Algorithm.

(1988) work has been to exploit the fact that motor impedance provides a unique solution to the configuration of a mechanism given an externally imposed motion to its end-effector. In the next section we will review a previously proposed algorithm, referred to as the *Error Vector Algorithm*, for solving kinematic redundancies (independently developed by Berkinblit et al. 1986a, and Hinton 1984), and we will show that this algorithm is a special case of the relation derived in (19).

4 Error Vector Algorithm

A current hypothesis in motor control is that motor behaviors are organized of compartments or segments that can be combined in different ways to form new movement patterns (e.g., Berkinblit et al. 1986b, Fentress 1987, Viviani and Terzuolo 1982). The essential component of this hypothesis is that "every limb joint is subserved by a set of individual control systems which interact in the process of solving a common motor task" (Berkinblit et al. 1986b). These organizations are said to exist in locomotion of cats—where it has been argued that limb joints are controlled by a set of generators which interact with each other to produce an overall locomotor pattern (Grillner 1975)—as well as in reaching movements (Hinton 1984), and the wiping reflex in the frog (Berkinblit et al. 1986a).

In this hypothesis, control of each joint is accomplished in parallel while information about the position of each joint relative to the end-effector is shared between all controllers in order to perform a common motor task. Each controller "produces an individual movement in the corresponding joint, based on the information on the position of the target and on the result of collective work of all the limb's joints, in particular, the knowledge of the position of the limb's tip relative to the target" (Berkinblit et al. 1986a). We refer to this as the *Error Vector Algorithm*. In this section we will generalize the algorithm for a robot with n joints, and show that the algorithm uses the transpose of the robot's Jacobian at the end-effector to simulate attachment of an imaginary spring between the target and the end-effector. Our discussion will indicate that is a special case of the relationship in (19).

Consider a planar, multi-joint limb such as the one in Fig. 5. Assume that each joint has one degree of freedom (along the axis perpendicular to the plane of motion) and joints are connected by a rigid link of some length l_i. Point (x_i, y_i) is the location of joint i, (x_e, y_e) is the position of

the end-effector, (x_t, y_t) is the position of the target, and a limb of length l_i connects to joints $i-1$ and i. Assuming that $(x_1, y_1) = (0, 0)$, then the forward kinematics are described by the following:

$$x_i = x_{i-1} + l_{i-1}\sin(\theta_{i-1} - \theta_{i-2} + \theta_{i-3} - \ldots + (-1)^i\theta_1) \quad (23)$$

$$y_i = y_{i-1} + (-1)^i l_{i-1}\cos(\theta_{i-1} - \theta_{i-2} + \theta_{i-3} - \ldots + (-1)^i\theta_1) \quad (24)$$

The iterative algorithm proposed by Hinton (1984) and Berkinblit et al. (1986a) describes how a particular target can be reached when the number of joints exceed the number of coordinates that define the target. The idea is to plan changes in joint angles as calculated by an error vector that points from the tip of the limb to the target position, as in Fig. 5: For each joint i, a vector r_i points to the current position of the end-effector. There is also a vector e which points from the end-effector to the target position. If c_i is a constant, and α_i is the angle between r_i and e, then the change in the angle of joint i is a vector that points along an axis perpendicular to the plane of movement, and is defined by Berkinblit et al. (1986a) as:

$$\Delta\theta_i = c_i\,|\mathrm{r}_i||\mathrm{e}|\,\sin(\alpha_i) \quad (25)$$

where $|\mathrm{x}|$ is the magnitude of the vector x.

We will examine the rationale for this algorithm first intuitively, and then rigorously by finding the Jacobian of the robot at the end-effector position. In (25), a rotation from r_i to e through an angle α_i will change θ_i about an axis perpendicular to the plane of motion. It is desirable to make the change in θ_i proportional to the magnitude of the error vector e. Also, $\Delta\theta_i$ is reasoned to be proportional to $|\mathrm{r}_i|$ since the longer this vector is, the more effectively the position of the end-effector can approach the desired target.

To show the reasoning behind (25) in a rigorous fashion, we need to initially express $\Delta\theta_i$ in terms of the end-effector coordinates. By use of geometry (Law of Cosines), α_i can be eliminated from (25). In general, for the notation that was introduced in Fig. 5, and using equations (23) and (24), we can express (25) as follows:

$$\text{if } i \text{ is even:} \quad \Delta\theta_i = c_i((x_e - x_i)(y_t - y_e) - (y_e - y_i)(x_t - x_e)) \quad (26)$$

$$\text{if } i \text{ is odd:} \quad \Delta\theta_i = c_i((y_e - y_i)(x_t - x_e) - (x_e - x_i)(y_t - y_e)) \quad (27)$$

For the 4-joint limb of Fig. 5, let us derive the Jacobian matrix J_S, where $\mathrm{J}_S = \partial\mathrm{x}/\partial\theta$. From (23) and (24), substituting (x_e, y_e) for (x_5, y_5) and differentiating with respect to θ we have:

$$\begin{aligned} dx_e = {} & l_1\cos(\theta_1)d\theta_1 + l_2\cos(\psi_1)d\psi_1 + l_3\cos(\psi_2 + \theta_1)(d\psi_2 + d\theta_1) \\ & + l_4\cos(\psi_3 + \psi_1)(d\psi_3 + d\psi_1) \end{aligned} \quad (28)$$

$$\begin{aligned} dy_e = {} & -l_1\sin(\theta_1)d\theta_1 + l_2\sin(\psi_1)d\psi_1 + l_3\sin(\psi_2 + \theta_1)(d\psi_2 + d\theta_1) \\ & + l_4\sin(\psi_3 + \psi_1)(d\psi_3 + d\psi_1) \end{aligned} \quad (29)$$

where $\psi_i = \theta_{i+1} - \theta_i$. For example, the element in the first row of the first column in the Jacobian matrix J_S will be:

$$J_{S\,(1,1)} = l_1\cos(\theta_1) - l_2\cos(\theta_2 - \theta_1) + l_3\cos(\theta_3 - \theta_2 + \theta_1) - l_4\cos(\theta_4 - \theta_3 + \theta_2 - \theta_1) \quad (30)$$

By comparing (24) with (30), we see that the (30) is in fact y_e. Similarly, we can show that:

$$\mathrm{J}_S = \begin{pmatrix} y_e & y_2 - y_e & y_3 - y_e & y_4 - y_e \\ -x_e & x_e - x_2 & x_e - x_3 & x_e - x_4 \end{pmatrix} \quad (31)$$

By substituting the above Jacobian in (26) and (27), we can rewrite (26) and (27) in the following format:

$$\Delta\theta = \mathrm{C}\,\mathrm{J}_S^T\,\mathrm{e} \tag{32}$$

where C is a diagonal matrix made up of c_i (one constant for each joint), and e is the error vector, $\mathrm{e} = [(y_t - y_e), (x_t - x_e)]^T$. The relation in (32) is another way to write the Error Vector Algorithm that has been suggested by Berkinblit et al. (1986a) and Hinton (1984).

Now compare the relation in (32) with (19). In (19), C_J is the joint compliance matrix, and K_S is the end-point stiffness. The relation in (19) assigns changes in joint angles as a function of small displacements in the end-effector and limb impedance. In (32), the error vector e represents the affect of the displacement after it interacts with end-point stiffness (to become a force acting on the end-effector).

It turns out that when we simulated movements with a kinematic model, the algorithm in (32) worked only if the matrix C had elements that were all very small (on the order of 1% of $|\mathrm{e}|$ before onset of movement), otherwise, the limb would either oscillate about the target position, or become unstable. Therefore we conclude that the algorithm proposed by Hinton (1984) and Berkinblit et al. (1986a) in equation (25), is a special case of the relation in (19)—the special case being that (25) assumes an identity matrix for end-point stiffness K_S.

The idea that every joint has a controller which interacts with other joint-controllers in the process of solving a common motor task can be described by the relation in (19) when one realizes that (19) is a precise formulation of (25). Each controller "works" by imitating the effect of a displacement at the tip of the limb on the joint that it controls. To do this, the controller needs to know three kinds of information: (1) where the tip of the limb is with respect to the target, (2) where the joint is with respect to the tip of the limb, and (3) what the impedance of the limb is. Since the algorithm is iterative and highly dependent on initial conditions, it is likely that quite different joint angles may be observed when the end-effector cycles through a given trajectory. Another reason for this phenomenon is that the mapping in (19) is not integrable, meaning that after tracing a circle, the joint angles do not return to their original position at the start of the trace. A final reason is that we have only mapped the kinematics of movement here. The actual trajectory of limb is affected by forces linked to its motion.

In the next section we will consider some of the dynamic forces inherent in motion. Our goal will be to show how to rapidly learn to compensate for muscle and limb dynamics in order to produce a precise end-effector trajectory.

5 Learning System Dynamics

Assume that we have a task that requires the end-effector to follow a trajectory $\mathbf{x}(t)$, with a stiffness profile $\mathrm{K}_J(t)$. Based on our discussion in the previous two sections, we have an algorithm in (22) which specifies what the muscle lengths should be during this task. This algorithm's purpose is to specify the kinematics of the task in the same coordinate system as the actuators. The kinematics of the task are, however, only a static representation of the dynamics of the system. For example, consider that the skeleton has non-linear and coupled dynamics, meaning that the relation in (16) is only valid for small disturbances from equilibrium since it has not taken into account the effect of centripetal, Coriolis, or gravitational forces. In order for the end-effector to produce the desired trajectory, a trajectory of muscle activations will have to be arrived at which compensates for these forces.

We propose that the dynamics of the muscle-load-feedback system can be learned with a

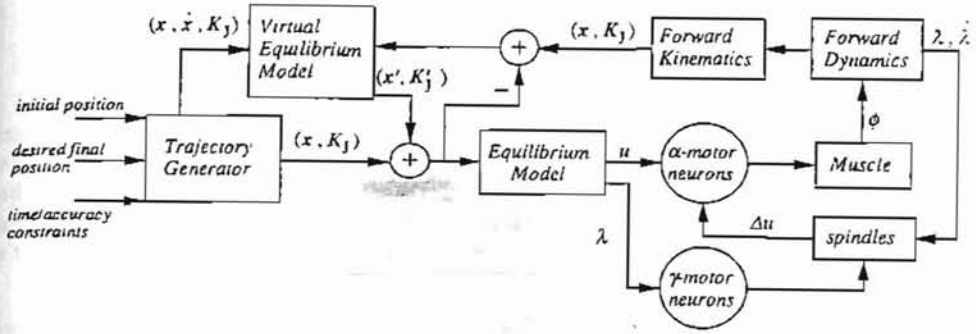

Figure 6: A schematic for learning dynamics of a biological limb. The trajectory generator specifies end-effector position x, velocity $\dot{\mathbf{x}}$, and stiffness K_J for the desired movement. The Equilibrium Model produces a trajectory of muscle lengths and activations without regard to dynamics of motion. The error between the observed and desired paths of the limb leads to formation of a Virtual Equilibrium Model that augments the equilibrium path.

control structure that adapts its model of the system with information from efferent copy and proprioceptive receptors, as shown in Figure 6. In this figure there are actually two motor maps: one is the static equilibrium map that is hypothesized to reside in the spinal cord (see Giszter et al. in this book) which maps a desired end-effector position and limb stiffness onto muscle lengths and activation: $(\mathbf{x}, \mathrm{K}_J) \rightarrow (\lambda, \mathbf{u})$, and the other is an adaptive motor map which attempts to compensate for the limb's dynamics during the movement. This adaptive model interacts with the equilibrium model to produce a *virtual equilibrium trajectory*. Formally, the mapping that is learned by the adaptive model is: $(\mathbf{x}, \dot{\mathbf{x}}, \mathrm{K}_J) \rightarrow (\mathbf{x}', \mathrm{K}_J')$, where the resulting variables represent error terms that add to the original trajectory in order to produce a new trajectory that compensates for limb dynamics.

For example, consider a two joint limb that we wish to move along a trajectory such as the one specified in Fig. 7A. The equilibrium trajectory specifies the end-effector position and stiffness profile during movement: the mapping to muscle activation and muscle lengths is performed in the spinal cord, e.g., in a region analogous to L4 and L5 in the cat—the area thought to be responsible for pattern generation in the scratch reflex (Berkinblit et al. 1978). For a particular stiffness profile (Shadmehr 1990), the resulting movement of the limb (referred to as the actual movement) is plotted by the dotted lines in Fig. 7A. Note that in this case, the limb lags the equilibrium trajectory and oscillates about the desired end-point before it is damped out. When a learning mechanism is in place, a virtual equilibrium trajectory can be fed to the spinal mechanism so that the actual trajectory is identical to the one that was desired. In Fig. 7B, the virtual equilibrium trajectory is the dotted set of lines. Note that in order to begin the movement, the virtual trajectory needs to accelerate the arm beyond the amount that is specified by the equilibrium trajectory. Correspondingly, to stop the motion, the virtual trajectory needs to reverse the movement to activate antagonist muscles and brake the motion. In Shadmehr (1990), we used a gross computational model of the Cerebellum, the Cerebellar Model Articulation Controller (Albus 1975), to rapidly learn this virtual equilibrium trajectory.

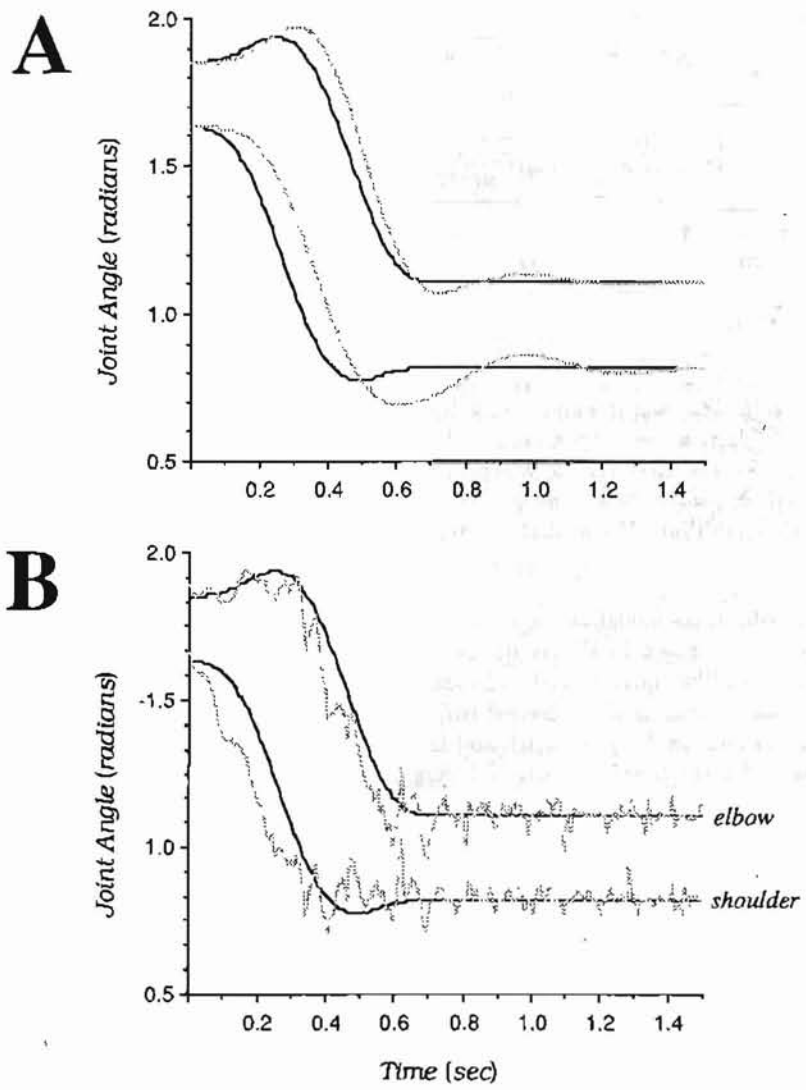

Figure 7: A learned virtual equilibrium joint trajectory that compensates for limb dynamics of a two joint arm. A) The equilibrium trajectory (solid lines) is the desired joint trajectory, while the dotted lines show the actual trajectory of the limb. B) The learned virtual equilibrium trajectory (dotted, more "noisy" lines) compensates for the limb dynamics. For this case, the actual trajectory is identical to the desired one.

6 Discussion

A fundamental question in motor control is how the CNS interacts with limbs that possess kinematic and actuator redundancies, for there are many more system parameters (e.g., α- and γ-motoneuron activation rates) than there are independent variables (e.g., position of the end-effector and limb stiffness). Our goal has been to discuss issues of actuator and kinematic redundancy within the framework of robotics in order to understand how to represent a movement so that it can uniquely specify how to perform it.

The issue of actuator redundancy arises because assignment of muscle forces to produce a given joint torque is not possible since many degrees of co-activation can lead to generation of identical joint torques. To deal with this issue, we suggested that description of a task must include not only the trajectory of the end-effector, but also, at least the trajectory of the stiffness at the end-effector. Within the framework of the *equilibrium trajectory* hypothesis (Flash 1987), this suggests that, for example, an equilibrium trajectory of the hand for reaching movements is an incomplete description of the task since the same trajectory may be performed with various degrees of joint stiffness. With reference to the λ-model (Feldman 1966), where λ is a threshold length for activation of a muscle, joint stiffness must be decided upon before the distance of the thresholds from the equilibrium joint angle can be assigned. Interestingly, for a task that requires rotating a joint from a horizontal to a vertical position, we showed that the equilibrium hypothesis predicts a *reduction* in the activity of both the agonist and the antagonist muscles, given that the limb's stiffness remains at a level just sufficient to ensure stability. This result is in sharp contrast to the muscle activity that is expected if a dynamic model of movement is used: for the same movement, this model predicts a sharp initial increase in the activity of both muscles, then a decrease and finally a braking pattern of activity by the antagonist muscle.

Mussa Ivaldi et al.'s (1988) work suggests that the kinematic redundancies of the limb can be overcome when the elastic properties of the system are considered. When an external agent displaces the end-effector, the resulting changes in joint angles can be determined if the stiffness characteristics of the limb are known. The same line of analysis can be used to relate displacements in the position of the end-effector to changes in muscle lengths (as in equation (22)). By "imitating" the effect of a foreign agent on the end-effector, the CNS can produce a trajectory in terms of joint angles and muscle lengths. The notion of an independent controller for each joint (the error vector algorithm) was shown to fit well with in this framework: the only information that is necessary for each controller is the distance of the tip of the limb to the target, and the distance from the center of the controlled joint to the end-effector. However, this mapping only describes the kinematics of the task: if muscle activations are assigned without regard to, for example, the inertial forces that act on the skeleton, then the observed trajectory of the end-effector will deviate significantly from the desired path. Learning dynamics of the muscle-load-feedback system is essential especially for execution of ballistic or precise movements.

In the adaptive control scheme that we proposed, the kinematic requirements of the task (as specified by (22)) are used by an equilibrium model, located in the spinal cord, to produce an equilibrium trajectory. This trajectory is augmented by a supra-spinal center to produce a virtual equilibrium trajectory that compensates for the dynamics of the limb. Our current work is exploring the usefulness of this approach in predicting muscle activation patterns and end-effector trajectories for control of redundant limbs.

7 References

Albus, J. S. (1975) A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Trans. ASME J. Dynamic Syst. Meas. Contr.*, 97:220–227.

Aubert, X., Roquet, M. L., and Van der Elst, J. (1951) The tension-length diagram of the frog's sartorius muscle. *Arch. Intern. de Physiol.*, 59:239–241.

Berkinblit, M. B., T. G. Deliagina, A. G. Feldman, I. M. Gelfand, and G. N. Orlovsky (1978) Generation of scratching. I. Activity of spinal interneurons during scratching. *J. Neurophys.*, 41:1040–1057.

Berkinblit, M. B., I. M. Gelfand, and A. G. Feldman (1986a) Model of the control of the movements of a multijoint limb. *Biophysics* 31(1):142–153.

Berkinblit, M. B., A. G. Feldman, and O. I. Fukson (1986b) Adaptability of innate motor patterns and motor control mechanisms. *Behav. Brain Sci.*, 9:585–599.

Feldman, A. G. (1966) Functional tuning of the nervous system with control of movement or maintenance of a steady posture—II. Controllable parameters of the muscles. *Biophysics*, 11:565–578.

Fentress, J. C. (1987) Compartments and cohesions in adaptive behavior. *J. Comp. Psychol.*, 101:254–258.

Flash, T. (1987) The control of hand equilibrium trajectories in multi-joint arm movements. *Biol. Cybern.*, 57:257–274.

Gasser, H. S., and Hill, A. V. (1924) The dynamics of muscular contraction. *Proc. Roy. Soc. B.*, 96:398–437.

Grillner, S. (1975) Locomotion in vertebrates: central mechanisms and reflex interaction. *PhysiolRev.*, 55:247–304.

Hinton, G. E. (1984) Parallel computations for controlling an arm. *J. Motor Behavior*, 16:171–194.

Hogan, N. (1984) An organizing principle for a class of voluntary movements. *J. Neuroscience*, 4:2745–2754.

Inbar, G. F., and Adam, D. (1976) Estimation of muscle active state. *Biol. Cybern.*, 23:61–72.

Mussa-Ivaldi, F. A., N. Hogan, and E. Bizzi (1985) Neural, mechanical, and geometric factors subserving arm posture in humans. *J. Neuroscience*, 5(10):2732-2743.

Mussa-Ivaldi, F. A., J. McIntyre, and E. Bizzi (1988) Theoretical and experimental perspectives on arm trajectory formation: A distributed model for motor redundancy. In *Biological and Artificial Intelligence Systems*, Eds: E. Clementi and S. Chin, Escom Press.

Sciavicco, L., and B. Siciliano (1988) A solution algorithm to the inverse kinematic problem of redundant manipulators. *IEEE J. Robotics and Automation*, 4(4):403–410.

Shadmehr, R. (1990) Learning virtual equilibrium trajectories for control of a robot arm. *Neural Computation*, 2(4):in press.

Viviani, P., and Terzuolo, C. (1982) Trajectory determines movement dynamics. *Neuroscience*, 7:431–437.

MOTOR PATTERN GENERATORS IN ANURAN PREY CAPTURE

Ananda Weerasuriya

Department of Basic Medical Sciences

Mercer University School of Medicine

Macon, GA 31207, U.S.A.

SUMMARY. Anuran prey capture, released by specific stimuli, consists of a sequence of motor synergies. This series of steps includes an approach or orientation toward the prey stimulus, a fixation of the prey in the frontal visual field and the consummatory event of snapping at the prey and swallowing it. The key stimulus that elicits prey capture is either visual, tactile or olfactory, and the outputs of their respective sensory analyzers share common access to motor pattern generators responsible for the elaboration of the appropriate motor outputs. An approach path can be altered midstream in response to movement of the prey, but an orienting turn once initiated cannot be modified. Thus these two motor components (approach and orient), while being similar in that they are produced by adjustable pattern generators, are dissimilar with respect to the degree of modifiability during execution. This distinction is largely attributable to the differences in the speed of execution and duration of the two motor patterns. The snapping stage of prey capture is a swift, stereotyped, and ballistic motor pattern with very little variability. It is the component with the smallest degree of variability in its duration and execution, is the most rapidly executed, and has several subcomponents. Snapping starts with the lunge of the head toward the prey, followed by opening of the mouth, tongue projection toward the prey, retraction of tongue with prey, closure of mouth, and swallowing of prey. The neuronal substrate responsible for this chain of events can be modeled as a linearly operating sequence of modules controlling the different subcomponents of snapping. Variations in neck muscle contractions are considered to provide the parametric control necessary to match the lunging distance with the relative location of the prey. On the other hand, jaw and tongue movements appear to be relatively invariant and stereotyped. A model with three integrator networks is proposed for the control of tongue muscles during prey capture.