A98-37039

AIAA-98-4139

ELECTROTSTATIC CONTROL OF A MEMBRANE USING ADAPTIVE FEEDBACK LINEARIZATION

Daniel P. Scharf[*] David C. Hyland[†] and Peter D. Washabaugh[‡]
Deptartment of Aerospace Engineering
University of Michigan, Ann Arbor, MI

Abstract

Inflatable, ultra-lightweight antennas using metallized membranes for reflectors are being investigated for use in orbital telescopes and solar energy applications. Simulation results for the control of a simplified model of a membrane using an electric field are reported. The non-linear mapping from position and acceleration of the center of the membrane to control input is learned on-line, and simultaneously used to approximately feedback linearize the SISO plant. Two types of inverse plant model structures are studied: neural nets with sigmoid non-linearities and adaptively constructed look-up tables. In both cases, using only input-output data, the simplified membrane model was successfully stabilized about an unstable equilibrium.

Introduction

Presently, the Hubble Space Telescope is the only space-based observatory. Since its worth has been abundantly proven and observing time is in great demand, it would be desirable to put more telescopes into orbit. However, the cost of such a venture would be very large.

In an effort to reduce launch and manufacturing costs, inflatable structures have been considered to construct ultra-lightweight orbital telescopes. A schematic is shown in Figure 1(a). Presently, three configurations are being considered for the mirror. Two of the cases use a metallized membrane as the reflector. In these cases either pneumatic pressure or an electric field deforms the membrane into the desired shape (Figure 1(b,c)). The third option uses a spin-cast, meniscus mirror, and will not be discussed here [1, 2, 3].

*PhD Candidate
†Professor and Chair
‡Associate Professor

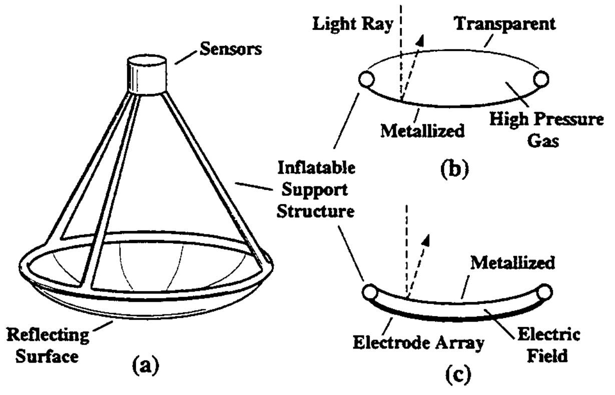

Figure 1: Inflatable Telescope Concepts.

A great deal of work has been done on pneumatically formed membrane mirrors, with a proof of concept mission flown in May of 1996 [4, 5]. However, there are concerns that pneumatic pressure may not be sufficient to smooth out the wrinkles in the membrane (the membrane must be folded and stowed in a launch vehicle) to the degree necessary for observing optical wavelengths.

To address this concern, an active control scheme using an electric field to shape the membrane is considered. The control system would first deform the membrane until it is roughly in the correct shape. Then, sensing the the shape of the membrane, differential voltages applied to an array of electrodes would correct for deviations.

Work led by Dennis Mihora [6] (mid-1970's) and Goslee [7] (1978-82) looked at a concentric arrangement of electrodes. Work done by Lang [8] (1978-82) considered a faceted arrangement of electrodes and investigated an LQG based controller.

In this study, the use of electric fields to roughly shape the membrane is considered. That is, only the displacement of the center of the membrane is considered important. However, instead of a taking a model based approach, an adaptive control scheme is developed. The control architecture uses a learning algorithm to determine the inverse map from plant states (position and velocity of the center of the membrane) and their derivatives to control input. This learned inverse map then approximates the control input necessary to achieve a desired acceleration. The desired acceleration is generated by a linear-state feedback controller. Therefore, if the inverse map learned is accurate, the controller will feedback linearize the plant.

The goal is to deform the membrane so that its center is deflected approximately 70% of the original membrane–electrode gap distance. The difficulty arises in that there are no stable equilibria beyond a deflection of 50% of the original gap distance [6]. Ideally, the membrane should be deflected as much as possible, decreasing the focal length and hence, the support structure. A 70% deflection was chosen as an initial goal for this study.

Neural nets and adaptively constructed look-up tables were both investigated as inverse plant model structures. The neural net model was updated using variants of back-propagation algorithms and the look-up table model was updated using a set of simple rules. Both the neural net and look-up table structures performed well in stabilizing a simplified membrane model. The adaptively constructed look-up table structure will be referred to as the look-up table structure.

Control Architecture

As mentioned in the Introduction, the strategy is to approximately feedback linearize the plant by learning the inverse mapping from plant states and their derivatives to control input. Since only the displacement of the center of the membrane is considered, the plant is SISO. Further, it was assumed that all plant states are available.

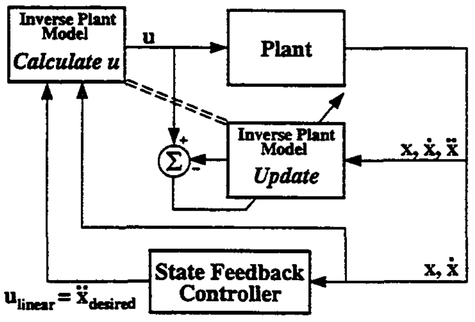

Figure 2: Approximate Feedback Linearization Control Architecture.

Figure 2 shows a block diagram detailing the control architecture. The inverse plant model is the map $x, \dot{x}, \ddot{x} \mapsto u$. The plants states and their derivatives (hereafter, the derivatives will be omitted, but implied by the phrase 'plant states') are measured, and using the known control input, the inverse map is updated. Then the position and velocity are used to calculate a desired acceleration via the state feedback controller (i.e. $u_{linear} = -\gamma_1(x - x_{desired}) - \gamma_2\dot{x}$). Finally, since the aim is to feedback linearize the plant, the inverse plant model is used to calculate the actual control input by $x, \dot{x}, u_{linear} \mapsto u$. If the inverse plant model is accurate, then $\ddot{x} \approx u_{linear}$.

Inverse Plant Structures

Two inverse plant model structures were considered: a neural net and a look-up table.

Neural Net

The design of the neural net was dictated by the complexity of the function to be approximated. It has been proven by Cybenko [9] that a single hidden layer is sufficient to approximate any map $\Re^n \mapsto \Re$ arbitrarily well. Albeit, the number of hidden neurons needed to do so must be determined from simulation. When damping was ignored, the map was $x, \ddot{x} \mapsto u$. In this case a neural net consisting of two input nodes, a hidden layer of ten neurons with hyperbolic tangent sigmoid functions, and an output layer of one linear neuron sufficed (see Figure 3).For a brief discussion of neural nets see Haykin, pp. 8-13 [10].

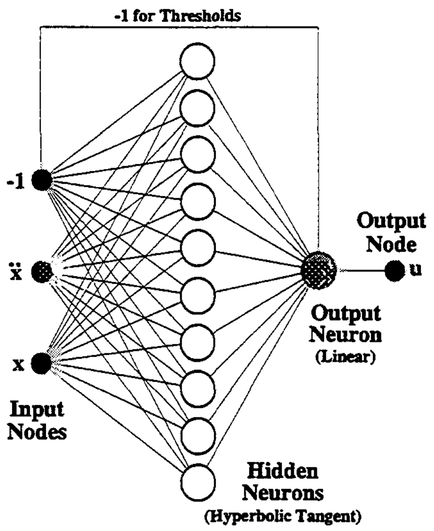

Figure 3: Neural Net for Mapping of $\Re \times \Re \mapsto \Re$.

The learning algorithm used for the neural net was back-propagation with momentum and a

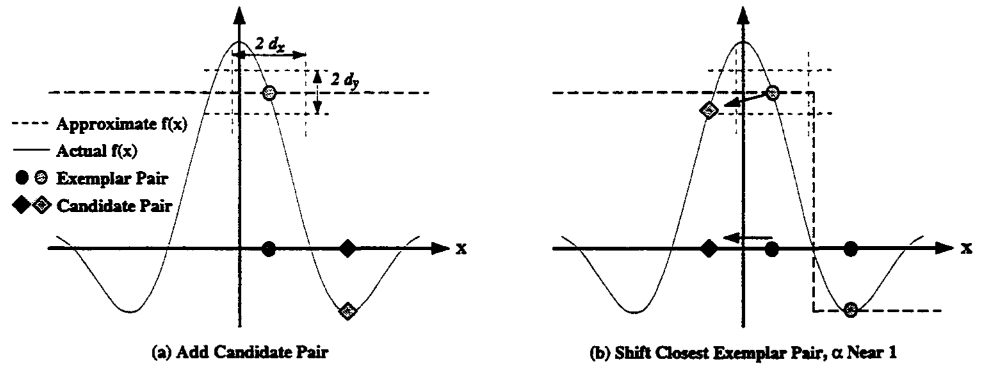

Figure 4: Various Exemplar Pair Update Cases.

variable step-size. The training set consisted of a single pair of inputs and desired outputs: the measured plant states at the end of the previous time step and the control applied during the previous time step. This pair was back-propagated until the squared error fell beneath a specified tolerance, up to a maximum of ten times. If the squared error did not reach the tolerance, then the plant was fed a random input for that time step.

The rationale behind feeding the plant a random input is as follows. Assume that at a certain time step the neural net approximates the inverse map well. Then the neural net output should be close to the desired output in the training set. Hence, the squared error is small, i.e. beneath the specified tolerance. The converse is not necessarily true. Nonetheless, it is a reasonable assumption that if the squared error does not fall beneath the tolerance the neural net is not approximating the inverse map well. In this case, near-term controller performance must be sacrificed for improved long-term performance. That is, the controller must compromise between controlling the plant in the present and searching for new information to use in the future (i.e. improving the inverse map approximation). This is similar to the dual problem from stochastic adaptive control [11]. The term dual arises from the controller providing the dual functions of controlling the plant and searching for new information about it.

Another strategy would have been to form a larger training set for back-propagation. As mentioned previously, the only pair of inputs and desired outputs in the training set were the plant states measured at the end of and the control used during *the previous time step.* If instead the pairs from the last N time steps were stored and used in the training set, one might expect the back-propagation to impart more information to the neural net, and, as a result, that the controller would perform better. This strategy was implemented, but did not work nearly as well as when the training set consisted of just one pair.

Adaptive Look-Up Table

In an attempt to make the update of the inverse model as fast as possible, the simplest function approximation technique was considered, a look-up table. Assuming a set of stored input-output pairs, the value of the function at any point is approximated by first finding the stored input nearest the point and then taking the associated stored output. The stored inputs and outputs are referred to as exemplars and exemplar-outputs, respectively, and collectively as an exemplar pair.

The update of the exemplar pairs proceeds as follows. For a new input-output pair the closest exemplar to the input is found. We place a neighborhood of radius d_x (i.e. a ball in $\Re^n$) around this exemplar and define a coarseness level, $d_y > 0$ (the effect of increasing d_y will be to reduce the fineness of the approximation). If the new pair is such that the input lies outside the exemplar's neighborhood and the output differs from the associated exemplar-output by more than d_y, the pair is added to the list of exemplar-pairs. Otherwise, the nearest exemplar-pair is shifted to include the new information contained in the input-output pair, but a new exemplar-pair is not added.

Figure 4 elucidates the update algorithm. In each part of the figure the d_x and d_y bounds, the actual function, the approximation to the function generated by the algorithm and the next candidate pair are shown. In (a) the candidate pair is added, in (b) the candidate pair causes the nearest exemplar pair to be shifted (the arrows indicate direction and magnitude of the shift).

Formally, given $f(\mathbf{x}) : \Re^n \mapsto \Re$, let $\hat{\mathbf{x}}$ be an input vector. $\hat{y} = f(\hat{\mathbf{x}})$ is to be approximated. Define $\xi_M \triangleq \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ as the set of exemplars with maximum size M, and $\mathcal{Y} \triangleq \{y_1, y_2, \ldots, y_M\}$ as the set of corresponding exemplar-outputs. Let $k^* = \min_k ||\mathbf{x}_k - \hat{\mathbf{x}}||$. Finally, take $\hat{y} = y_{k^*}$.

To update ξ_M and $\mathcal{Y}$, consider $(\bar{\mathbf{x}}, \bar{y})$ as the next candidate pair. Let $\bar{k} = \min_k ||\mathbf{x}_k - \bar{\mathbf{x}}||$. Then for $d_x, d_y > 0$

- If $||\bar{\mathbf{x}} - \mathbf{x}_{\bar{k}}|| > d_x$ and $||\bar{y} - y_{\bar{k}}|| > d_y$, then add $\bar{\mathbf{x}}$ and $\bar{y}$ to ξ_M and $\mathcal{Y}$, respectively. If necessary, an element of each set is deleted to keep M elements in each.
- Otherwise, for $0 \leq \alpha \leq 1$ shift the exemplar and exemplar output by

$$\begin{bmatrix} \mathbf{x}_{\bar{k}} \\ y_{\bar{k}} \end{bmatrix} \mapsto \begin{bmatrix} \mathbf{x}_{\bar{k}} \\ y_{\bar{k}} \end{bmatrix} + \alpha \begin{bmatrix} \bar{\mathbf{x}} - \mathbf{x}_{\bar{k}} \\ \bar{y} - y_{\bar{k}} \end{bmatrix}$$

As noted in the first bullet, when a new exemplar is added, it may replace a previous one. This deletion enables the inverse plant model to adapt to plant changes. The model 'forgets' old information.

Controlling vs. Learning

The dual problem was mentioned in regards to the neural net structure. Due to the complexity of the neural net, it is difficult to determine the conditions under which the back-propagation will fail to drive the squared error beneath the tolerance. Having implicitly assume that this failure means that the present inverse map approximation is insufficient, the plant is fed white noise in attempt to gain new information. In the simulations to follow this brute force approach to the dual problem is sufficient, but more refined methods are being considered.

The conflict between control and learning is seen much more easily in the look-up table structure. Assume the plant has been fed white noise for some time so that an initial inverse plant model has been learned. For simplicity, consider Figure 5. In this one-dimensional case (i.e. only an acceleration measurement is necessary for the update), the actual plant is given by $\ddot{x} = u$. The controller has learned three exemplar pairs and the approximation to the inverse plant, $u = \ddot{x}$, generated by these exemplar pairs is shown by the dashed line.

If the controller is now used to calculate control inputs (instead of just learning), then nothing new will be learned. To see this, let the output of the state-feedback controller be the dark diamond

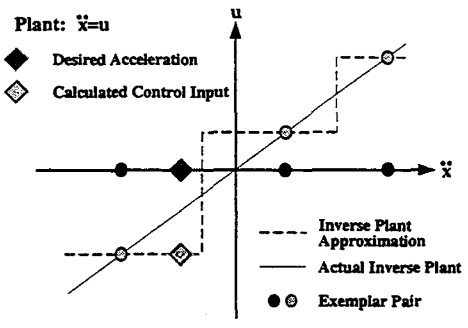

Figure 5: Illustration of Learning/Controlling Conflict.

shown in Figure 5. The light diamond is the control calculated by the control algorithm (i.e. the same value as the light circle to the far left in the figure). This control is stored for the update step and then fed into the plant (see Figure 2). The acceleration resulting from this control input is measured. Note that it will be the same value as the dark circle to the far left in the figure. This acceleration and the stored control input form a candidate pair which is passed to the update algorithm. But this candidate pair is exactly the exemplar pair. Therefore, the update algorithm does nothing and the candidate pair is discarded.

In fact, for any desired acceleration the calculated control and the resulting acceleration measured will always be one of the three initially learned exemplar pairs. As a result, during the update step new exemplar pairs will never be added. *The inverse plant model is frozen.* In higher dimensional cases (i.e. where the position and velocity are included in the domain), the learning does not cease altogether. The position and velocity may vary enough to cause an exemplar to shift. However, all the control values (exemplar outputs) will be those learned initially. No new control values will be tried.

To correct this situation, the controller recognizes when the current inverse plant model is insufficient and starts learning again. The controller determines this insufficiency by comparing the last measured acceleration to the acceleration that was desired (i.e. $\ddot{x}$ vs. u_{linear} in Figure 2). If the difference is greater than a specified error tolerance a search for new information is initiated. Performance based control resumes when a new, suitable control is found.

The information search consists of varying the last control that produced an acceleration within tolerance, called u_{last}, by greater and greater amounts. For example, the control is randomly varied within ±10% for ten steps and then within ±20% for another ten steps. If at any time the randomly varied control produces an acceleration within tolerance, then it becomes u_{last} and control calculation in the standard way resumes. If at the end of the twenty steps an acceptable control has not been found, then the plant is fed white noise until one is found. The number of different percentages tried before reverting to white noise, the number of steps during which each percentage is used and the percentages themselves are all adjustable.

Simulation Results

Plant Model and Simulation Parameters

The membrane is modeled as a linear spring and mass with the control input being the voltage difference between the electrode and membrane (see Figure 6). The equation of motion for this system is

$$\ddot{x} + k(x-1) + \beta\frac{u^2}{x^2} = 0 \qquad (1)$$

where the spring equilibrium is at $x = 1$, k is the spring constant and β contains electromechanical constants. The equilibria are unstable for $x \leq \frac{2}{3}$. While this is different than the $x < \frac{1}{2}$ predicted by more accurate models and confirmed by experiment [6], this simplified model captures the salient aspect—the instability of equilibria at large deflections.

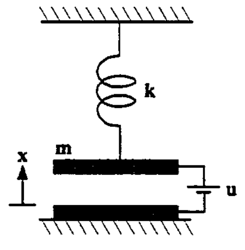

Figure 6: Simplified Membrane Model.

For the simulations, $k, \beta = 1$ and the sample frequency is 100 Hz. Initial estimates of the run-time of the look-up table and neural net algorithms were made to justify this frequency. The initial conditions for all the simulations were the same: $x = 0.98$ and $\dot{x} = -0.01$. These initial conditions are indicative of a small oscillation about the control-off equilibrium position, $x = 1$. The goal is for the algorithms to stabilize the plant about $x = 0.3$. This location is well within the unstable range.

In practice arcing is a concern. In the simulations the breakdown electric field magnitude was taken to be 1. If the controller calls for a voltage which would give rise to an electric field greater than a safety factor × the breakdown electric field, the control is set to a safe, maximum value. The safety factor in these simulations was 0.9.

The figures to be presented include the position vs. time and the desired acceleration (output of state feedback controller) compared to the actual acceleration vs. time. In the look-up table algorithm, the plant is fed white noise from 0s to 1s so that the controller can develop an initial model. The acceleration history of this time segment is not shown for reasons of clarity.

Neural Net Structure

The neural net controller results for controlling Equation 1 are shown in Figure 7. The plant is successfully driven to $x = 0.3$. There is a 5% offset at the end of 10 s, but this can be reduced by reducing the sum squared error tolerance. The state feedback controller used was $u_{linear} = -(x-0.3)-\dot{x}$.

Shown for comparison in the Figure 7 Position Plot is the response of the system $\ddot{x} = -(x - 0.3) - \dot{x}$ (i.e. the system to be emulated through feedback linearization) with initial conditions taken to be the same as the plant state at 0.82 s. This time was chosen for comparison since it is when the controller initially stops feeding white noise to the plant. As can be seen from the figure, the plant is approximately feedback linearized as desired.

Also apparent in the Acceleration Plot of Figure 7 are the time steps when the controller reverted to white noise because of insufficient convergence in

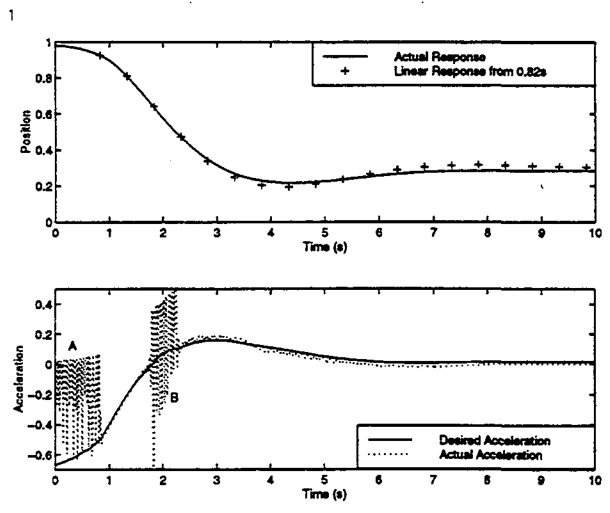

Figure 7: Neural Net Simulation Results.

Since the neural net structure controls the plant so well, it is expected that the inverse surface learned by the neural net should closely approximate the actual inverse surface. Shown in Figure 8 and Figure 9 are the inverse surface generated by the neural net and the actual inverse surface, respectively. The flat area in Figure 9 is a result of the fact that the electrode is assumed to always be at a lower potential then the membrane. Therefore, the electrode cannot push on the membrane. All positive accelerations are a result of the spring restoring force which gives a maximum positive acceleration $\ddot{x} = k(1 - x)$ (i.e. $u = 0$). Larger positive accelerations are not achievable.

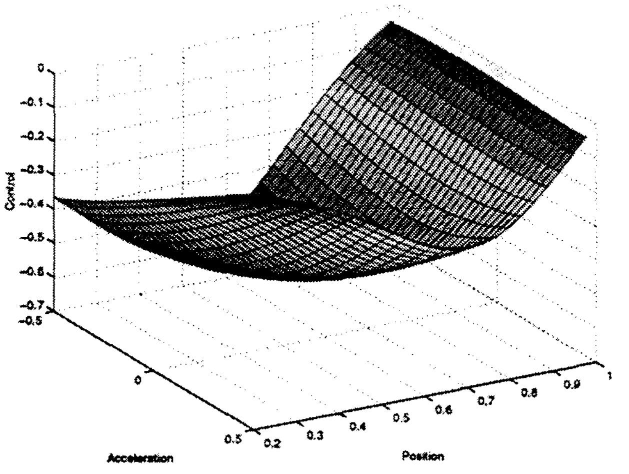

Figure 8: Neural Net Approximation of Inverse Surface.

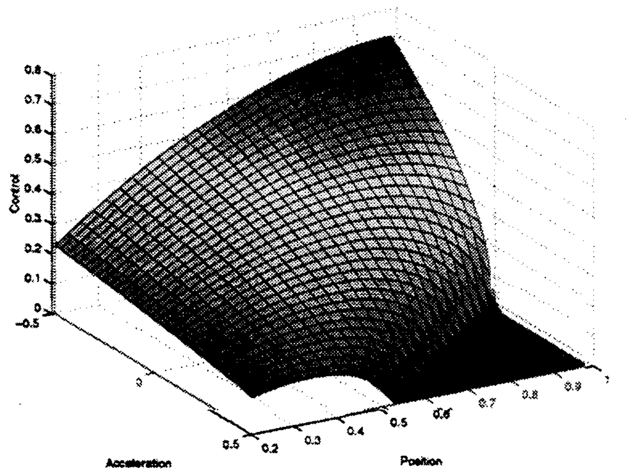

Figure 9: Actual Inverse Surface for Equation 1.

The neural net approximation appears to be terrible. But note that the control enters Equation 1 through u^2. The sign of u does not matter. In Figure 10 the absolute value of the neural net approximation of Figure 8 is shown. Now the neural net approximation appears a little better in some areas, but it is still not a very good global approximation.

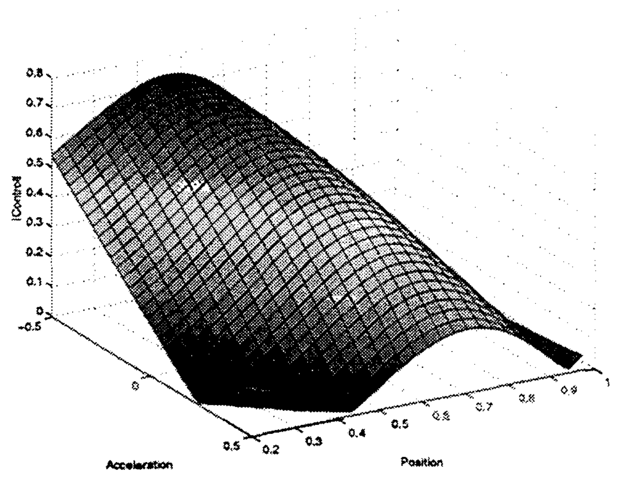

Figure 10: Absolute Value of Neural Net Approximation of Inverse Surface.

However, the neural net does not need to accurately approximate the inverse surface globally for the algorithm to function; it need only do it along the trajectory of the plant in position-acceleration space. And this it does. Figure 11 is a contour plot of the error surface, i.e. the difference between the absolute value of the neural net approximate inverse surface and the actual inverse surface. The path of the plant in position-acceleration space is overlaid. The bursts in the plant trajectory, labeled A and B, correspond to the bursts labeled A and B in Figure 7.

At the end of the simulation (in Figure 11 the hook in the plant trajectory ending at an acceleration of 0 and a position of 0.3), the neural net approximation is almost exact as evidenced by the nearness of the zero error contour. The neural net is locally approximating the inverse surface accurately. Recall that the training set consists only of information from the previous time step. As a result, the region where the neural net approximation is accurate enough for control purposes will change with time.

The plant knowledge needed to design the neural net controller consisted solely of position and acceleration measurements. Impulse and step responses were used to determine position and acceleration magnitudes used in initializing the neural net. Once this was done, trial runs commanding only mild deflections were used to determine the optimal number of hidden neurons and sum squared error tolerance. The state-feedback controller was designed to give accelerations in the range observed in the impulse and step responses.

In some cases when poor acceleration bounds were used in initialization the neural net would fail to converge. To address this problem, the algorithm was modified so that it would recognize when the neural net had failed to converge and reinitialize. A scaling of the acceleration bounds in response to acceleration measurements remains to be added, otherwise subsequent initializations will be just as bad as the first.

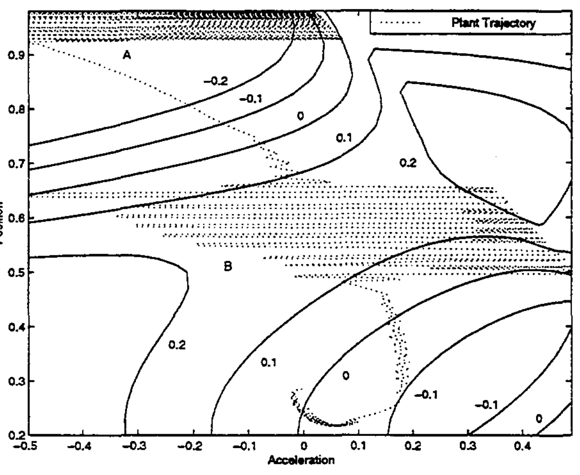

Figure 11: Error Surface at Final Time of Neural Net with Plant Trajectory.

Adaptive Look-Up Table Structure

The simulation results for the look-up table structure applied to Equation 1 are shown in Figure 12. The plant is successfully controlled to $x = 0.3$, with an offset of 1% at the end of 18s. The state feedback controller used was $u_{linear} = -(x-0.3)-\dot{x}$. For comparison, the response of the linear system $\ddot{x} = -(x-0.3) - \dot{x}$ starting with initial conditions identical to the plant states at 1.0 s is also shown in the Position Plot. The comparison was begun at 1.0 s because this is when the controller began controlling as opposed to only learning. The plant is not as accurately feedback linearized as in the neural net case. The information searches contributed to this degradation, but it is also a result of discrete nature of the look-up table structure itself.

The information searches can be seen in the acceleration plot of Figure 12. For example, at approximately 10 s, the controller reverts to a white noise search after failing to find a suitable control by varying u_{last} The small burst at 7.5 s is an information search that found a suitable control and did not revert to white noise. After 14s, enough exemplars have been learned to keep the acceleration within the specified error tolerance. The controller begins alternating between using one exemplar and switching between two (the small deviations between 16 and 17 s).

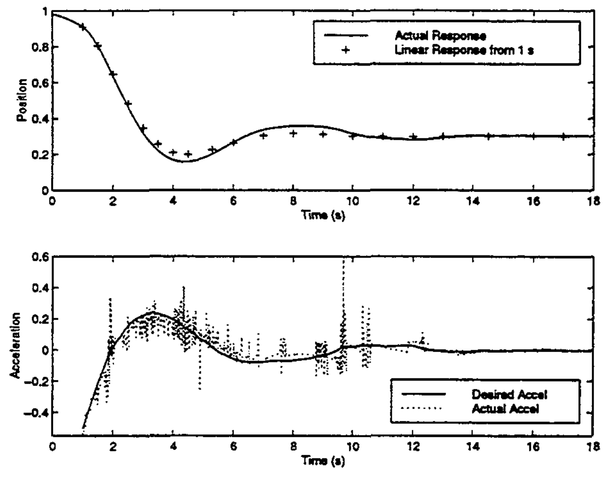

Figure 12: Look-Up Table Simulation Results.

The inverse surface learned by the controller at the end of the simulation is shown in Figure 13. The learned inverse surface compares favorably to the actual inverse surface shown in Figure 9.

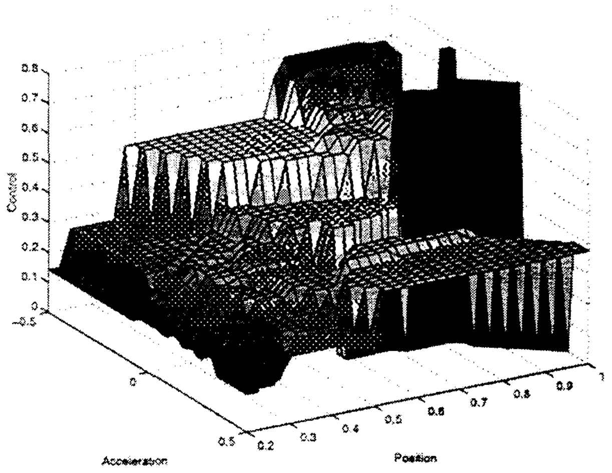

Figure 13: Look-Up Table Approximation of Inverse Surface.

Shown in Figure 14 is a contour plot of the error surface (look-up approximate inverse surface minus actual inverse surface) at the end of the simulation with the exemplar pairs overlaid. Note that the zero-error contour generally stays near the exemplar pairs. This is as expected for at these points the inverse surface is initially known exactly. The exemplars are then shifted from their exact values.

From comparing Figure 14 to Figure 11 it is seen that the look-up table structure globally approximates the inverse surface of Equation 1 more accurately than the neural net structure. The look-up table structure retains general knowledge about the points in position-acceleration space it has passed through, except when those points are overwritten by new points. At the end of the simulation, the look-up table has 142 exemplars, well below the specified maximum of 200. No points were lost. The neural net, however, has a fixed number of parameters (the weights and thresholds) which vary each time the neural net is trained. By training on only one input-output pair, the neural net obtains a very accurate approximation for that training pair at the cost of accuracy at other points.

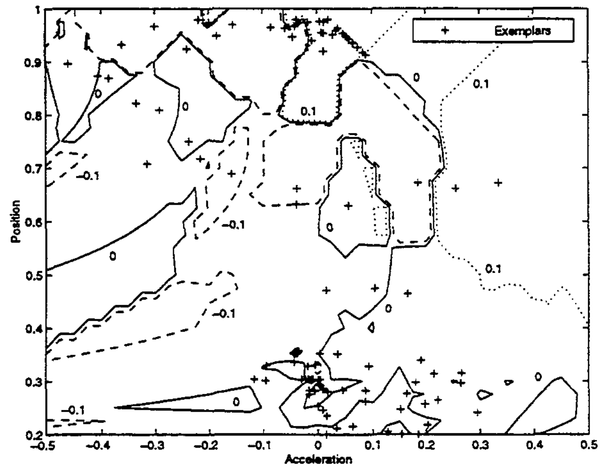

Figure 14: Error Surface at Final Time Look-Up Table with Exemplars.

Instability Due to Control Saturation

In certain cases an instability was noted in the look-up table structure algorithm. In these cases, the information search failed to find a suitable control and the algorithm reverted to feeding the plant white noise. The difference in the simulations which generated the instability was that the white noise was fed when x was very small (i.e. the plates were very close together). As the controller searched for a suitable control the spring caused the plate to 'snap back' which in turn gave rise to desired accelerations ($\ddot{x}_{desired}$ in Figure 2) outside the achievable range of accelerations. Recall that the electric field is limited because of arcing. As a result, the controller kept feeding the plant white noise until the plant states were such that the desired acceleration was again achievable. Unfortunately, this did not occur until the plate was back near its equilibrium position. At this point the algorithm would successfully regain control of the plant, drive x to near 0.3 and then either hold the plate there or loose control again.

An example of this phenomenon is shown in Figure 15. From 0s to just before 4s, the algorithm proceeds as expected. At just before 4s, an information search fails and white noise is fed into the plant. Looking at the Position Plot of Figure 15, we see the plate begin to 'snap back'. In the Acceleration Plot the approximate acceleration bounds are shown during the white noise search. The downward sloping bands of the actual acceleration are bounded above by $u = 0$ resulting in an acceleration given by $\ddot{x} = 1 - x$ and below by the maximum electric field $\frac{u}{x} = 0.9$ (safety factor) implying $\ddot{x} = 0.19 - x$. The values when $x = 0.6$ are shown. It is clear where the desired acceleration is not achievable.

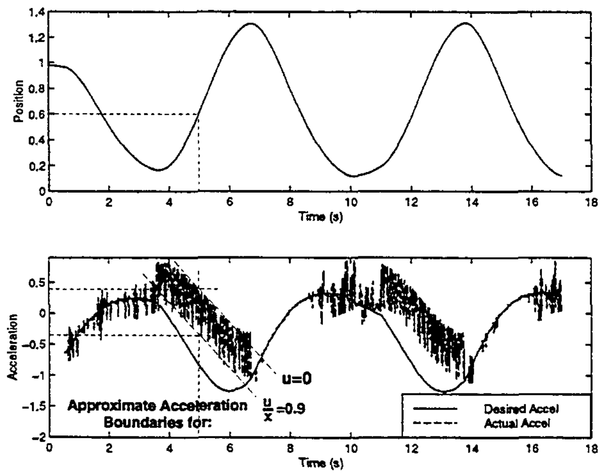

Figure 15: Example of Closed Loop Instability Due to Saturation.

Note that the initial conditions for the unstable simulations were exactly the same as those in simulations where the plant was successfully stabilized. It is important to remember that this algorithm has a random nature due to the information searches. Therefore, for two runs, one may successfully control the plant while the other drives the plant unstable.

However, this instability can be avoided by carefully choosing the information search parameters in concert with those of the state feedback controller, as was done for the simulation in Figure 12. This balance between the state feedback and the information search parameters is illustrated in Figure 16. In this figure, the acceleration limits of the information search (grey bands in the figure), which are a function of the search percentages selected, are too small for the desired acceleration generated by the state feedback controller. Choosing the state feedback controller parameters small enough to give a slow desired response and the search percentages large enough to give an appreciable acceleration variation greatly reduces the probability of the search reverting to white noise and from there possibly to an instability. Though this is by no means proof, once the look-up table algorithm's parameters were properly tuned, numerous simulations were conducted without a single instability occurring.

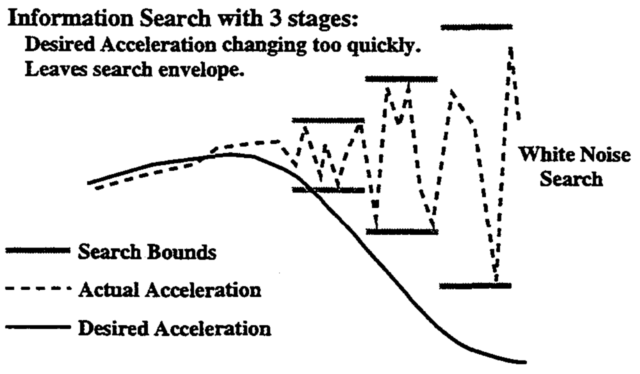

Figure 16: Illustration of Information Search Failure.

The same unstable behavior was noted in the neural net structure when the squared error tolerance was set too low, causing the controller to feed white noise into the plant at inopportune times. However, setting the squared error tolerance relatively high initially and then reducing it as the magnitude of the desired accelerations decrease eliminates this saturation instability. Provisions were, of course, made for increasing the sum square error tolerance should the desired accelerations suddenly increase.

As a final note, the applications for which these control algorithms are being considered are such that in the unlikely event of an instability, the control could simply be reinitialized without damage to hardware or people.

Design Effort

Considerably more design effort is needed for the look-up table structure than for the neural net structure. As alluded to, the determination of the information search parameters (i.e. the best acceleration error tolerance (used to initiate information searches), the number of steps spent varying u_{last} within each search percentage and the search percentages themselves) required considerable simulation. The remaining parameters in the look-up table structure, the exemplar update algorithm parameters, were chosen as follows. d_x was taken to be a fraction of the acceleration error tolerance. For d_y we need to estimate the fineness of the inverse approximation we need. If d_y is too large, then the plant is not feedback linearized, and if it is too small, then we may throwing away information by storing too many exemplar pairs (i.e. deleting older, varied exemplar pairs in favor of repetitive, newer pairs). Since we wish to stabilize about $x = 0.3$, various fractions of the control range at this location (i.e. 0 to $0.9x_{desired}$ or 0.27) were tried. After some experimentation, $1/25^{th}$ of the control range was found to work well.

While all the parameters in the look-up table structure could conceivably be chosen from only input-output data, much more extensive testing would be necessary than in the neural net structure to fine tune the control search parameters. One avenue of future research is developing auto-scale features for the acceleration measurements and state feedback controller parameters. It is the hope that with this features a general set of control search parameters could be chosen for a wide range of systems.

Finally, Figure 17 illustrates the learning vs. controlling conflict discussed previously and the necessity for having some type of information search. Shown in the figure are the position and acceleration results for a look-up table structure simulation without a information search algorithm. The controller drives the membrane into the electrode. Training longer does not upgrade the performance since, regardless of the length of time, the training occurs in only a small region of the applicable position-acceleration space.

Similar behavior is noted for the neural net structure when the option to feed white noise into the plant is removed.

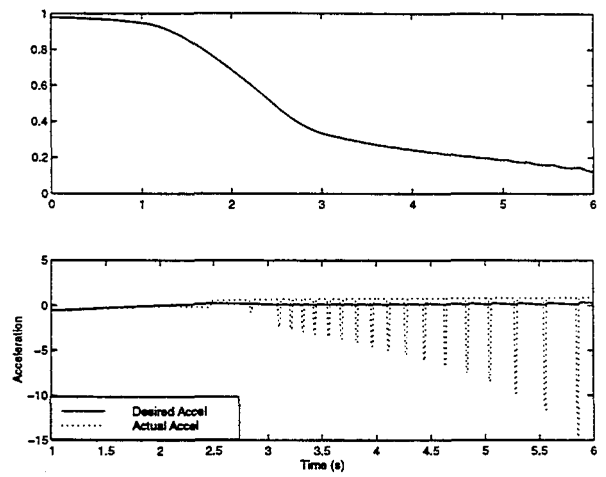

Figure 17: Look-Up Table Structure without Information Search Algorithm.

Conclusion

Two very different inverse plant structures were considered for use in an approximate feedback linearizing control architecture: neural nets and look-up tables. Both structures performed extremely well in stabilizing a simple nonlinear system about an unstable equilibrium.

The neural net structure required less design effort and performed better in terms of feedback linearing the plant than the look-up table structure. However, the inverse map these structures were learning was a map $\Re^2 \mapsto \Re$, that is, the acceleration did not depend on the velocity. Initial research into damped systems suggests that the neural net structure has trouble converging in the present architecture, and for this reason, the look-up table structure may prove more feasible for more complicated systems.

Subsequent research will concentrate on reducing the design effort for the look-up table structure through auto-scaling features and making the neural net algorithm more robust to poor initializations. The disturbance rejection abilities of both structures will also be investigated. Finally, experimental work has begun on a 16 in. diameter membrane.

Acknowledgements

This work is supported by an NSF Graduate Fellowship and by the Rackham Graduate School at the University of Michigan.

References

[1] Rapp, Donald,"Ultra Lightweight Telescopes", JPL Report D-13975, JPL, 4800 Oak Grove Drive, Pasadena, CA, 91109, September 30, 1996.

[2] Reibaldi, G.G., Bernasconi, M.C., "QUASAT Program: The ESA Reflector," *Acta Astronautica,* Vol. 15(3), pp. 181-187, 1987.

[3] Cassapakis, C. and Thomas, M., "Inflatable Structures Technology Development Overview," AIAA 95-3738, Proc. of AIAA 1995 Space Programs and Technologies Conf., Sept. 26-28, Huntsville, AL, 1995.

[4] Freeland, R.E., Bilyeu, G.D., Veal, G.R., "Development of Flight Hardware for a Large, Inflatable-Deployable Antenna Experiment," *Acta Astronautica,* Vol. 38(4-8), pp. 251-260, 1996.

[5] Veal, G., Freeland, R.E., "IN-STEP Inflatable Antenna Description," AIAA 95-3739, Proc. of the AIAA 1995 Space Program and Technologies Conf., September 26-28, Huntsville, AL, 1995.

[6] Mihora, D.J., Redmond, P.J., "Electrostatically Formed Antennas," *Journal of Spacecraft,* Vol. 17(5), pp.465-473, 1980.

[7] Goslee, J.W., "Electrostatic Membrane Antenna Concept Studies," Large Space Systems Technology, NASA Conf. Pub. 2168, Nov. 18-20, 1981.

[8] Lang, J.H., Staelin, D.H., "Electrostatically Figured Reflecting Membrane Antennas for Satellites," *IEEE Transactions on Automatic Control,* Vol. 27(3), pp. 666-670, 1982.

[9] Cybenko, G., "Approximation by Superposition of Sigmoidal Functions," *Mathematics of Control, Signals and Systems,* Vol. 2(4), pp. 303-314, 1989.

[10] Haykin, S., *Neural Networks: A Comprehensive Foundation.* New York: Macmillian College Publishing Company, Inc., 1994.

[11] Åström, K.J., Wittenmark, B., *Adaptive Control, 2nd ed.* Reading, MA: Addison-Wesley Publishing Company, Inc., 1995.