ADAPTIVE SPACECRAFT ATTITUDE CONTROL USING NEURAL NETWORKS

Andrew Turner – ajturner@vt.edu
Dr. Christopher Hall – cdhall@vt.edu
Aerospace & Ocean Engineering
Virginia Tech, Blacksburg, VA 24061 USA

Abstract

Neural networks offer a unique approach to controlling a dynamically changing system. By updating synaptic weights of an interconnected, algebraic framework, desired system objectives can be reached despite an unknown operating environment. This research implements a neural network for controlling the attitude of a satellite with unknown system disturbances. The training and updating of the controller occurs on-line in real-time, increasingly minimizing pointing error during mission operations. The design of the neural network controller is discussed, as well as preliminary results from computer simulations.

Introduction

The majority of current satellite control technologies employ classical methods that are determined *a priori*. However, due to changes in the satellite system or unknown flight characteristics, these systems may prove to be unreliable. Adaptive control systems provide a robust means of controlling a satellite in the face of these uncertainties and failures.

In this research project we develop a control system that adapts to the satellite's flight characteristics. By using 'soft-computing' techniques such as a neural network, the control system learns and adapts to changes in the controllability of the satellite.

Using existing methods of neural network development [1], an adequate controller can be designed that will control the three axis angles, and three rotation rates of a satellite's attitude [2][3][4][5].

The use of an adaptive attitude control system can greatly reduce the costs of satellite algorithm and software development. An adaptive control system can learn from a model of the satellite and from the actual flight mission instead of having to specify the classical control law. Furthermore, a well-designed neural network can be reused for many missions, without having to change the base structure of the system. This reuse can greatly reduce the cost of having to develop a control system from the ground up. Also, the adaptive control system can recover from system failures that may otherwise cripple a spacecraft such as those that occurred on Mariner 2 and PanAmSat Corporation's Galaxy 4 [6]. The robust satellite can recover from these failures without having to wait for human intervention and updating of its software.

Spacecraft Model

In order to design a controller for spacecraft attitude, a model of the system must be determined. The nonlinear spacecraft model includes the spacecraft attitude kinematics, rotational dynamics and control torques. It is assumed that the spacecraft is measured in body-orbital coordinates, with a nominal nadir pointing attitude. That is, the body axes are initially aligned with the orbital axes, and the body z-axis points towards the earth.

The 4-element quaternion set, $\bar{\mathbf{q}}$, is used to specify the attitude, as there are no singularities similar to those found in an Euler angle set. The quaternion can be determined from the Euler-axis parameters set (**a,** ϕ) as follows:

$$\mathbf{q} = \mathbf{a}\sin\frac{\Phi}{2} \tag{1}$$

$$q_4 = \cos\frac{\Phi}{2} \tag{2}$$

$$\bar{\mathbf{q}} = \begin{bmatrix}\mathbf{q} & q_4\end{bmatrix}^T \tag{3}$$

Therefore, for a nadir-pointing spacecraft, the body-orbital quaternion is $\bar{\mathbf{q}} = [0,0,0,1]^T$.

The equations of motion for the attitude are:

$$\dot{\bar{\mathbf{q}}} = \frac{1}{2}\begin{bmatrix} \mathbf{q}^x + q_4 \cdot \mathbf{1} \\ -\mathbf{q}^T \end{bmatrix} \cdot \omega \qquad (4)$$

The rotational dynamics are characterized by the absolute angular velocity vector, ω. The differential equation for the angular velocity is based on Euler's equation:

$$I\dot{\omega} = N - \omega \times I\omega \,. \qquad (5)$$

where $\boldsymbol{I}$ is the spacecraft moment of inertia matrix (assumed to be measured along the principle axes, and therefore diagonal), ω is the body angular velocity, and $\boldsymbol{N}$ is the spacecraft torques, which is the sum of the gravity-gradient, $\boldsymbol{N_{gg}}$, and control, $\boldsymbol{N_c}$, torques. Then the body-inertial angular velocity satisfies:

$$\dot{\omega}^{bi} = \mathbf{I}^{-1}\left[N_{gg} + N_c - \omega^{bi} \times \left(I\omega^{bi}\right)\right] \qquad (6)$$

As mentioned previously, the equations of motion are used in the body-orbital coordinates. In order to do this, we must find a relation between the body-inertial, ω^{bi}, and body-orbital, ω^{bo}, angular velocities.

$$\begin{aligned} \omega^{bi} &= \omega^{bo} + \omega^{oi} \\ &= \omega^{bo} - \omega_c o_2 \end{aligned} \qquad (7)$$

therefore,

$$\begin{aligned} \dot{\omega}^{bi} &= \dot{\omega}^{bo} - \omega_c \dot{o}_2 \\ &= \dot{\omega}^{bo} + \omega_c \omega^{bo^x} o_2 \end{aligned}, \qquad (8)$$

where ω_c, is the orbital angular velocity. This derivation leads to the equation of the angular velocity in body-orbital coordinates:

$$\begin{aligned} \dot{\omega}^{bo} &= \mathbf{I}^{-1}\left[N_{gg} + N_c\right] \\ &\quad - \mathbf{I}^{-1}\left[\left(\omega^{bo} - \omega^{oi}\right) \times I\left(\omega^{bo} - \omega^{oi}\right)\right] \\ &\quad - \omega_c \omega^{bo^x} o_2 \end{aligned} \qquad (9)$$

where the gravity gradient torque is:

$$N_{gg} = 3\omega_c{}^2 o_3{}^x \mathbf{I} o_3 \qquad (10)$$

and the body-orbital rotation matrix:

$$R^{bo} = \left(q_4{}^2 - \mathbf{q}^T\mathbf{q}\right)\cdot\mathbf{1} + 2\mathbf{q}\mathbf{q}^T - 2q_4\mathbf{q}^x \qquad (11)$$

is required to determine the o_2 and o_3 vectors.

Neural Networks

Formulation of NN

A neural network (NN) is a computational structure composed of a collection of artificial neurons. Each neuron is an operator that processes, with a nonlinear activation function ϕ, the weighted sum of its inputs to produce an output signal. The connections between the artificial neurons define the behavior of the net, identify its applicability, and specify its training methods. In a multi-layer perceptron (MLP), the neurons are grouped in one or more layers with the output of each layer being the input to the next layer.

The training process consists of adjusting the neuron weights based on the expected output and some optimization rules. In a supervised scheme, the weights are adjusted interactively by comparing the output of the network with the desired value at each step. This scheme means that the training process teaches the net the expected output for each given input.

A feedforward MLP is a mapping function with n_0 input elements and n_L output elements. An MLP network is composed of L layers, with n_l (l = 1, 2, …, L) neurons in layer l. When the layer l is not 0 or L the layer is referred to as a hidden layer.

To formalize this description, if x_i^l is the output of the i^{th} neuron of layer l, w_{ij}^l is the weight of the j^{th} input (coming from the j^{th} neuron of the preceding layer) and ϕ^l is the activation function, then:

$$x_i^l = \phi^l\left(\bar{x}_i^l + b_i^l\right) = \phi^l\left(\sum_{j=1}^{n_{l-1}} w_{ij}^l x_j^{l-1} + b_i^l\right) \qquad (12)$$

where x_j^{l-1} is the output of the j^{th} neuron in the previous layer l-1 and b_i^l is the bias, introduced to allow the neuron to present a non-null output even in the presence of a null input [7].

While an MLP can have any number of layers, L, for most applications a single hidden layer, or two total layers, is sufficient as is shown in Figure 1. This system has p inputs, q hidden nodes, and r outputs. In matrix form, the complete expression can be formulated as:

$$y = \mathbf{w}_2\left[\phi\left(\mathbf{w}_1 x + b_1\right)\right] + b_2 \,. \qquad (13)$$

Training Algorithm

Training a neural net generally consists of applying a method to adjust or estimate the neuron weights. The training process minimizes the neural net output error through the application of an optimization method. All methods need to know how the net output varies with respect to the variation of a given neuron weight.

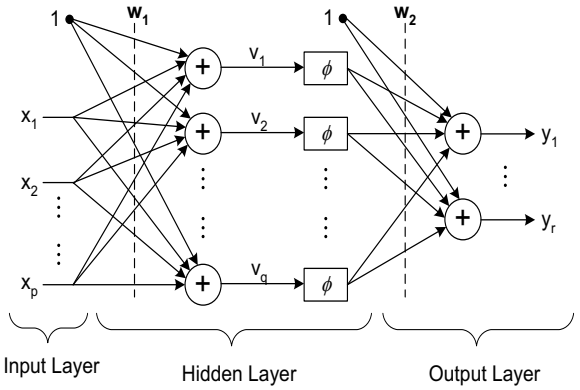

Figure 1: Typical Multi-Layer Perceptron Neural Network Structure

The backpropagation algorithm is an algorithm for learning in feedforward networks using mean squared error and gradient descent. The error that we choose to minimize for our training algorithm is:

$$J = \frac{1}{2}\sum_k e_k^2(n) \qquad \textbf{(14)}$$

calculated at a given time step, n.

The calculated error at the output layer is:

$$e_k(n) = d_k(n) - y_k(n) \qquad (15)$$

where d_k is the desired output of element k and y_k is the hidden layer output response specified by:

$$y_k(n) = \phi(v_k(n)) \,. \qquad (16)$$

In equation 16, v_k is the output layer activation level:

$$v_k(n) = \sum_j w_{kj}(n) y_j(n) \,. \qquad (17)$$

This continues through the hidden layer to the input layer:

$$y_j(n) = \phi(v_j(n)) \qquad (18)$$

and:

$$v_j = \sum_i w_{ji}(n) x_i \,. \qquad (19)$$

To calculate the gradient of the error function, we use the chain rule to get:

$$\begin{aligned} \frac{\partial J}{\partial w_{kj}} &= \frac{\partial J}{\partial e_k}\frac{\partial e_k}{\partial y_k}\frac{\partial y_k}{\partial v_k}\frac{\partial v_k}{\partial w_{kj}} \\ &= e_k(n)(-1)\phi'(v_k) y_j(n) \end{aligned} \qquad (20)$$

This derivative allows the weight update to be calculated, using a learning parameter, η_1:

$$\Delta w_{kj} = -\eta_1 \frac{\partial J}{\partial w_{kj}} \,. \qquad (21)$$

The resulting weight update is then:

$$\begin{aligned} w_{kj}(n+1) &= w_{kj}(n) + \Delta w_{kj} \\ &= w_{kj}(n) + \eta_1 e_k(n)\phi'(v_k) y_j(n) \end{aligned} \qquad (22)$$

To simplify the equations, we calculate an intermediate delta value, δ, for the output layer:

$$\delta_k \overset{\Delta}{=} e_k \phi'(v_k), \qquad (23)$$

which gives the final weight update equation for the output layer:

$$w_{kj}(n+1) = w_{kj}(n) + \eta_1 \delta_k y_j(n) \qquad (24)$$

This operation is repeated for the hidden layer:

$$\begin{aligned} \frac{\partial J}{\partial w_{ji}} &= \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial v_i}\frac{\partial v_i}{\partial w_{ji}} \\ &= -\sum e_k \phi'(v_k) w_{kj}) \phi'(v_j) y_i(n) \end{aligned} \qquad (25)$$

and again defining an intermediate δ:

$$\delta_j \overset{\Delta}{=} -\frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial v_j} = \sum e_k \phi'(v_k) w_{kj} \phi'(v_j) \qquad (26)$$

gives the final weight update:

$$w_{ji}(n+1) = w_{ji}(n) + \eta_2 \delta_j y_i(n). \qquad (27)$$

This training is carried out over a large training set, then over multiple epochs of that training set. For the spacecraft application, this training happens off-line to initialize the weights, and then continues on-line during flight operations.

Activation Function

The nonlinear neuron computes its activation level by adding all the weighted activations it receives. It then transforms this activation level into a response using an activation, or squashing, function ϕ. Several squashing functions can be used. The most common one is the logistic function:

$$\phi(x) = \log(x) = \frac{1}{1+e^{-x}} \qquad (28)$$

This function maps the set of real numbers into the [0, 1] range. The derivative is easy to compute:

$$\phi'(x) = \phi(x)(1-\phi(x)) \qquad (29)$$

which is useful for computing the backpropagation error.

The hyperbolic tangent is also often used when the response range is the interval [-1, +1]:

$$\phi(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad (30)$$

Its derivative is also easy to compute:

$$\phi'(x) = 1-\phi^{2}(x)\,. \qquad (31)$$

There are many such activation functions that can be used. Choice of which to use in which layers is left to the designer of the network based on the dynamics of the system.

Neural Network Control

The objective of the attitude control system design is to achieve attitude tracking maneuvers in the presence of mass parameter uncertainties and external disturbances.

As mentioned previously, a neural network controller has many benefits over a predetermined controller. However, there are still many methods in which to employ the use of neural networks in a control system. Most of these techniques, similar to more typical adaptive control systems, require prior knowledge of the system dynamics or a preconception of the control being used [8] [9].

It is the aim of this research to develop a neural network control system that would be solely based on the observed attitude motion of the spacecraft being controlled, and continue this training during flight operations [10]. Therefore, a self-tuning neural network controller was developed [11]. An overview of the neural network control design steps is given in Figure 2.

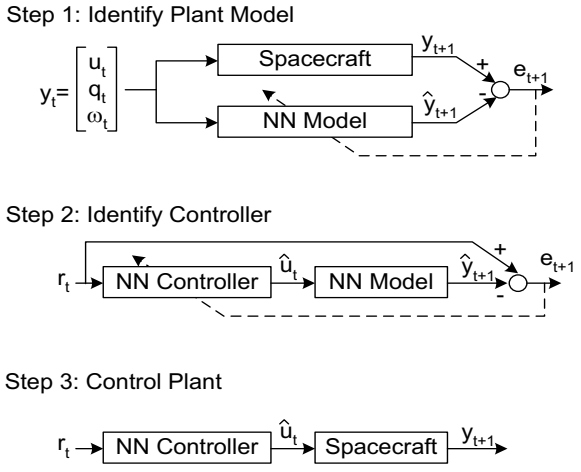

Figure 2: Training NN model and self-tuning controller

The following subsections outline each of these steps.

Identify Plant Model

Since the spacecraft can be considered an unknown system, in which only the inputs and outputs are known, a neural network model is created that is trained using the inputs and outputs from the nonlinear plant.

A hidden layer activation function $\phi(x) = tanh(x)$, is chosen due to its range corresponding well with the range of the quaternion vector. For the output layer, a linear activation function $\phi(x) = x$ is used.

Because the system is dynamic, the neural network is given a compilation of the control inputs, current state, and a past history, up to T, of states [9]. The expected plant output is computed:

$$\begin{bmatrix} q_{t+1} \\ \omega_{t+1} \end{bmatrix} = \mathbf{w}_2{}^m \tanh\left(\mathbf{w}_1{}^m \begin{bmatrix} u \\ q_t \\ \omega_t \\ \vdots \\ q_{t-T} \\ \omega_{t-T} \end{bmatrix} + b_1{}^m \right) + b_2{}^m \quad (32)$$

and compared with the actual spacecraft output.

Identify Controller

The self-tuning controller is connected in series with the newly trained neural network model. The output of the controller is the set of attitude torques, u, which are given to the model which calculates the system output, $\hat{y}$ as shown above in equation 32.

The input to the control system is r, the desired reference trajectory. The desired trajectory and resulting trajectory, $\hat{y}$, are compared and the result is the network error. However, this error is due to the dual-network of the model and controller. Therefore, the error must be backpropagated through the entire model without updating, and then propagated through the controller model with updating of the controller weights ($\mathbf{w_1^c}$, $\mathbf{w_2^c}$).

$$[u_t] = \mathbf{w}_2{}^c \tanh\left(\mathbf{w}_1{}^c \begin{bmatrix} r_t \\ q_{t-1} \\ \omega_{t-1} \\ \vdots \\ q_{t-T} \\ \omega_{t-T} \end{bmatrix} + b_1{}^c \right) + b_2{}^c \quad (33)$$

Control Plant

The entire system is combined as shown in Figure 3.

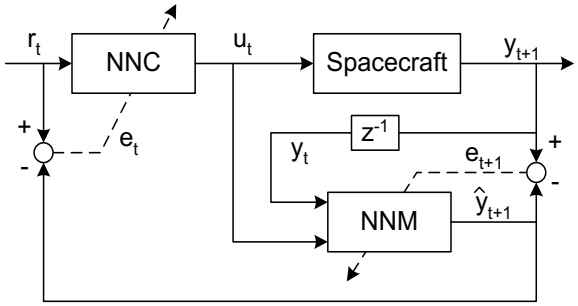

Figure 3: Neural network self-tuning controller with on-line training

The system runs real-time, determining control outputs, u, while updating its Neural Network Model (NNM) and Neural Network Controller (NNC).

Testing

Preliminary results show satisfactory modeling and control of a simple spacecraft with gravity gradient stability as seen in Figure 4.

Once software testing of the control system is completed, the adaptive attitude control system will be implemented on a spacecraft simulator table, and finally on Virginia Tech's Nanosatellite, HokieSat and compared with a LQR gain-scheduled controller [12].

Comparison

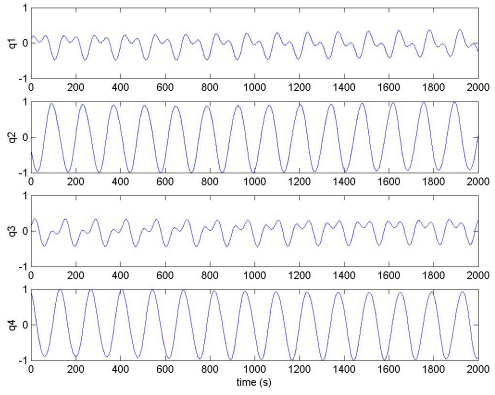

Figure 4: Neural network model of nonlinear spacecraft dynamics (solid-desired, dashed-neural net model) (nearly similar)

Conclusion

A neural network model and controller for use in adaptively controlling spacecraft attitude is developed. The self-tuning controller requires no prior knowledge of the spacecraft dynamics other than an estimated set of input-output training pairs. From this initialization, the model and controller continue to optimize on-line during flight operations, compensating for unknown external disturbances and system failures. Further work is to be done in verifying the robustness of this controller under severe system failures and for a variety of spacecraft control mechanisms and varying mass properties.

References

[1] Nguyen, D.H., Widrow, B. "Neural Networks for Self-Learning Control Systems." *International Journal of Control*. Vol. 54, No. 6, 1991, pp. 1439.

[2] Antsaklis, P.J., Passino, K.M., *et al*. "Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues*." Journal of Intelligent Robotic Systems*. Vol. 1, pp. 315-342, 1989.

[3] KrishnaKumar, K. "Adaptive Neuro-Control for Spacecraft Attitude Control." *Neurocomputing*, Vol. 9, No. 2, pp. 131-148. Oct 1995.

[4] Sheen, J.J., Bishop, R.H. "Adaptive Nonlinear Control of Spacecraft." *Proceedings of the American Control Conference*. Baltimore, Maryland, pp. 2867-2871. June 1994.

[5] Noriega, J.R., Wang, H. "A Direct Adaptive Neural-Network Control for Unknown Nonlinear Systems and Its Application," *IEEE Transactions on Neural Networks*, Vol. 9, No. 1, pp. 27-34. Jan 1998.

[6] Space News, 25-31 May 1998, p. 3

[7] Carrara, V., Varotto, S.E.C., Neto, A.R. "Satellite Attitude Control using Multilayer Perception Neural Networks." *Advances in the Astronautical Sciences, Spaceflight Dynamics.* Vol. 100. pp. 565-579. 1998

[8] Nardi, F., Calise, A.J. "Robust Adaptive Nonlinear Control using Single Hidden Layer Neural Networks." *Procs of the 39th IEEE Conference on Decision and Control.* Sydney, Australia. Dec 2000.

[9] Choi, M., Flashner, H. "Neural-Network-Based Spacecraft Attitude Control and Momentum Management." *AIAA Guidance, Navigation, and Control Conference and Exhibit.* Denver, CO, Aug 14-17, 2000.

[10] Hadaegh, F.Y., Lin, C., *et al.* "Neural Intelligent Control for Maneuvering Spacecraft." pp.259-263

[11] Cabrera, J.B.D., Narendra, K.S. "Issues in the Application of Neural Networks for Tracking Based on Inverse Control." *IEEE Transactions on Automatic Control.* Vol. 44, No. 11. pp. 2007-2027. Nov 1999.

[12] Makovec, K. Turner, A. "Design & Implementation of a Nanosatellite Attitude Determination & Control System." *2001 AAS/AIAA Astrodynamics Specialists Conference*, Quebec City, Canada, Jul 30 – Aug 2, 2001.