PERCEPTUAL CONTROL THEORY

Proceedings of the 1st European Workshop on Perceptual Control Theory
Gregynog, The University of Wales
23rd – 27th June 1994

Edited by
Marcos A. Rodrigues and Mark H. Lee
Centre for Intelligent Systems
Department of Computer Science
The University of Wales

The University of Wales, Aberystwyth

ISBN 0 903878 19 4

Contents

An "Artificial Cerebellum" Adaptive Stabilization of a Control System

William T. Powers

The Control Systems Group
73 Ridge Place, CR 510
Durango, CO 81301-8136, USA
Powers_W%FLC@vaxf.Colorado.edu

Abstract

This paper gives a preliminary view of a practical method of producing a real-time adaptation of a control system to its load characteristics. This method requires no teacher and no knowledge of the properties of the external environment; it is based strictly on the error signal inside the control system. A computer simulation of the behavior of the adaptive system is shown.

1 Introduction

Adaptive stabilization is needed in living control systems because they must operate successfully in a changing environment. A control system that can produce walking movements under normal circumstances can adapt to wading through water, or to walking while carrying another person or an off-center suitcase, while towing a ricksha, while walking downhill, or while walking on the Moon in a spacesuit. Pointing to a target with a stick uses the same motor control systems as pointing to it with a fingertip. The same control systems, with the hands holding a control yoke, are used to control the angle of bank of a Boeing 747.

The basic adaptations required under these different circumstances are to changes in the direction and amount of effect that an action has, and to changes in the dynamical effects of an action in terms of producing accelerations and velocities in the perception being controlled. A control system tuned for best control under one circumstance will be either too sluggish or too overreactive in another. What is required, and what human and animal control systems apparently possess, is a type of control mechanism that can automatically alter its own sensitivity and dynamical characteristics to minimize errors of control over a wide range of environmental properties – without having a Ph.D. in physical dynamics and control theory.

The title of this paper refers to the fact that in the human cerebellum there are architectures that suggest computations involving what appear to be delay lines and summations of delayed and weighted signals. The method of adaptive stabilization to be presented here, reduced to a simple block diagram, has some resemblance to the cerebellum's architecture, although the actual cerebellar computations are not known (Eccles, Ito, & Szentagothai, 1967). The term Artificial Cerebellum, or AC, provides a handy label for this method, and also refers to the fact that the cerebellum seems to be involved in the stabilization of motor control systems in higher organisms, just as this method is used here to stabilize a simulated control system.

No attempt is made here to achieve mathematical generality. There are many questions about the conditions under which this method will work that will remain unanswered. Perhaps this practical demonstration will attract the interest of those with a deeper understanding of the underlying mathematical considerations.

2 Convolution

The response of a linear system to a sudden input impulse (infinite amplitude, zero duration, finite energy) can be used to characterize the way the system will react to a continuing input. If the input waveform is thought of as a series of impulses with no spaces between them, the response of the system to the continuous wave can be conceptualized as an unending series of responses of the same sort that occur for a single impulse, all superimposed. At any moment, the total output of the system is the sum of the effects remaining from previous impulses beginning with the current one and extending some time into the past, covering the total time over which an impulse produces any residual effects.

If the input waveform is represented as an arbitrary function $g(t)$, the values of that waveform at previous times are given as $g(t-\tau)$, where τ is the distance into the past (the time t is the momentary present) at which the waveform is measured. Plotting the effects of impulses at various times τ in the past against τ, we obtain a function $f(\tau)$ which is the shape of the response to a single impulse. So the part of the present output due to a signal at time τ in the past is written as $g(t-\tau) * f(\tau)$. Summing over all values of τ, for a given time t, is called "convolution" of the functions g and f. This method is used in analysis as one way of representing the response characteristics of the system.

In the Artificial Cerebellum model, we employ the convolution theorem the other way: not to analyze an existing system, but to synthesize a function with particular response characteristics. In fact the process of synthesis itself becomes a method by which the output function of a control system can have its characteristics altered so as to achieve stability of control.

The principal feature of the AC is that the output function of a control system is constructed quite literally as a model of the convolution theorem. The output of the control system $O(t)$ is derived from the error signal $E(t)$ by a convolution process so that

$$O(t) = \sum_{\tau} E(t-\tau) * f(\tau) \qquad (1)$$

The details of the basic output function are shown in Figure 1.

The error signal enters an analog shift register (like a charge-coupled device or delay line) from which copies of the signal delayed by increasing times τ are tapped. The signals crossing from left to right in Figure 1 correspond to $E(t-\tau)$. Each signal is multipled by a weighting factor $k1 \ldots k7$ before being summed to produce the output signal $O(t)$. The values of k are the values of $f(\tau)$. The use of only 7 delays is for illustration only.

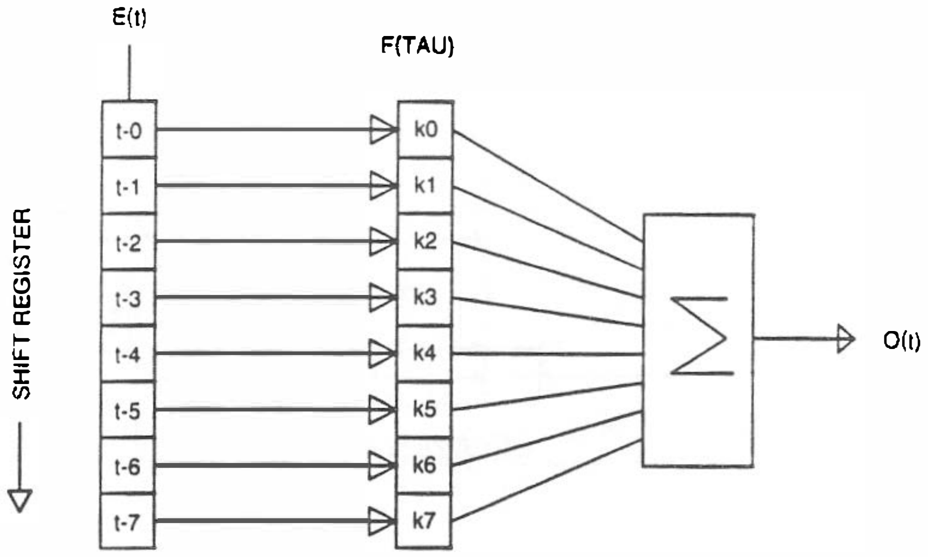


Figure 1: Circuit for implementation of convolution theory in an output function.

3 Deconvolution

The basic problem to be solved is this. For a given control situation (a particular kind of load), choose values of $f(\tau)$ in the output function that will result in the minimum possible average error signal. This process is called deconvolution.

The adaptive aspect of this output device is a method for adjusting the weights $k1 \dots k7$ to minimize the error measure. I have used this method in several applications, one of which was to construct a transfer function to match the input-output characteristics of a real system. Figure 2 illustrates the procedure.

Coming in from the left in Figure 2 is a time-varying input signal $I(t)$, which enters the real system. The time-varying output of the real system, $O'(t)$, exits to the right. It is desired to find a transfer function $f(\tau)$ which, when convolved with the input signal as above, will create an output signal $O(t)$ that matches $O'(t)$. The adaptive criterion is thus to minimize the difference $d(t) = O'(t) - O(t)$. When this is accomplished, the weights representing values of $f(\tau)$ will then represent the impulse response of the real system, in numerical or graphical form. This method does not require that the impulse response be any analytical function.

Shown in Figure 2 is the arrangement for one value of the delay, τ. The delayed input signal $I(t-\tau)$ is multipled by a value $k\tau$ to produce the τ-th contribution to the synthesized output signal $O(t)$. If the real system consisted of a pure transport lag with a value of τ, then this would be the only non-zero value of $k\tau$ required, and all that would be needed would be a way of raising or lowering $k\tau$ until $d(t) = 0$.

The solution is represented as an analog computation. The difference $d(t)$ is produced by a comparator that receives $O'(t)$ and $O(t)$. The signal $d(t)$ is multiplied by the signum function, $sgn[I(t-\tau)]$ – that is, by 1, 0, or -1 depending on the sign and magnitude of $I(t-\tau)$. The time integral of the resulting value is the value of $k\tau$. This negative feedback loop will increase or decrease $k\tau$ until, on the average, $d(t) = 0$. The signum function is required to keep the sign of the feedback negative regardless of the delay, τ.

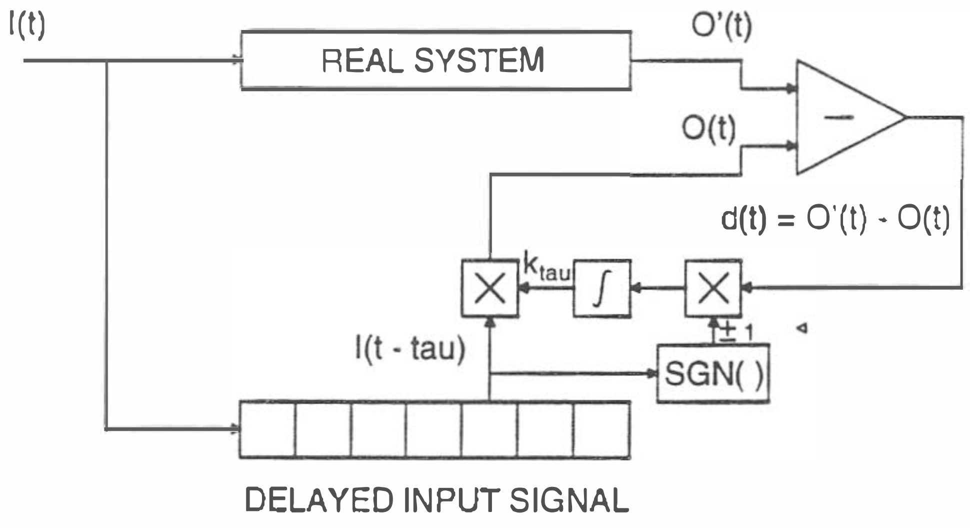


Figure 2: Example of deconvolution method. Used to match simulated function to real system characteristics when only output and input are known. See text.

Strictly speaking, the correction applied to $k\tau$ should be the integral of $d(t)/I(t-\tau)$, for the correct value of k is simply $O'(t)/I(t-\tau)$. However, in general the waveforms involved will contain noise, and using the ratio of output to delayed input for the correction would give the largest weight to the smallest signals, which will contain the highest proportion of noise. Using the signum function instead keeps the sign of the corrections right, but weights them in effect by the magnitude of $I(t-\tau)$. It also avoids the problem of division by zero when $I(t-\tau)$ is zero.

If the true transfer function involves nonzero values for all entries in $f(\tau)$, we can duplicate the correction part of the feedback loop for each value of τ, summing the outputs to produce $O(t)$ and driving all the corrections simultaneously with $d(t)$. To sum up in mathematical form:

$$O'(t) = [Some\ function\ of]I(t), \tag{2}$$

$$O(t) = \sum I(t-\tau) * f(\tau), \tag{3}$$

$$d(t) = O'(t) - O(t). \tag{4}$$

For all τ:

$$d[f(\tau)]/dt = GAIN * d(t) * sgn[I(t-\tau)]. \tag{5}$$

The GAIN factor (not shown in Figure 2) is made small enough so that the correction requires a long time compared with the normal variations of $I(t)$ and $O'(t)$. If a smoothed random signal is used as a driving function $I(t)$, the values of $f(\tau)$ will gradually change until $O(t)$ is tracking $O'(t)$ very closely. At that point, $f(\tau)$ will closely approximate the impulse response of the real system.

There is a tendency of the solutions to oscillate when nearing asymptote. This can be corrected by applying an exponential decay factor to each value of $f(\tau)$. The more rapid the decay, the more rapidly the final value can be approached (the $GAIN$ factor can be larger), but the larger the uncorrected difference $d(t)$ will be.

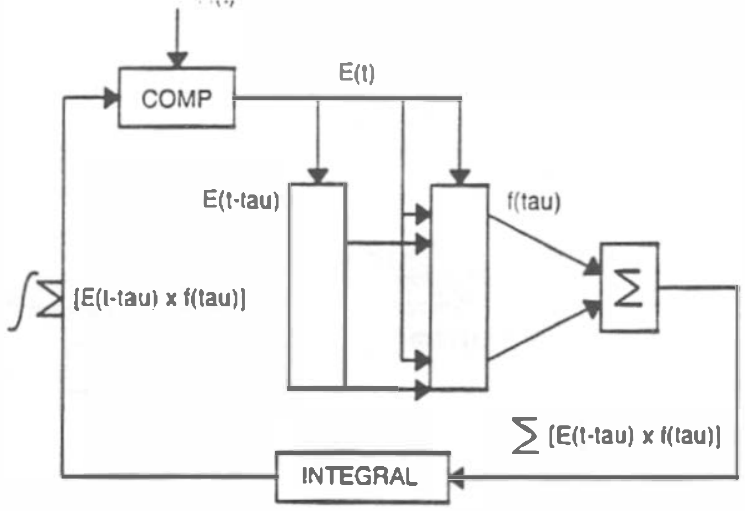

COMP
E(t)
E(t-tau)
f(tau)
Σ
$\int\Sigma$ [E(t-tau) x f(tau)]
Σ [E(t-tau) x f(tau)]
INTEGRAL

Figure 3: Embedding the Artificial Cerebellum as the output function of a control system. See text.

4 Application to Stabilizing a Control System

Figure 3 shows a control system with feedback from its output through an integral load (like applying a force to an object and sensing its velocity). The comparator subtracts the sensed value of the integral from the reference signal $R(t)$, with the error signal $E(t)$ entering a stylized diagram of the "artificial cerebellum".

The error signal in the model is

$$E(t) = R(t) - \int \sum_{\tau} sgn(E(t-\tau)) * f(\tau) \tag{6}$$

The error signal in the "real system", which is now the system we desire to obtain, is

$$E'(t) = 0. \tag{7}$$

The difference between model and real system is

$$d(t) = E'(t) - E(t) = -E(t) \tag{8}$$

and the correction algorithm is

$$d[(f(\tau)]/dt = GAIN * E(t-\tau)] * E(t). \tag{9}$$

The middle line descending from $E(t)$ in Figure 2 represents the $E(t)$ on the right in (9), used in the correction process inside the box labelled $f(\tau)$. That box in Figure 4 now includes the simple deconvolution computation, not shown in detail.

Because our adaptation criterion is now $E'(t) = 0$, the fact that we are using an integral in the feedback path has no effect on how the adaptation method is applied: any function can go in the feedback path. I don't know what the limitations on the feedback function must be for this deconvolution method to work.

This adaptation method uses only the information in the error signal: there is no teacher and no information about the external world is needed.

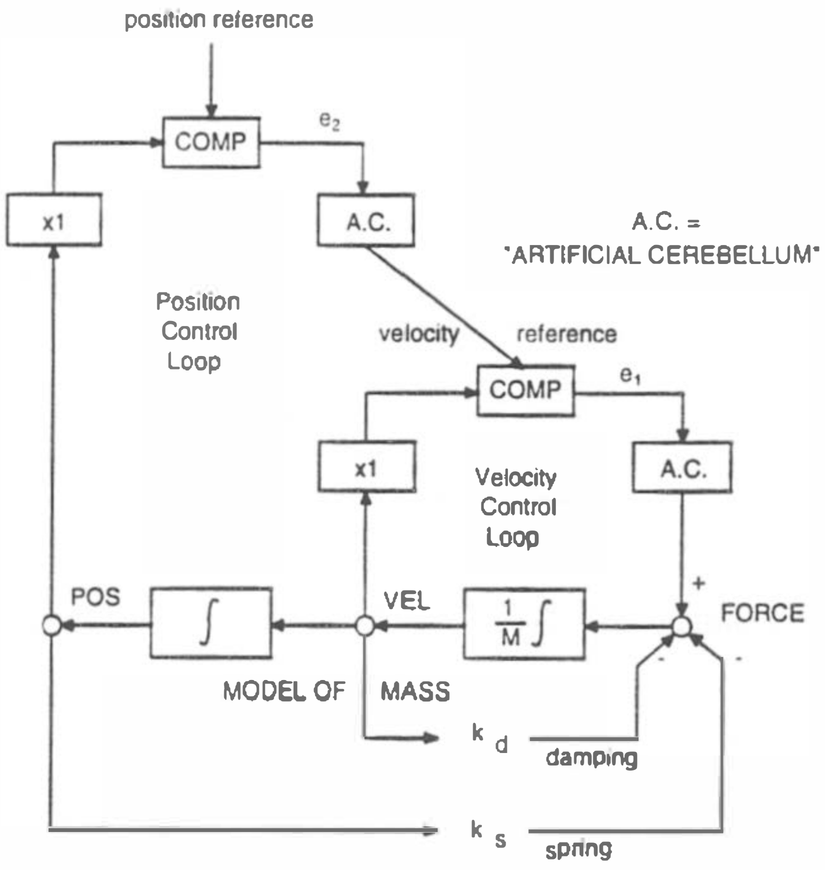


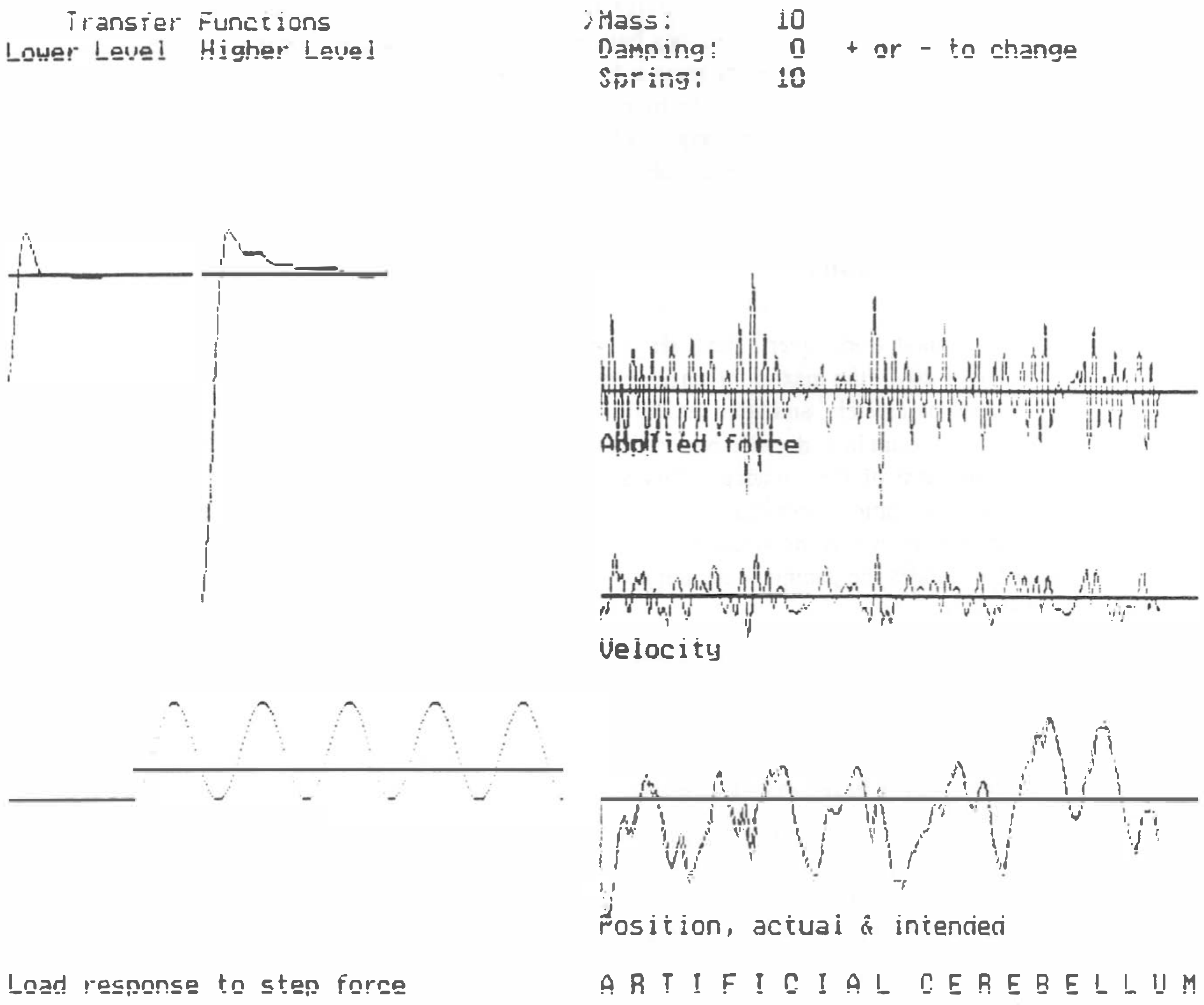
Figure 4: A two-level control system using two A. C. components as output functions, with a load having variable mass, restoring spring constant, and viscous drag constant. See text.

5 Demonstration: Controlling a Mass on a Spring with Damping

As a test of this approach to adaptively stabilizing a control system, I wrote a program that simulates a two-level control system controlling the position of a mass on a spring with viscous damping. The lower level of control senses and controls the velocity of the mass by applying a force to it, and the upper level senses and controls the position of the mass by varying the reference signal for the lower-level system. Both levels have an output function that is an artificial cerebellum model. The program is written so that all functions operate effectively at the same time. See Figure 4.

There are three user-adjustable parameters: the mass, M, can be changed between 1 and 20 units, the spring constant, ks, between 0 and 20 units, and the coefficient of damping, kd, between 0 and 100 units. These can be changed from the keyboard while the program is running. Changing these parameters drastically changes the dynamical characteristics of the load being controlled. In addition, pressing the "z" key sets the two $f(\tau)$ tables to zero, causing adaptation to begin over again. The program runs on a 80486DX 33MHz computer. The reference signal is continuously produced by a smoothed random- number generator. This demonstration is like a baby learning to move its arms, using only the difference between what it wants to experience and what it does experience as the basis for learning motor control. The random variations in the reference signal simply exercise the control system while its parameters adapt to the external world.

The screen display is shown in Figure 5 after the program has run for about 30 seconds. In the upper left are plotted the two $f(\tau)$ tables, which are negative-going. If a sign had been reversed in the adaptation step, the tables would have contained positive values. The actual values automatically



```
>Mass:        10
 Damping:      0    + or - to change
 Spring:      10
```

Transfer Functions
Lower Level
Higher Level
Applied force
Velocity
Position, actual & intended
Load response to step force
A R T I F I C I A L C E R E B E L L U M

Figure 5: Behavior of two-level system after 30 seconds. Upper left, plots of $f(\tau)$ for each level. Lower right, plots of applied force, velocity, and position (actual and reference). Lower left, behavior of load with a step force applied, without control.

become those that are correct for negative feedback. The tables have a length of 120 time units. Below on the left is a plot of the way the mass would behave when subject to a step-force (without control); since there is no damping in this run, the mass would simply oscillate on its restoring spring. The plots on the right, with the system controlling, show the applied force (upper trace), the velocity (middle trace), and the reference signal behavior and actual position of the mass (lower double trace). The position follows the reference signal very closely.

Figure 6 shows the result when "z" is struck to reset the $f(\tau)$ tables to zero just after the start of one plot (the plots repeat endlessly); the screen is printed when that plot ends. The $f(\tau)$ tables are set to zero at the arrow. As they build up again, in around 700 time units (the plots are 800 time units long), the position of the mass at first shows oscillations, then quickly begins to conform to the reference signal variations. If the time scaling were such that each unit equalled 1/60 second (as in the actual program), the time required to regain reasonable control would be about 11 seconds, and good control would be restored in about 20 seconds.

6 Conclusions

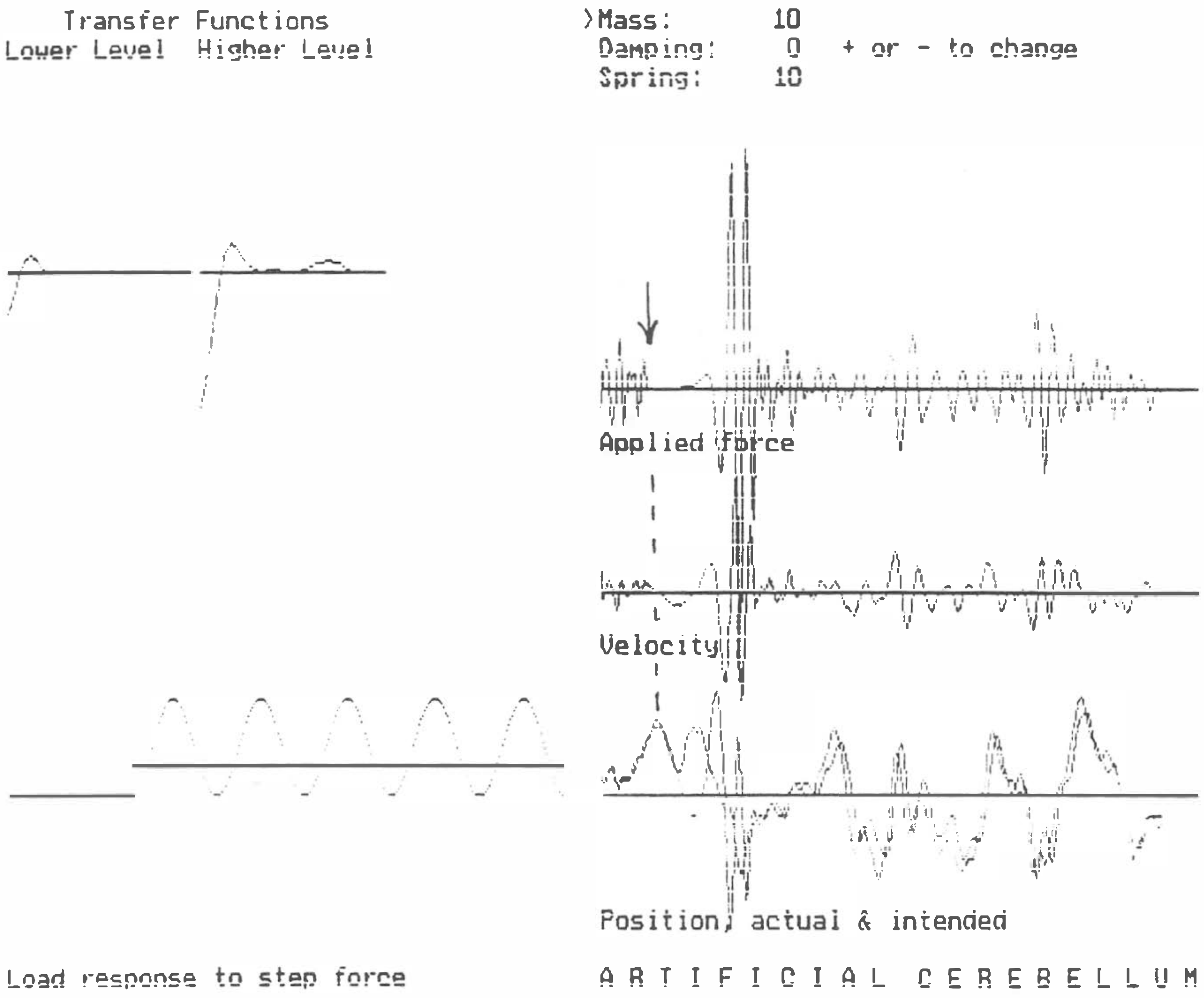
This method works over a mass range of 20–1, a damping range of 100–0, and a spring constant range of 20–0 (the maximum ranges provided for). Stability is quickly recovered after any change in these parameters, and after setting the $f(\tau)$ tables to zero. Occasionally, an abrupt reduction in mass will result in a "division by zero" error; this can be avoided by reducing the mass slowly enough for the forms of $f(\tau)$ to adapt. This is a "feature" of the program, not a flaw in the method. With this demonstration successfully completed, the next step will be to install artificial cerebellum output functions in an existing simulation of a human arm. This model employs full physical dynamics, and simulates the combined tendon and stretch reflexes, the muscle, and visual control of pointing behavior.

There are many possible variations of this method, including running an artificial cerebellum in parallel with a fixed output function to trim the control system's properties (there are some indications that the real cerebellum relates in this way to neuromotor control systems). The range of load types with which this method can work is not known. A proper mathematical analysis of the artificial cerebellum will have to await the efforts of others with more mathematical ability than the author has.

NOTE: Copies of this program, with Turbo Pascal 5.5 and assembler source code included, can be obtained from the author by sending a formatted IBM PC compatible floppy disk (any capacity) with an adequately stamped self-addressed disk mailer to the author at 73 Ridge Place, CR 510, Durango, CO 81301. The program is released to the public domain and can be copied freely.

References

[1] Eccles, J. Ito, Masao, and Szentagothai, J. (1967), *The cerebellum as a neuronal machine*. New York: Springer-Verlag.


Transfer Functions
Lower Level
Higher Level
+ or - to change
Applied force
Velocity
Position, actual & intended
Load response to step force
A R T I F I C I A L   C E R E B E L L U M

```
>Mass:      10
 Damping:    0
 Spring:    10
```

Figure 6: Behavior of two-level system with both $f(\tau)$ functions reset to zero immediately after start of plot (arrow). In the bottom right plot of position and reference position, note that control is quickly restored.