References

Allgower, E. L. & Georg, K. (1990). *Numerical Continuation Methods.* Berlin: Springer-Verlag.

Fun, W. & Jordan, M. I. (1992). *The moving basin: Effective action-search in forward models.* MIT Computational Cognitive Science Technical Report #9204, Cambridge, MA.

Jordan, M. I. & Rumelhart, D. E. (in press). Forward models: Supervised learning with a distal teacher. *Cognitive Science.*

Møller, M. F. (1990). A scaled conjugate gradient algorithm for fast supervised learning. (Tech. Rep. PB-339), Computer Science Department, University of Aarhus, Denmark.

Narendra, K. S. & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, *1*, 4-27.

Rosen, J. B. (1967). Optimal control and convex programming. In J. Abadie, (Ed.), *Nonlinear Programming.* New York: John Wiley.

Werbos, P. (1983). *Energy Models and Studies.* B. Lev. Ed., North Holland.

amount.[3] As shown in Figures 5(b) and 55(c), the trajectory is completed stably when $\boldsymbol{\theta}$ is scaled by as much as 1.65 or displaced by as much as 1 radian. This is significant robustness, given that the span of the joint angles are ± 1.57 radians.

3. *Performance with errors in joint velocity sensors.* In this set of experiments the readings of $\dot{\boldsymbol{\theta}}$ were either scaled by a constant factor or displaced by a constant amount. The target is reached stably with a scaling factor of up to 50 (Figure 5(d)), or displacements of as much as 14.5 rad/s (Figure 5(e)).

6 Discussion

In this paper we have described a computational technique that allows gradient-based methods to be used more effectively in nonlinear adaptive control problems. By utilizing the moving-basin search procedure, the control system in our simulations is able to avoid the control boundaries and thereby stably track the reference trajectory. The simulations also show that the control system is robust with respect to inaccuracies in the forward model of the plant.

Although our simulations studied a simple version of the moving-basin algorithm in which the search procedure was initialized with the zero torque vector at each time step, it is straightforward to cache the results of the search at one time step to initialize the search at the following time step. Because reference signals tend to change smoothly in time, the cached value will often be sufficiently close to a solution such that the local gradient search will succeed. This implies that the moving-basin search will generally be required only at the onset of a movement and at other points where the desired trajectory changes unpredictably.

Acknowledgements

This project was supported by a grant from ATR Auditory and Visual Perception Research Laboratories, by a grant from Siemens Corporation, by the McDonnell-Pew Program in Cognitive Neuroscience, by a grant from the Human Frontier Science Program. Wey Fun is a recipient of a Defense Technology Training Award from the Ministry of Defense, Republic of Singapore. Michael I. Jordan is an NSF Presidential Young Investigator.

[3] Note that in the experiments with perturbations to the sensor readings, the information provided to the trajectory planner remains accurate, such that the arm would reach the target if the desired accelerations at each time step were achieved.

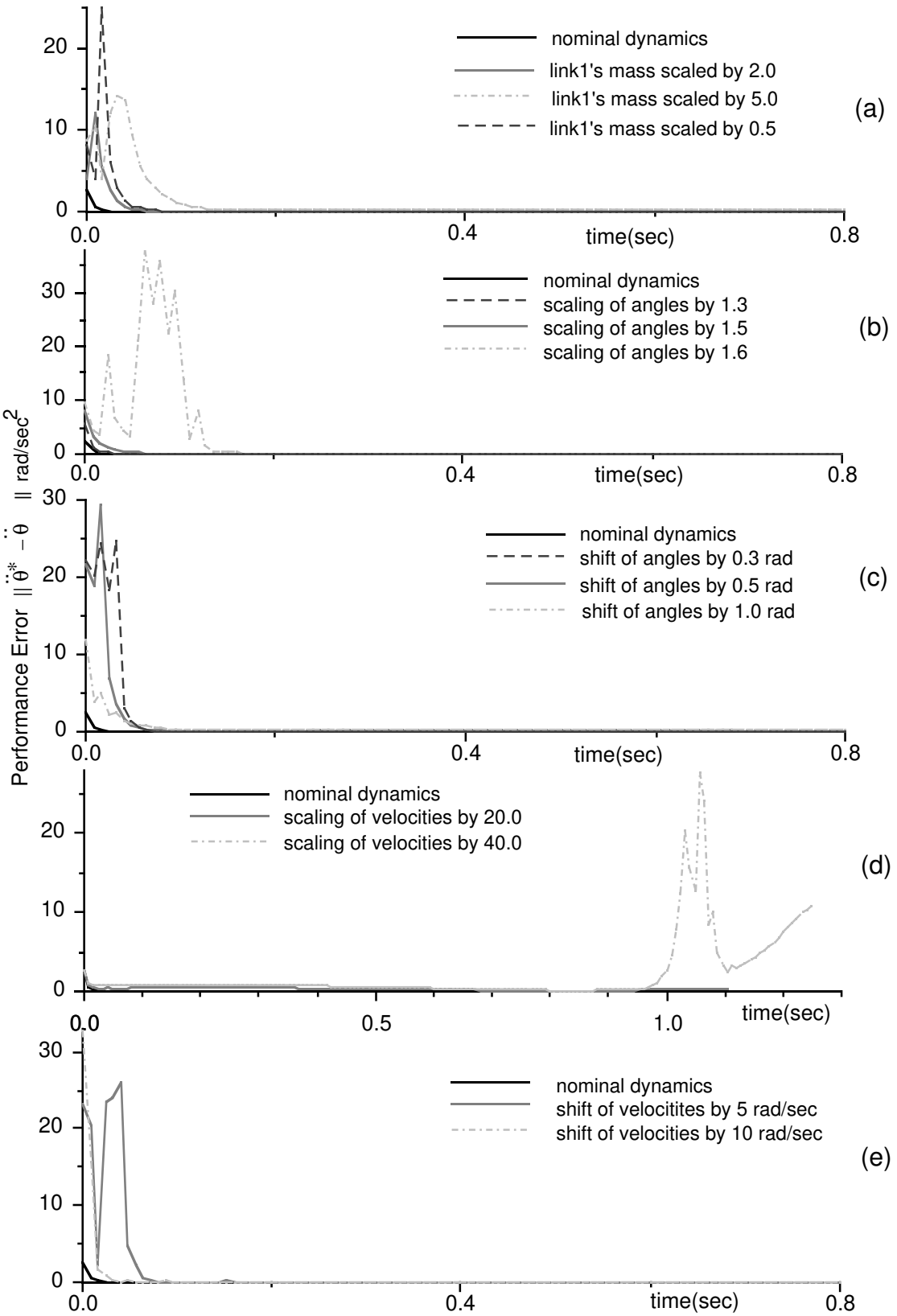

Figure 5: Performance robustness under various disturbances. The target is reached within 1.3 seconds in all cases.

gradient algorithm of Møller (1990) was used for off-line training, on-line adaptation and action search.

5.2 Controller

For the experiments reported here, no separate controller was used. Rather, the moving basin search process was initialized with a zero torque vector at each time step and the process was iterated until a control signal was found.

The reference signal for the controller was the desired acceleration, computed as follows:

$$\ddot{\boldsymbol{\theta}}^*[t+1] = 3.5(\boldsymbol{\theta}^* - \boldsymbol{\theta}[t]) + 1.31(\dot{\boldsymbol{\theta}}^* - \dot{\boldsymbol{\theta}}[t]), \tag{8}$$

where $\boldsymbol{\theta}^*$ and $\dot{\boldsymbol{\theta}}^*$ are the desired endpoint joint angle and velocity respectively.

The moving basin algorithm utilized a linear interpolatory process with a variable step size. A recursive procedure was used to refine the step size up to a certain maximum resolution before the algorithm gave up and issed the feedforward action that was predicted to achieve the virtual target closest to the commanded goal, plus a small stochastic component.

5.3 Results for nominal dynamics

The results for the reaching task under nominal dynamics are shown in Figure 4(b). As is seen the system moves the arm stably along the desired trajectory.

5.4 Robustness to perturbed dynamics

We performed a series of experiments investigating the robustness of the control system with respect to inaccuracies in the forward model. The experiments involved learning a forward model for the nominal dynamics and then altering the plant dynamics in various ways. Three experiments were performed:

1. *Performance with changes of the links' masses.* Figure 5(a) shows the performance with changes in the mass of the links. The trajectory is completed stably with up to 510 percent increase of the mass of the first link or 5 percent increase of the mass of the second link, or with 50 percent decrement of either mass. Beyond these values the system becomes unstable. The smaller tolerance with respect to increases in the mass of the second link is due to the fact that near-boundary torque is required at the second joint to achieve the desired initial acceleration.

2. *Performance with errors in the joint angle sensors.* In this set of experiments the values of $\boldsymbol{\theta}$ were either scaled by a constant factor or displaced by a constant

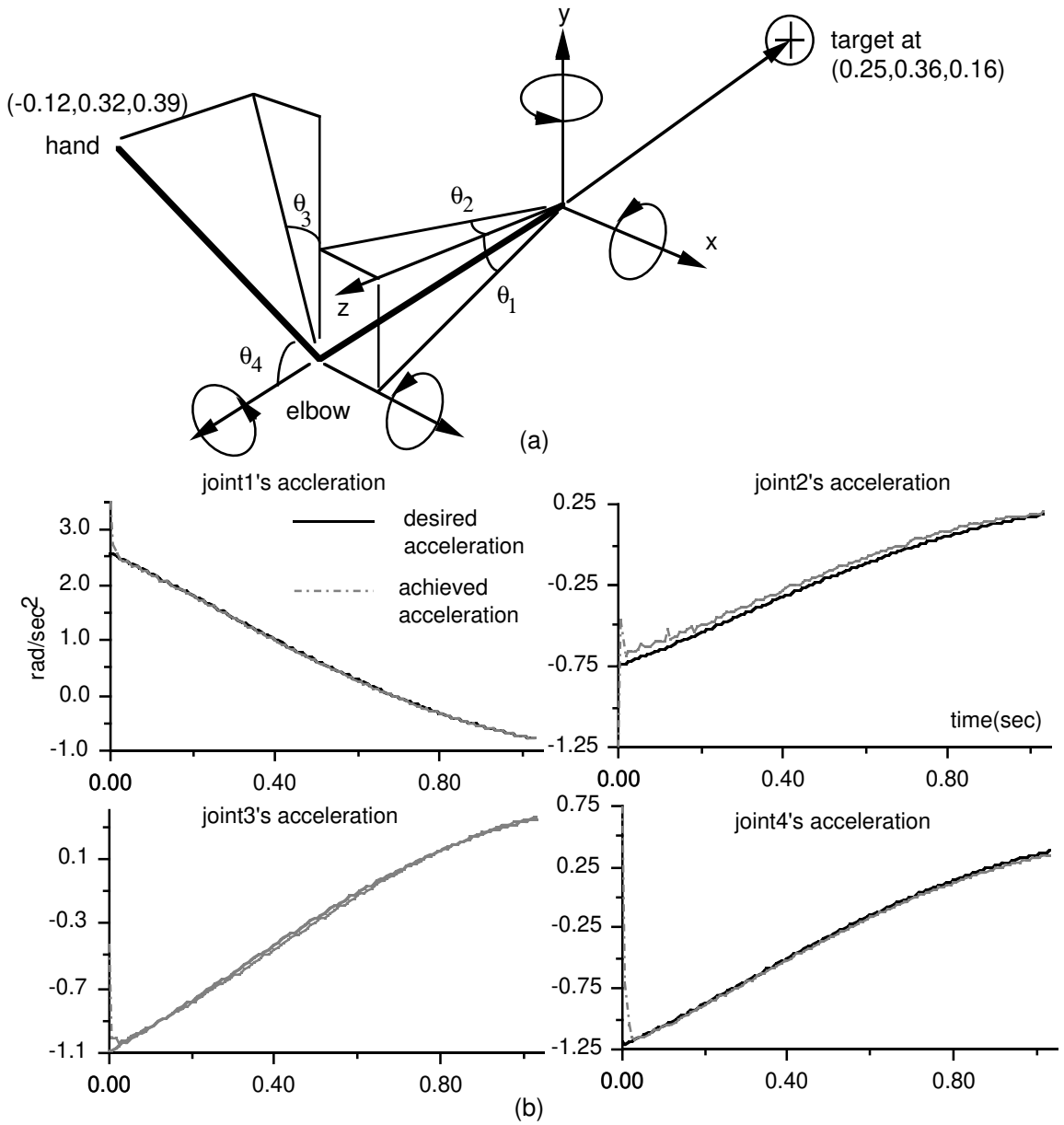

Figure 4: (a) The arm and the reaching task. The length of each link is 0.3 m. The center of mass of each link is at the center of the link. The maximum torques at the shoulder are 91.3 N m, whereas those at elbow are 15.6 N m. The experiments on robustness involved a reaching movement from the point $(-0.12, 0.32, 0.39)$ to the point $(0.25, 0.36, 0.16)$. The joint angles in the initial configuration were $(\theta_1, \theta_2, \theta_3, \theta_4) = (0.14, 0.24, 0.04, 1.04)$. (b) Performance of the controller with nominal dynamics in the reaching task. Note that the scales of the graphs of each joint are not the same. The performance error at the beginning of the trajectory is 2.66 rad/s^2. The target is reached in 1.05 seconds.

action-search process is utilized to find a locally-optimal control action. This action is then used to initialize the search at the next step. As shown in Figure 3, this process has an interpretation in terms of a moving basin of the cost function that is tracked by a linked sequence of action searches.

By setting intermediate targets along the line joining the commanded goal $\mathbf{y}^*[t+1]$ and the grounding goal $\hat{\mathbf{y}}_g[t+1]$, the system reduces the effect of the nonlinearity of the search path. By following a straighter path, the system reduces the probability of meeting the boundary of the predicted achievable target set (see Figure 2).

There are advantages to choosing the *null* or *neutral* action as the grounding action. When the predicted achievable target set is nonconvex, a neutral grounding action produces a grounding goal which is near the geometric center of the set, and the straightened search path has a better chance of reaching all of the points in the set. Also, in problems involving a redundant plant, choosing a grounding action at the origin of the control space increases the tendency of the action-search process to find solutions in which the norm of the control signal is small (i.e., "least-effort" solutions).

5 Control of a four degree-of-freedom arm in 3-D space

In this section we report the results of experiments on the control of a simulated four degree-of-freedom arm in 3-D space under the influence of gravity. The arm is shown in Figure 4(a). The shoulder and elbow are ball joints and each has two degrees of freedom. The sampling time was set at eight milliseconds, which is coarse relative to the amount of time required for a full-span traverse at maximum acceleration (approximately 100 milliseconds).

5.1 Forward model

The forward model was a feedforward network of three hidden layers of sigmoidal units, the first with 20 nodes and the second and third with 15 nodes. The input consisted of 21 linear units while the output consisted of 4 linear units. The network had 1005 connections and learned to perform the following mapping:

$$\hat{\ddot{\boldsymbol{\theta}}}[t+1] = \hat{f}(\boldsymbol{\tau}[t], \boldsymbol{\theta}[t], \dot{\boldsymbol{\theta}}[t], \mathbf{w}[t]), \tag{7}$$

where $\hat{\ddot{\boldsymbol{\theta}}}$ is the vector of estimated joint accelerations, $\boldsymbol{\tau}$ is the torque vector, and the state variables $\boldsymbol{\theta}$ and $\dot{\boldsymbol{\theta}}$ are the vectors of joint positions and velocities. The latter three variables were scaled and presented as inputs to the network. The input also consisted of sines and cosines of the components of $\boldsymbol{\theta}$.

The forward model was trained off-line with 30,000 exemplars, at which point the mean length of the prediction error was approximately 8.04 rad/s^2. We also continued training the model on-line during the reaching task. The scaled conjugate

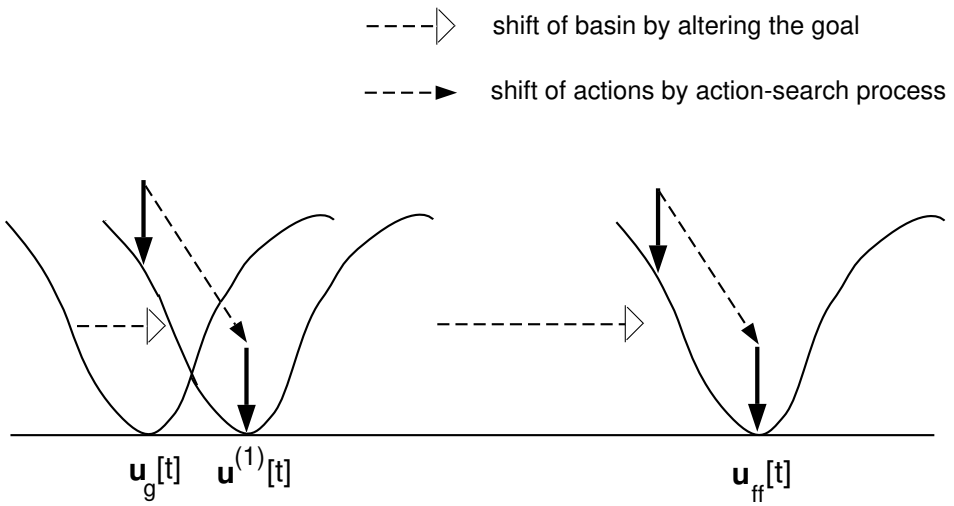

Figure 3: The basin on the left corresponds to the cost J when $\hat{\mathbf{y}}_g[t+1]$ is the target, and the basin on the right corresponds to the cost J when $\mathbf{y}^*[t+1]$ is the target. The basin of J is shifted by slowly shifting the virtual target from $\hat{\mathbf{y}}_g[t+1]$ to $\mathbf{y}^*[t+1]$, and the action-search finds the locally-optimal action $\mathbf{u}^{(i)}[t]$ for each virtual target until the feedforward action $\mathbf{u}_{ff}[t]$ is found. Note that the state and the forward model's parameters are invariant during the search process.

We have developed a very simple approach for retrieving the wanted information from the forward model in an efficient way that avoids the control boundary problem. This idea is somewhat analogous to Werbos' notion of "moving targets" (Werbos, 1983). Werbos used the term "moving targets" to describe the qualitative idea that the network should "set itself intermediate objectives, and vary these objectives as information is accumulated on their attainability and their usefulness for achieving overall objectives." Our algorithm works as follows. For a fixed value of t and a fixed value of the state $\mathbf{x}[t]$, the system chooses a *grounding action* $\mathbf{u}_g[t]$. This can be chosen arbitrarily or can be taken to be the output of the feedforward controller. The forward model is used to estimate the *grounding output* $\hat{\mathbf{y}}_g[t+1]$ that would be obtained if that action were used as a control signal:

$$\hat{\mathbf{y}}_g[t+1] = \hat{f}(\mathbf{u}_g[t], \mathbf{x}[t], \mathbf{w}[t]). \tag{6}$$

As shown in Figure 3, the pairing of the grounding action and the grounding output essentially induces an error surface whose minimum is located at the known location $\mathbf{u}_g[t]$. That is, if the system wished to achieve $\hat{\mathbf{y}}^{(1)}[t]$ as a target, then the grounding action $\mathbf{u}_g[t]$ would be the appropriate action. The system now effects a series of steps in which a sequence of virtual targets are interpolated along the path between the grounding output and the desired output $\mathbf{y}^*[t+1]$. At each step, the

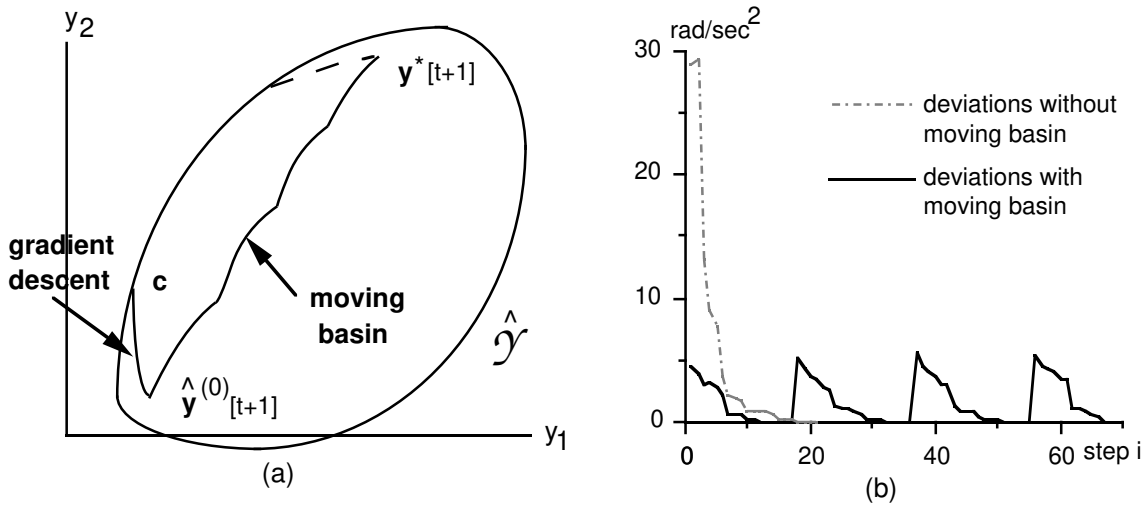

Figure 2: (a). The path taken by steepest descent is generally nonlinear and often reaches the boundary of the predicted achievable target set $\hat{\mathcal{Y}}$. The moving basin mechanism tends to straighten the path, allowing the target to be reached. (b) A plot of the deviations of $\hat{\mathbf{y}}^{(i)}[t+1]$ from the line joining $\hat{\mathbf{y}}^{(0)}[t+1]$ and $\mathbf{y}^*[t+1]$ in an actual action-search process, with and without the use of moving basin mechanism. With a 4-step target-shift the deviation is reduced by approximately 83 percent.

For a fixed value of t, the action-search process yields a sequence of control vectors $\mathbf{u}^{(i)}[t]$:

$$\mathbf{u}^{(i)}[t] = \mathbf{u}^{(i-1)}[t] + \left.\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{u}}^T\right|_{i-1} (\mathbf{y}^*[t+1] - \hat{\mathbf{y}}^{(i)}[t+1]), \tag{4}$$

where $\mathbf{u}^{(0)}[t]$ is the initial control vector provided by the feedforward controller and $\hat{\mathbf{y}}^{(i)}[t+1]$ is the corresponding sequence of outputs obtained from the forward model:

$$\hat{\mathbf{y}}^{(i)}[t+1] = \hat{f}(\mathbf{u}^{(i)}[t], \mathbf{x}[t], \mathbf{w}[t]). \tag{5}$$

Because the search process utilizes the transpose Jacobian $(\partial \hat{\mathbf{y}}/\partial \mathbf{u})^T$ rather than the inverse Jacobian $(\partial \hat{\mathbf{y}}/\partial \mathbf{u})^{-1}$, the path followed in the output space is generally nonlinear. We often find in practice that these nonlinear paths reach the boundary of the predicted achievable target set $\hat{\mathcal{Y}}$, as illustrated in Figure 2(a). One approach to dealing with the problem would be to project the gradient onto $\hat{\mathcal{Y}}$ and to continue the search along the boundary (cf. Rosen, 1967). However, because it cannot be guaranteed that an approximate forward model yields a convex predicted achievable target set, this approach is not feasible.[2]

The control boundary problem is problematic for the distal supervised learning approach in both on-line adaptation and off-line training of the controller. In on-line adaptation, because the system fails to move along the path leading to the goal, the forward model fails to receive useful training data. An even more serious problem is that the controller tends to have its weights set to values of high magnitude in order to produce actions that are near the boundary. The control boundary problem therefore tends to persist in subsequent time steps and the system can become unstable.

The control boundary problem is an example of a larger class of problems that are encountered when using gradient search techniques for nonlinear adaptive control. Gradient techniques are based on estimates of the partial derivatives of the plant output with respect to the control variables (cf. Equation 4). Control variables tend to be small in number and therefore act as a bottleneck for the gradient search process. This leads to performance surfaces that are complex as viewed through the bottleneck. The approach often taken to deal with complex performance surfaces–that of increasing the dimensionality of the search space–is not an option in the adaptive control problem because the dimensionality of the control space is fixed by the nature of the plant.

4 The moving basin mechanism

[2]Indeed we find empirically that the set $\hat{\mathcal{Y}}$ is generally nonconvex for a partially-trained forward model.

It is important to note that in principle a perfect controller can be learned even if the forward model is inaccurate. This is due to two factors: first, although an inaccurate forward model may preclude steepest descent it need not prevent the algorithm from moving downhill; and second, the true output from the environment can be substituted for the inaccurate estimate in the calculation of the error term (see Figure 1). That is, the system can use the *performance error* ($\mathbf{y}^*[t]-\mathbf{y}[t]$) rather than the *predicted performance error* ($\mathbf{y}^*[t]-\hat{\mathbf{y}}[t]$), (where $\hat{\mathbf{y}}[t]$ is the output of the forward model and $\mathbf{y}[t]$ is the output of the plant). In the current paper, however, we are concerned with finding an appropriate control signal *before* observing the output from the environment, thus we use the predicted performance error. The cost function that we utilize is the squared norm of the predicted performance error:

$$J = \frac{1}{2}(\mathbf{y}^*[t+1] - \hat{\mathbf{y}}[t+1])^T(\mathbf{y}^*[t+1] - \hat{\mathbf{y}}[t+1]). \tag{2}$$

To search for a control signal, the state, the reference signal, and the weights in the forward model are held fixed while $\mathbf{u}[t]$ is varied in a direction that decreases J_{pp}.

3 The control boundary problem

In preliminary simulation experiments involving the control of a four-joint robot arm, we have found that the action-search process often causes one (or more) degree-of-freedom to be required to produce a torque that is greater or smaller than possible. One possible cause for this problem is that the desired output is actually not achievable. We have found, however, that the problem exists even in cases where the desired output is achievable, and would be predicted to be so by an accurate forward model if the initial predicted output were sufficiently near to the desired output. Moreover, the problem is not due to the presence of local minima, because it can be shown from a Lagrangian formulation of robot arm dynamics that joint accelerations are *monotonic* functions of the joint torques (Fun & Jordan, 1992). Hence local minima cannot appear in the action-search process if the forward model has been well-trained.

The control boundary problem can be understood by considering the dynamics of the action-search process in the forward model's output space. Define the *predicted achievable target set* $\hat{\mathcal{Y}}$ of a forward model $\hat{f}$ at time t as:

$$\hat{\mathcal{Y}} = \{\hat{\mathbf{y}} = \hat{f}(\mathbf{u}, \mathbf{x}[t], \mathbf{w}[t]) \mid u_j^{min} \leq u_j \leq u_j^{max}, 1 \leq j \leq n\}. \tag{3}$$

where $\mathbf{u} = (u_1, ..., u_n)^T$ is the control vector, u_j^{min} and u_j^{max} are the minimum and maximum values of u_j respectively, $\mathbf{w}[t]$ is the current weight vector of the forward model, and $\mathbf{x}[t]$ is the state at time t. For a system whose dynamics is monotonic, at least one of the u_j's is equal to its minimum or maximum value at any point on the boundary of $\hat{\mathcal{Y}}$.

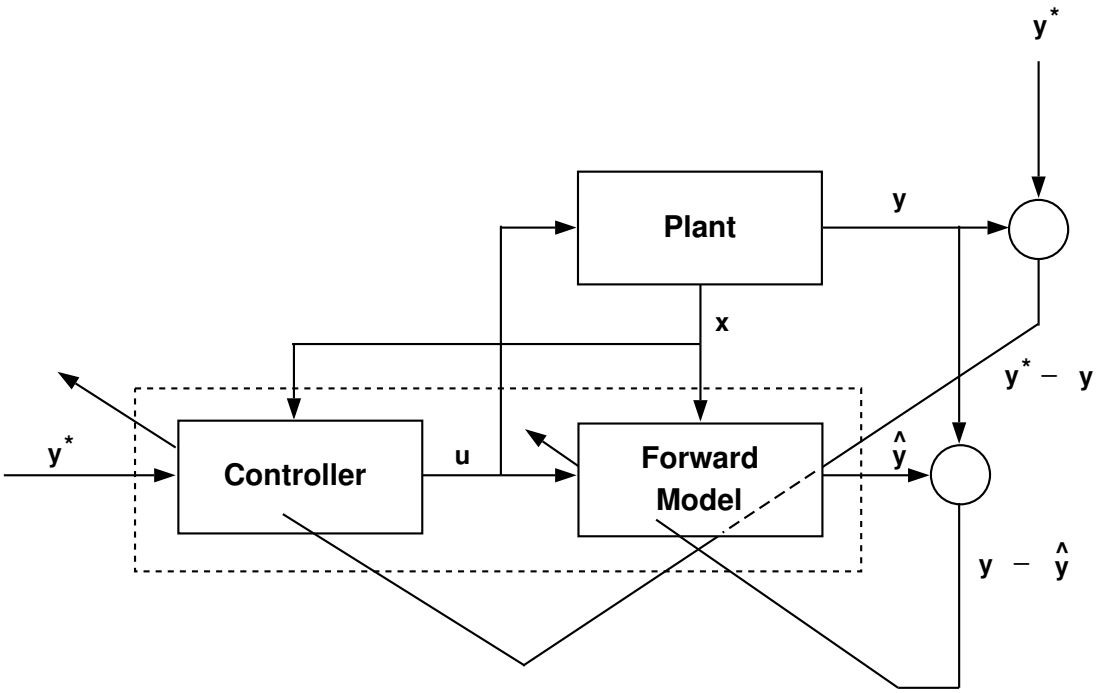

Figure 1: The distal supervised learning approach. The forward model is trained using the *prediction error* $\mathbf{y}[t] - \hat{\mathbf{y}}[t]$. The controller is trained by propagating the *performance error* $\mathbf{y}^*[t] - \mathbf{y}[t]$ backward through the forward model to yield an incremental adjustment to the controller. It is also possible to propagate the *predicted performance error* $\mathbf{y}^*[t] - \hat{\mathbf{y}}[t]$ backward through the forward model. This can be done iteratively to find a locally optimal control signal based on the initial control signal proposed by the controller.

where the state vector $\mathbf{x}[t]$ and the control vector $\mathbf{u}[t]$ are defined at time t, and the reference signal $\mathbf{y}^*[t+1]$ is the desired output at time $t+1$.

Jordan and Rumelhart (1990) discuss the problem of learning a feedforward controller using supervised learning methods. They propose an indirect approach for training a feedforward controller in which the controller is placed in series with an internal *forward model* of the controlled system. The forward model is itself trained using supervised learning algorithms. With the forward model held fixed, the composite system is trained using the reference signal as both the input and the target (Figure 1). This procedure solves implicitly for a control signal at the interface of the controller and the forward model.[1]

[1] Although we utilize gradient-based techniques in the current paper, it is worth noting that the distal supervised learning approach is not restricted to such techniques.

1 Introduction

This paper introduces a new computational technique for the adaptive control of nonlinear multi-degree-of-freedom systems. The problem that we address concerns the process of searching for appropriate control signals during on-line control. Nonlinear adaptive control generally involves an identification procedure in which the control system forms an approximate internal model of the system that is being controlled, coupled with a search process in which the model is used to search for control signals (Jordan & Rumelhart, 1990; Narendra & Parthasarathy, 1990). In nonlinear problems the requirements of the identification procedure and the search procedure tend to be in conflict: whereas identification requires the ability to represent complex nonlinear dynamics, effective search requires simple model structures. Using powerful nonlinear learning algorithms for identification may therefore run aground because of ineffective search. Indeed, in simulations of the adaptive control of four-joint robot manipulators, we have observed that gradient-based neural network algorithms are not always able to find control signals that achieve specified endpoint accelerations throughout the state space. The problem is particularly acute early in learning, when the controller is often unable to propose an initial control signal that lies sufficiently close to a solution. As we show below, in robotic control the problem is not due to the presence of local minima, but arises instead because the search process reaches the boundary of the search space. Whether the problem is caused by local minima or by control boundaries, failure to find an appropriate control signal can lead to instability.

In this paper we describe an approach for achieving more effective search in gradient-based adaptive control. The approach is related to continuation methods in optimization theory (Allgower & Georg, 1990). We propose that the system begins by setting a target that is known to be achievable at the current point in the state space. The system then sets a succession of intermediate targets that can be achieved by a linked succession of "easy" searches. Within the framework of gradient-based optimization, this approach has an interpretation in terms of a "moving basin" of the cost function. We will show that the moving basin mechanism can solve adaptive control problems that simple gradient descent cannot.

2 Distal supervised learning

To control a system requires solving for a control input to the system that will yield a desired result at its output. A *feedforward controller* solves for a control signal on the basis of future desired values of the output. As shown in Figure 1, a feedforward controller is a mapping from reference signals and states to control signals:

$$\mathbf{u}[t] = g(\mathbf{y}^*[t+1], \mathbf{x}[t]). \tag{1}$$

The moving basin: Effective action-search in adaptive control*

Wey Fun
Michael I. Jordan
Department of Brain and Cognitive Sciences
Massachusetts Institute of Technology

Abstract

We present a novel technique that allows neural networks to be used more effectively in nonlinear adaptive control problems. The technique avoids some of the difficulties associated with gradient-based methods by adaptively creating subgoals that lead the system to the desired goal. We describe the results of using this technique for the adaptive control of a simulated three-dimensional, four degree-of-freedom robot arm. Experiments are reported that study the robustness of the control system with respect to inaccuracies in the system's internal model of the arm dynamics.

*Submitted to: *Neural Computation*. A longer version of this paper will appear as MIT Computational Cognitive Science Technical Report #9204.