JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 21, No. 1, January–February 1998

Neural Network Control of Space Vehicle Intercept and Rendezvous Maneuvers

Elisabeth A. Youmans* and Frederick H. Lutze†
Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061

Neural networks are examined for use as optimal controllers. The effect of the addition of noise to the neural network input measurements is investigated to determine the performance robustness of the neural network controllers. These techniques are applied to the autonomous control of interceptor-to-target rendezvous missions. For this example, the target lies in a circular orbit and remains passive throughout the maneuver. The linearized Clohessy–Wiltshire equations with thrust are used to describe the relative motion of the two vehicles. Parameter optimization is used to generate the training data for the neural network designs. A combination of open-loop and closed-loop control is shown to work effectively for this problem.

Nomenclature

$\boldsymbol{a}$	= vector of neural network output values
E	= total sum-squared error between the neural network outputs and the desired outputs
H	= variational Hamiltonian of the system
J	= cost function to be minimized
m	= total mass of interceptor vehicle, kg
n	= total number of input–output training pairs
r	= radial distance between interceptor and target, km
T	= thrust magnitude, N
$T_{\max}$	= maximum thrust level, N
t	= time, s
$\boldsymbol{u}$	= control vector
V_e	= exit velocity of the exhaust gases, km/s
v_x, v_y	= components of interceptor velocity vector, m/s
x, y	= components of the interceptor position vector, m
$\boldsymbol{x}$	= state vector
$\boldsymbol{z}$	= vector of desired neural network output values
α	= line of sight angle between target and interceptor, deg
θ	= thrust angle, rad
λ_i	= Lagrange multiplier associated with the ith state
ν	= multipliers associated with the final state conditions
σ	= thrust switching function
τ	= nondimensional time, identical to ωt
ϕ	= function of the final time and the state at the final time
Ψ	= final state-variable constraints
ω	= angular rate of the target vehicle in its orbit

Subscript and Superscripts

f	= final time
T	= transpose of vector or matrix
$'$	= differentiation with respect to nondimensional time, τ

Introduction

SINCE the early 1980s, neural networks have been used in a wide variety of applications. Lisboa[1] summarized many areas where neural networks have helped solve problems, including medical risk prediction, airline revenue management, and nonlinear chemical process modeling. Neural networks are also being used as controllers as well as parts of control systems. Iiguni et al.[2] used two neural networks with a linear quadratic regulator to estimate and compensate for regulator modeling uncertainties.

Nguyen and Widrow[3] designed a neural network truck backer-upper system. Their approach was interesting because they initially used a neural network to emulate the truck's kinematics. The outputs of that network served to design a second neural network controller to perform the actual operation of the truck. Grogan[4] applied neural networks to the aircraft control allocation problem. He demonstrated how neural networks can learn to model the available moment subset for an aircraft.

Finally, there has been growing interest in using neural networks to solve optimal linear and nonlinear control problems. Lee and Smyth[5] designed a neural network to approximate minimum-time state-to-controls mapping. They used optimal time histories of the states and controls as training data so that the neural network can operate in a closed-loop feedback mode. This technique was applied to a low-order approximation of the minimum-time orbit injection problem with one control variable. Similar ideas for designing and applying neural networks are used in this paper. For this application, however, there are five states and two control variables, and the behavior of the controller is examined in the presence of noise. Basic neural network concepts and definitions are discussed next, followed by their application to the minimum-fuel intercept and rendezvous problem.

Neural Networks

A neural network is a simple system of interconnections and mathematical computations. It takes inputs, processes them, and produces an output. The general structure of a neural network contains an input layer, a hidden layer of processing elements called neurons, and an output layer (Fig. 1). Neurons have a squashing, or transfer, function[6] associated with them to process the input signals they receive. A squashing function, such as the hyperbolic tangent function used in this work, takes the input data and maps their range, for example, from -1 to 1. These transfer functions aid in the training of the neural network, which will be discussed later. The neural network designer does not know a priori how many neurons will be required in the hidden layer to provide the best performance. Therefore, a series of trial and error designs must be done, but they are intelligently selected based on the experience of the neural network designer.

Each neuron can receive any number of inputs. The inputs have a weight, or multiplier, associated with them that emphasizes or deemphasizes the input's contribution to the system's evaluation process. The outputs of the neurons inside the hidden layer have weights associated with them, too. Constant signals, called biases, can also be included in the network. All weights and biases compose the memory of the network.

Determination of the weights and biases is done in a training process. In this process, the neural network is exposed to a representative set of inputs it will encounter during its operation and the

Received June 13, 1995; revision received Aug. 19, 1997; accepted for publication Sept. 23, 1997.

*Research Assistant, Aerospace and Ocean Engineering; currently Manager and Analyst, Systems Planning and Analysis, Inc., 2000 North Beauregard Street, Suite 400, Alexandria, VA 22311. Member AIAA.

†Professor, Aerospace and Ocean Engineering, 215 Randolph Hall. Associate Fellow AIAA.

Table 1 Hidden layer structure of the neural networks

Neural network	Number of neurons in hidden layer
Open loop: computes first thrust burn time	30
Open loop: computes first thrust angle	25
Open loop: computes coast time	30
Open loop: computes second thrust burn time	30
Open loop: computes second thrust angle	35
Closed loop: computes sine of thrust angle	25
Closed loop: computes cosine of thrust angle	25

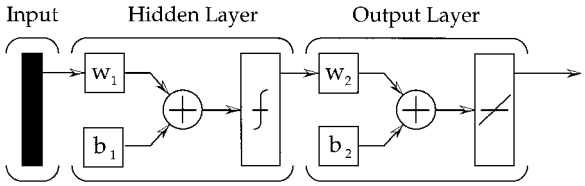


Fig. 1 Neural network structure.

outputs it will be expected to approximate. Training involves adjusting the weights and biases at the network input and at the output of the hidden layer to reduce the sum-squared error between the network output and the desired output. That is, the training process seeks to minimize

$$E = \sum_{i=1}^{n} (z_i - a_i)^2 \tag{1}$$

where z_i is the ith desired output, a_i is the ith neural network output, and n is the number of input–output training pairs for a particular neural network. Weights and biases are adjusted according to a learning method, and their values are updated using the backpropagation technique.[6]

Backpropagation was a breakthrough in neural network design. It allows for the error signal between the output of the neural network and the desired output to propagate back through the network so that the weights and biases can be adjusted. It allows for a variety of learning methods. For this paper, the Levenberg–Marquardt learning method[7] was used. This method combines elements of the steepest descent and Gauss–Newton methods to select the weights and biases that minimize the error E.

Neural networks can be trained on-line and continue learning, or adjusting weights and biases, during their operation. The neural networks designed in this paper, however, are trained off-line using what is called supervised learning. Once the training is completed, the networks perform solely as interpolating devices. The neural network performance is then tested with inputs it has never been given before to see how well the output compares to known desired outputs.

The "MATLAB Neural Network Toolbox"[8] was used to design the neural networks. This software contains many programs necessary for neural network design, including network initialization, training, and testing. The structure of the neural network, choice of processing element transfer functions, and necessary training data preparation are determined by the designer.

The general structure of all of the neural networks designed in this paper is given in Fig. 1. The hidden layer neurons have the hyperbolic tangent squashing function associated with them. The networks designed differ only in the number of neurons chosen to compose the network's hidden layer. Trial and error determined the best number of hidden layer neurons. Best is defined by that number of hidden layer neurons that, when increased, does not improve the performance of the network in minimizing Eq. (1). Table 1 summarizes the hidden layer composition of each neural network designed for the intercept and rendezvous application. All of the weights and biases for each neural network are given in Ref. 9. The specific problem to be addressed will now be introduced.

Minimum-Fuel Intercept and Rendezvous Problem

Neural networks are applied to the minimum-fuel intercept and rendezvous problem. Two vehicles are orbiting above the Earth. It is desirable to actively control one vehicle, called the interceptor, to autonomously rendezvous with another, called the target, using minimal-fuel expenditure. Figure 2 shows the two vehicles and the coordinate system representing their relative positions, including the relative line-of-sight angle α. The target vehicle is assumed to be in a circular orbit 400 km above the Earth, and it remains passive throughout the maneuver. The interceptor is assumed to lie in the same orbital plane as the target.

Table 2 Interceptor vehicle properties

Specification	Value
Total mass of vehicle	20.62 kg
Mass of fuel	1.50 kg
Thrust level	410 N
Exhaust velocity	2.352 km/s

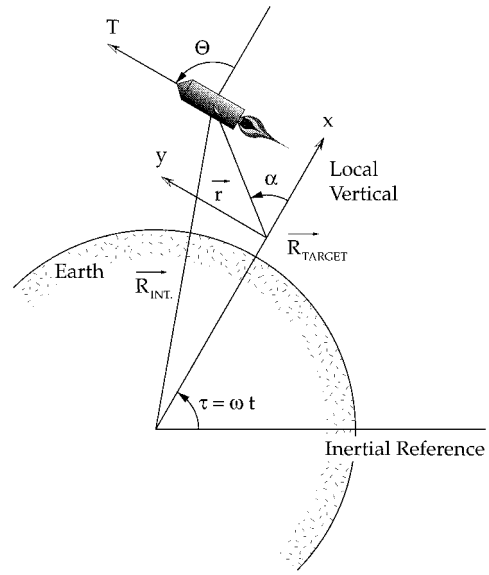


Fig. 2 Visualization of the interceptor–target relative motion.

The properties of the intercept vehicle are given in Table 2. This satellite is very small and was not originally intended for autonomous rendezvous missions. This work is an effort to expand the use of this kind of satellite to include retrieval and rendezvous missions.

The in-plane linearized Clohessy–Wiltshire[10] equations with thrust are used to describe the relative motion between the interceptor and the target. The state vector is $\boldsymbol{x} = \{x \;\; y \;\; v_x \;\; v_y \;\; m\}^T$, and the control vector is $\boldsymbol{u} = [\theta, T]^T$. The governing equations of motion are

$$x'(\tau) = v_x(\tau) \tag{2a}$$

$$y'(\tau) = v_y(\tau) \tag{2b}$$

$$v_x'(\tau) = 2v_y(\tau) + 3x(\tau) + \frac{T\cos\theta}{m(\tau)\omega^2} \tag{2c}$$

$$v_y'(\tau) = -2v_x(\tau) + \frac{T\sin\theta}{m(\tau)\omega^2} \tag{2d}$$

$$m'(\tau) = -(T/V_e\omega) \tag{2e}$$

with initial conditions

$$0 \le x(0) \le 34 \text{ km} \tag{3a}$$

$$0 \le y(0) \le 34 \text{ km} \tag{3b}$$

$$v_x(0) = 0 \tag{3c}$$

$$v_y(0) = 0 \tag{3d}$$

$$m(0) = 20.62 \text{ kg} \tag{3e}$$

For this satellite, the thrust is either on or off, but when on, the thrust direction is variable.

We want to conserve fuel during the rendezvous maneuver; hence, the cost function to be minimized is the negative of the final mass:

$$J = \phi[\boldsymbol{x}(\tau_f), \tau_f] = -m(\tau_f) \tag{4}$$

with final time constraints

$$\Psi = [\boldsymbol{x}(\tau_f), \tau_f] = \begin{bmatrix} x(\tau_f) \\ y(\tau_f) \\ v_x(\tau_f) \\ v_y(\tau_f) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

The variational Hamiltonian for this problem is given by

$$H = \lambda_x v_x + \lambda_y v_y + \lambda_{v_x}\left[2v_y + 3x + \frac{T\cos\theta}{m(\tau)\omega^2}\right] + \lambda_{v_y}\left[-2v_x + \frac{T\sin\theta}{m(\tau)\omega^2}\right] - \lambda_m\left(\frac{T}{V_e\omega}\right) \tag{6}$$

The necessary conditions for optimality are given by

$$\lambda_i' = -\frac{\partial H}{\partial x_i} \tag{7}$$

Pontryagin's maximum principle states that the optimal control will always maximize the Hamiltonian:

$$\boldsymbol{u}^* = \max_{\boldsymbol{u}} H(\boldsymbol{x}, \lambda, \tau, \boldsymbol{u}) \tag{8}$$

Finally, the transversality conditions require

$$\left(\lambda^T + \frac{\partial \phi}{\partial x_i} + \nu^T \frac{\partial \Psi}{\partial x_i}\right)\Bigg|_{\tau=\tau_f} = 0 \tag{9a}$$

$$\left(\frac{\partial \phi}{\partial \tau_f} + \nu^T \frac{\partial \Psi}{\partial \tau_f} - H\right)\Bigg|_{\tau=\tau_f} = 0 \tag{9b}$$

Because the Hamiltonian is linear in the control T, the necessary conditions lead to the definition of a switching function,

$$\sigma = \frac{\partial H}{\partial T} = \lambda_{v_x}\frac{\cos\theta}{m(\tau)\omega^2} + \lambda_{v_y}\frac{\sin\theta}{m(\tau)\omega^2} - \frac{\lambda_m}{V_e\omega^2} \tag{10}$$

that dictates the thrust and coast requirements to maximize the Hamiltonian. The optimal thrust scheme is, therefore, given by

$$T = \begin{cases} T_{\max} & \sigma > 0 \\ 0 & \sigma < 0 \end{cases} \tag{11}$$

Because the thrust is either constant at its maximum value or zero, we do not consider intermediate thrust levels; that is, we rule out singular control where $\sigma = 0$ for finite time. We determine the optimal thrust direction from Eq. (8).

The complete solution of this problem is given in Ref. 9. Solutions to the described problem were obtained for various initial conditions. It was observed that the optimal trajectories were composed of a thrust–coast–thrust control scheme. The solution to this problem for any given initial point consists of time histories of the thrust angle when the thrust is on and the switching times for turning thrust on or off.

The motivation for considering a neural network for this problem was to implement a closed-loop controller. One of the drawbacks of nonlinear optimal control is that the solutions are open loop and, hence, are not suitable for the case where there are possibilities of measurement or system modeling errors. Consequently, the idea was to solve a large range of optimal control problems that give open-loop solutions based on the maximum principle. Because the optimal control is known for special combinations of states along these solutions, they can be used to generate input for training a neural network. If we use a large range of trajectories as training data, then the neural network will provide an approximation to a closed loop optimal controller.

Unfortunately, we find that for long distances away, the number of state combinations becomes staggering; therefore, some simplifications and assumptions must be made. A neural network controller was, therefore, designed to operate open loop to bring the interceptor vehicle to within a small neighborhood of the target. Following the open-loop segment, a second neural network controller operating closed loop will complete the terminal phase of the rendezvous mission.

Because the control is constrained to be on or off with a fixed magnitude when it is on, linear regulator theory does not apply directly, and no closed-loop solutions are available. Other nonlinear closed-loop control methods might include a Lyapunov-type controller, but these were not investigated. The design of the open-loop and closed-loop neural network controllers are discussed in the next sections.

Neural Network Training Data and Open-Loop Design

For this problem, the times for thrust-on segments were relatively short, and the thrust angles could be approximated with constant values. The difference between the fuel expenditure of the maneuver using the approximate constant-angle scheme and that using the optimal scheme was $\ll 1\%$. Consequently, the minimum-fuel rendezvous trajectories could be characterized by five parameters. These parameters were the initial and final thrust burn times, their associated constant thrust angles, and the intermediate coast time. Rather than solving the true optimal control problem, a Schittkowski[11] sequential quadratic programming (SQP) algorithm was used to solve the now nonlinear programming problem for the parameters that characterize these near-optimal trajectories for different sets of initial conditions.

Neural networks were trained for control of the aforementioned open-loop problem. The near-optimal open-loop parameters were computed for more than a thousand initial conditions. The parameters for remaining initial conditions were interpolated by the neural network controller. Five neural networks were designed, each computing one of the five parameters that characterize the near-optimal trajectory. One neural network could not compute all five parameters simultaneously with the required accuracy. The neural networks took the distance of the interceptor vehicle from the target and the line-of-sight angle α (see Fig. 2) as inputs and generated the open-loop parameters required for rendezvous from that position. Unlike Ref. 3, where neural networks were designed to model the dynamics of the problem, the neural networks in this paper were used to implement a set of learned responses as determined by the optimal control formulation and the parameters that represent the trajectory.

To reduce the number of combinations of the initial state, it was assumed that there is zero relative velocity between the two vehicles when the rendezvous maneuver commences [Eqs. (3c) and (3d)]. Furthermore, at small line-of-sight angles, the optimal trajectory requires more fuel than available on this small satellite for distances larger than 34 km between the target and the interceptor; therefore, the interceptor for this problem were assumed to lie within 34 km from the target.

Symmetry holds for the controller, too. The open-loop neural network controller needed only be trained on two of four quadrants for line-of-sight angles α (see Fig. 2). The open-loop thrust angle parameters were shifted by π radians to accommodate initial conditions with line-of-sight angles lying in the remaining two quadrants.

Programs were written in the MATLAB environment to simulate the open-loop maneuver. Fourth- and fifth-order Runge–Kutta integrators were used to update the state variables.

Table 3 compares the neural-network-generated open-loop parameters with those of the near-optimal solution. Although the parameters compare well, differences in the nondimensional burn time $\mathrm{d}\tau = \tau_{\mathrm{des}} - \tau_{\mathrm{nn}}$ actually correspond to differences in real time $\mathrm{d}t = \mathrm{d}\tau/\omega \cong 885 \times \mathrm{d}\tau$ for target angular rate $\omega = 0.00113$ rad/s. This difference in burn time has a significant effect on the interceptor vehicle's state after completing the open-loop segment. Therefore, a closed-loop controller was designed to complete the mission after the end of the open-loop thrust–coast–thrust segments.

Table 3 Comparison of neural network (NN) computed open-loop parameters with optimal (Opt) parameters

r, km/α, deg	$\tau 1$, NN/Opt	$\tau 2$, NN/Opt	$\tau 3$, NN/Opt	Thrust 1, rad, NN/Opt	Thrust 2, rad, NN/Opt
8/40	0.06555/0.0654	2.1555/2.1422	2.1724/2.1548	−1.9784/−1.9806	−1.7651/−1.7335
4/−65	0.01993/0.0200	4.4440/4.4351	4.4426/4.4449	−1.6995/−1.70275	−3.1442/−3.1196
20/−30	0.2004/0.2001	4.0802/4.0930	4.1643/4.1714	−1.8734/−1.8747	−2.9773/−2.9782
27/75	0.07197/0.07182	3.7784/3.7719	3.7823/3.7836	−1.4525/−1.4508	−1.5541/−1.5492
10/25	0.09967/0.0998	1.9766/1.9716	1.9941/1.9953	−2.1301/−2.1307	−1.8985/−1.9085
17/50	0.12658/0.1271	4.2254/4.2161	4.2713/4.2714	−1.8161/−1.8149	−3.0415/−3.040
15/50	0.10271/0.1026	2.4332/2.4304	2.4391/2.4474	−1.9026/−1.902	−1.6874/−1.6883

Open-Loop/Closed-Loop Neural Network Design

The aim of the open-loop controller was to drive the interceptor vehicle to within a small neighborhood of the target. The number of input–output training pairs needed to design a closed-loop neural network controller to complete the rendezvous maneuver was greatly reduced. This new closed-loop controller used continuous thrust, i.e., $T = T_{\max}$ at all times, and updated the thrust angle at intervals of 0.001τ, or 0.885 s, during the maneuver.

We used continuous thrust for two reasons. One, because the third open-loop segment used $T = T_{\max}$, it was logical to extend this segment until tolerance was met. The second reason is the neural network controller does not know its history. Each update must treat the current conditions as initial conditions. Such a controller would need to compute thrust level (on or off) and when on, compute the thrust angle at each update. The training set for this kind of controller would contain similar state vectors with different thrust levels. A neural network controller designed for a one-dimensional bang-bang control problem[9] was shown to have difficulties determining thrust levels for states close to a switching point. A closed-loop, regularly updating thrust–coast–thrust neural network controller was, therefore, not feasible. The fixed-thrust controller designed required updating the thrust angle only for its current state.

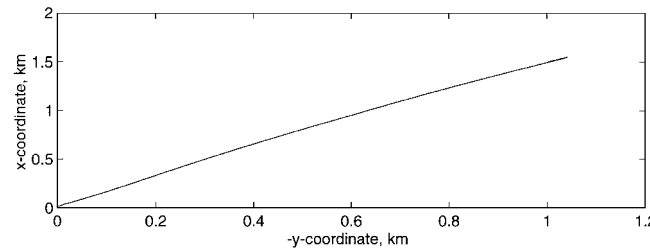
The continuous-thrust trajectory was approximated with three parameters: two constant thrust angles and a switching time. SQP was used once again to generate the state-control training pairs. Two training pairs were generated for each initial condition: the initial state and its corresponding thrust angle and the state at the switching time with its corresponding thrust angle. The range of the position and velocity components for training and an average total mass value were determined based on 10 open-loop simulation results. The maximum position range after these simulations was about 2-km radial distance, and the maximum velocity range was about 10 m/s. A constant value representing about 97% of the total mass was used as the initial mass for all trajectories computed for training the closed-loop controller.

Because the thrust angles were updated on a regular basis, the switching time did not enter into determining how the neural network controls the vehicle. At each update, the closed-loop controller computed the thrust angle as if the current state were the initial condition of the vehicle. Two neural networks were designed to control the closed-loop segment of the rendezvous problem. These neural networks generated the sine and cosine of the desired thrust angle. A normalized combination of the outputs of these networks determined the actual vehicle thrust angle.

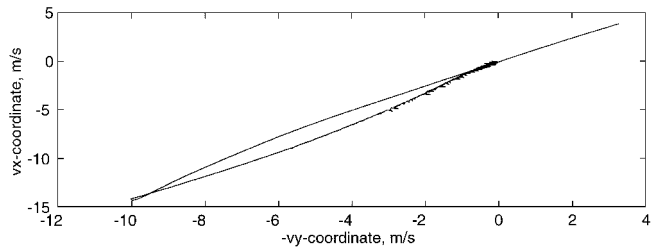
Figure 3 gives the neural-network-controlled position and velocity trajectories for one set of initial conditions within the closed-loop training range. These initial conditions were not part of the training set; the neural networks were interpolating to determine the control parameters. All cases tested within the closed-loop control range resulted in the neural networks successfully driving the interceptor to a point of rendezvous within tolerance to the target. Tolerance for this problem was set at 15-m radial distance from the target vehicle with a relative velocity of 0.015 m/s.

The complete open-loop/closed-loop combination of control systems were examined for 50 simulations. Figures 4 and 5 show the neural-network-controlled rendezvous maneuvers for two of these cases. The initial conditions for Fig. 4 were a training pair, but those of Fig. 5 required interpolation between training pairs. Because the closed-loop segment used continuous thrust, the fuel expenditure of the complete maneuver departed from the optimal fuel expenditure. An average of two to three times more fuel was expended compared to the optimal solution. For time-critical rendezvous situations, however, it may be more desirable that the controller reach tolerance than conserve some of its fuel.

All but 2 of the 50 open-loop/closed-loop test cases reached tolerance. The open-loop segment of the two failure cases left the interceptor outside of the training range of the closed-loop controller, thus revealing that neural networks do not extrapolate very well

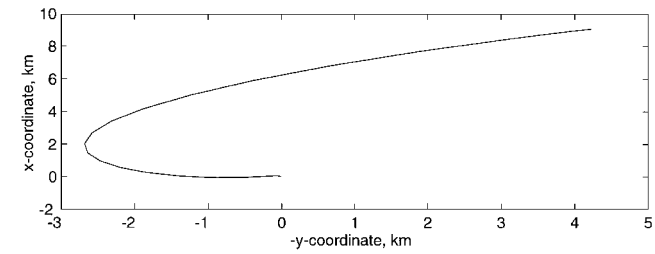


Inner neighborhood position trajectory

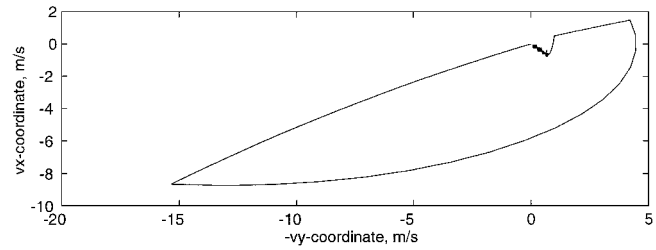


Inner neighborhood velocity trajectory

Fig. 3 Closed-loop neural-network-controlled intercept and rendezvous maneuver for initial conditions $\{x_0, y_0, v_{x_0}, v_{y_0}\}$ = {1.5 km, 1 km, 3.95 m/s, 3.39 m/s}.

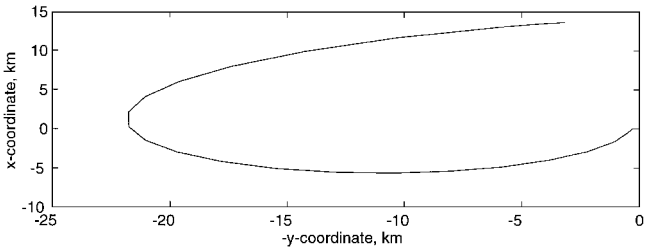


Neural network position trajectory

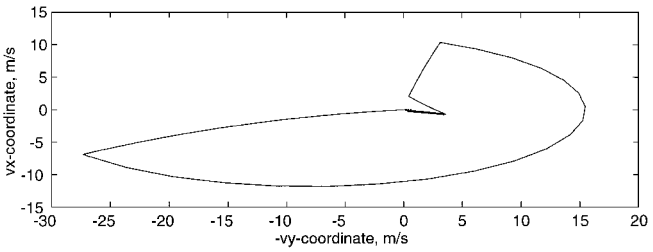


Neural network velocity trajectory

Fig. 4 Open-loop and closed-loop neural-network-controlled intercept and rendezvous maneuver for initial conditions r = 10 km, α = 25 deg, and zero relative velocity.

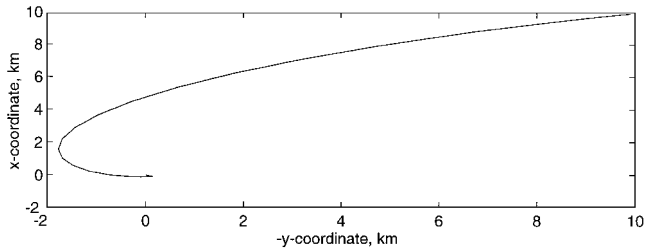


Neural network position trajectory

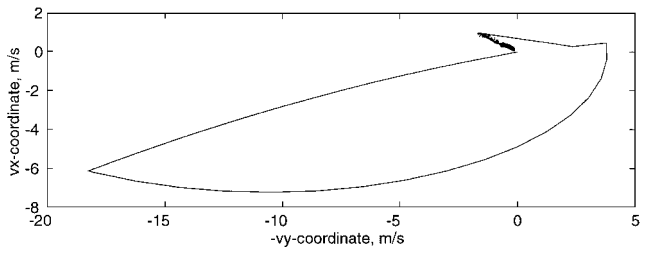


Neural network velocity trajectory

Fig. 5 Open-loop and closed-loop neural-network-controlled intercept and rendezvous maneuver for initial conditions $r = 14$ km, $\alpha = -13$ deg, and zero relative velocity.



Position trajectory with noise



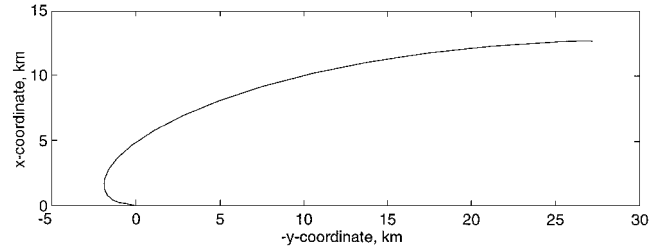
Velocity trajectory with noise

Fig. 6 Open-loop and closed-loop neural-network-controlled intercept and rendezvous maneuver with noise for initial conditions $r = 14$ km, $\alpha = 45$ deg, and zero relative velocity.

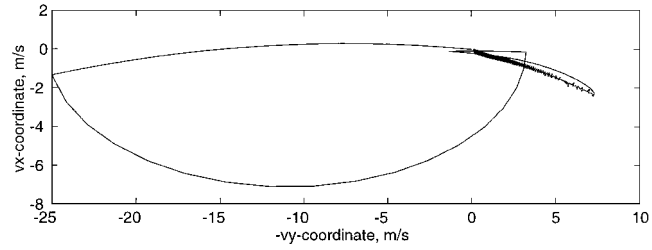
beyond their training range. If initial efforts to drive the interceptor to rendezvous leave the interceptor within a small neighborhood of the target, then the closed-loop neural network controller could be used to home in on the target vehicle for the remainder of the mission. One problem encountered with the closed-loop controller, however, was that there were rapid changes in thrust direction, which can be very difficult to control and can cause wear on the hardware.

Neural Network Controllers with Noise

Thus far, we have assumed that the neural network inputs are measured perfectly. It is interesting to see how well these networks operate in the presence of noise, or slightly inaccurate state variable measurements. We have done a rudimentary analysis of the effect of noise on the controller performance. To simulate the noise in the measurements, a random-number generator was used to provide noise elements with zero mean and standard deviation of one. The random number generated was multiplied by 0.0025, and the state variables were modified by this resulting quantity. The state variables could, therefore, be modified by as much as 0.25% of their original value.



Position trajectory with noise



Velocity trajectory with noise

Fig. 7 Open-loop and closed-loop neural-network-controlled intercept and rendezvous maneuver with noise for initial conditions $r = 30$ km, $\alpha = 65$ deg, and zero relative velocity.

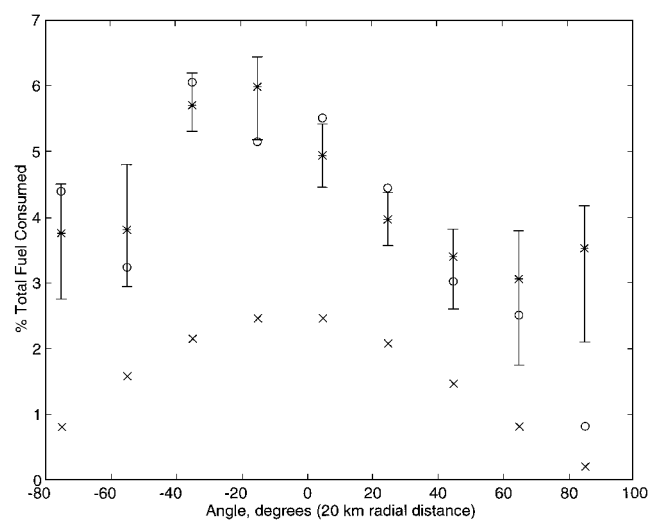


Fig. 8 Comparison of fuel consumption of neural-network-controlled maneuvers with noisy measurements to the noise-free and optimal cases for initial conditions $r = 20$ km and $\alpha = -75, -55, -35, -15, 5, 25, 65,$ and 85 deg; mean = 0, and standard deviation = 0.0025* × (i).

Figures 6 and 7 give the trajectories for cases containing noise for initial conditions of radial distance 14 km and line-of-sight angle 45 deg (Fig. 6) and radial distance 30 km and line-of-sight angle 65 deg (Fig. 7). For the case of Fig. 6, the fuel expenditure for the noise-free case was 0.12% higher than the same case with noise. This result can be expected because the neural networks are interpolating between training points. Slight errors in the interpolation can cause the neural networks to give more accurate parameter outputs, but the reverse is also true. For the case of Fig. 7, the noise case used 0.77% more fuel than the noise-free case. It is important to note, however, that both cases resulted in the vehicle reaching tolerance, even in the presence of noisy measurements.

To recognize any trends in the behavior of the neural networks in the presence of noise, the initial conditions tested were along constant radial distances with different line-of-sight angles. For each initial condition, 10 sessions were executed, and the average fuel expenditure was computed. Figure 8 shows the fuel expenditure for radial distance $r = 20$ km. The error bars give the extreme fuel consumption values for the 10 sessions, the ○s correspond to the noise-free cases, and the ×s give the optimal fuel expenditures.

At least one noise case for each initial condition resulted in improved performance compared to the noise-free case. On the whole, however, the noise did not cause a significant change in the neural network controller operation. These results demonstrate how the controllers are performance robust.

Conclusions

The feasibility of neural networks for open-loop and closed-loop control system design was examined and applied to the minimum-fuel intercept and rendezvous problem. Initial condition constraints were required to have a reasonably sized training set for the neural network design. For higher dimensional control problems, this approach is probably not feasible. It was shown that approximations to the optimal solution are needed because of the complexity of the rendezvous problem. Parameter optimization was shown to be an excellent method of generating parameters and, hence, the training data for the near-optimal solution.

The open-loop controller successfully drove the interceptor to within a small neighborhood of the target for most cases tested. The closed-loop controller drove the interceptor to a point of rendezvous for every initial condition tested within the closed-loop grid training set. This terminal segment was no longer fuel efficient; rather, it sought to drive the vehicle to within tolerance in minimum time. Although the closed-loop controller caused the interceptor vehicle to expend two to three times more fuel on average than the optimal case, this may still be acceptable for real-life applications. Neural network operation in the presence of noise was not significantly altered, thus demonstrating performance robustness of the controllers.

References

[1]Lisboa, P., *Neural Networks: Current Applications*, Chapman and Hall, New York, 1992, Chap. 1.

[2]Iiguni, Y., Sakai, H., and Tokumaru, H., "A Nonlinear Regulator Design in the Presence of Uncertainties Using Multilayered Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 2, No. 4, 1991, pp. 410–417.

[3]Nguyen, D. H., and Widrow, B., "Neural Networks for Self-Learning Control Systems," *IEEE Control Systems Magazine*, Vol. 10, No. 2, 1990, pp. 18–23.

[4]Grogan, R. L., "On the Application of Neural Network Computing to the Constrained Flight Control Allocation Problem," M.S. Thesis, Aerospace and Ocean Engineering, Virginia Polytechnic Inst. and State Univ., Blacksburg, VA, May 1994.

[5]Lee, A. Y., and Smyth, P., "Synthesis of Minimum-Time Feedback Laws for Dynamic Systems Using Neural Networks," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 4, 1994, pp. 868–870.

[6]Hecht-Nielsen, R., *Neurocomputing*, Addison–Wesley, Reading, MA, 1990, Chaps. 2 and 5.

[7]Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic, London, 1981, pp. 136, 137.

[8]Demuth, H., and Beale, M., "MATLAB Neural Network Toolbox," MathWorks, Inc., Natick, MA, 1994.

[9]Youmans, E. A., "Neural Network Control of Space Vehicle Orbit Transfer, Intercept, and Rendezvous Maneuvers," Ph.D. Dissertation, Aerospace and Ocean Engineering, Virginia Polytechnic Inst. and State Univ., Blacksburg, VA, May 1995.

[10]Clohessy, W. H., and Wiltshire, R. S., "Terminal Guidance System for Satellite Rendezvous," *Journal of the Aerospace Sciences*, Vol. 27, Sept. 1960, pp. 653–658.

[11]Schittkowski, K., "NLPQL: A FORTRAN Subroutine Solving Constrained Nonlinear Programming Problems," *Annals of Operations Research*, Vol. 5, 1985–1986, pp. 485–500.