[illegible]

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

AIM 412 April 1977

Control and Learning by the State Space Model: Experimental Findings

by Marc Raibert

ABSTRACT This is the second of a two part presentation of a model for motor control and learning. The model was implemented using a small computer and the MIT-Scheinman manipulator. Experiments were conducted which demonstrate the controller's ability to learn new movements, adapt to mechanical changes caused by inertial and elastic loading, and generalize its behavior among similar movements. A second generation model, based on improvements suggested by these experiments, is suggested.

This report describes research done in the Department of Psychology and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research is provided in part by grant NIH-5-T01-6M0-1064-14 supervised by H.L. Teuber.

Table of contents

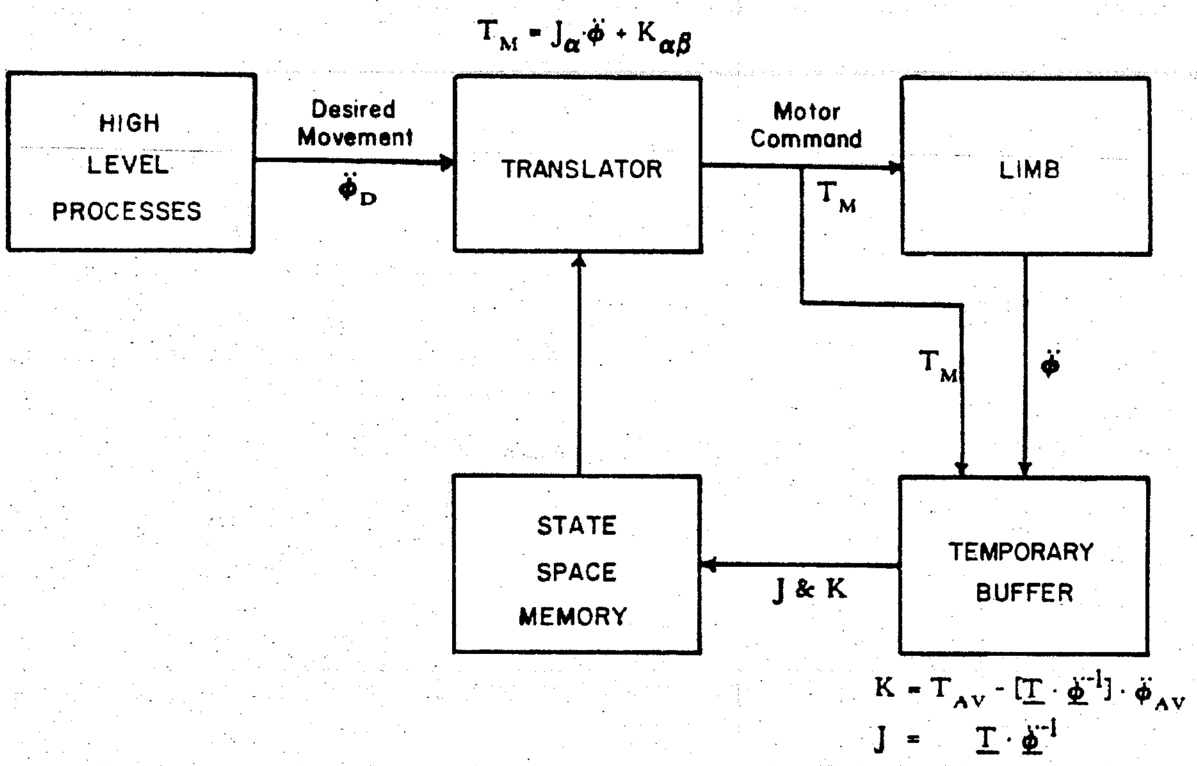
Introduction

The human motor system is organized into high- and low-level processes [7,17]. High-level processes produce descriptions of desired movements which are independent of the mechanical considerations necessary for production of the movement. Low-level processes, each associated with the kinematic and dynamic properties of a particular effector system, translate the high-level descriptions into motor commands. When man improves in motor performance through practice or adapts to changes in the mechanical properties of his limbs, part of his improvement is due to a more effective translation of the high-level descriptions.

A model for sensorimotor control has been proposed which helps to account for this behavior, and presents the possibility of designing machines which exhibit these characteristics [17]. Such a model for the low-level processes is based on the following considerations: An 'internal *inverse* dynamic model' translates descriptions of desired trajectories into motor plans. The high-level processes providing input to the translator do not have to deal with the non-linear mechanical properties of the manipulator; properties which vary with the static and dynamic configuration of the limb. The specified trajectories may be expressed in a coordinate system appropriate to the available sensors (eg. visual coordinates). The translator's outputs are motor commands suited to the dynamic properties of a particular manipulator and its actuators.

The translation is based on a form of the equations of motion which allows a tabular representation of a manipulator's dynamic behavior. The table, actually a quantized multi-dimensional memory organized by state variables, is supplied with data derived from the analysis of 'practice' movements. The analysis performed is based on 'measurement' rather than search -- error correction and hill-climbing are avoided. Fig. 1 summarizes the model's essential componenets. (See [17] for more details and [22] for a discussion of learning based on search techniques.)

We predict that a device which performs this translation process in the manner outlined, will have the following desirable properties: Each limb motion will contribute to the improvement of subsequent motions. When a particular movement is practiced, the controller's ability to generate the desired trajectory will rapidly improve. Other movements, similar to the one practiced should also improve. No a priori information about the particular kinematic and dynamic properties of the manipulator need be supplied, and a large range of sensory, actuator, and kinematic non-linearities will be acceptable. The choice of coordinate system in which to specify desired movements will be quite flexible.



$T_M = J_\alpha \cdot \ddot{\phi} + K_{\alpha\beta}$
HIGH LEVEL PROCESSES
Desired Movement
$\ddot{\phi}_D$
TRANSLATOR
Motor Command
T_M
LIMB
T_M
$\ddot{\phi}$
STATE SPACE MEMORY
J & K
TEMPORARY BUFFER
$K = T_{AV} - [\underline{T} \cdot \underline{\ddot{\phi}}^{-1}] \cdot \ddot{\phi}_{AV}$
$J = \underline{T} \cdot \underline{\ddot{\phi}}^{-1}$

Fig. 1 Major componenets of the model. The translator converts descriptions of desired trajetories into motor commands suited to the kinematic and dynamic properties of a particular limb. This operation makes use of the tabular equations of motion in conjunction with the state space memory. Each movement of the limb generates data which, when processed by the inversion equations, contribute to the state space memory.

Adaptation to changes in the mechanical properties of the limb will also take place.

These predictions are based only on intuition; this paper examines these predictions experimentally. We would like answers to the following questions:

1) How well does the translator perform vis-a-vis its expected desirable properties?
2) How does the behavior of each component of the translator influence overall performance and contribute to successes and failures?
3) What relationships can be drawn between the behavior of the translator and that of the human?
4) How might the processes described here coexist with other models for control?

Our answers to these questions depend on data obtained using a variety of tests applied to an implementation of the model which we hope adequately reflects its power and its weaknesses. These data include measures of overall performance during learning and adaptation, as well as information about the behavior of internal variables.

Implementation

In order to evaluate the power of this model we developed a set of computer programs which embody its various elements. These programs are used to control a mechanical arm in order to study the detailed nature of the resulting movement. We have introduced a number of notions -- practice movements, a temporary buffer, inverse computations, a discretized state space memory, desired trajectories, and a translation process [17] -- which now have to be made concrete.

A PDP-11/45 computer is used to perform all computations, to issue commands to the manipulator, and to make measurements. The manipulator is the MIT-Vicarm, manufactured by Victor Scheinman. It has six degrees of freedom; the three joints used in this study, (N=3), allow the wrist to be positioned arbitrarily within the arm's work space. See Fig. 2. Each joint is powered by a DC torque motor and provided with a clutch-type brake which can be used to hold the arm stationary when no movement is in progress. The PDP-11 may, through suitable circuitry, specify the current delivered to each motor. DC torque motors have the characteristic that the torque they deliver is proportional to the winding current, independent of armature velocity. Since the currents for each motor may be specified independently and simultaneously the PDP-11 computes a vector which

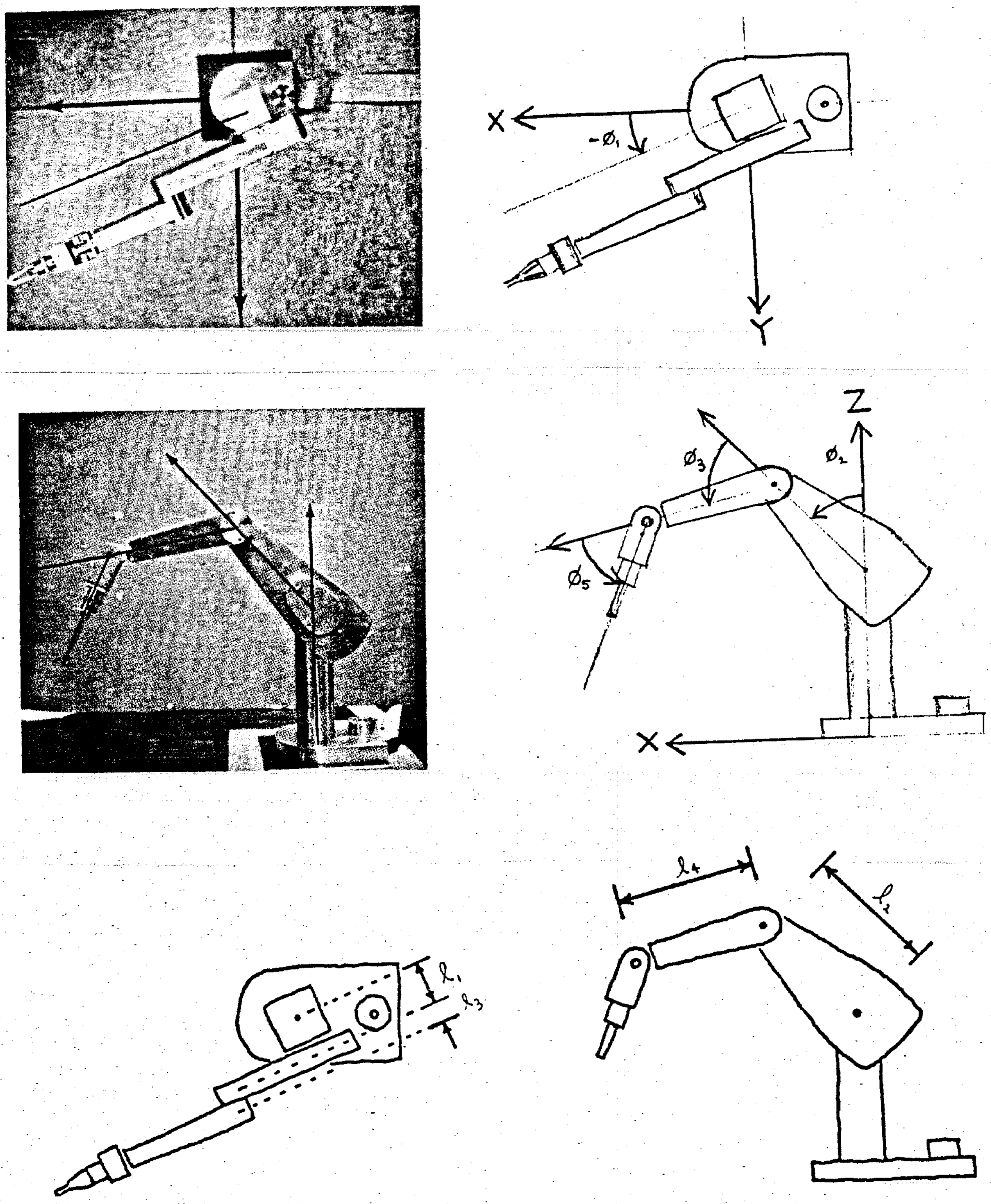


Fig. 2 Layout of the first three joints of the MIT-Vicarm manipulator are shown. ϕ_1 acts about the vertical axis. The manipualtor is about the size of a human arm; l_0=.273m, l_1=l_3=.059m, l_2=l_4=.203m. Each joint is provided with a DC torque motor, a potentiometer, a tachometer, and a clutch-type brake. The diagram is from [10] with modifications.

determines the torques applied to the joints of the arm.

Signals proportional to angular position and velocity are available from potentiometers and tachometers provided for each joint. When a movement is made the computer makes position and velocity measurements every 10 msec. In addition, the velocities, sampled every .5 msec., allow the accelerations to be estimated using least mean square error techniques. A record of each movement can be saved for future use where each record represents up to 1.2 seconds of movement and contains position, velocity, acceleration, and motor current information for each of the three joints.

The mechanics of the joints are completely backdrivable -- the torque produced by the motor plus externally induced torques sum to determine the motion of the joint. Consequently, the motions of each joint are a function of the torques applied to all the joints. This fact, which is generally true of biological limbs, is illustrated in Fig. 3. A step of current applied to the motor which drives joint 2 caused changes in the trajectories of joints 1 and 3. For some manipulators, these interactions may be ignored [15].

Two types of movement

Each manipulator movement may be classified according to the way the translator processes it. Practice movements are those which generate data for the state space memory via inversion (see below). The sources of the commands used to produce practice movements are unrestricted. Test movements are those produced by translation of desired trajectories and they are used to assess performance. The trajectories are converted into sets of motor commands using data from the state space memory. When these commands are issued to the arm a test movement results. In principle a movement can be both test and practice, but for the sake of clarity no such overlap was permitted here.

In order to make use of a practice movement it must be divided into short duration pieces called sections. The duration of a section was chosen to resolve two conflicting factors. The state of the arm, its mechanical properties, and its accelerations are more nearly constant during very short time intervals. On the other hand, longer intervals allow more precise estimates of acceleration because more samples can be used. We examined sections of 40, 50, 60, 80, and 120 msec. in order to find an acceptable compromise. Sixty msec. sections were used for all the data reported here. By superimposing 10 trajectories Fig. 4a shows that the repeatability of typical acceleration measurements is very good, indicating an adequately long estimation interval. Fig. 4b, showing a rather faithful reconstruction of an original movement from its estimated accelerations, substantiates the use of piecewise constant accelerations. Occasionally, accelerations violate this assumption, but they are small

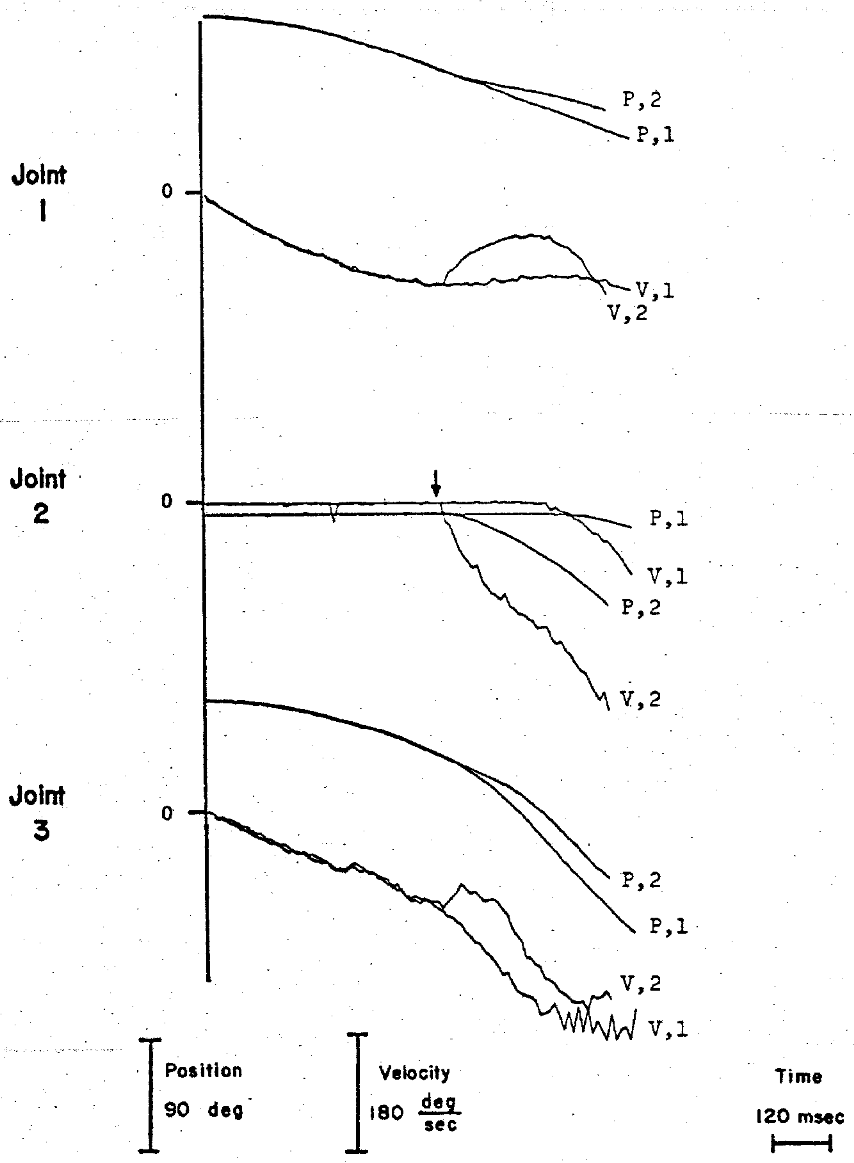


Fig. 3 The manipulator was used to demonstrate the potential for mechanical interactions among joints of a limb. These graphs show how torques applied to one joint of the manipulator influenced the other joints. Each movement labelled *1* was made by applying constant torque to each joint. In movement 2 the torques at joints 1 and 3 were unchanged, but a step was applied to joint 2 after 500 msec. (at arrow). Note that the position and velocity trajectories of all three joints were affected. P-position; V-velocity.

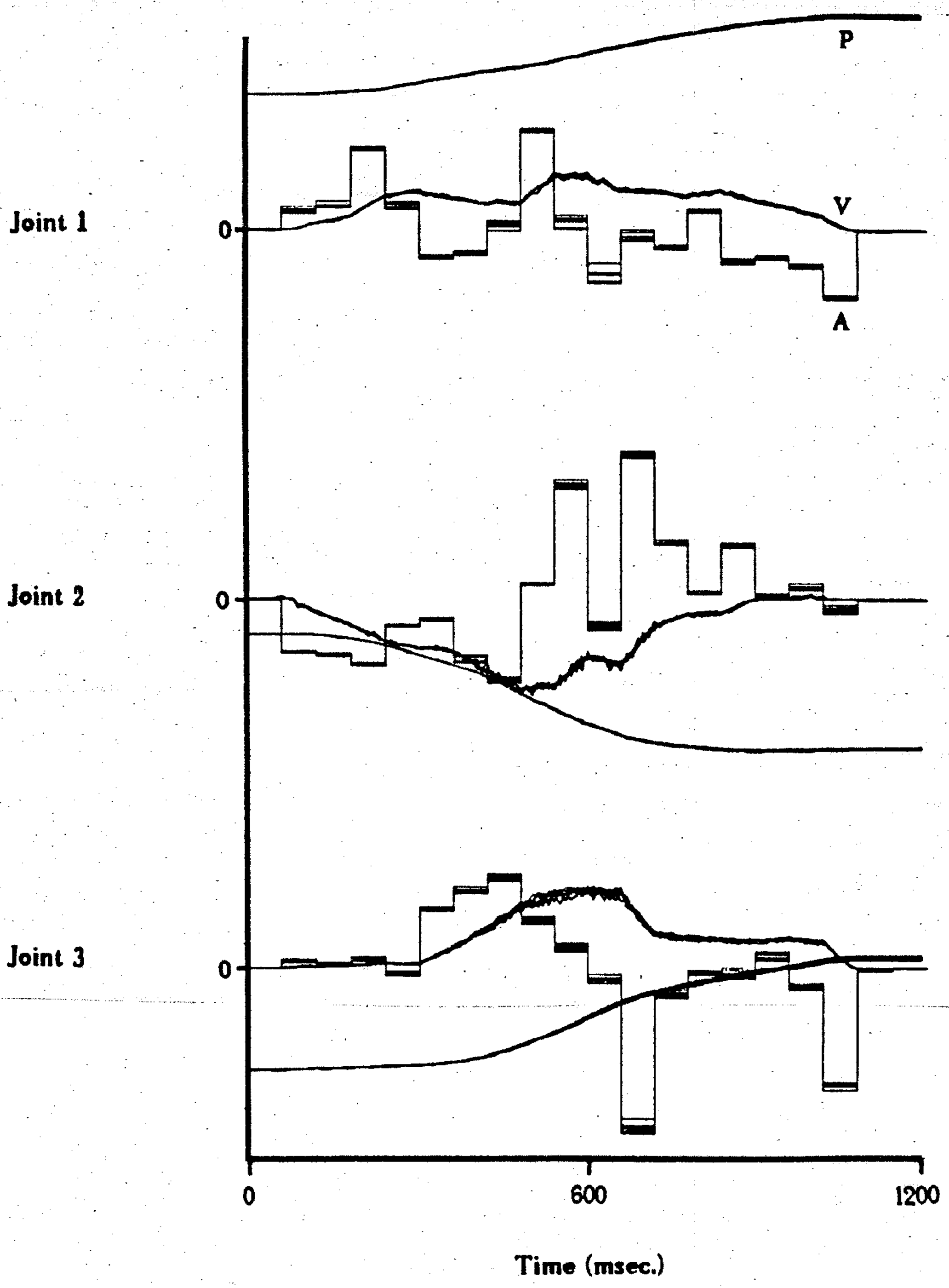


Fig. 4a In order to determine the variability of acceleration estimates, the manipulator executed ten repetitions of the same movement. Most sections only experienced small variations in estimated acceleration, indicating an adequately long sampling interval.

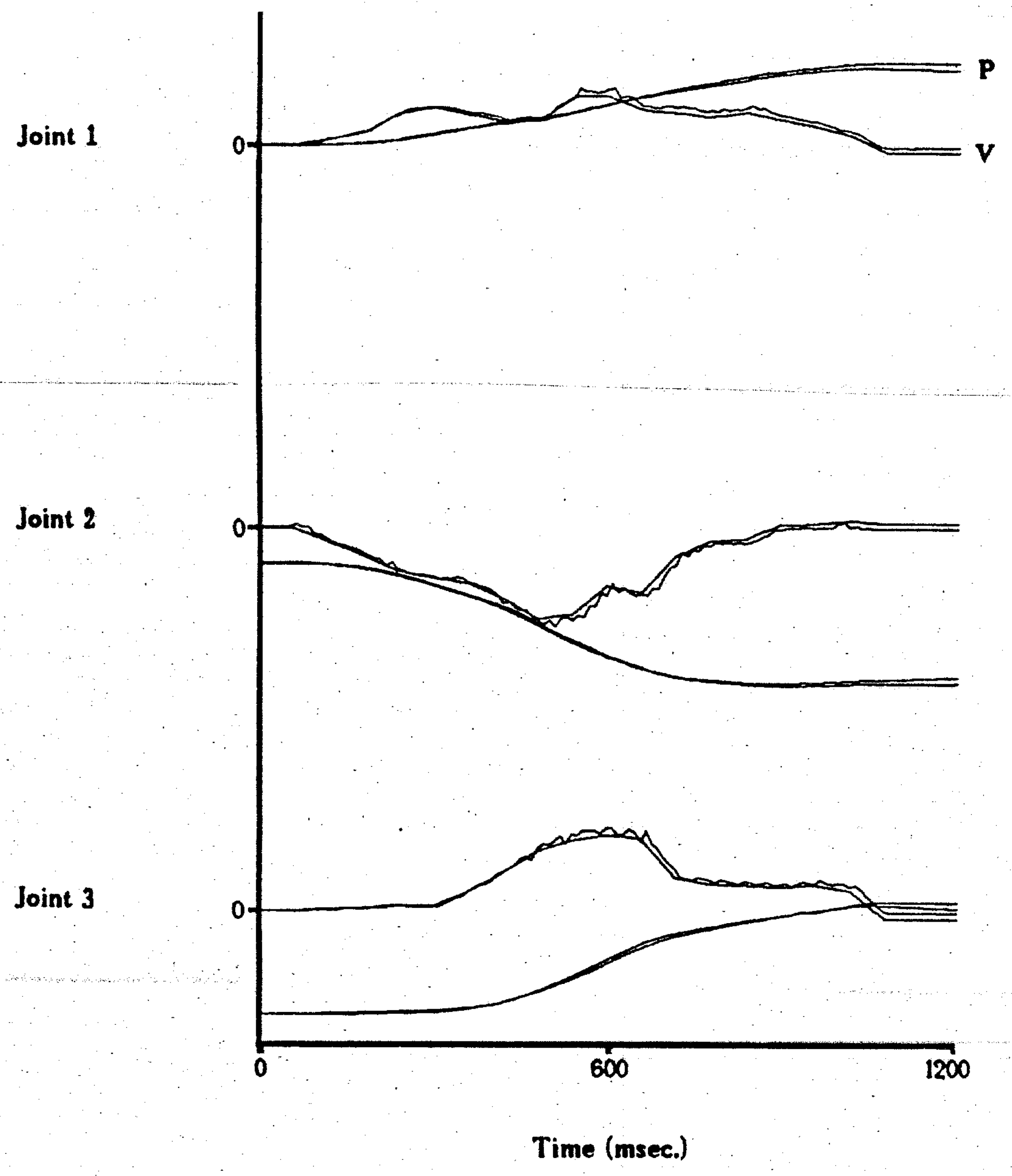


Fig. 4b These curves show that 60 msec. piecewise-constant estimates of acceleration adequately describe a typical movement. During execution of the movement, position, velocity, and estimated acceleration were recorded. The reconstructed position and velocity trajectories were computed by integrating the estimates of acceleration. The comparison between the recorded and reconstructed trajectories is quite good.

in number.

Once a practice movement has been divided into sections, vectors are produced and stored in the temporary buffer. These vectors contain a record of the motor currents, one for each joint, a set of acceleration estimates, and information regarding the state of the limb prevailing during the section. These measurement vectors are collected in the temporary buffer until enough are present to perform an inverse computation.

Computation of the inverse

In order to calculate the constants of mechanical description, which are the data stored in the state space memory, it is necessary to invert an N dimensional matrix of acceleration measurements. (Actually these are acceleration differences taken from N+1 sets of measurements. See Eq. 5. in the Appendix). One may not invert a matrix if it is singular, but N sets of N measurements taken from a physical system are unlikely to be strictly linearly dependent. We must take care that the matrix of acceleration estimates are well conditioned, for inversion of an ill-conditioned matrix amplifies noise [14]. In order to avoid the potentially disastrous effects of inverting an ill-conditioned set of noisy measurements, two precautions are taken.

Each group of vectors are screened before inversion by a conditioning index, x, which determines the degree to which a set of measurement vectors are independent well conditioned:

$$x = \frac{|A'_1 \vdots A'_2 \vdots A'_3|}{\|A'_1\|+\|A'_2\|+\|A'_3\|}$$

where:

$A'_i = A_i - A_4$

A_i is a column vector of dimension $N=3$

$\|A\|$ is the norm of A.

The numerator of this index will be near zero if the matrix, A, is nearly singular. The denominator insures that very large vectors do not make a nearly singular matrix appear to be non-singular. Only sets of measurements that meet a criterion value of the index contribute to the state space memory. When a set of vectors fail the conditioning test, the two least contributory vectors (smallest cross-product) are averaged together and replaced in the temporary buffer.

In theory, the calculations that produce data for the state space memory can be performed when only N+1 measurement vectors have been collected. (See Eq. 5 of the Appendix.) When the computations are performed in this perfectly constrained manner, the effects of noise can be quite large. The resulting inverse rigidly applies to the measurement data, analogous to the way a straight line fits only two data points. More than N+1 measurements can be used to reduce the effects of noise, much the way a straight line fit to more than two data points minimizes the influence of noise present at each point. To perform the computation on more than N+1 measurements we have to use the generalized inverse, since a matrix must be square to be inverted in the usual sense. Using the generalized inverse any number of measurements can be regressed, and the procedure is quite analogous to the line fit mentioned above. This inverse does, in fact, minimize the error of the inverse in the mean square sense.

The value of using more than N+1 measurements is demonstrated in Fig. 5. These histograms were made by generating a set of measurement vectors relevant to one region in state space and inverting them N' at a time, where N' was 4, 6, and 8. The value distributed is one element of the resulting matrix. The figure shows that when more than the minimum number of measurements is used, N'>N+1, the results are more consistent and less subject to extreme variations. Values of N'=12 and 16 were also tested, but the additional computational burden was not justified by the resulting improvements in noise rejection. For this report N'=8. (A discussion of the generalized or pseudo inverse can be found in [1]. The particular algorithm used here, an extension of an orthogonalization method, is given in [20].)

<u>The state space memory</u>

The state space memory is organized into a large number of small regions, each corresponding to a different state of the manipulator. Two factors determine the effective *size* of these hyperregions; the parameter M, the number of divisions along each dimension of the state space, and the range of values each state variable is permitted to assume. For the present implementation with M=10, (M^{2N} or 10^6 defined states (for N=3)), each hyperregion measures $(15 \text{ deg})^3$ by $(25 \frac{\text{deg}}{\text{sec}})^3$. These regions are actually quite small, and the mechanical properties of the arm are fairly constant throughout.

An assumption of the state space model is that memory is initially tabula rasa. But what does that mean? For a neuronal mechanism it might be connections of zero strength, connections of random strengths, no connections, or nothing to do with connections. Here we have distinguished between no data and zero values. The two situations where this question arises, entering new data into the memory and applying data from the memory, are

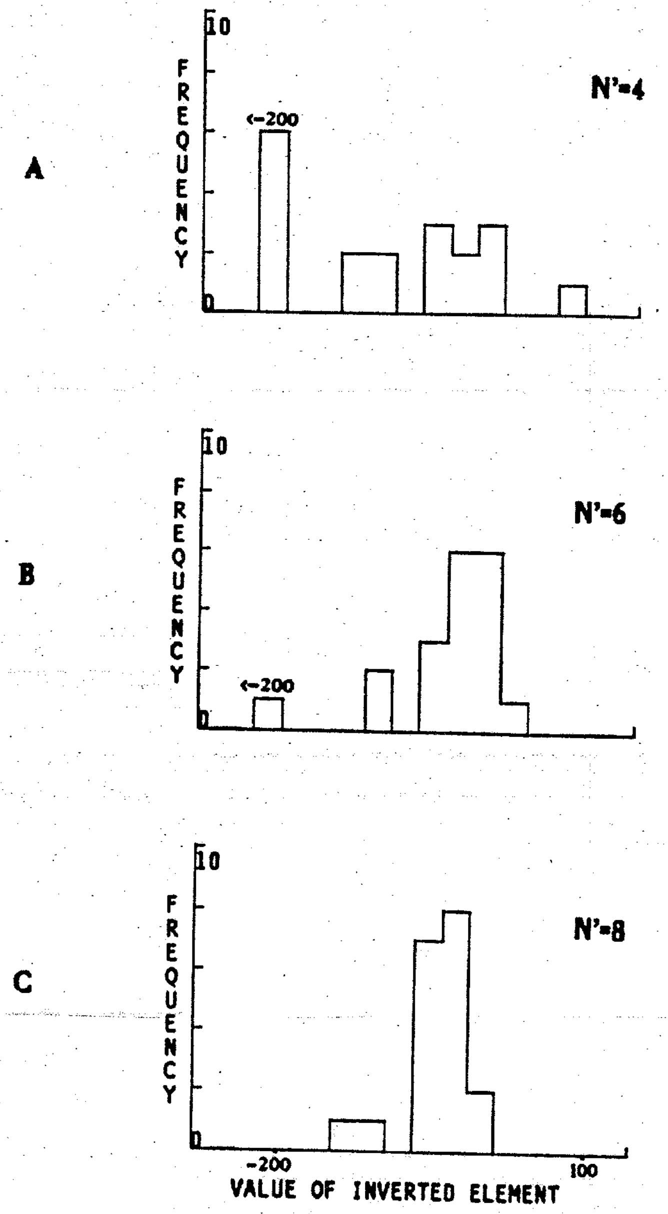


Fig. 5 Use of the generalized inverse with more than N' vectors reduces variability in the resulting computed data. These histograms show the variation in computed values for one element of the J matrix, for different values of the parameter N'. a) N'=4, (ordinary inverse); b) N'=6; c) N'=8, (used for reported data).

treated separately and explained below.

When new data are computed for regions of the memory, they must be stored in combination with data that are already present. Many procedures which combine new and old data will produce adaptive behavior. Without a great deal of consideration it was decided to combine data as follows:

> If a hyperregion has been updated *k* times, where $k<l$ and *l* is a parameter, each new datum is weighted by unity and the old value is weighted by *k* -- the first *l* values are weighted uniformly. When $k>l$ the new value is weighted by unity and the old value by $l-1$.

l should be chosen to give good immunity to noise while providing rapid adjustments to changes in the dynamics of the manipulator. There is a direct tradeoff between these two goals. Fig. 6 demonstrates the time course of the weighting factor for each piece of data stored in the memory for various values of *l*. One can see that small values of *l* give a large weight to new data, and the effectiveness of the data are rather transitory. Larger values of *l* result in small weights, but longer lasting effects. Unless otherwise noted, $l=10$ in this report. The effects of varying *l* are described later. (It should be understood that the reference to time in this context is indirect. Time is only a factor in that more data enter the memory as time passes. Procedures for treating time explicitly are discussed later.)

Use of the state space data or 'Translation'.

When presented with the description of a desired movement, the translator uses the tabular equations of motion with data which describe the mechanics of the limb to produce a set of motor commands. First the desired trajectory is divided into 60 msec. sections, just as is done to practice movements. Using data from the appropriate regions of the state space, the computation defined by Eq. 3 of [17], (see the Appendix), is performed in order to determine a set of motor currents.

What are the *appropriate* regions of the memory? Surely, data from the desired hyperregion are appropriate, but data from nearby states can also be useful. Use of data from neighboring states is justified since the mechanical behavior of our manipulator varies smoothly throughout the state space. Data from these neighbors can be used to advantage whenever the desired hyperregion has never been updated with information about the prevailing mechanical properties of the limb. This situation arises when data generated in the learning of one part of a movement are used to replicate other parts of the same movement, or when a new movement makes use of data originally derived from a separate but similar movement. The distinction between these two cases can not be drawn very

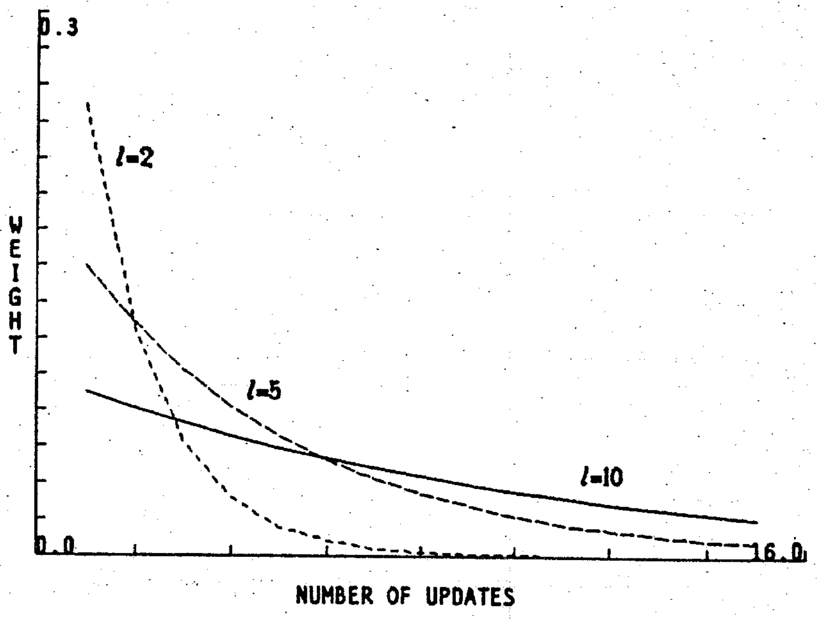


Fig. 6 Each piece of data added to the state space memory is weighted and averaged with older data. As subsequent data are added to the memory the effective weight of an entry is reduced. The magnitude of initial contribution and rate of decay are determined by the paramter *l*. Note that time is not a direct factor, but rate of decay depends on rate of subsequent memory updates.

sharply.

Experimentation with a number of data combination algorithms lead to the following simple and effective choice:

> Each of the desired hyperregions' first order neighbors is accessed. (A first order neighbor differs from the desired region by only one unit on one dimension.) The contents of each are given a weight of one. The contents of the desired hypercube are given a weight equal to the number of times that region has been updated with new information. The weighted average is used.

The range of the *neighborhood function* can have important effects on the generalization behavior of the system. These effects are discussed more fully later.

Practice

The program which generates practice movements is not actually a part of the controller. Since the behavior of the translator during testing is so intimately affected by the details of the practice algorithm, we decided to describe its operation here along with that of the implementation's other components. Once a movement is designated as the desired movement the practice routine takes the following steps:

> On each trial, for each section and joint, the Newton-Raphson method is used to choose a motor current predicted to achieve the desired acceleration. Only the previous two trials are used in making this prediction. Whenever the acceleration errors on the previous trial are within a set of limits for the section for all three joints, the motor currents for that section are not changed.

An example of seven consecutive practice trials are shown in Fig. 7, where the nature of progressive improvements is demonstrated. Since the duration of each practice movement varies, the number of trials of practice does not precisely indicate the amount of data generated for subsequent analysis.

It must be stressed that although the practice program relies on error correction procedures to ensure convergence, the learning displayed by the controller does not rely on error data in any way. The selection of this particular practice algorithm was made to simulate, in a simple way, the short term behavior of humans when practicing. Levine has preliminary data which suggest that a similar iteration method may be used by the cat when learning to make an optimal jump [11].

Originally it was assumed that the details of the practice algorithm would have

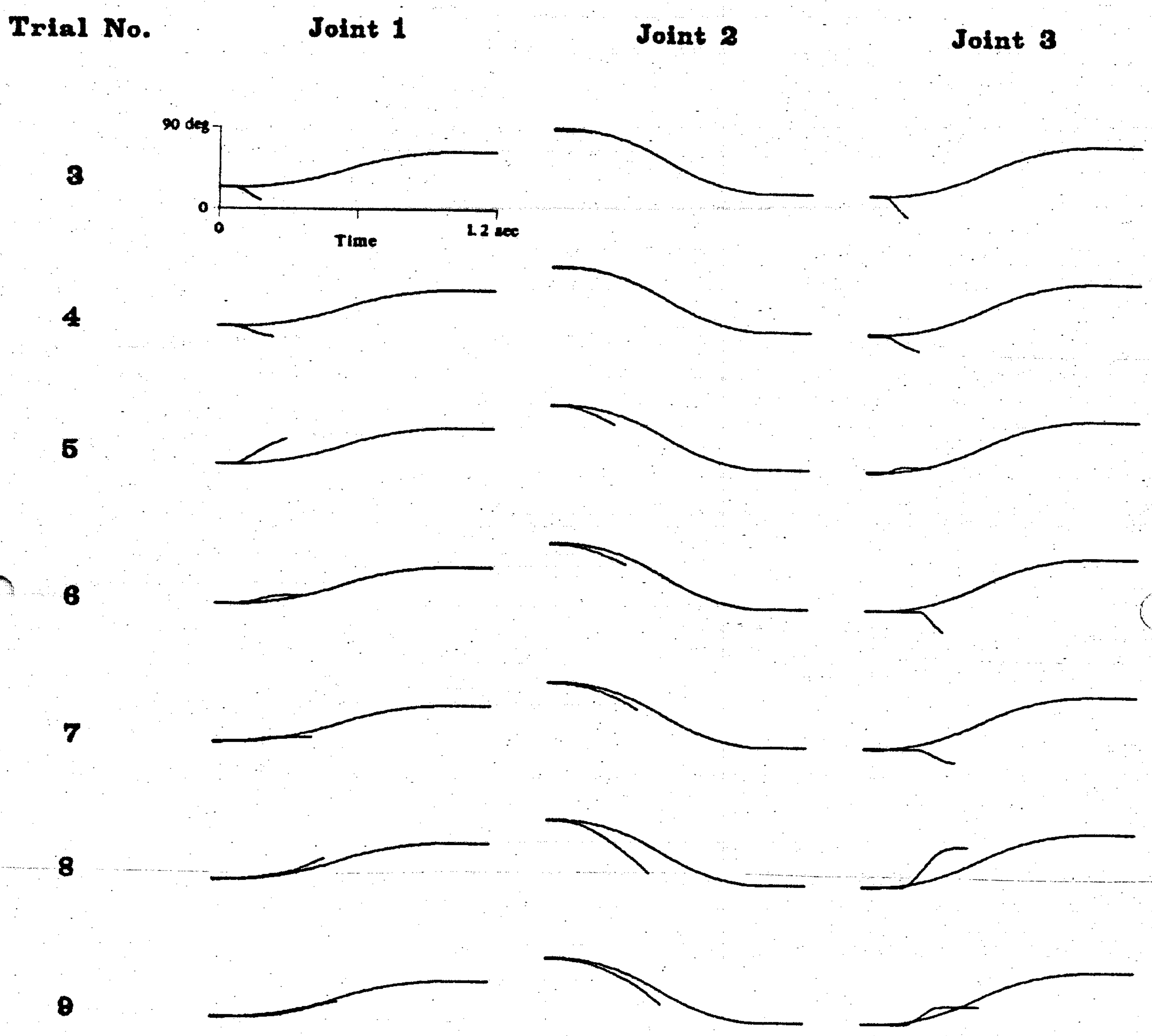


Fig. 7 Seven consecutive trials from a typical practice session are shown. Each curve shows the practice prototype, PR-1, and the attempted movement. At the beginning of each trial the manipulator is servoed to the correct starting position for the prototype. In order to avoid damage to the arm, most practice movements had to be terminated before completion.

little effect on the performance of the system, but this turned out to be quite false. The behavior of the translator depends on how many data are generated, how variable they are, and to which regions of the memory they apply. The rules which govern the former two of these factors are in direct conflict, at least for the simple algorithm used in this project. In order to produce useful data the practice algorithm has to produce movements which vary the output currents independently and by significant amounts. Once the practice algorithm converges upon an acceptable set of accelerations, ('acceptable' means within the acceleration limits; AL), the output currents are not changed for that section. Therefore, the same set of output currents are produced repeatedly -- not good for calculating mechanical constants.

These acceleration limits (AL) do insure, however, that once the correct values for the output currents are found, they are maintained so that subsequent sections can be practiced and processed. When the allowable error for a section is reduced, a larger variety of movements is produced, but those sections late in the movement rarely receive enough attention to produce adequate measurement vectors. The effects of using AL = ±50, 75, 90, and 115 are shown in Fig. 8. Limits of AL=±75 provide a good tradeoff between variety of data and number of sections practiced.

This limiting effect can be overcome to some degree by practicing the movement in parts. A simple servo program moves the arm to the correct initial conditions for a point in the middle of the movement, after which the practice program continues with a normal practice movement. People are known to use such a strategy when they break a complicated movement into parts during learning [5,23]. Fig. 9 shows how this procedure can redistribute the effects of practice which would normally generate data primarily for the initial sections of the practice movement (Fig. 9a). When the servo is used to start the movement, sections in the middle of the movement also receive data (Figs. 9b, c, and d).

The choice of parameters has been described. In summary: Three manipulator joints are used for testing (N=3). Each dimension of the state space memory is divided into ten intervals, resulting in 10^6 hyperregions (M=10). Movements are processed in terms of 60 msec. sections and eight measurements are inverted at a time (N'=8). When new data are stored into the memory they receive a weight of one tenth, and old data receive a weight of nine tenths (*l=10*). When the memory is accessed, data from the desired hyperregion and from the first order neighbors are used in combination. The practice algorithm has been adjusted to generate moderately variable data while remaining near the prototype trajectory (AL=±75).

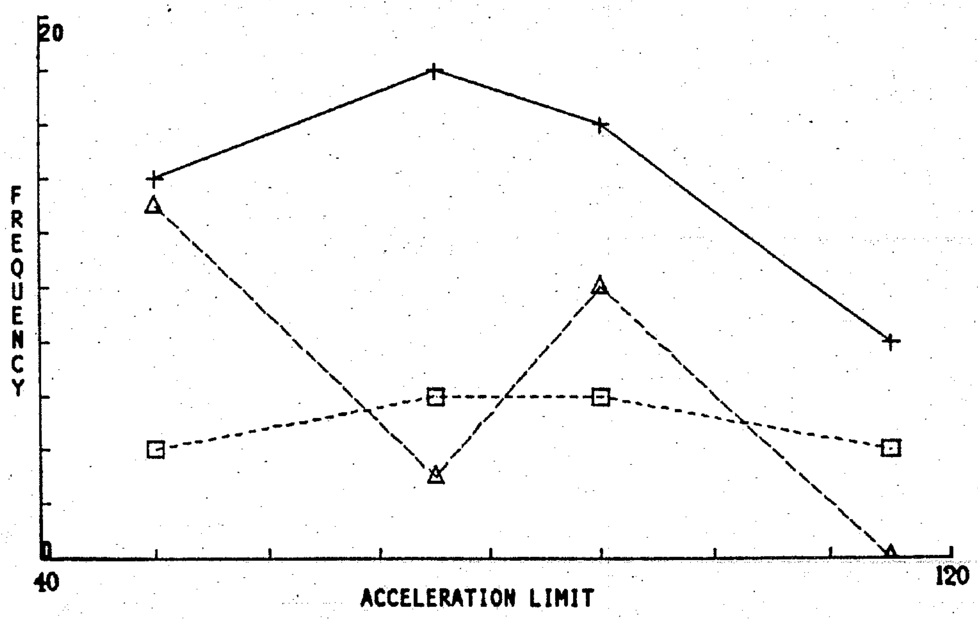


Fig. 8 Manipulating the practice algorithm's acceleration limit affects the algorithm's effectiveness. Three measures are plotted vs. the value of this limit: 1) Number of inverses computed for regions accessed directly by practiced prototype. (+) 2) Number of inverses computed for neighbors of practice prototype. (triangles) 3) Number of prototype sections for which data are provided. (squares) Acceleration limit is ±75 for data in this report.

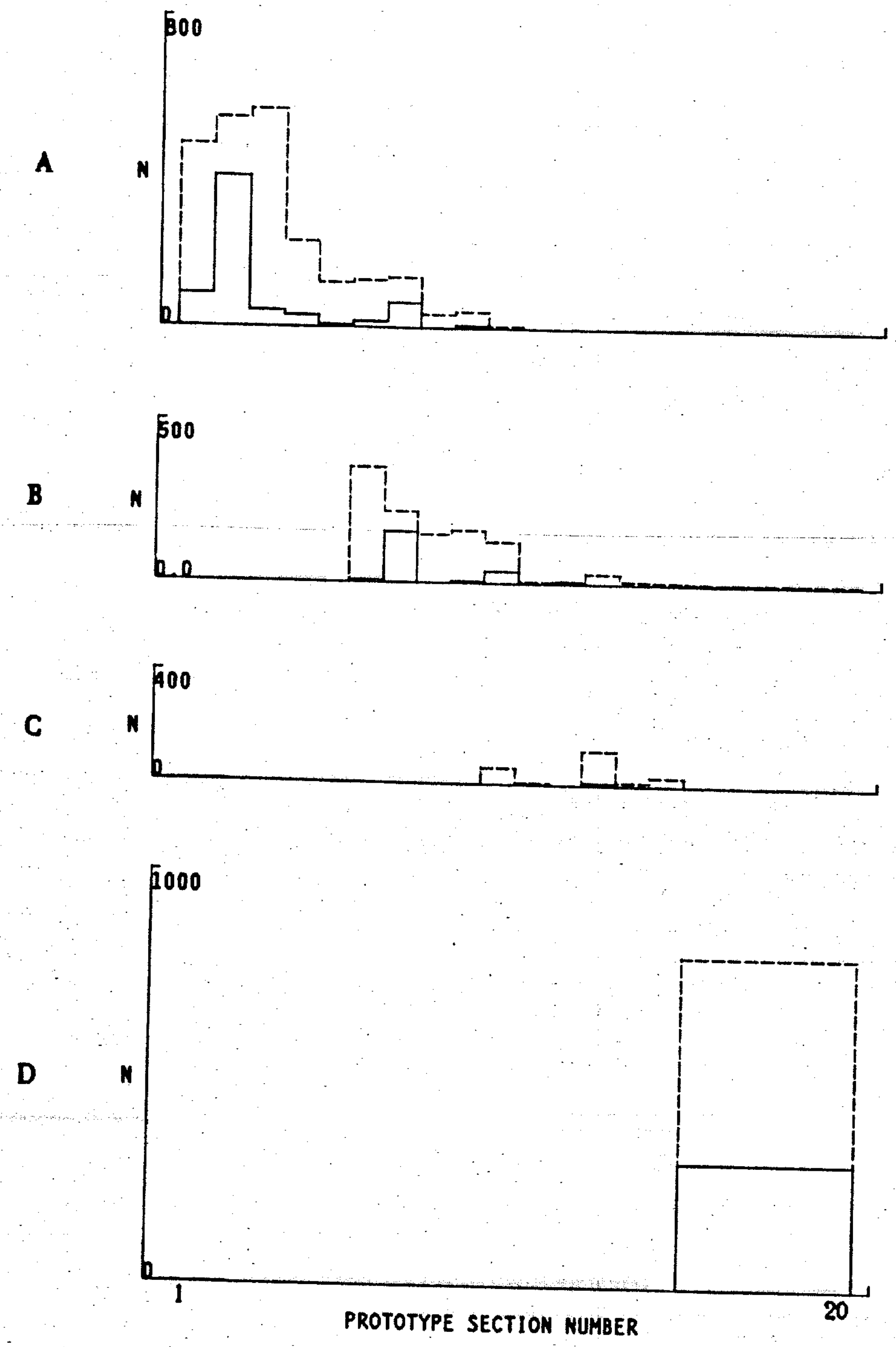


Fig. 9 The effect of changing the starting section for a practice session are shown here. For each histogram 250 practice trials were executed. At the start of each trial a servo routine was used so that the movements could start with section: a) 1, b) 6, c) 11, d) 16. The histograms show how many usable measurement vectors were generated for each section of the practice prototype, PR-1. The solid bars indicate vectors which apply to sections in the prototype, while the open bars indicate vectors which apply to first order neighbors (see text).

Methods

Prototypes

Prototypes are internal representations of ideal movements. They are used to specify desired trajectories to the tranlslator in the production of test movements. They are also used as target movement during practice sessions. Each prototype, is produced in one of three ways:

1) The arm is moved manually by the experimenter while position and velocities are recorded from each joint, and accelerations are estimated.
2) A set of currents are selected by the experimenter and the arm is driven by these currents during which time the positions and velocities are measured, and accelerations are estimated.
3) A set of acceleration trajectories are selected by the experimenter and they are integrated to obtain position and velocity trajectories.

Each of these three methods produces position, velocity, and acceleration trajectories for each of the joints. Method (1) has the advantage that it facilitates the generation of complicated spatial patterns, which are difficult to decompose into the sensory system's coordinates. It is also important because programming industrial robots often makes use of this method. Method (2) has the advantage that the experimenter knows, a priori, what set of currents will produce the movement. This can be useful in conducting tests of competence rather than performance. Method (3) has the advantage that a set of prototype movements can be generated which vary in carefully controlled ways (eg. starting position, final position, maximum velocity, duration, etc.) Although all three techniques were used at some point in this study, most of the prototypes used to generate the data included in this report were produced using method (3).

Using method (3) a set of seven prototypes were generated and used to test the controller's performance. Among these there are two series graded in similarity. The movements in one series share initial and final positions, but differ in duration (and peak velocity). The other series also shares initial position and duration, but the final position is varied. These sets are used to assess the model's ability to generalize among movements. Curves representing one of the prototypes are shown in Fig. 10.

Procedure

During each practice session, the practice program, using one of the prototypes as a

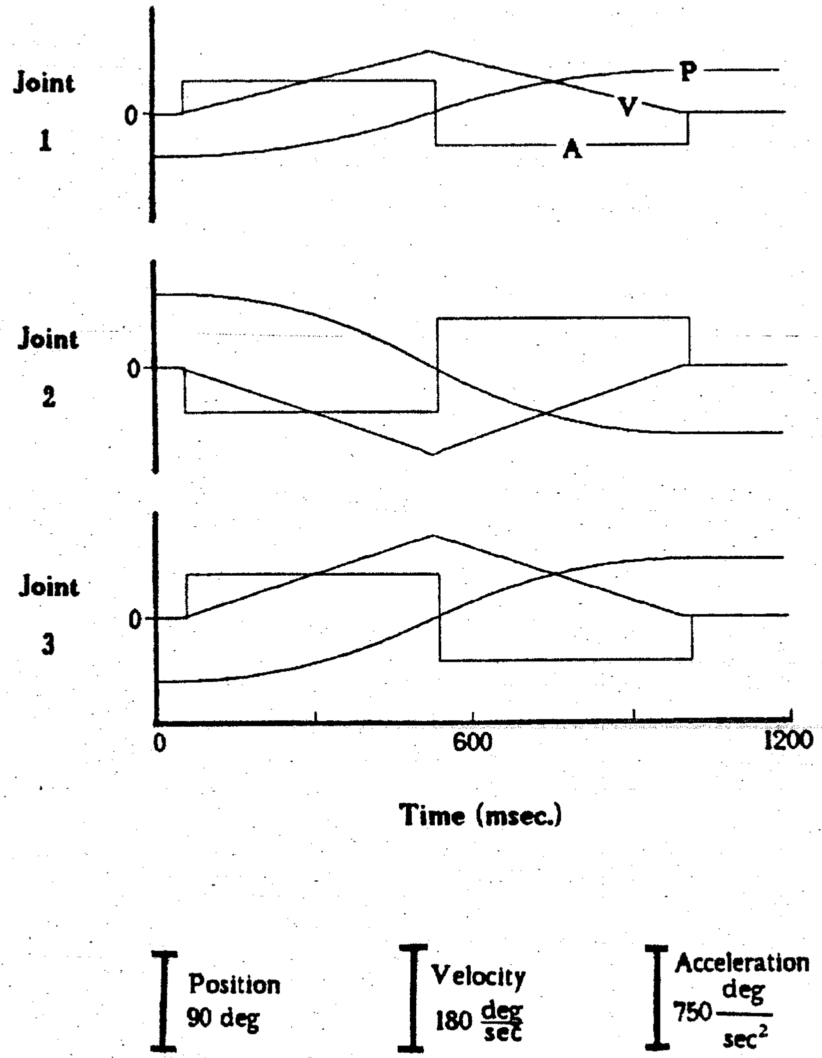


Fig. 10 These curves represent the prototype used for practice in this report, PR-1. It was generated using method (3). P-position, V-velocity, A-acceleration.

goal movement, generates 100 practice movements. At the end of each of these 100-trial blocks, the movements are processed by the translator, creating data which describe the mechanics of the manipulator. The seven prototypes are then used to plan test movements using the tabular equations of motion and the data from the state space memory. After the test movements are executed measurements are made of the performance and competence of the translator. These are used to construct learning curves which plot the values of a performance index as a function of experience.

Performance indices

The behavior of the model is measured by applying performance indices to the test movements. Learning curves are created by plotting the values of one or another of these indices against the number of practice movements made by the system at the time of the test. Each index is applied to an error curve found by comparing the movement to the test prototype. These error curves are only used for analysis and do not effect performance of the system. The indices in use are:

1) Root mean square position error - The mean square position error for each joint and for all three joints is cumulated for the entire trajectory. The square roots of these values are reported. (RMS P)
2) Final position error - The position error at the end of a specified time interval is found for each joint. The total position error is found by taking the square root of the sum of squares of the errors for each joint. Since the joint coordinates are not orthogonal, this total measure is not equivalent to a resultant error in cartesian space. (RMS FP)
3) Root mean square acceleration error - Same as RMS P (1), but acceleration error is found. (RMS A)
4) Root mean square velocity error - Same as RMS P but the velocity error is found. (RMS V)

Prototypes and movement plans have 1.2 sec. duration, but each performance index used in this paper is only applied to the first 500 msec. of the test movement. This was done for practical and theoretical reasons:

1) Many movements made early in learning must be stopped before completion to avoid damage to the manipulator. Therefore, they are shorter than 1.2 sec. This argument does not apply to competence indices.
2) Most of the data produced by the practice algorithm are only useful for

planning the first half the test movements. (See Fig. 11.) The extra investment of time needed to generate practice data for all sections of a prototype seemed unjustified.

3) All available evidence indicates that open loop segments longer than 300 msec. are not necessary for good control and are not found in nature [4,9,13,16].

Competence index

It is useful to distinguish between performance of the system and performance of the manipulator under control of the system. The latter is measured by the performance indices given above, while we feel the former should be assessed by a competence index. Drawing this distinction between competence and performance allows us to ignore *extraneous* factors related to the production of movement not under the influence of the controller. Furthermore, we can evaluate the controller's behavior in terms of variables more closely related to its internal workings. Of course, the only good controller is one which causes production of quality limb movements, but our success in finding such models and modifying existing ones is improved by measuring these internal variables in addition to terminal behavior. After all, do we want to casually reject a controller which produces very nearly the *right* control just because the manipulator behaves poorly under that control? (This can occur, for instance, when the mechanics of the manipulator include discontinuous non-linearities like stiction.)

One added feature of a competence measure is that it can be used to evaluate the entire 1.2 sec. duration of a movement while performance indices must be restricted. We feel that the following index conveys information about the competence of the system to generate motor plans while de-emphasizing the problems of production:

Root mean square motor current error - Same as RMS P but the motor current error is found. This index gives a measure of competence, but can only be used when the currents which will reproduce the prototype trajectory are known. This is normally the case only for prototypes generated from motor-current plans (see section on prototypes), but it was possible to estimate the currents for the prototypes used here. (RMS MC)

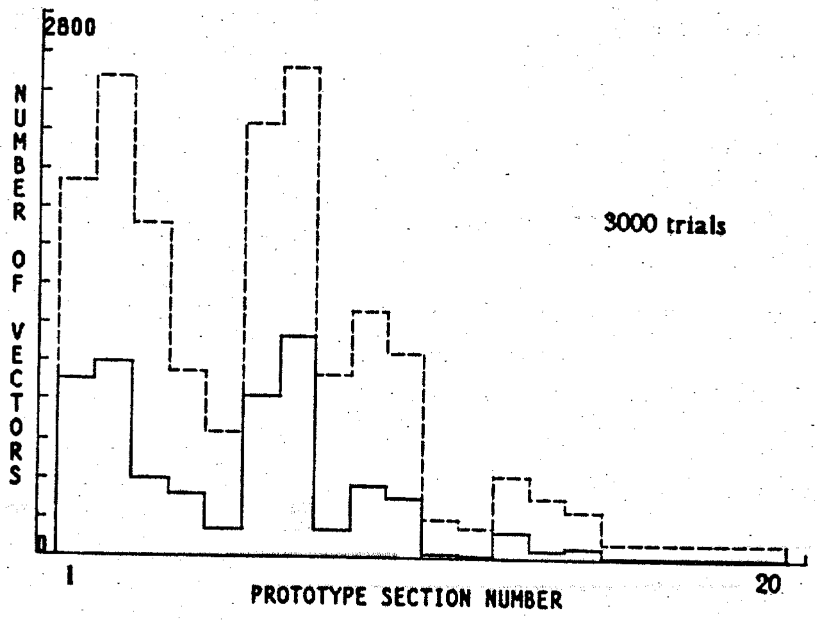


Fig. 11 The measurement vectors produced by 3000 practice trials are shown distributed on the sections of practice prototype PR-1. Note that most of the data apply to the first 10 sections.

Results

Acquisition

Figs. 12a and b show a series of test movements made during a learning session, each separated by 500 practice trials. The prototype which described the desired movement to the translator is also shown. Each succeeding test movement is a better replica of the desired movement than the previous one, and the last movement shown is very similar to the prototype. Most of the remaining error after 3000 practice trials is caused by deviations from the desired trajectory of joint three. Stiction forces in this joint are especially large, and are thought to be responsible for the observed deviations.

These same results are given in quatitative form by Fig. 13. The performance index, root mean square position error (RMS P), was evaluated for each test movement and plotted against the number of practice trials processed by the system. The learning curve shows a rapid initial improvement with subsequent asymptotic behavior. The learning curve shown in Fig. 14a presents a more dramatic example of acquisition. It was produced by testing with prototype PR-2, though the practice was the same as used above.

One apparently peculiar result is that a set of test movements may show that the FP error and RMS P error are converging nicely to small values while the RMS A error shows somewhat disorderly conduct. (See Fig. 14b). Although one might suppose that these two measures are closely linked, some thought shows that a very small error in acceleration near the beginning of a movement may result in a very large final position error. So the acceleration for one section near the beginning of a movement may contribute a great deal to the final position error, and the acceleration for a section near the end may produce almost no effect on position error. On the other hand, there may be no change in acceleration error, or perhaps even a net improvement, while the final and RMS P errors have become quite large.

Fig. 14c shows that during the practice session the motor currents planned by the translator approach those which are known to produce the desired movement. The competence index, (RMS MC), was used to produce this learning curve.

These figures provide substantiation for our basic claim; the state space model can acquire control of a limb-like mechanical device by processing data produced by movements of that device, without the use of error information.

Generalization

In addition to learning to perform new movements when they are practiced, we have

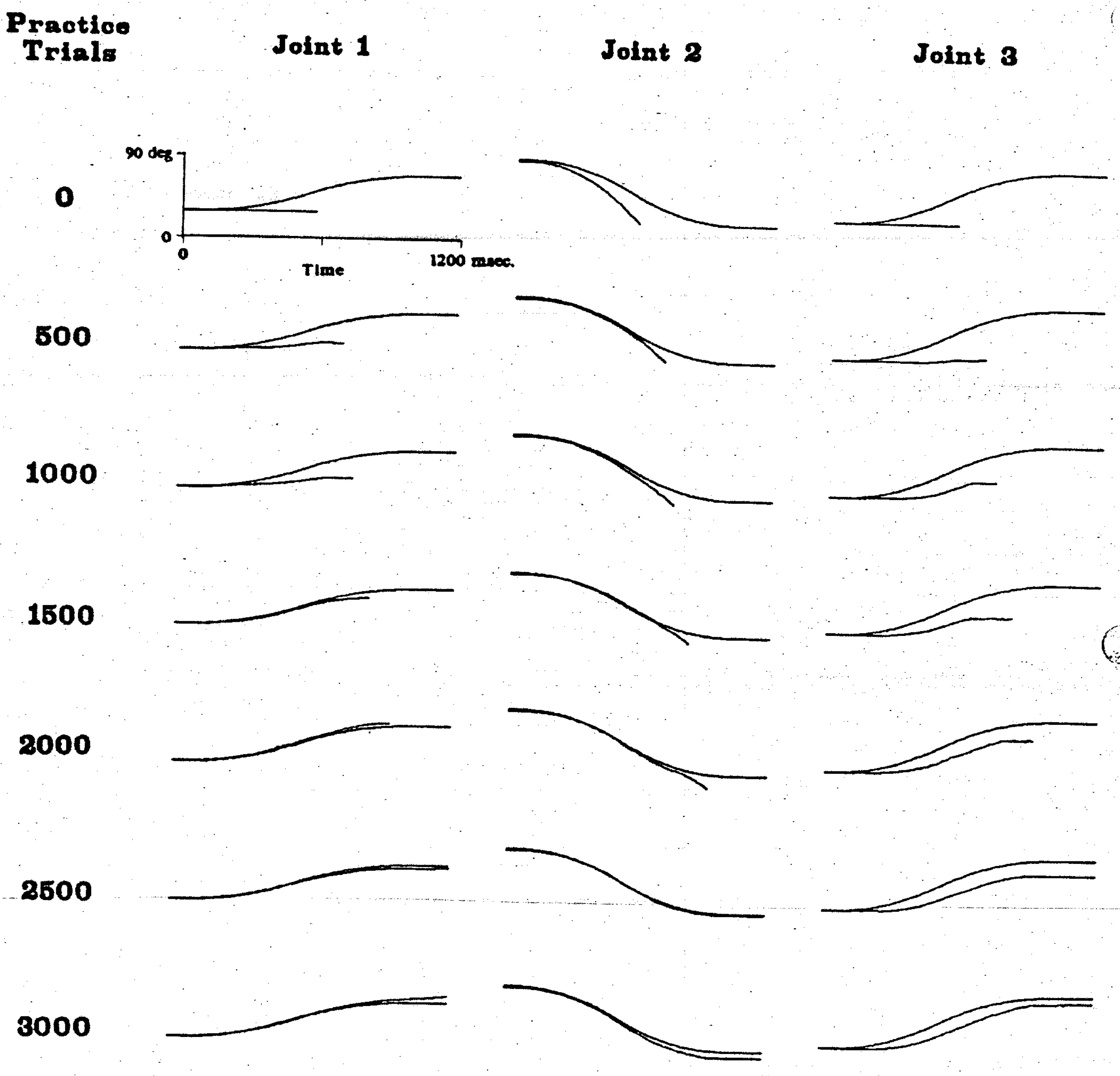


Fig. 12a Seven attempts to replicate prototype PR-1 are shown. Each is separated by 500 practice trials and is plotted along with the desired trajectory. (Position trajectories.)

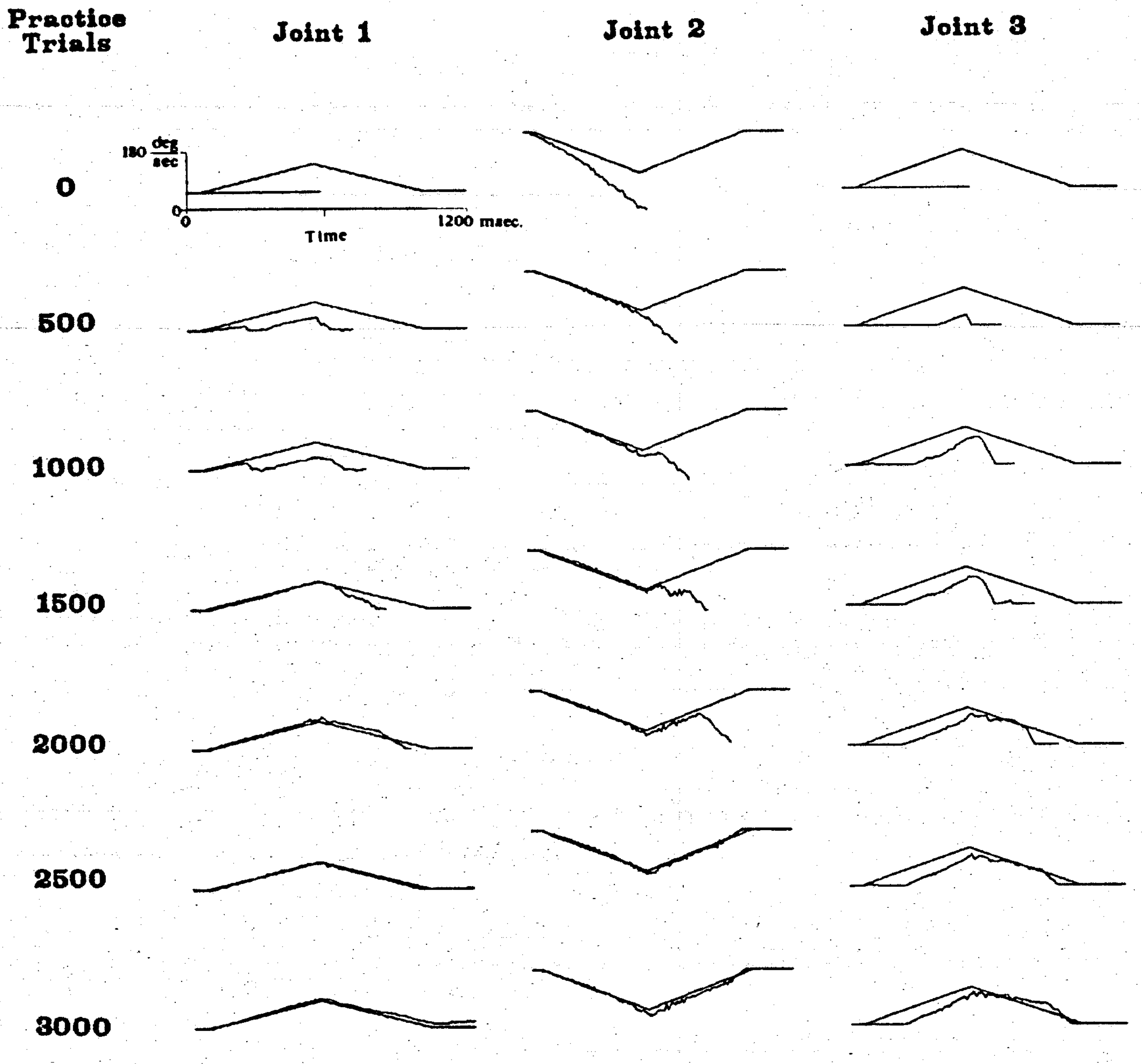


Fig. 12b Same movements as in Fig. 12a but velocities are plotted. The effects of stiction in joint 3 are apparent.

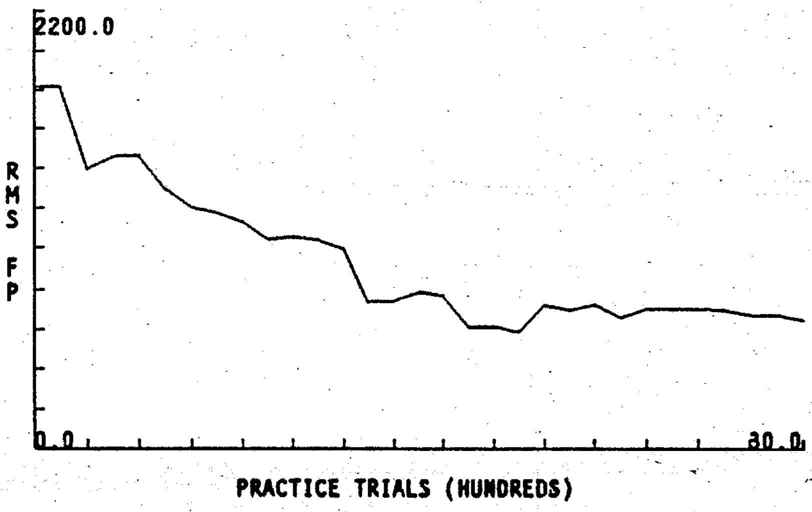
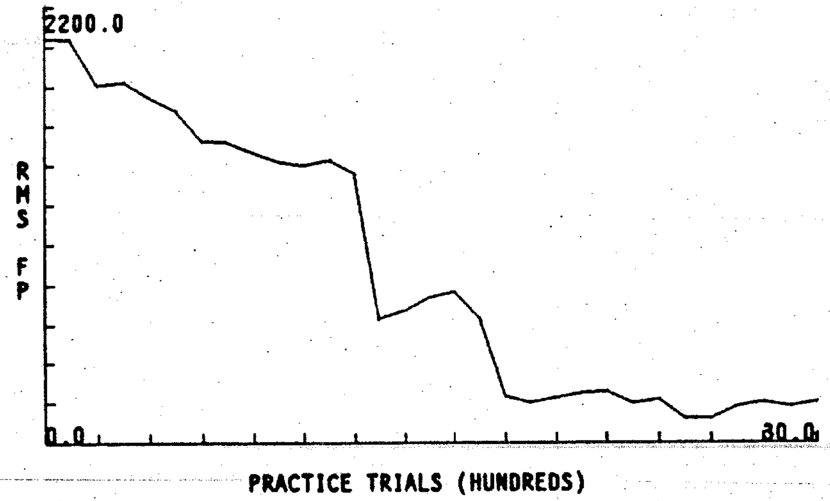


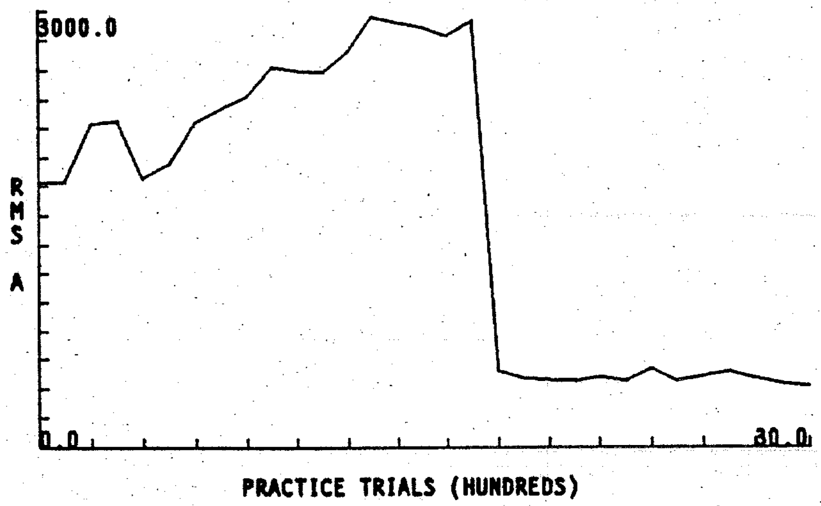
Fig. 13 After every 100 practice trials a set of test movements are produced and the performance indices are applied. This learning curve shows the performance index, RMS FP plotted against trials of practice. These data also appear in Fig. 12.

Fig. 14 (Next page) These learning curves were produced by practicing PR-1, and testing performance of PR-2. Two performance indices and the competence index were used. a) RMS FP, performance. b) RMS A; Note the irregular, non-monotonic behavior. c) RMS MC, competence.

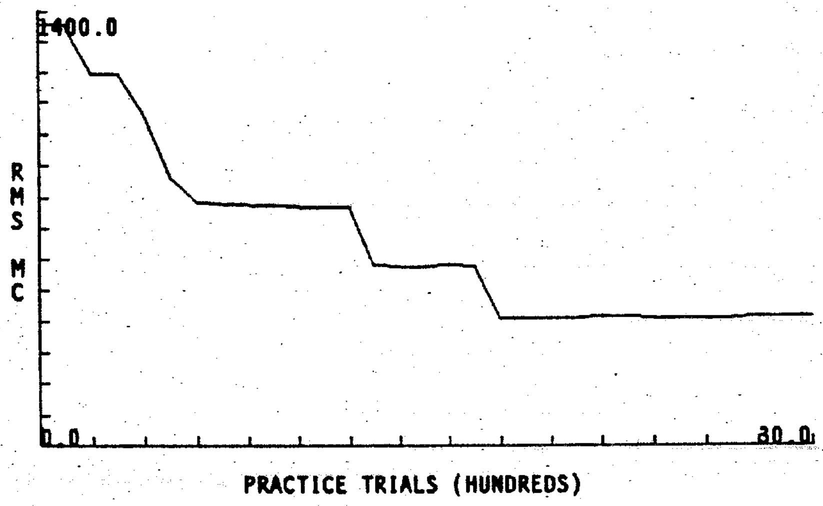
A


2200.0
R M S F P
0.0
30.0
PRACTICE TRIALS (HUNDREDS)

B


3000.0
R M S A
0.0
30.0
PRACTICE TRIALS (HUNDREDS)

C


1400.0
R M S M C
0.0
30.0
PRACTICE TRIALS (HUNDREDS)

claimed that the state space model will generalize from practiced movements to movements which have never been practiced, provided they are sufficiently similar. So far, this claim has been difficult to verify convincingly. Both technical and substantive consideration have contributed to this situation.

Fig. 15a shows the learning curves for a set of test movements. The movements in this set were chosen to be graded in similarity to the practice prototype, PR-1. Each had the same initial and final positions, but varied in duration. These curves show how performance was improved when PR-1 was practiced, though the amount of improvement for each prototype is variable. Fig. 15b shows the same data after normalization. Here, each point on each curve gives the percent improvement up to that point in the learning session. By a strict interpretation of the definition of generalization, these data qualify, but they are peculiar in certain respects. Normally, we would expect a gradual deterioration in performance as the test movements become more and more different from the practice movement. Such a systematic variation was not found among these performance curves. Examination of learning curves for the competence index applied to these same data reveals more orderly behavior. (See Fig. 15c.) The highest level of competence after 3000 practice trials was exhibited by the practice prototype, though the differences between prototypes was fairly small.

These differences, between performance and competence, reveal the presence of effects related to the learnability and replicability of individual movements. The lack of large differences between pairs of prototypes for performance measures and competence measures, indicates a poor choice of test movements. We postpone a more complete discussion of these results to a later section of this report.

Adaptation

We claim that the model will adapt its motor commands to compensate for changes in the mechanics of the manipulator or limb. Fig. 16 illustrates how the arm is modified to test this property. In one case (Fig. 16a) we attach a .75 kg weight to link 3 of the arm to increase the moments of inertia for all links and the effect of gravity on links 2 and 3. In the other case a spring, having a constant of 1.85 kg/m, is attached between link 2 and ground. (Fig. 16b.)

Our general finding is that application of a mechanical load causes a temporary disruption of motor control, but control is restored after practice with the new mechanical situation. This result is demonstrated by the data shown in Fig. 17. These curves were produced by establishing a 3000 practice trial baseline upon which the effects of

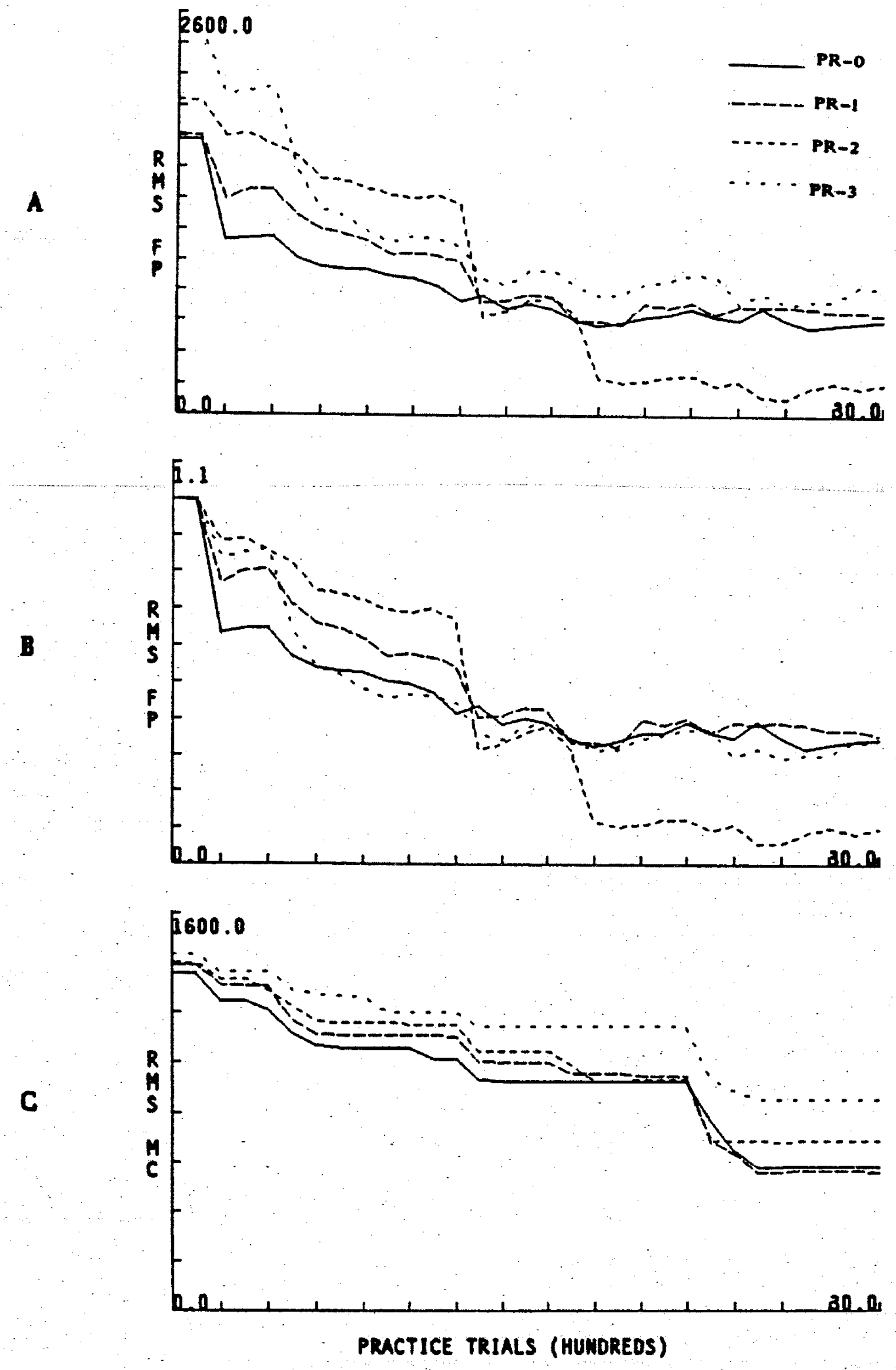
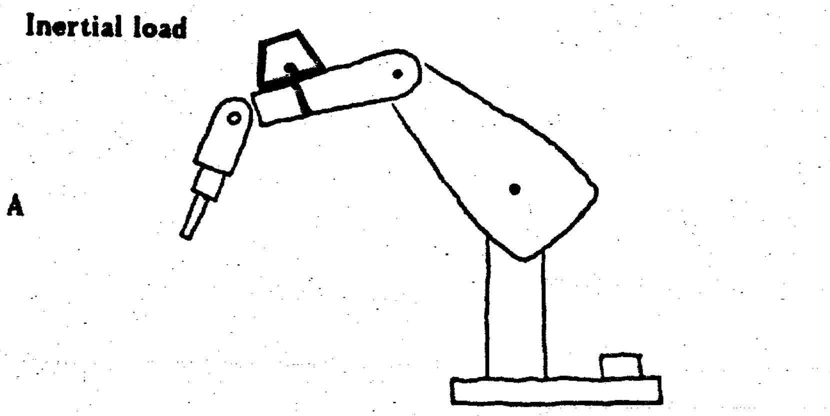


Fig. 15 The effects of generalization are determined by examining learning curves for movements different from the practiced prototype. a) These performance curves do not vary systematically across prototypes. b) The data for each curve are normalized so that percent improvement can be readily determined. Again, no systematic variations are apparent. c) Learning curves for the competence index reveal a gradient of competence in moving away from the practiced prototype.



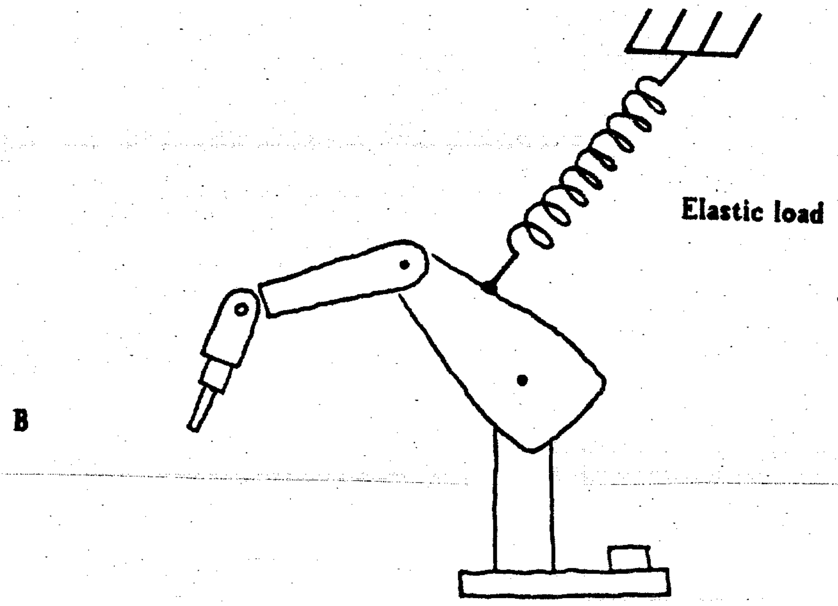


Fig. 16 Two methods of applying loads in order to disturb the manipulator's behavior are shown. a) A .19 kg. weight is attached to the third link of the manipulator. b) A 1.85 kg/m spring is attached from the second link to 'ground'. When movements start the spring is stretched .83m and runs from coordinates (.17,.00,.25) to (.02,.7,1.20) in meters; see Fig. 2.

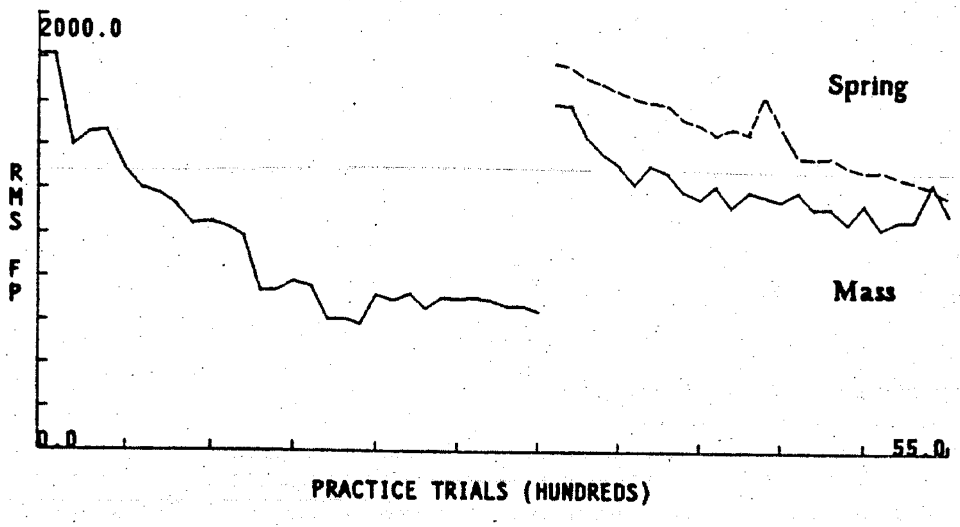


Fig. 17 Adaptation to two types of load are shown. First, a 3000 trial baseline is established. After practice trial 3000 the load is applied and the time course of adaptation is recorded. l=10.

disturbances are assessed. The figure shows that both types of load cause a large increase in error which is subsequently reduced. Although these results satisfy our minimal expectations for adaptation, we experimented with manipulations designed to improve the rate of adaptation. Two factors might be responsible for retarding adaptation:

One factor arises because measurement vectors remain in the temporary buffer until they are used in an inversion. Therefore, data generated during the period following application of a mechanical disturbance are likely to result from computations based on combinations of measurements taken before and after application of the load. The constants obtained from these *interim* calculations may attain values which are quite different from either pre- or post-adaptation values -- they do not necessarily attain intermediate values. Fig. 18 demonstrates this counter-intuitive effect.

Eight measurement vectors were recorded in each of two different states. For each state a different set of mechanical conditions prevailed. As the ratio of number-of-vectors-from-state-A to number -from-B changes monotonically, the data produced by inversion vary non-monotoniclly. If one were <u>averaging</u> data from two groups, however, the transition would be monotonic. To assess the effects of these interim calculations, an adaptation test was conducted in which all data from the temporary buffer were removed when the load was applied. The heavier dotted line in Fig. 19 shows that this procedure produces no clear improvement in rate of adaptation.

The rate of adaptation is also retarded when old and new data are combined in the state space memory by averaging. This factor can be adjusted by reducing the value of l, the averaging parameter discussed earlier. (See Fig. 6.) The results of such a manipulation are shown by the curves in Figs. 20a and b. Each successively smaller value of l results in more rapid adaptation to the mechanical disturbance imposed by the spring. While reduction of l improves adaptation rate, it may reduce the system's resistance to the effects of noisy measurements. (Data presently available do not substantiate this point, but it is strongly expected to be true based on our understanding of the model's operation.)

While reductions in l decrease the effects of old state space memory data, they do not eliminate them. An experiment was done in which all previous state space data were eliminated at the time the load was applied. The temporary buffer was also zeroed. The lightly dotted curve in Fig. 19 reveals an extremely rapid adaptation. Unfortunately, this improvement in adaptation rate is accompanied by an initial loss of control. The level of performance following application of the load was temporarily lower than that initially achieved when the memories are left intact (solid curve in Fig. 19). A more severe loss of control resulting from initialization of the memories is shown in Fig. 21.

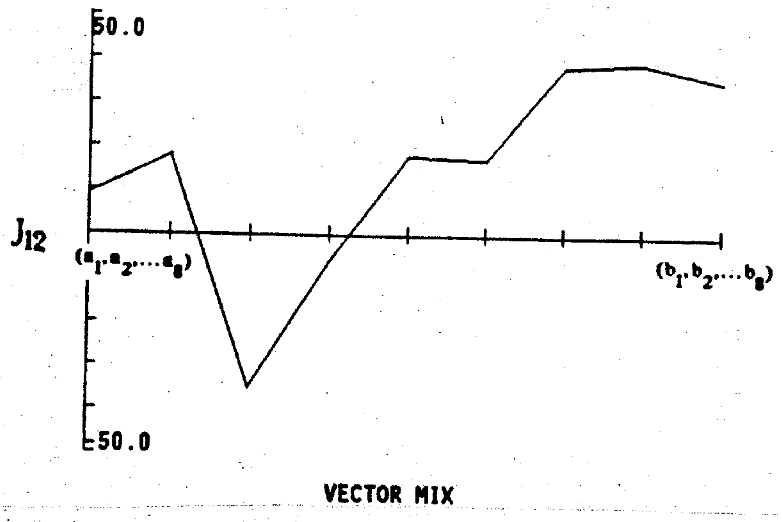


Fig. 18 A demonstration of the deleterious effects of 'mixing' measurement vectors when inverting. Two sets of vectors were used, (sets A and B), each consisting of 8 vectors generated for a mechanical situation. Nine inversions were performed where the vectors contributing to each inversion were:$(a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8)$, $(a_1,a_2,a_3,a_4,a_5,a_6,a_7,b_8)$, $(a_1,a_2,a_3,a_4,a_5,a_6,b_7,b_8)$, . . . $(b_1,b_2,b_3,b_4,b_5,b_6,b_7,b_8)$. The value plotted on the ordinate is one element of the resulting J matrix.

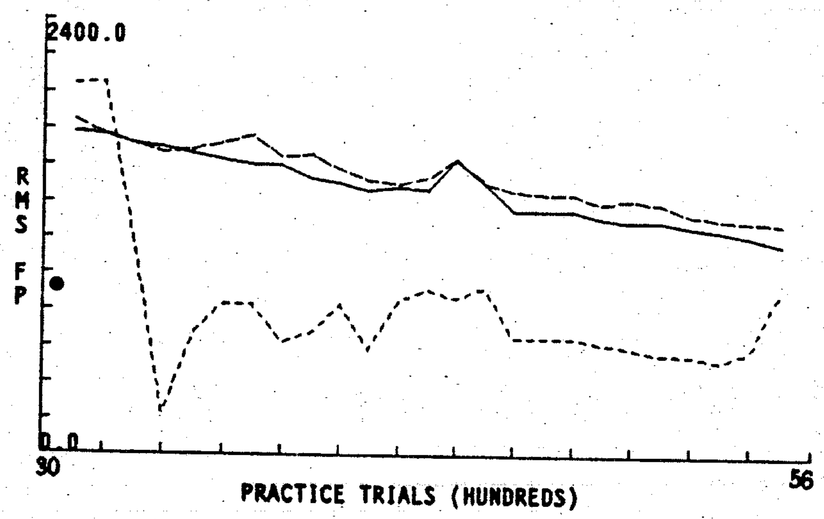
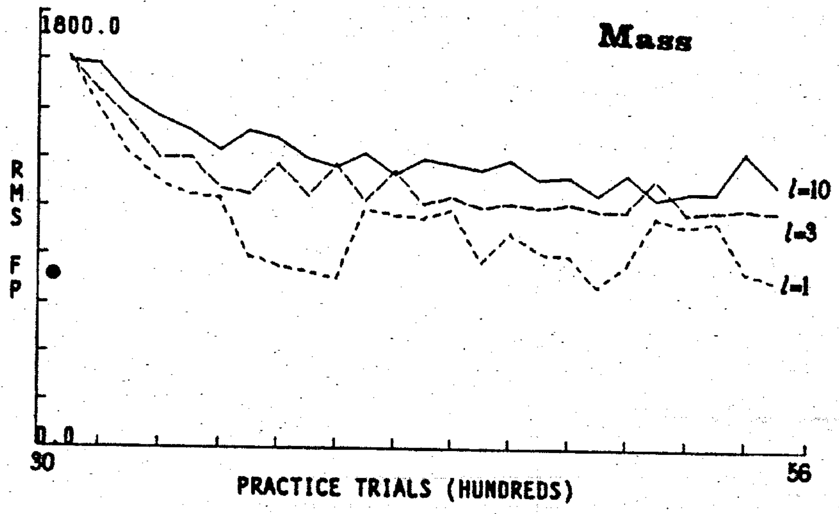


Fig. 19 Two manipulations were performed to improve the rate of adaptation. The solid curve is reproduced from Fig. 17 for comparison (l=10, no memory manipulations). The heavy dotted curve was produced by removing all measurement vectors from the temporary buffer when the load is applied. The light dotted line was produced by removing all data from both the temporary and the state space memories. (PR-1); (See text.) Closed circle indicates pre-adaptation level.



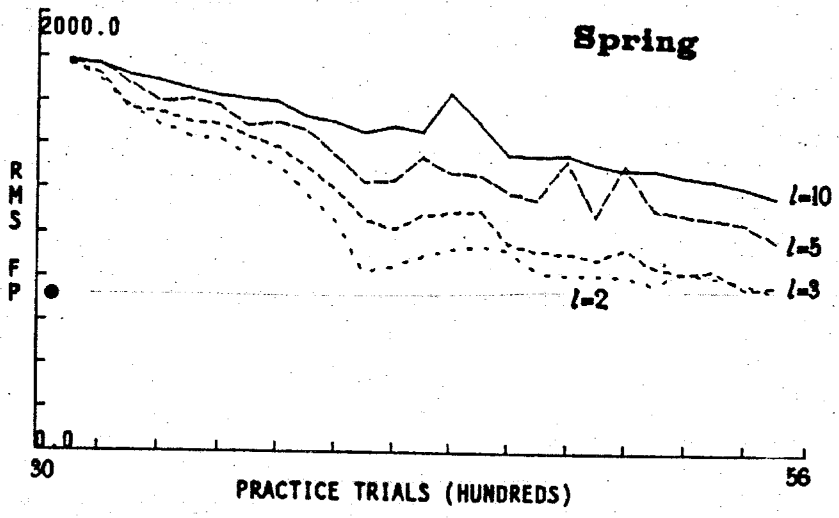


Fig. 20 The averaging parameter, *l* is systematically varied. Smaller values of *l* yield more rapid, but noisier adaptations. a) inertial load; b) spring load; (PR-1). Closed circle indicates pre-adaptation level.

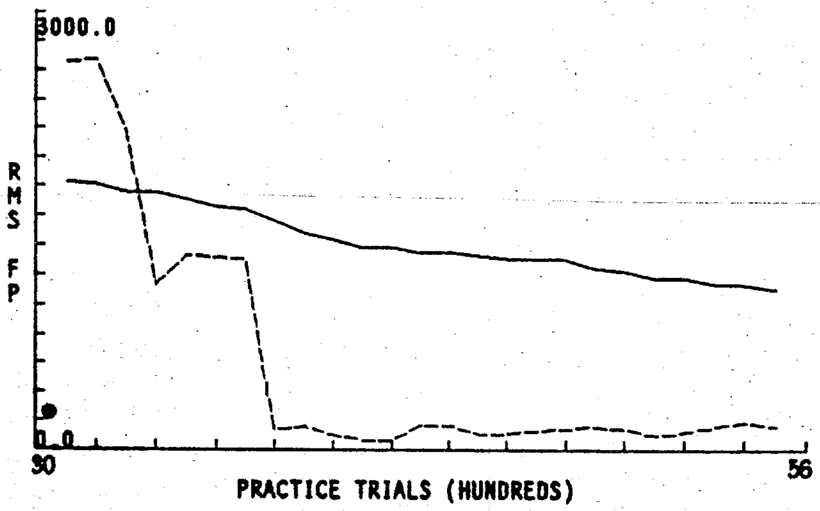


Fig. 21 When the state space memory is initialized in order to improve the adaptation rate, a loss of control may insue. The solid line was produced with l=10 and no memory manipulations. l was not changed for the dotted curve, but the state space memory and the temporary buffer were zeroed when the spring load was applied. Adaptation is made more rapid, but only after an initial period of poor performance. Closed circle indicates pre-adaptation level.

Disoussion

The acquisition and adaptation data presented above substantiate our basic claims: A controller using an internal inverse dynamic model can translate desired trajectories into motor plans. The analysis of measurements made during practice movements makes translation possible and allows the controller to compensate for changes in the dynamic properties of the manipulator. Error correction and iteration are not necessary if the tabular form of the equations of motion are used wi:h a state space memory.

Generalization

The tests used to demonstrate generalization were not so successful. Indeed, practicing one movement facilitated performance of others, but we expected a gradual deterioration of performance as the test movements became more and more different from the practice movement. Along the two dimensions tested, performance did not vary systematically. (See Fig. 15.)

Perhaps we need to clarify why we are unhappy with results which indicate that the controller does not generalize poorly among dissimilar movements. After all, we are looking for efficient solutions for learning, and a controller which performs well with little practice is desirable. There are two very general principles which govern such solutions. A controller should:

1) Maximize the use of available data by providing access to them whenever possible.
2) Minimize the misuse of data by restricting access to them whenever necessary.

Since the dynamic behavior of the manipulator varies smoothly throughout state space, data generated for one region of the state space memory can be made available when planning movements through other, nearby parts of the space. But this 'sharing' of data cannot go too far or the state dependent variations in mechanical properties will lead to the generation of very poor trajectories.

Each of these principles should induce a generalization gradient in the controller's behavior. The first because inappropriate use of available data will produce bad trajectories. The second because unavailability of data will produce bad trajectories. For these reasons our data must show a gradient in performance in order to verify our analysis of generalization.

Factors closely related to these principles may be held responsible for the lack of observed regularity. One such factor caused each trajectory to be replicated with roughly equal precision. (Eg. replication of prototype PR-3 was as good as PR-1.) Since the range of movements chosen for the tests was relatively small, none were sufficiently different from the practice movement to involve substantially changed mechanical behavior. Nor were much data required from unfilled regions of the state space memory. This problem may be solved by selecting test movements which span a larger range of movement space.

Another factor, related to the individual characteristics of certain trajectories, caused some to be replicated much more faithfully than the practice movement. (Eg. replication of prototype PR-2 was better than PR-1.) A movement may be *easier* or *harder* to replicate for a number of reasons. It may have fewer low velocity components, require more data from the memory, bear a particular relationship to the practice algorithm, etc.

Experiments carefully designed to eliminate both of these problems are proposed in [18].

The amount of generalization exhibited by the controller is largely determined by the range of the neighborhood function and the map that determines the discretization of the state space memory. But these parameters should be chosen with some knowledge of the behavior of the manipulator. That does not mean that a naive controller must have a priori knowledge of correct values for these parameters, but *optimum* choices must be postponed until some experience with the plant variables is gained. This issue has not received systematic attention here. The neighborhood function and state space memory map were chosen and adjusted to give good performance. It is my opinion, however, that simple mechanisms exist which will perform these selections automatically. Furthermore, such automatic selection could design memory maps and neighborhood functions which compensate for the rate at which the limb's properties vary with state.

Another difficulty in studying generalization is the lack of a good, general classification scheme for movement. As a consequence, the concept of 'similar movements', necessary for a precise study of generalization, is not well developed. Each pair of movements can be easily classified if we are willing to limit our consideration to a particular model or theory, but the results are often quite unappealing. (See, for example, table 1.)

Distributed vs. massed trials

In a normal practice session, measurements vectors from temporally adjacent practice trials are often similar. The beginning of a practice block, however, is unlikely to include vectors similar to those at the end of the previous block. Since the invertibility index screens and combines sets of linearly dependent vectors, one would expect more inverses to be found, (more learning), near the beginning of a practice block than elsewhere. Taking this factor into account, one might expect more learning when 100 practice trials are broken down into 5 blocks of 20, than when they are practiced in one large block. This is reminiscent of behavior observed in humans and other animals: Learning is more efficient when trials are distributed in time than when long sessions of practice are employed [4,5,21,23].

A use for error data

It was shown that the rate of adaptation to mechanical disturbance was increased when outdated data were removed from the memory. (Fig. 19) Unlike reducing the value of *l*, however, this manipulation requires information indicative of the data's obsolescence. That information was provided by the experimenter for the tests described above, but a control system could provide those data for itself in a number of ways. For instance someday, high-level processes might visually ascertain that a coil shape object was now connected between arm and ceiling. Using its data base it could determine that such a device was probably a spring and would probably change the mechanical properties of the limb. . . Alternatively, a simple mechanism which merely examines error data could quickly determine a loss of control.

It is interesting to postulate a system that uses error data to determine that something went wrong, and measurement data to find out what went wrong. The expected behavior of such a system, rapid adaptation when error information is provided and moderately rapid adaptation when it is absent, is in agreement with experiments from the psychological literature [16]. This combination of feedback and feedforward may prove to be a powerful concept for future models of adaptation.

The translator and optimal control

The theory of optimal control describes how trajectories may be chosen to satisfy a set of movement constraints, while minimizing a cost function for a particular mechanical system [4]. The constraints might specify, for example, initial and final positions and execution time, while the cost function provides penalties for, say, errors in position, time of arrival, and expenditure of energy (the last of which is minimized by humans during at least one motor activity [19]).

We have supposed that the functions of motor control are divided into: High-level processes which plan trajectories without considering the mechanics of the motor apparatus, and low-level mechanisms which translate these trajectories into commands understood by the limb. Since the optimization process generates trajectories, one is inclined to include it with the high-level mechanisms mentioned. When variables related to the manipulator enter the cost function, however, as they do when energy is conserved, the optimization process threatens the presumed dichotomy between high- and low-level functions. In order to optimize energy consumption, the process which generates trajectories must know which motor commands will be required for production, and that is the business of the low-level translator.

This apparent merging of high- and low-level functions is avoided if the optimization process gets its information about energy costs, not from the translator, but from another source which remembers the measured costs associated with previous movements.

A fair test of the model?

The ranges of certain variables have to be limited to satisfy technical considerations. For instance, all velocities have to be below a maximum. When very large velocities are allowed the current/force relationship for the DC motors is no longer valid. This restriction is especially annoying, because some of the more important properties of the control system are most clearly exhibited at high velocities. For obvious reasons, the value of M, the state space quantizing factor, also has to be restricted, thereby reducing the attainable precision of control.

Many combinations of the model's parameters are possible. Limited time resources forced us to choose relatively few combinations for experimental investigation. In most cases the experimenter was guided by his intuition derived from previous experience, and the results were satisfactory. We have no way of knowing, however, what 'pockets' of unusual or revealing behavior may have gone undetected.

In spite of these difficulties, I feel that the tests presented here are representative of the model's abilities and power. From a research point of view, these drawbacks are compensated in a rather direct way by the degree to which each of the model's variables and parameters are available for examination and manipulation.

Next generation state space model

In the course of developing and testing this model, a number of ideas emerged which were not included in the implementation presented above. Some were available at the outset but were not used in order to keep things simple. Others presented themselves after the experimenter became more familiar with the system's operation. Since they might be valuable for future work in this field, this section presents a brief list of these ideas with some discussion of their motivation. Most of them are not well developed and no plans exist for their implementation or test.

Insuring the command-force relationship

Earlier it was pointed out that there are restrictions on the allowable relationships between the command issued by the controller and the net force or torque applied to the joint. The Scheinman manipulator used in these tests is powered by DC torque motors. They have the characteristic that, neglecting friction, the torque produced is proportional to the current through the motor at all speeds. These motors are driven by servo amplifiers which insure, for a certain range of inputs and velocities, that the current through the motor is proportional to the voltage applied to the amplifier. Since the amplifiers only have a finite voltage swing (±28v) and the motors produce a back emf when in motion, the amplifiers are not always able to drive the desired current. In order to check for this condition the actual voltages across the motors was monitored at all times. Whenever these values approached 28 volts during a measurement, that measurement was ignored because the amplifier might have been saturated and applied an unknown force.

A more systematic treatment of this problem could be developed if sensors were used to measure the actual force delivered by the actuator. Then the force measurement, rather than the command, could be stored with the resulting acceleration measurement. Measurements of actual force delivered would automatically adjust for any saturation effects in the actuator. On the other hand, the translator would only determine the force to apply to a joint, rather than command -- another piece of hardware would have to convert the desired force into a command which produced that force. But that one dimensional problem, involving data for only one joint, is easily solved. This type of arrangement would also have the advantage that changes in properties of the actuators which occur quickly, such as fatigue or warm-up, need not effect performance of the translator.

Practice improves practice

In the present implementation, each practice session is totally independent from every other practice session. Each session starts about the same, and often includes a number of wild trajectories that are very different from the desired movement. Therefore, although much of the data generated might be useful at some future time, or for replication of some other movement, they are useless for the task at hand -- learning to replicate the desired trajectory.

In man, on the other hand, experience influences practice. The sophisticated mover does not necessarily flail his limbs around each time he wishes to learn a new movement. On the contrary, he may begin by executing a reasonably good approximation to his goal on the very first try. After some practice he will be doing something very close to the desired response, and each attempt at that level may be rich in measurements usable by the learning mechanism.

This type of regenerative effect (practice→learning→better practice→more learning→etc.) could be quite important for future studies. In order to make use of this approach, the practice algorithm must use the translator's expertize when planning movements.

Decaying vectors

During the normal course of an organism's development, the mechanical properties of a limb will change in a number of ways. As new measurement data arrive describing these new properties, the translator's equations of motion will change. But there is a potential problem which impairs the translator's ability to adapt, and even allows for wildly deviant performance during the adaptation period.

At any given time there are usually a number of vectors stored in the temporary buffer awaiting the arrival of others, at which time inversion takes place. Each of these vectors may have been generated during different mechanical conditions, if the mechanical properties of the system are changing rapidly or measurement data are being generated slowly. If a single inverse computation includes vectors generated during changing mechanical conditions, the resulting state space data are not likely to be reliable. (See Fig. 18.) Even when the transition from old to new data is smooth, this effect tends to prolong the amount of time and practice needed to completely change the controls.

The translator described here does not know when its vectors are no longer applicable, though performance measures could be used to help obtain such information. Another solution is based on the notion of a decaying memory. If old measurement data are removed from the temporary buffer, or given reduced weights, there will be a reduction in the range of measurement dates found contributing to any one inversion. Of course, the same sort of temporal decay could also be applied to data in the state space memory. Observe that I am advocating a decay of weight, not a decay of value.

An interesting result comes from considering the consequences of applying a particular decay function to the state space memory. Suppose, first of all, that all data decay exponentially. During any time interval new, old, and intermediate data are all reduced by the same fraction. When data do not enter the memory, all weights are reduced, but the relationships among weights are the same and behavior remains unchanged. If an exponential with a growing time-constant is used, (see Fig. 22), new data will decay faster than old, and the system will tend to return to previously used values. Of course, the time-constant need not be continuously growing. An exponential decay function with a piecewise constant time-constants, (eg. short plus long term memory; see Fig. 22b), would also give a temporary large weight to recent data.

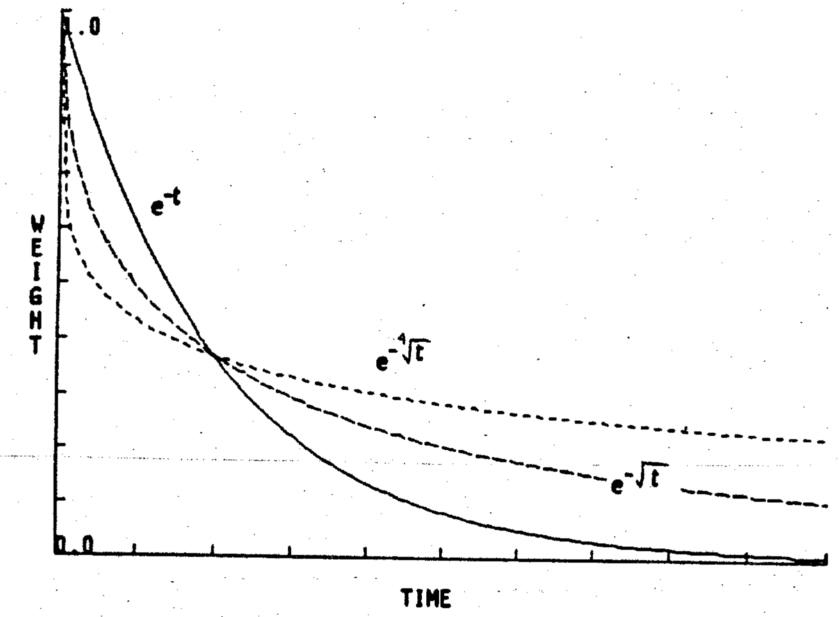
Now let us consider the following case: A large amount of data have been collected and stored. The mechanical properties of a limb are artificially manipulated and a practice period is permitted. If memory weights decay exponentially, the level of performance at any time after the adaptation period, but before new data are generated, should be the same as that found immediately after the adaptation period. A decay function which uses a growing time constant, however, should result in an initial improvement in performance, followed by a gradual return to pre-adaptation levels.

Hamilton and Bossom [8], and Choe and Welch [4] conducted prism adaptation experiments under these circumstances. Hamilton and Bossom suggest that their findings argue for a distinction between the mechanisms responsible for initial acquisition and those for adaptation. Their results, however, are consistent with the alternative notion of a variable time-constant memory.

Smaller state space memories

The worst drawback of the controller presented here is the size of the state space memory. The size of the memory increases as a power of the number of state variables; two for each degree of freedom, M^{2N}. For a few degrees of freedom this number is managably large, but soon reaches unreasonable proportions, even for the renowned capacity of the central nervous system. (By some analyses the human arm and hand have a total of 35 degrees of freedom.) There are, however, a number of ways in which the memories required for each limb or manipulator can be kept to practical sizes.

A



B

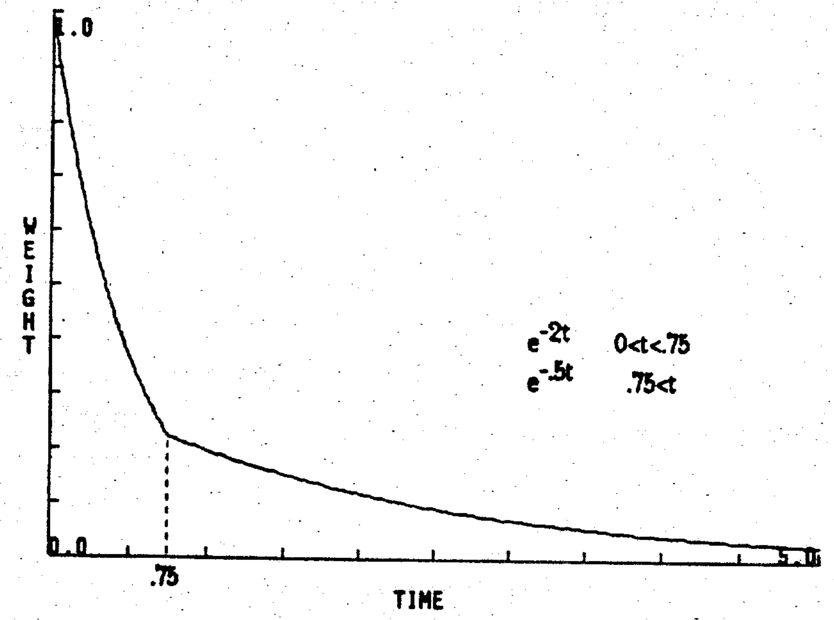


Fig. 22 Examples of decay functions with variable time-constants. Examples of two classes of such functions are shown a) Continuously varying time-constants: $\tau=-1.$, $\tau=-t^{-.5}$, and $\tau=-t^{-.75}$; b) Piecewise constant time-constant: $\tau=\{-2., 0<t<.75; -.5, .75<t\}$. A combination of short and long term memory falls into this second class.

If hash coding is used, the controller can capitalize on the tradeoff between size of memory, and the number of different types of movement learnable at one time. If few movements are to be learned, the number of different state space hyperregions involved will be small compared to all the possibilities. A hashing algorithm maps these few hyperregions from the very large state space into the available memory. In general, this technique does not reduce the dimensionality of the problem, and may not be adequate. Albus [2] employs hashing methods in an implementation of a version of Marr's model for cerebellar function [14].

A review of the terms entering the equations of motion for a manipulator reveals that, except for the Coriolis force, each is a function of the position state vector or the velocity state vector, but not both. (See Eq. 1 in the Appendix.) Gravitational forces and moments of inertia are dependent only on the position vector, while friction is primarily a function of velocity. Since Coriolis forces are typically small, one might propose the use of two memories; one with N positional dimensions and one with N velocity dimensions. Development of this approach could lead to practical control applications, especially if used together with hashing techniques.

Under certain circumstances a controller that uses one memory with dimensionality 2N can be replaced by two controllers, each of which employs a memory with fewer than 2N dimensions. This can be done whenever the mechanical properties of the plant may be decomposed into non-interacting subsections, or when the interactions are constrained to a few degrees of freedom. Then each controller will have state variables which represent the net influence of coupling with the other mechanical component. One could imagine the use of this type of arrangement in controlling the interactions between the trunk of the body and each limb. Suppose that each of two limbs had 4 degrees of freedom, and the trunk had another 3. Further suppose that all the coupling at each shoulder could be represented in terms of 3 degrees of freedom. If the entire system were controlled from one memory it would require 2(4+4+3)=22 dimensions. If three separate controllers were used, however, 3 memories would be required, of dimension 14, 14, and 12.

An improved version

The following formulation incorporates a number of improvements into the design of the controller, while maintaining its desirable properties. The major differences in this new formulation are:

1) The state space memory and temporary buffer are combined.

2) Hyperregions are no longer discrete entities.

3) Each measurement vector is labelled with its time of generation.
4) The neighborhood function, applied before inversion rather than after, gives each measurement vector a weight determined by its distance from the desired state. The age of the measurement also contributes to its weight.
5) During movement planning the generalized inverse is used to invert all measurement vectors found in the neighborhood of the desired state.

This arrangement allows for a simpler, less ad hoc specification of the neighborhood function. When data are taken from the memory they are weighted as a function of their distance from the desired state. The neighborhood function can be continuous and the broader it is the more the data in the memory are shared in the production of different movements. A sharper function enables better replication of movements which travel through non-linear portions of the movement space. Since each measurement vector carries information about its age, the advantages of a decaying memory can be realized.

An invertibility index would no longer be needed because each use of the generalized inverse yields an optimal estimate of the constants of mechanical description in the mean square error sense. The procedure described in this paper, averaging optimal estimates, is only sub-optimal. The proposed new method is truly optimal in that each computation of the mechanical constants relies on the maximum possible number of measurements -- all those present.

Conclusions

A computer and manipulator were used to implement a model for motor control, and transformation of the model into a working system verifies the model's explicitness. A discussion of the considerations which contributed to the selection of parameters for the implementation was presented. A number of parametric variations received experimental attention, though the space of all possible parametric combinations could not be explored. It is felt, however, that the results are representative of what can be expected from the state space controller.

Experiments were performed which demonstrated the controller's ability to improve its control vis-a-vis a set of test movements by practicing one movement of the set. Rudimentary examples of generalization were presented, though more extensive experimentation in this area is needed.

A number of improvements to the model were proposed and discussed. They included:

1) Employment of a subsidiary controller which maintains the command-force relationship for each actuator.
2) The use of a more powerful practice algorithm
3) Inclusion of a decay factor in operation of the controller's memory.
4) Methods for reducing the size of the state space memory.
5) Elimination of the discrete nature of the memory.

For those readers who are concerned with understanding man, the contents of this paper may supply no direct information. We hope, however, that examination and testing of these models will lead to the development of intellectual tools with which such direct information may be sought.

Acknowledgments

This work could not have been done without the patient help of Fred Drenckhahn. He kept the Scheinman manipulator in good health and working order despite experimentation which often lead to its abuse. My thanks go to Tomaso Poggio for introducing me to the generalized inverse. Thanks also to Berthold Horn, David Marr, Matt Mason, and Ron Pankiewicz for their critical readings, and especially Whitman Richards and John Hollerbach for their help.

References

1. Albert, A., *Regression and the Moore-Penrose Pseudoinverse*, (Academic Press, N.Y., 1972)

2. Albus, J.S., "Theoretical and Experimental Aspects of a Cerebellar Model", Ph.D Thesis, Biomedical Engineering, University of Maryland, 1972.

3. Bryson, A.E., and Ho, Y.C., *Appied Optimal Control*, (Ginn and Company, Waltham, 1969).

4. Choe, C.S., and Welch, R.B., "Variables Affecting the Intermanual Transfer and Decay of Prism Adaptation", *J. of Experimental Psychology*, 1974, 102, pp. 1076-1084.

5. Cratty, B.J., *Movement Behavior and Motor Learning*, (Lea & Febiger, Philadelphia, 1964).

6. Evarts, E.V., Bizzi, E., Burke, R.E., Delong, M., and Thach, W.T., Jr., "Central Control of Movement", *Neuroscience Research Program Bulletin*, Vol. 9, No. 1, 1970.

7. Gelfand, I.M., Gurfinkel, V.S., Tsetlin, M.L., and Shik, M.L., "Some Problems in the Analysis of Movement", In: *Models of the Structural-Functional Organization of Certain Biological Systems*, Gelfand, I.M., (ed.), (MIT Press, Cambridge, Mass., 1971) p. 335.

8. Hamilton, C.R., and Bossom, J., "Decay of Prism Aftereffects", *J. of Experimental Psychology*, 1964, 67, pp. 148-150.

9. Hammond, P.H., "The Influence of Prior Instruction to the Subject on an Apparently Involuntary Neuro-muscular Response", *J. Physiol.*, 127, 1956, pp. 17-18.

10. Horn, B.K.P. and Inoue, H., "Kinematics of the MIT-AI-Vicarm Manipulator", *MIT Working Paper 69*, May 1974.

11. Levine, W.S., of University of Maryland, Department of Electrical Engineering, Personal Communication of 9/30/75.

12. Marr, D., "A Theory of Cerebellar Cortex", *J. Physiol.*, 1969, 202, p. 468.

13. Melvill-Jones, G. and Watt, D.O.D., "Observations on the Control of Stepping and Hopping Movements in Man", *J. Physiol.*, 1971, 219, pp. 709-727.

14. Noble, B., *Applied Linear Algebra*, (Prentice-Hall, Inc., New Jersey, 1969, pp. 229-273.

15. Paul, R., "Modelling Trajectory Calculation and Servoing of a Computer Controlled Arm", *Stanford Artificial Intelligence Memo No. 177*, November, 1972, p. 57.

16. Pew, R.W., "Human Perceptual-Motor Performance", In: *Human Information Processing: Tutorials in Performance and Cognition*, Kantowitz, B.H., (ed.), (John Wiley & Sons, N.Y., 1974) pp. 1-39.

17. Raibert, M.H., "A State Space Model for Sensorimotor Control and Learning", *MIT Artificial Intelligence Memo No. 351*, January, 1976.

18. Raibert, M.H., "Doctoral Thesis Proposal", February 1977.

19. Ralston, H.J., "Energetics of Human Walking", In: *Neural Control of Locomotion*, Herman, R.M., Grillner, S., Stein, P.S.G., and Stuart, D.G., (eds.), (Plenum Press, New York, 1976) pp. 77-98.

20. Rust, B., Burrus, W.R., and Schneeberger, C., "A Simple Algorithm for Computing the Generalized Inverse of a Matrix", *Communications of the ACM*, 1966, Vol. 9, 5, 1966, pp. 381-387.

21. Taub, E. and Goldberg, I., "Prism Adaptation: Control of Intermanual Transfer by Distribution of Practice", *Science*, 1973, 180, pp. 755-757.

22. Tsypkin, Y.Z., *Adaptation and Learning in Automatic Systems*, Translated by Nikoloic, Z.J., (Academic Press, New York, 1971).

23. Welford, A.T., *Fundamentals of Skill*, (Methuen & Company, LTD, London, 1968) pp. 291-292.

Appendix from [15] with revisions.

Eq. 1 Equations of motion:

$$T_m - G(\phi) - B(\dot{\phi}) - C(\phi,\dot{\phi}) = J(\phi)\cdot\ddot{\phi}$$

where:

T_m is the motor torque vector
G is the gravitational torque vector
B is the frictional torque vector
C is the coriolis torque vector
J is the moment of inertia matrix
ϕ, $\dot{\phi}$, and $\ddot{\phi}$ are the position, velocity, and acceleration vectors.

Eq. 2 Equations of motion for a given state:

$$T_m - G_\alpha - B_\beta - C_{\alpha\beta} = J_\alpha\cdot\ddot{\phi}$$

where:

$F_{\alpha\beta}$ is the vectored function $F(\phi,\dot{\phi})$ evaluated at $\phi=\alpha$, $\dot{\phi}=\beta$.

Eq. 3 Equation 2 simplified further:

$$T_m = J_\alpha\cdot\ddot{\phi} + K_{\alpha\beta}$$

where:

$K_{\alpha\beta} = G_\alpha - B_\beta - C_{\alpha\beta}$.

Eq. 4 One dimensional inversion equation (N=1):

$$k = t_1 - \frac{t_1 - t_2}{\ddot{\phi}_1 - \ddot{\phi}_2} \cdot \ddot{\phi}_1$$

$$j = \frac{t_1 - t_2}{\ddot{\phi}_1 - \ddot{\phi}_2}$$

where:

k, j, and t are one dimensional versions of K, J, and T.

Eq. 5 The inversion equations (dropping the $\alpha\beta$ subscripts):

$$K = T_{av} - [\underline{T} \cdot \underline{\ddot{\phi}}^{-1}] \cdot \ddot{\phi}_{av}$$
$$J = \underline{T} \cdot \underline{\ddot{\phi}}^{-1}$$

where:

$$\underline{T} = [T_1 \mid T_2 \mid \ldots \mid T_N] - [T_{N+1} \mid T_{N+1} \mid \ldots \mid T_{N+1}]$$
$$\underline{\ddot{\phi}} = [\ddot{\phi}_1 \mid \ddot{\phi}_2 \mid \ldots \mid \ddot{\phi}_N] - [\ddot{\phi}_{N+1} \mid \ddot{\phi}_{N+1} \mid \ldots \mid \ddot{\phi}_{N+1}]$$

Eq. 6 Torque-command constraint:

$$t = a(\phi,\dot{\phi}) \cdot u + b(\phi,\dot{\phi})$$

where:

t is the force applied to the tendon

u is the motor command

$a(\phi,\dot{\phi})$ and $b(\phi,\dot{\phi})$ are state dependent constants.