Planning to Reach or Point II: A Next-State Planner

Overview: Smooth hand trajectories may be an emergent property of a feedback control system that plans for a desired change in the limb's state based on an estimate of its current location and goal. Called a next-state planner, such a system allows the CNS to respond smoothly, as if on autopilot control, to unexpected changes in goals or perturbations to the limb. Evidence indicates that people carrying the gene for Huntington's disease, a disorder primarily of the basal ganglia, do not make these computations efficiently.

Neville Hogan\(^1\) noted that minimum-jerk trajectories do not explain "the cause of the behavior they describe but rather [provide] a distillation of its essence." He recognized that although minimum jerk provides a concise description of a wide variety of movements, it fails to address how the CNS might produce such movements or the advantages of doing so. This chapter considers those topics.

19.1 The Problem of Planning

Chapter 18 described the smoothness of reaching and pointing movements in visual coordinates: At their fastest, they have the least possible jerk. One idea about how your CNS controls these trajectories holds that when you decide to make a movement, your CNS computes a minimum-jerk trajectory, called a desired trajectory. Perhaps your CNS stores this trajectory somewhere and plays it out like a tape. At each instant of time, the tape provides the desired state of the limb—its location, velocity, and acceleration—and internal models transform these desired states into motor commands.

The notion of a desired trajectory entails two assumptions: (1) that your CNS plans the details of its movements far in advance and (2) that a mechanism keeps track of time. Both assumptions are problematic. For example, imagine that you want to stir a pot of soup with a spoon. As you begin stirring, it seems unlikely that your CNS would plan the number of times your hand would circle the pot and the entire trajectory of your hand during that motion. Or, to take another example, imagine that as you are reaching to a target, the target jumps halfway into the movement
or something perturbs your hand. Your CNS would need to reevaluate the
desired trajectory because such changes render the original plan obsolete.

Two groups of investigators—first Bruce Hoff and Michael Arbib,
and later Stefan Schaal and his colleagues—noted these problems and de-
developed models that do not depend on a precomputed desired trajectory.
They suggest that the CNS produces movements by evaluating a goal in
relation to the limb's current state, in real time, and then generates a small
desired change in state. A smooth movement that minimizes jerk might be
an emergent property of such an autonomous feedback system. A system
like that does not plan the entire desired trajectory at or prior to the onset of
the movement. Rather, it monitors the current state of the end effector and
the target, and continuously formulates a desired change in end-effector
state for the immediate future. The system acts as a next-state planner.

The next-state planner receives a high-level goal: bringing an end
effector to the target. It iteratively accomplishes this goal by breaking the
movement down into a sequence of small end-effector displacements: not
in advance, but as the movement progresses. Put another way, it always
plans the next state based on an evaluation of the current state with
respect to the goal. This approach not only produces smooth trajectories
when an end effector reaches to a stationary target, but it also explains
why movements remain smooth when a target changes location during
movement or some perturbation unexpectedly affects the arm. As you will
see, this framework also does away with the notion of a timekeeper, some-
thing the desired-trajectory hypothesis requires. This general concept is
often called "autopilot control" (see section 17.3).

While this approach clarifies a lot about how the CNS plans move-
ments, it does not, at first, seem to explain the smoothness of end-effector
trajectories. A theory put forth by Chris Harris and Daniel Wolpert sug-
gests that two factors may combine to make smooth trajectories a good
plan, and a next-state planner would produce them. First, neurons in the
motor system produce a noisy output in which the standard deviation of
the noise grows as the output increases. Second, the CNS needs to mini-
mize variability at the end of movement in order to achieve goals. In a
system that has these two characteristics, minimum-jerk movements work
well because they do not require large motor commands and produce ac-
curate endpoints. The next-state planner does not generate a motor com-
mand to reach the target in one go, so the commands at any given time
remain small (and therefore relatively low in noise). But it always gets to
the target, no matter what happens to the limb or the goal along the way.
The final stages of a reaching or pointing movement always involve small
motor-command signals, so the system can minimize noise-related errors.
Section 19.4 explores these ideas in more detail.

19.2 Transforming a Displacement Vector into a Trajectory

The model depicted in figure 14.1 starts by computing an estimate of
target location \( \hat{x}_t \) and end-effector location \( \hat{x}_{ee} \) in fixation-centered coor-
coordinates, and then computes a difference vector $x_{dv}$. How can your CNS gradually transform this vector into a change in joint angles $\Delta \theta$ so that your CNS produces a smooth movement to the target? And how can it specify the intention to move fast at one time but slow at another?

Dan Bullock and Steve Grossberg proposed that in addition to the spatial variables $x_{dv}$ and $\Delta \theta$, another variable describes intended movement speed. They considered this variable to be a scalar function of time $\gamma(t)$ and called it a "go" function. To derive some of the properties of this function, they noted that $x_{dv}$ decreases monotonically during a movement. They proposed that in order to compute $\Delta \theta$ at any instant of time, $\gamma(t)$ should scale $x_{dv}$ to produce a time-dependent difference vector, which is transformed into a joint-rotation vector. They proposed that $\gamma(t)$ should be a monotonically increasing function of time and should specify how fast you intend to make the movement.

To see the consequence of this approach, consider the behavior of the system over a small time interval $t = 0 \rightarrow \Delta t$. Initially, $x_{dv}$ begins as a relatively large vector and $\gamma(t)$ begins small. Multiplying the two factors leads to the scaling of $x_{dv}$; hence the product begins as a relatively small vector. To map $x_{dv}$ into $\Delta \theta$, you might use the Jacobian that relates joint rotations to changes in end-effector location:

$$x = f(\theta)$$
$$J = \frac{dx}{d\theta}.$$  

Using this Jacobian, you can compute the joint rotations needed to move the end effector along a small displacement $\Delta x$:

$$\Delta x(t) = \gamma(t)x_{dv}$$
$$\Delta \theta(t) = J^{-1}\Delta x(t).$$  (19.1)

Note that the joint-rotation vector now corresponds to a velocity. In the final step, according to this model, your CNS transforms this velocity into a motor command and produces the forces that move the arm. As it computes these commands and transmits them to the spinal cord, it also sends an efference copy somewhere (perhaps to the PPC) to estimate current end-effector location $\hat{x}_{ev}$, which in turn produces a new $x_{dv}$. This series of computations produces a monotonically decreasing function $x_{dv}(t)$ multiplied by a monotonically increasing function $\gamma(t)$. This multiplication results in a function $\Delta x(t)$ that first increases and then decreases during movement. $|\Delta x(t)|$ is the speed profile for the movement. If $\gamma(t)$ is 0 at $t = 0$, $\Delta x(t)$ grows from 0 to some peak value, then declines to 0 as the vector $x_{dv}$ becomes smaller. By manipulating $\gamma(t)$, this model can produce fast or slow movements.

Figure 19.1 shows an example of a movement generated by this model. For this simulation, $\gamma(t) = 0.1t^{1/4}$, where $t$ has units of seconds. The end effector moves along a straight line to the target (figure 19.1A) with a unimodal speed profile (figure 19.1B). Figure 19.1C and 19.1D show the
Joint rotations corresponding to this movement, as calculated through an iteration of equation 19.1.

Three important ideas emerge from this model. First, the model assumes that your CNS always computes an estimate of the current end-effector location. Second, it assumes that your CNS continuously estimates the vector that points from the current end-effector location to the target. And third, it scales the estimate by a time-dependent function $\gamma(t)$. At any time $t$, the resulting vector corresponds to the desired displacement for the end effector, $\Delta x(t)$. A transformation of $\Delta x(t)$ to joint coordinates, and a further transformation to forces, produces the motor commands that eventually move the limb. The so-called invariant properties of limb trajectory (i.e., its straightness in visual coordinates and minimum-jerk smoothness) represent emergent properties of both the feedback system and the "go" signal.

Scaling the difference vector $x_{de}$ by a time-dependent function $\gamma(t)$ has some limitations, however. Consider what should happen to $\gamma(t)$ if conditions change during the movement. For example, imagine that as you start reaching to pick up a pencil (figure 17.1B), someone accidentally bumps the table and the pencil begins rolling away from you. Target location $x_t$ thus changes during the execution of the movement. Because
the model continuously estimates \( x_{dv} \), the change in target location affects
the difference vector. However, does \( \gamma(t) \) also change? Does it need to be
reset, for example? Recall the discussion of autopilot control in section
17.3. When a target jumps during a movement, the movement adjusts
smoothly, only slightly lengthening movement duration. But by the time
that happens, the go function would have reached a high level, which
would create problems. Imagine further that somehow the pencil keeps
rolling away at the same velocity as your reaching movement so that
\( x_{dv} \) remains constant. The motor commands would get larger and larger.
The "go" function of Bullock and Grossberg shows one way in which a
smooth hand trajectory can emerge from a state-feedback system, but it
relies on a feedback-independent function of time and an exponential rise
to infinity, neither of which seems realistic. The model nevertheless illus-
trates that a control system does not need to plan for the entire trajectory
at the onset of a movement.

19.3 The Next-State Planner

Instead of planning an entire trajectory in advance of movement onset, a
controller can just plan for what to do next. This approach requires an
appropriate control policy, one with a measure of target and end-effector
locations. As the CNS executes its plan, and therefore the limb's state
changes, the planner responds by updating the plan of what to do next.
The planner has a policy (i.e., a set of equations that describe how to pro-
duce an output, your desired end-effector location in the near future),
given some inputs (estimates of current limb configuration and target loca-
tion). Through this approach, the planner becomes an autonomous sys-
ystem. If your policy involves smooth movements, the problem of planning
becomes one of figuring out the autonomous system that will produce
smooth movements regardless of the state of the limb or the location of
the target.

Two theoretical developments have suggested a way to solve the
problem of planning. First, Bruce Hoff and Michael Arbib showed how to
derive the dynamics of the autonomous system with a policy of moving
smoothly. Second, Stephan Schaal and his colleagues showed that what-
ever your policy may be, you can produce an autonomous system that
implements that policy. That is, whether you intend to move to a target
smoothly, in a zigzag, or more variably, a policy can describe those move-
ments by setting the static parameters of an autonomous system. These
parameters describe your intention, at an abstract level, of how to move.
The next-state planner, then, computes the details of the plan during the
actual movement.

19.3.1 The Hoff–Arbib Model

Recall that the Bullock–Grossberg model had a feedback-independent, but
time-dependent, "go" function \( \gamma(t) \) that scaled motor output. Bruce Hoff
and Michael Arbib developed a model that produces smooth movements and also relies on feedback, like the Bullock–Grossberg model, but to some extent does away with a time-dependent function that scales $x_{d_{fr}}$. In dispensing with a "go" signal, Hoff and Arbib considered a system having a current estimate of $x_{d_{fr}}$, as well as a current estimate of end-effector location, velocity, and acceleration $(\dot{x}_{ee}(t), \ddot{x}_{ee}(t), \dddot{x}_{ee}(t))$. This system would produce the desired state of the end effector at some time in the future: $[x_{d}(t+\Delta), \dot{x}_{d}(t+\Delta), \ddot{x}_{d}(t+\Delta)]$. Their model computed this desired state so that the trajectory of desired end-effector states would follow a smooth transition to the target. That is, they forced the transition of the states from one time to another to follow a criterion described by the minimum-jerk function.

Hoff and Arbib began by computing the solution to the minimum-jerk functional (see section 18.2.1) for arbitrary initial and final limb states. For example, if the initial state at $t = 0$ is something other than 0 velocity and 0 acceleration, then acceleration at time $t$ becomes a function of these initial conditions:

$$
\ddot{x}(t) = \ddot{x}_o (1 - 9\delta + 18\delta^2 - 10\delta^3) + \ddot{x}_o (-36\delta + 96\delta^2 - 60\delta^3) / D
$$

$$
+ \left( \ddot{x}_o - x_o \right) (60\delta + 180\delta^2 + 120\delta^3) / D^2,
$$

where at $t = 0$, $x = x_o$, $\dot{x} = \dot{x}_o$, and $\ddot{x} = \ddot{x}_o$. Variable $D$ represents the desired duration of the movement and, therefore, corresponds to the difference between the final time of movement $t_f$ and the original time of movement $t_o$ such that $D = t_f - t_o$. The proportional elapsed time is $\delta = (t - t_o) / D$. The time derivative of the function described in equation 19.2 is a description of how acceleration changes at any instant of time (i.e., jerk). If you find the jerk $\dddot{x}(t)$ and in its expression set $\delta = 0$ and $D = t_f - t$, you have a rule with which you can optimally change the desired state of your system. You have

$$
\Delta x_{d}(t) = \dot{x}_d(t)
$$

$$
\Delta \dot{x}_{d}(t) = \ddot{x}_d(t)
$$

$$
\Delta \ddot{x}_{d}(t) = \left( -\frac{60}{D^3} (x_t - \dot{x}_{ee}(t)) - \frac{36}{D^2} \dot{x}_d(t) - \frac{9}{D} \ddot{x}_d(t) \right).
$$

Equation 19.3 describes a next-state planner that conforms to an imposed policy of minimum jerk. It specifies how much change should take place in the desired location, velocity, and acceleration of the end effector. It computes this change as a function of the current estimate for end-effector location $\dot{x}_{ee}$ and the estimated time $D$ remaining to reach the target. You can express the vector that specifies desired change in end-effector location as $\Delta \dddot{x}_{d} = [\Delta x_{d}, \Delta \dot{x}_{d}, \Delta \ddot{x}_{d}]$.

The Hoff–Arbib model not only generates a smooth end-effector trajectory to the target but also changes this desired trajectory in the face of perturbations either to the target or to the end effector. For example, you can use equation 19.3 to simulate behavior of the end effector when the target jumps to a new location during a reach. In figure 19.2A, the target
Figure 19.2
Using feedback to generate a smooth desired trajectory. (A) Equation 19.3 was used to simulate desired end-effector location. Dots indicate end-effector location at 10-msec intervals, \( D = 0.97 \) sec. (B) At \( t = 500 \) msec, the target jumped to a new location. Desired end-effector trajectory changed smoothly to reflect the change in target location, as expected for autopilot control. At the time of target jump, time remaining \( D \) increased by \( 200 \) msec. (C) Target jumped at \( t = 0 \) msec, \( t = 100 \) msec, \ldots, \( t = 800 \) msec.

remains stationary and the desired trajectory for the end effector has a smooth speed profile. In figure 19.2B, at 500 msec into the desired trajectory, the target jumps to a new location. The system responds by modifying the desired trajectory, producing a second peak in the speed trace. The end-effector trajectory depends on when this perturbation takes place (figure 19.2C). If the target jumps early in the movement, the end effector simply moves toward the new location of the target. If it takes place late in the movement, the model generates two movements in sequence. In between these extremes, a family of trajectories guides the end effector smoothly to the new target location.

Notice that a change in the target location often increases the duration of the movement. Bruce Hoff noted that this change in movement time correlates with the distance that the target moved. In equation 19.3, \( D \) (time remaining to target) is an input to the system. You have to provide this input by estimating the time remaining to the target when a change occurs in target location. Therefore, the system needs a timekeeper. Hoff asked whether this change in \( D \) could be optimally determined by another estimator, one that predicts the optimum length of time it takes to perform that movement.

To incorporate a movement-duration estimator, Hoff introduced a cost function that took into account both jerk and time. The faster the movement, the smaller the time cost but the higher the jerk cost. This led to a function that estimated time remaining to target \( (D) \) as a function of
the current state of the end effector, the distance to target $x_t - \hat{x}_v$, and a constant $r$ that described the trade-off between moving fast versus moving smoothly. For example, Hoff found that for a movement that started from static initial conditions, movement time should be

$$D = (60(x_t - \hat{x}_v))^1/3 r^{1/6}.$$ 

In this formulation, $r$ "weighs" the relative cost of a movement's smoothness versus the cost of its duration. If you imagine that you can "afford" jerkier movements for larger movement targets, then this relationship leads to Fitts's Law: a systematic trade-off between movement duration and target size.

$$D = a + b \log_2(2x_{dc}/S),$$

where $S$ is the size or width of the target (i.e., the accuracy requirement), and $a$ and $b$ are regression coefficients. If you wish to move very accurately, the movement will take systematically longer and vice versa. Hoff showed that you can measure movement duration for one target, estimate $r$ from that movement, and then predict movement durations for other movements, including perturbed movements. His approach produced movement durations and end-effector trajectories that matched psychophysical data in a number of experiments.

The Hoff–Arbib model combines optimization with control and shows that a "large-scale goal" of bringing the end effector to the target can be accomplished through a sequence of small goals $\Delta \hat{x}_v(i)$. The system that plans for the next state of the limb, the next-state planner, does not plan very far in advance. When the target appears, the planner does not compute a desired end-effector trajectory for the entire movement, nor does it use a signal, such as the "go" signal of Bullock and Grossberg's model, that is an explicit function of time. Rather, the planner keeps track of the current state of the end effector, the target's location, and the amount of time needed to reach the target, and then estimates the desired next state for the end effector. The ability to change movement plans in response to perturbations of the end effector or displacements of the target emerges from the properties of this next-state planner.

### 19.3.2 Planning for Arbitrary Trajectories

With the Hoff–Arbib model, you can abandon the notion of planning an entire reaching or pointing movement from beginning to end. Instead, you can rely on an autonomous feedback-control system to plan a movement through a next-state planner. In each instant, the next-state planner evaluates the goal of the task as well as an estimate of the current state of the limb, and then generates a desired change in that state for the immediate future. However, what if you do not want simply to reach straight to a target, but desire to move your hand in a zigzag path or take some other route to the target? To make such movements, you would need a different next-state planner.
Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal considered this problem and proposed a way to represent most trajectories in terms of a parameterized, autonomous system. In their model, the next-state planner consists of a set of nonlinear differential equations. The parameters of these equations describe the "landscape" over which the system brings the limb to the goal. Therefore, as the system learns to plan a movement, it learns the parameters of the autonomous system.

For example, imagine that you have a robot (figure 15.1A) and you want to guide its gripper to a target. However, in this case you do not want it to go straight to the target; rather, you want the gripper to take a zigzag path along the way. You need to set the parameters of a next-state planner. At any instant, it evaluates the end effector's location with respect to the target and generates a small desired change in state. When the changes in state accumulate over a long period, the gripper moves in a zigzag pattern.

In this model, the next-state planner has two inputs, \( x_t \) and \( \dot{x}_{ee} \) (target location and estimated end-effector location), and one output, \( \dot{x}_d \) (desired change in end-effector location). It has four internal states, labeled \( z, v, x, \tilde{x} \), and a few time constants \( a_z, b_z \), and so on. A set of differential equations describes the dynamics of this system:

\[
\begin{align*}
\dot{z} &= a_z(b_z(x_t - x_d) - z) \\
\dot{v} &= a_v(b_v(x_t - x_d) - v) \\
\dot{x} &= v(1 + a_{px}(\dot{x}_{ee} - x_d))^{-1} \\
\dot{\tilde{x}} &= (x - x_o)/(x_t - x_o) \\
\dot{x}_d &= z + \frac{\sum_{i=1}^{n} w_i g_i(\tilde{x})}{\sum_{i=1}^{n} g_i(\tilde{x})} v + a_{py}(\dot{x}_{ee} - x_d).
\end{align*}
\]

(19.4)

In this formulation, \( x_o \) represents end-effector location at the start of the movement, and \( \tilde{x} \) represents the fraction of the distance progressed to the target. A set of nonlinear basis functions \( g_i \) encodes the distance \( \tilde{x} \):

\[
g_i = \exp\left(-\frac{1}{2\sigma_i^2}(\tilde{x} - c_i)^2\right).
\]

Each basis has a "center" at \( c_i \). The parameters of this system are the weights \( w_i \) and the time constants \( a_z, b_z \), and so on. If the time constants are all positive, the system will settle at \((z, v, x, x_d) \rightarrow (0, 0, x_t, x_t)\). That is, the desired end-effector location will go to the target, without doubt. However, by manipulating the time constants and the weights of the basis functions, you can dictate how the system brings the end effector to the target. The task of learning to plan a movement becomes one of learning the parameters of this dynamical system.

For example, consider a movement around a single joint, where \( x_t = 1, \dot{x}_{ee} = 0 \). You want to complete the movement in approximately 1 sec. The desired movement time dictates the time constants of equation 19.4. In this case, you set the time constants \( a_z = a_v = 12 \) and \( b_z = b_v = \)
\( a_z/4 \). You do not know for sure how you want to move to the target (i.e., whether smoothly or in a zigzag), so you simply set the weights \( w_i \) to be some random numbers. Further, assume that your planned movements have no error (i.e., \( \dot{x}_a = x_d \)). The resulting movement (figure 19.3A) gets to the target in approximately the right time but not with a smooth trajectory: The velocity profile has a number of peaks. If you want to plan a smooth movement (e.g., one with a minimum-jerk profile), you need to change the weights in the model so that the next-state planner produces such a movement through its own dynamics. Once you find the “right” weights (the web document *schaalmodel* provides the procedure for finding them), the end effector moves smoothly to the target (figure 19.3B). By design, the next-state planner easily handles changes in the target location during the movement or perturbations to the limb. Accordingly, figure 19.3B displays behavior of the system when the target jumps at various times in the movement.

Now consider a reaching movement that takes a zigzag path. To plan the zigzag, you set the weights of your dynamical system so that its “landscape” produces the desired behavior. In this case, you set the weights so that the end effector moves slightly toward the target, then moves away from the target, and then moves to the target again. Figure 19.3C shows that the system can produce such a behavior. More interestingly, you can ask the system to display its desired trajectory when the target jumps. You find that the trajectory has a single zigzag if the target jumps at the very start of the movement (at time 0) but a double zigzag if it jumps at the middle of the movement (at time 0.5). If the target jumps at around 1 sec, when the movement to the first target is nearly complete, you will see a smaller second zigzag.

People may not plan their movements using exactly this kind of mechanism, but it illuminates certain principles. The formalism introduced by Schaal and his colleagues suggests that movement planning can be represented as the static parameters of an autonomous system. When someone switches the target of the movement, the autonomous system generalizes from a movement that its weights have been trained on to another movement. The pattern of generalization depends on the basis functions \( g_i \).

Figure 19.4 embeds this model in a control system. In this figure, the displacement map transforms the change in desired end-effector state \( \dot{x}_a \) into a desired change in joint state \( \dot{\theta}_a \). Mathematically, the transformation involves an estimate of the inverse of the Jacobian, \( J^{-1}(\dot{\theta}) \). \( \dot{\theta}_a \) is then transformed into motor commands (torques) using an internal model of inverse dynamics. Chapters 20–22 explain limb dynamics in some detail. For now, you need only recognize that these motor commands eventually reach the limb, causing it to move. Because of delays in the system, the next-state planner relies on eference copy and a model of the forward dynamics of the limb to estimate current end-effector location. This aspect of the model captures the remapping function presented in figure 17.10.
Figure 19.3
Generating arbitrary trajectories with a next-state planner. (A) Equation 19.4 was simulated with 25 basis functions having centers uniformly distributed over $\tilde{x}: 0 \rightarrow 1$ and random weights $w$, sampled over an interval from $-1$ to $1$. Top: From left to right, trajectories of $z, v$, and $x$. Bottom: From left to right, trajectories of $x_d$ and $\dot{x}_d$. (B) Weights $w$ were found that produced a minimum-jerk trajectory $x_d$. At 0.2-sec intervals into the movement (including at time 0), the target jumped from location 1 to location 2 for $x_d$ (left) and $\dot{x}_d$ (right) in both unperturbed and perturbed conditions. (C) Weights $w_i$ were found that produced a zigzag trajectory to the target. Format as in B.
Figure 19.4
A system for planning and control of reaching. Schematic representation of the next-state planner in conjunction with the system that executes the plan using internal models of dynamics. $\hat{x}_t$ represents estimated target location; $\hat{x}_{te}$, the estimated end-effector location; $x_d$, the difference vector; $\hat{x}_d$, the desired change in hand state, and $\hat{\theta}_d$, the desired change in joint angles; $\tau$, the torque needed to accomplish the plan. Abbreviations: $f$, force; $J$, the Jacobian; $\theta$, joint angles; $\Delta$, time delay. (Drawn by George Nichols.)

19.3.3 Learning to Plan a Movement

The next-state planner implements a control policy, expressed through the selection of the static parameters of an autonomous system. This approach suggests that in order to learn how to perform a task, you first need to learn a control policy in terms of the parameters of the next-state planner. Emo Todorov, Shadmehr, and Emilio Bizzi studied this possibility by asking right-handed participants to hold a paddle in their left hand and hit a ball accurately. A training group began by watching a video monitor that displayed the movements of their paddle superimposed on the movements of a teacher’s paddle. A control group had no such experience. The experimenters sought to teach the training group the trajectory that an expert teacher performed. Although the training group received 50% less actual practice than the control group, they nevertheless improved performance to about the same level as controls.

19.4 Minimizing the Effects of Signal-Dependent Noise

If you set the weights of the next-state planner appropriately, it can produce state transitions that smoothly bring the end effector to the target.
The resulting movement minimizes jerk. Why should the nervous system want to optimize such a cost function?

Chris Harris and Daniel Wolpert provided a theory that answers this question. They argued that if your goal is to bring your end effector to the target in some desired amount of time $D$, then a more natural cost is something that measures the error in end-effector location $x_{e}$ with respect to the target $x_{t}$ at time $D$. To minimize this error, it seems reasonable to start the movement with high acceleration and then slow down near the end, in order to home in on the target. That is, given that you have $D$ time to get to the target, you should get close to the target quickly, and then spend most of the remaining time slowly fine-tuning end-effector location. The result of this high initial acceleration will not be a smooth movement because end-effector speed would rise quickly but fall slowly.

How, then, can you account for the finding that reaching movements have a symmetrical speed profile? Wolpert and his colleagues suggested that the reason might have something to do with noise. Neurons that convey motor commands to the muscles have a certain amount of noise in their signal. Perhaps this noise results from the fact that a neuron’s response to an input varies from one trial to the next. In their experiment, participants maintained a target force by flexing their thumb muscles. The experimenters then measured the standard deviation of the resulting force. They found that it grew roughly linearly as a function of mean force output (figure 19.5). The growth in force variability did not result from properties of the muscles: Electrical stimulation of the thumb flexor muscle did not cause increasing variance with stimulation magnitude. The signal-dependent noise came from the neural, motor-command signals.

In exploring the significance of this signal-dependent noise, Harris and Wolpert modeled a two-joint arm with pairs of shoulder and elbow muscles. They gave each muscle a motor neuron that had a noise property. The motor neuron’s output equaled the sum of its input and a noise term that had 0 mean and a standard deviation that grew linearly as a function of the input. The ratio of standard deviation to the mean is known as the coefficient of variation, which Harris and Wolpert set to about 15%. The larger the input to the motor neuron, the larger its mean output and the larger its standard deviation around this mean.

Because the modeled motor neurons had noise, two identical inputs to the modeled arm did not produce identical movements. In fact, the larger the input, the larger the variance in the movement. Therefore, if you tried to move this model limb with a high initial acceleration, and then you evaluated $x_{t} - x_{e}$ at the end of the movement (i.e., at $t = D$), you would have the endpoint error. If you repeated this measure over a number of trials, you would notice a large variance.

Harris and Wolpert suggested that your CNS learns to generate motor commands so that—across repeated movements to the same target—it minimizes endpoint error. Given a desired movement duration and a target location, they began by assuming some desired hand trajectory and then worked back through the equations of motion and estimated the
Figure 19.5
The standard deviation of noise grows with mean force in an isometric task. Participants produced a given force with their thumb flexors. In one condition (labeled “voluntary”), the participants generated the force, whereas in another condition (labeled “NMES”) the experimenters stimulated their thumb flexors artificially to produce force. To guide force production, the participants viewed a cursor that displayed thumb force, but the experimenters analyzed the data during a 4-sec period in which this feedback had disappeared. (A) Force produced by a typical participant. For each two-column panel (voluntary and NMES), the period without visual feedback is marked by the horizontal bar at the top, labeled “4 sec.” These data are expanded to make up the right half of each panel. (B) When participants generated force, noise (measured as the standard deviation) increased linearly with force magnitude (with a slope of ~1). Abbreviations: NMES, neuromuscular electrical stimulation; MVC, maximum voluntary contraction. (From Jones et al.9 with permission.)

pattern of inputs to the motor neurons. Harris and Wolpert then produced that pattern a number of times and recorded the variance in endpoint error. They searched for the desired hand trajectory that produced the smallest variance and found that in the optimal trajectory, the end effector moved with a smooth, bell-shaped speed profile along an essentially straight line to the target (figure 19.6). Therefore, perhaps end-effector trajectories have smooth accelerations and decelerations because such a pattern minimizes endpoint error in the presence of signal-dependent noise.

19.5 Online Correction of Self-Generated and Imposed Errors in Huntington's Disease

The approach outlined in figure 19.4 combines optimization with control so that at any given instant, the next-state planner produces a desired
19.5. Online Correction of Self-Generated

Figure 19.6
Comparison of observed end-effector trajectories and those predicted in a system with signal-dependent noise. (A) Observed end-effector paths for a set of point-topoint movements. (B) Optimal end-effector paths for a simulation of a two-joint, four-muscle arm, in which the inputs to the muscles contained signal-dependent noise. (C) Observed end-effector speed for movements between T1 and T3 in A, normalized to have a maximum of 1. (D) Normalized end-effector speed in all the movements shown in part B. (From Harris and Wolpert, with permission.)

state for the limb that takes into account both the target’s location and the current state of the limb. Despite the fact that the control policy remains unchanged (i.e., parameters of the next-state planner remain static throughout the movement), the system responds smoothly to unexpected events (e.g., the target jumping midway through the movement or the end effector hitting some obstacle). However, even if nothing like those perturbations occurs, you should expect some amount of unexpected behavior because of the noise that exists in the system. If the next-state planner or the forward model functions poorly, then the feedback system of figure 19.4 cannot deal with these errors effectively.

This section presents the idea that the basal ganglia needs to function properly for error corrections of this kind (i.e., normal autopilot control requires healthy basal ganglia). To investigate this idea, Maurice Smith, Jason Brandt, and Shadmehr examined the movements of patients with Huntington’s disease (HD). They also studied people called *asymptomatic gene carriers* (AGCs), who had the gene for Huntington’s disease but had not developed symptoms. But before we present the results, it might be useful to consider the nature of HD and its causes.
19.5.1 Why Do Striatal Cells Die in Huntington’s Disease?

Significant, progressive atrophy of the basal ganglia occurs in HD and its hallmark motor sign involves rapid, irregular, involuntary movements. The gene that causes HD has an abnormal lengthening of a particular sequence of nucleotides which codes for the amino acid glutamine. Within the protein encoded by this gene, called huntingtin, the long stretches of glutamine appear to compromise both exocytosis and endocytosis, which affect synaptic plasticity. Thus, some of the initial problems in HD might reflect disruption of the cells’ plasticity mechanisms. In addition, the long run of glutamine in the huntingtin protein weakens its interaction with another protein, Hip1, to the extent that the latter disassociates from a huntingtin–Hip1 complex. Hip1 then runs riot in the cell, interacting with other proteins, some of which degrade proteins and others of which compromise the function of mitochondria, eventually causing cell death. The effects of this mutation appear relatively late in life, when aging processes have compromised the ability of cells to withstand the insult.

For some reason, striatal cells have a special susceptibility to the effects of this mutation, and in HD the striatum degenerates dramatically. The earliest stages of the disease begin with degeneration of the striatal patches, or striosomes. It has been established that the patches provide the outputs from the striatum to the dopaminergic cells of the midbrain (see also section 23.4.4). In addition to degeneration in the patches, the dorsal and lateral aspects of the caudate and the putamen seem to be affected earlier than more ventral and medial parts, which in turn degenerate before the ventral striatum. In late stages of the disease, the degeneration becomes more widespread and includes the cerebral cortex in addition to the striatum. Nevertheless, most experts treat HD as a basal ganglia disorder, and especially in its early stages and for ACGs, this assumption seems reasonable. HD does, however, affect signal processing in the cerebral cortex.

19.5.2 Huntington’s Disease and Cortical Function

Many experts view the autopilot mechanism outlined in chapter 17 and above in this chapter as a function of the PPC. This section presents the idea that this mechanism does not work properly in HD and in ACGs, which seems to point to basal ganglia rather than to the PPC. However, as emphasized in section 6.2.1, the cortex and basal ganglia work together in cortical–basal ganglionic modules, and the PPC participates in such networks. The PPC sends massive projections to the striatum and the basal ganglia sends outputs to the PPC. Therefore, you should not be surprised that striatal dysfunctions in HD and ACGs include problems with autopilot control.

Recall from section 8.8.2 that somatosensory evoked potentials are diminished in patients with HD, as are long-loop reflexes. These findings indicate that proprioceptive signals have a smaller amplitude in HD patients, at least at the level of the cortex. Proprioceptive feedback plays a critical role in the autopilot mechanism and in the computations of the
19.5. Online Correction of Self-Generated

Figure 19.7
Best and worst movements of patients with the gene for Huntington’s disease (HD) and control participants. Both groups exhibited small initial errors in some movements (arrows), but movements in the HD patients had numerous sharp, jerky changes in direction as the hand neared the target. (From Smith et al.\textsuperscript{10} with permission.)

forward model because they depend on it to predict the limb’s state in the near future. Among other relevant findings, the early component of the somatosensory evoked potential decreases in correlation with the number of glutamine repeats in the huntingtin protein,\textsuperscript{14} the electrical sources of the somatosensory evoked potential are reduced in HD,\textsuperscript{15} and corticostriatal pathways are the first to show abnormalities in mice that have the huntingtin gene.\textsuperscript{16}

19.5.3 Adjustments to Noise in Huntington’s Disease

Smith et al.\textsuperscript{10} examined the feedback-control process in HD by asking patients to reach to a set of targets (figure 19.7). They noticed that in HD patients, many movements had abrupt changes in direction as the hand approached the target. In some trials, movements began in slightly the wrong direction and failed to stop smoothly at the target. However, sometimes movements appeared quite normal from beginning to end. When Smith et al. examined movements of control participants, they also saw movements that began in slightly the wrong direction. Unlike the HD patients, however, control participants smoothly corrected movements that began with small errors.

To understand these differences better, Smith et al. measured movement jerk by comparing the raw squared jerk profiles in the two groups. Early in movement, movement jerk in the HD patients matched that in the control group. Later, ~300 msec after movement onset, it began to exceed that of the controls (figure 19.8A). Interestingly, that was about the time
Figure 19.8
A deficit in error feedback control in HD. (A) The mean squared jerk ± SE on a logarithmic scale, as a function of movement time in each speed range, in three participant groups. The second peak in the squared jerk profile corresponded approximately to the peak in the speed profile. Note that groups had quite similar jerk profiles until 300–400 msec, but diverge after this point. Abbreviation: AGC, asymptomatic gene carriers for HD. (B) Average movement jerk near the end of movement (i.e., after peak speed), as a function of the amount of error early in the movement. (From Smith et al.10 with permission.)
the hand reached peak speed. Near the end of movement, HD patients showed several times the amount of jerk as controls did. In ACGs, movement jerk fell between the values for controls and HD patients, but significantly exceeded control levels.

Why might the movements of HD patients begin to become irregular 300 msec into their course and not before? Smith et al. concluded that control participants smoothly correct the errors that occur early in a movement, but in HD patients, the feedback process that suberves these corrections malfunctions. To quantify error early in the movement, Smith et al. assumed that regardless of the intended speed of movement, hand location at peak hand speed should be near the center of the movement path, a prediction of the minimum-jerk model. They estimated the error, therefore, as the difference between hand location at peak speed and the midpoint of the movement. The experimenters found that the error early in movement did not differ between the groups. However, compared with the control group, in HD patients and in ACGs a given error early in the movement led to a higher jerk in the remainder of the movement (figure 19.8B). Therefore, in HD, the motor control system appeared to respond poorly to small, self-generated errors. This finding suggested that damage to the basal ganglia in HD somehow reduced the integrity of the feedback-control system, perhaps due to some dysfunction of the next-state planner.

How could a problem in the next-state planner produce an inappropriate motor command only when there is substantial error in the movement and not when the movement is error free? One possibility is that the error-dependent increase in jerkiness results from sudden, midmovement changes in the control policy (i.e., changes in the parameters of the next-state planner). Whereas in a healthy individual, a static control policy suffices for responding to self-generated errors during a reach, it is possible that in HD, the control policy changes in the middle of movement, producing corrective “submovements.” Even in healthy individuals, however, there must be a threshold of error beyond which the initial control policy is abandoned. Perhaps this threshold is much lower in HD.

The same general result for HD patients and ACGs was also observed when perturbations were imposed on the hand during movement. Similar results have been obtained in a different motor task by Kevin Novak, Lee Miller, and Jim Houk. As with the results on control participants by Smith et al., Novak and his colleagues found that corrective submovements began at about the time of peak velocity, near 300 msec after movement onset. They also suggested that the basal ganglia played an important role in these corrective aspects of the movement. Basal ganglia disease appears to lead to an inability to correct for errors that result from the noisiness of motor commands.

19.6 Transforming Plans into Trajectories: The Problem of Redundancy

In the next-state planner, the transformation of a large-scale plan into a sequence of small, near-term plans preserves the coordinate system of the
large-scale plan. Both $x_{d0}$ and $\hat{x}_d(l)$ represent vectors in visual, fixation-centered coordinates. How do you transform a desired change in end-effector location in fixation-centered coordinates into joint rotations? Many changes in arm configuration can lead to the same displacements of the end effector.

You have only three degrees of freedom to describe end-effector location in fixation-centered coordinates because you need to deal only with the three dimensions of real space. Nevertheless, your arm configurations at the shoulder and elbow occupy at least a five-dimensional space. (If you include your wrist, they occupy a seven-dimensional space.) Because of this high dimensionality, the task of reaching or pointing involves many combinations of joint angles that can produce the same wrist or finger location. By analogy, many combinations of joint rotations correspond to the same end-effector displacement. Specifying what your end effector should do does not determine what your arm should do.

To make the problem a bit more explicit, imagine a professor pointing to a spot on the blackboard. When the professor wants to point to some other location, the tip of the pointer moves roughly parallel to the surface of the blackboard with a smooth, straight trajectory. For pointing movements of this sort $x_{d0}$ is a two-dimensional displacement. The movement retains its low dimensionality when transformed into a time sequence of end-effector displacements $\hat{x}_d(l)$. Imagine, for example, the professor walking parallel to the blackboard as he or she points to a series of locations along a horizontal line. The professor could do this with many different joint angles, and if some obstacle made him or her move away from the board, the trajectory of the end effector could continue unchanged by simply extending the arm more. The next section considers two ways to solve this problem, often called the problem of kinematic redundancy.

19.6.1 The Stiffness Approach

Imagine that you attach a pair of rubber bands to each joint of your robot's arm, configured in an antagonistic architecture. After the rubber bands are attached, the arm comes to rest at some unique configuration which depends on the stiffness and length of each rubber band. Sandro Mussa-Ivaldi, Pietro Morasso, and Roberto Zaccaria, and Mussa-Ivaldi and Neville Hogan, constructed such a system and observed that despite the limb's kinematic redundancy, if they pulled on the end effector, the joint angles changed in a consistent and reproducible way.

This happened because each rubber band acted like a spring, and springlike systems minimize potential energy. The equilibrium point of the arm represents the minimal potential energy when no one pulls on the arm's end effector. When someone does pull on the end effector, the system still has a minimal potential energy for that new location, although the energy level exceeds that of the equilibrium point at rest. At each point in the trajectory, some minimal potential energy exists, and the limb should move along that path regardless of its redundancy.
Of course, your arm also has stiffness because muscles and the associated reflexes act like springs. For example, recall from section 8.2 that if someone displaces your hand by $\Delta x$, the springlike arm muscles produce a restoring force. If you represent this restoring force with vector $f$, you have $f = K_{ee} \Delta x$.

In the above expression, $K_{ee}$ is a matrix that specifies stiffness of the arm as measured at the end effector (in this case, your hand). The force at the hand corresponds to a vector of joint torques specified by the arm's Jacobian matrix:

$$\tau = J^T f.$$ 

The muscles that produce these torques have a stiffness as measured at the joints. If you label this joint stiffness with matrix $K_{\theta}$, torque at the joints is related to joint displacement $\Delta \theta$ by

$$\tau = K_{\theta} \Delta \theta.$$ 

This expression leads you to a relationship between end-effector displacement and joint displacement:

$$\Delta \theta = K_{\theta}^{-1} J^T K_{ee} \Delta x. \quad (19.5)$$

The stiffness of muscles produces a potential energy "landscape" that can act as a minimization principle with which to solve the problem of redundancy. To displace the end effector in your redundant limb, your joints should move in a pattern specified by the stiffness of the arm and the energy "landscape" it creates.

### 19.6.2 Trade-Offs Between Controlled and Uncontrolled Variables

In the model of figure 19.4, the controlled variable is the end effector's location with respect to the target. In a redundant system, this controlled variable has fewer degrees of freedom than planned joint states or the resulting motor commands. This fact implies that as long as the end effector gets to the target, the system may not care about the precise configuration of the arm. For example, assume that there is noise in the transformation $x_d \to \theta_d$ so that the system introduces some errors in calculating the forces necessary to make the movement. The system, however, does not attend to these errors equally. The only relevant errors involve those affecting end-effector location with respect to the target. The system allows other errors to remain uncorrected.

Emo Todorov and Michael Jordan proposed that in a redundant system, the fluctuations in redundant joint-angle dimensions can be ignored unless they have an impact on the task-relevant variables. As long as those fluctuations have no relevance to the crucial variables, the redundant degrees of freedom remain largely uncontrolled.

Todorov and Jordan found that control laws for redundant tasks do not enforce a specific trajectory; instead, they obey a minimal intervention
principle, which states that only those deviations from the average behavior that interfere with the task-level goals should be corrected. As long as fluctuations among the redundant degrees of freedom do not interfere with task performance, an optimal control strategy leaves them uncorrected. The advantage of doing so results from the fact that correcting (and acting in general) is expensive: It increases both signal-dependent noise and energy consumption.

But what are the relevant variables that the CNS needs to control? The answer to that question depends on the task. In a complex task, such as tying your shoelaces, the relevant variables describe the configuration of the shoelaces, regardless of your hand movements. In a reaching or pointing task, it makes sense for the CNS to care about endpoint error because your goal involves reaching the target (as opposed to, say, making a movement along a particular trajectory). The theory of Todorov and Jordan explains trajectory smoothness as the outcome of minimizing endpoint error and energy consumption in the face of sensory and motor noise (the latter being signal-dependent). As Novak, Miller, and Houk note: “A biologically plausible optimization criterion is the minimization of the occurrence and amplitude of corrective submovements…. We postulate that... other criteria... are instead secondary.”

References


Reading List

If you have an interest in a detailed mathematical description of trajectory computations, especially the approach to resolving redundancy known as the Moore-Penrose pseudoinverse, you might consult two papers: Mussa-Ivaldi et al.\textsuperscript{19} and Mussa-Ivaldi and Hogan.\textsuperscript{20}