DUJASE Vol. 3(2) 155-159, 2016 (January)

Non-linear model inversion control for air vehicle system using neural networks and particle swarm optimization

Jakia Afruz[1] and M. S. Alam[2]

[1]*University of Liberal Arts, Dhaka, Bangladesh*

[2]*Department of Electrical and Electronic Engineering, University of Dhaka, Bangladesh*

E-mail: msalam@du.ac.bd

Received on 24.08.2015. Accepted for publication on 09.09.2015.

ABSTRACT

This paper investigates the development of a non-linear model inversion controller for air vehicle system utilizing artificial neural networks and particle swarm optimization (PSO). An adaptive neural network element is integrated with feedback control system to compensate for model inversion errors. Two control structures: direct inverse model and internal model control are considered in this work. A twin rotor multi-input multi-output system (TRMS) is considered as a test rig for air vehicle system. A non-linear inverse model is developed for the pitch movement of TRMS utilizing artificial neural network (ANN) which is used as one of the main components of control system for input tracking. A relatively new population-based, self-adaptive optimization, PSO, is used to train the inverse model in order to avoid premature convergence to local minima. The control scheme shows good tracking capability with satisfactory level of rise time, settling time and steady state error.

Keywords: Control, Inverse Modeling, Neural Networks, Particle Swarm Optimization, Twin Rotor System.

1. Introduction

Recent advances in air vehicle system have led to the development of many new concepts in aircraft design. Compared to conventional fixed-wing aircraft, helicopters are much more complex in terms of system dynamics and control because the inputs are not directly applied torques or forces, but rather aerodynamic torques and forces created by the main and tail rotors. Moreover, a significant cross-coupling exists between the rotors which in turn increases system nonlinearities and uncertainties. These system characteristics present formidable challenges in modelling, control design, and implementation. A scaled and simplified version of practical helicopter, namely twin rotor multi-input multi-output system, (TRMS)[1] can be perceived as "air vehicle" and is being used as an interesting 'test rig' for aerodynamic modelling and control problems.[2-4] Dynamic inversion[5] is an interesting feedback linearization control system design methodology that continues to find a wide acceptance in the control engineering community, with many applications and extensions, particularly in the field of flight control of air vehicle systems.[4-5]

Artificial Neural Networks (ANNs) are circuits, computer algorithms, or mathematical representations loosely inspired by the massively connected set of neurons that form biological neural networks.[6] The universal approximation capabilities of ANN have made it a popular choice for modeling nonlinear systems and for implementing general-purpose nonlinear controllers.[7] The neural network may often get stuck in a local minimum with standard backpropagation. To enable the ANN to slide through a local minimum, several modifications and new algorithms have been proposed and are used in practice.

Particle swarm optimization (PSO) is a population-based, self-adaptive search optimization technique.[8-9] PSO has proved to be efficient at training NNs and solving unconstrained global optimization problems mainly due to its simplicity, low memory requirement, low computational cost, fast convergence and its good overall performance.[10]

The objective of this paper is to design controller for pitch movement of TRMS using ANN and PSO algorithm. An inverse model of TRMS is developed using neural network which is trained by PSO algorithm. The designed inverse model is used as the main components of a feedback control scheme to control the movement of TRMS. Two control structures: direct inverse model and internal model control are designed for input tracking of TRMS and their performances are tested.

2. Neural Network

A generalized architecture of multi-layer perceptrons (MLP) representing its basic functions is shown in Figure 1. A basic unit of this network or a neuron performs two functions viz, the combining functions and the activation functions.

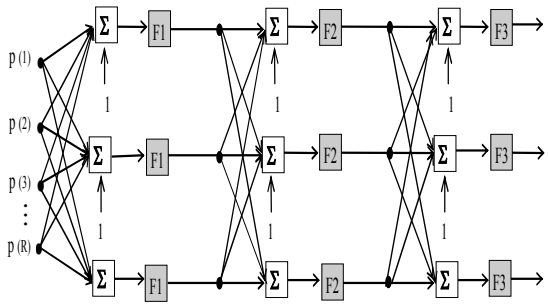

Fig. 1: Multiple layers of feedforward neural network

The combining function produces activation for the neuron of the form,

$$v_j^m(t) = \sum_{i=1}^{n_{m-1}} w_{ij}^m x_i^{m-1} + b_j^m \qquad (1)$$

where w_{ij}^m is the weight connection between the i th neuron of the $(m-1)$ th layer and the j th neuron of the m th layer, b_j^m is the threshold of the neuron and n_{m-1} is the number of neuron in the $(m-1)$ th layer. The activation function performs a non-linear transformation to give the output,

$$x_j^m(t) = f(v_j^m(t)) \quad (2)$$

where $f(\cdot)$ is the non-linear transformation or activation function. By combining equation (1) and (2) the output of a functional unit of the neuron can be expressed as,

$$x_j^m(t) = f(\sum_{i=1}^{n_{m-1}} w_{ij}^m x_i^{m-1} + b_j^m) \quad (3)$$

The functionality of the network is determined by specifying the strength of the connection paths, called weights, and the threshold parameter of each neuron. The input layer usually acts as an input data holder and distributes inputs to the first hidden layer. The inputs then propagate forward through the network and each neuron computes its output according to the learning rule chosen.

3. Control Structures

The control structures used in the present work are: Direct Inverse Control and Internal Model Control.[7, 11]

A. Direct Inverse Control (DIC)

Inverse control utilizes the inverse of the system model. The diagram below (Figure 2) is a simple example of direct inverse control. A neural network is trained to model the inverse of the process. When the inverse controller is cascaded with the process the output of the combined system will be equal to the set point.

B. Internal Model Control (IMC)

Internal Model Control is a structure that allows the error feedback to reflect the effect of disturbance and plant mismodeling. Internal model control based on direct inverse control is shown in Figure 3.

A neural network model is placed in parallel with the real system. The controller is an inverse model of the process. The filter makes the system robust to process-model mismatch. With the IMC scheme, the aim is to eliminate the unknown disturbance affecting the system. The difference between the process and the model *yhat(k+1)* is determined. If the ANN model is a good approximate of the process then the *yhat(k+1)* is equal to the unknown disturbance. The signal *yhat(k+1)* is the information that is missing from the NN-model and can be used to improve the control. The *yhat(k+1)* signal is subtracted from the input set point *(k+1)*.

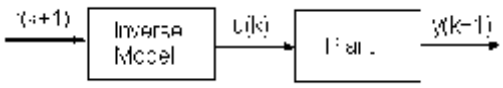

Fig. 2. Direct inverse control

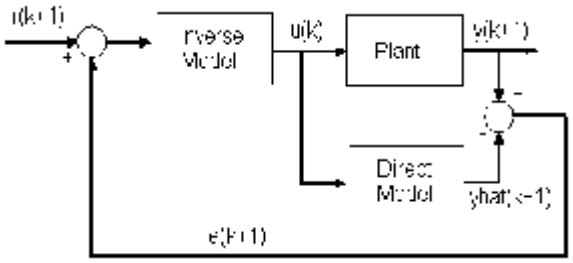

Fig. 3. Diagram of Internal model control

4. Twin Rotor System

The Two Rotor MIMO System (TRMS)[1] is a laboratory set-up designed for control experiments. Although the dynamics of the TRMS are simpler than those of a real helicopter, they retain the most important helicopter features such as couplings and strong nonlinearities.

The TRMS consists of a beam pivoted on its base in such a way that it can rotate freely in both its horizontal and vertical planes producing two rotating movements around yaw and pitch axes, respectively.[1] The schematic diagram of the TRMS is shown in Figure 4. At both ends of a beam, pivoting on its base, there are two propellers driven by DC-motors. The articulated joint allows the beam to rotate in such a way that its ends move on spherical surfaces. There is a counter-weight fixed to the beam and it determines a stable equilibrium position.

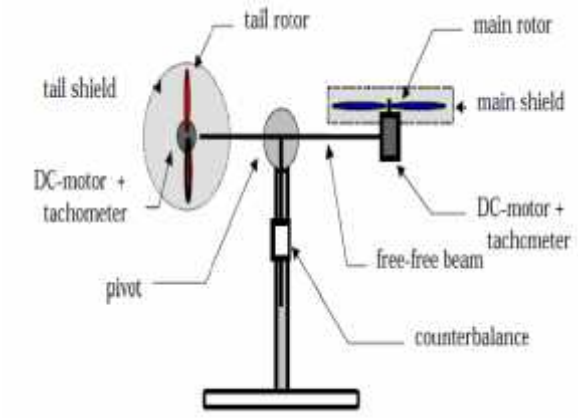

Fig. 4. Schematic diagram of TRMS

5. Modeling and Inverse Modeling Using ANN

Various modeling techniques can be used with neural networks to identify non-linear dynamical systems. These include state-output model, recurrent state model and non-linear autoregressive moving average process with exogeneous (NARMAX) input model. However, a model, without incorporating the noise term or considering the noise as additive at the output, namely NARX can also be used to identify system reliably. Schematic diagram of NARX[3,4,6] is shown in Figure 5 and mathematical expression is given below:

$$\hat{y}(t) = f[(y(t-1), y(t-2), \cdots, y(t-n_y), u(t-1), u(t-2), \cdots, u(t-n_u)] + e(t) \quad (4)$$

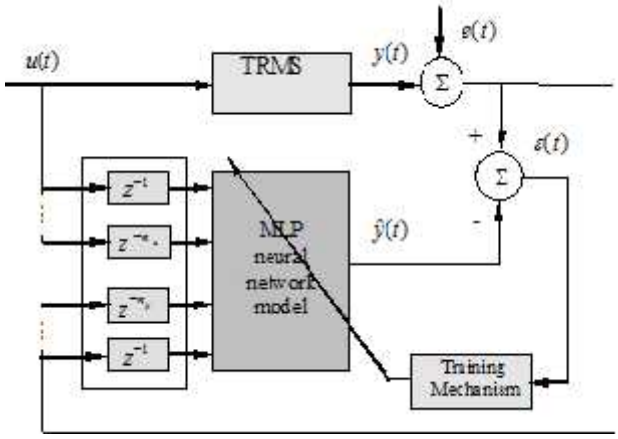

Fig. 5: NARX model identification with MLP neural networks

Once the network weights and biases have been initialized, the network is ready for training. During training the weights and biases of the network are iteratively adjusted to minimize the network performance function. The default performance function for feedforward networks is mean square error MSE - the average squared error between the network outputs and the target outputs.

Inverse modeling is used to generate the inverse of the process. The system output is used as an input to the network. The ANN output is compared with the training signal (the system input) and this error signal is used to train the network. This training method will force the neural network to represent the inverse of the system. The simplest approach of inverse modeling is shown in Figure 6. The system output is used as an input to the network. The ANN output is compared with the training signal (the system input) and this error signal is used to train the network. This training method will force the neural network to represent the inverse of the system.

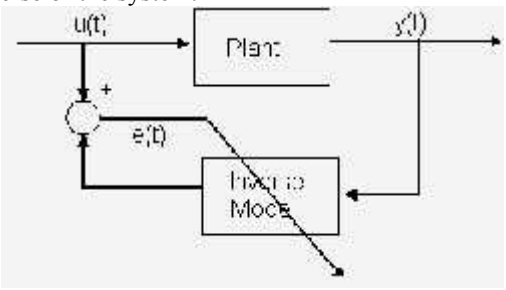

Fig. 6: Structure for inverse model training

There are certain problems associated with this approach. The TRMS is open-loop unstable, the training data would not show the dynamics of the system as the system moves/oscillates quickly. There may be plant-model mismatches. The training of the ANN for an inverse model may not yield an accurate model of the actual system which in turn, may lead to unknown disturbances in the system.

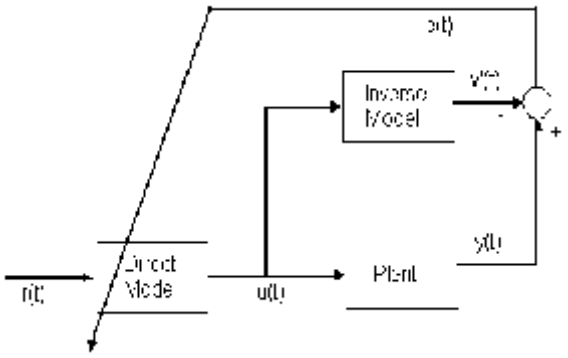

Fig. 7: Structure for specialized inverse model training

To avoid these problems another way of training inverse models is present in the literature, namely specialized inverse model training.[7,11] The models and plant are connected according to Figure 7. The inverse model is connected in series with the plant, but since usually the internal states of the Plant are unknown and do not allow performing the necessary calculations to report the error (between plant output and desired output) to the output of the inverse model, a direct model is placed in parallel with plant. This structure is supposed to overcome the problems mentioned for the previous type of training because the network is trained in a situation similar to the one that the NN will assume in a control situation.

6. Design of Neural Network Controller

The NN plant model and the NN controller (inverse plant model) are trained off-line, using data collected from plant operations. The direct inverse control and internal model control are investigated in this work. In the present NN design work, PSO training method is used.

PSO is a population-based optimization algorithm and is initialized with a population of random solutions, called particles and each particle in the PSO also has an associated velocity. One of the variants of PSO algorithm, used in this work, is stated as:[9,12]

$$v_{id} = \omega \times v_{id} + c_1 \times r \times (p_{id} - x_{id}) + c_2 \times R \times (p_{gd} - x_{id}) \quad (5)$$

$$x_{id} = x_{id} + v_{id} \quad (6)$$

Where ω is inertia weight, c_1 and c_2 are positive constants, and r and R are two random functions in the range [0,1]; $X_i = (x_{i1}, x_{i2}, ..., x_{id})$ represents the i-th particle; $P_i = (p_{i1}, p_{i2}, ..., p_{id})$ represents the best previous position (the position giving the best fitness value) of the i-th particle; the symbol g represents the index of the best particle among all the particles in the population; $V_i = (v_{i1}, v_{i2}, ..., v_{id})$ represents the rate of the position change (velocity) for particle i. Equation (5) describes the flying trajectory of a population of particles. It describes how the velocity is dynamically updated and equation (6)

gives the position update of the "flying" particles. The modeling problem has been re-formulated as minimization problem in the PSO process. This fitness function used in this case is mean squared error (MSE). Particles having relatively lower values of MSE will be selected as personal bests (P_i) and particle having lowest MSE will be selected as global best (P_g). These values will guide other particles towards better region in the search space. The algorithm was run with a swarm size (number of individuals in initial population) of 20, generated randomly. The acceleration coefficients c_1 and c_2 were set at 1.5 and inertia coefficient, ω was gradually decreased from 1.2 to 0.1 with generation. With these parameters, PSO algorithm was run for 500 generations.

In case of inverse modeling the same process has been used, only the input and output data were interchanged. As shown in Figure 2 and 3, the direct inverse controller and internal model controller have been implemented using Matlab/Simulink.[13] This method used the past values of input, u and output, y, the control signal required for producing the desired output is found. The difference between expected u and the neural model output, uN is the error, eN which was utilized for network learning. The input- output data obtained is divided into two parts containing 300 data and 200 data. The first 300 data are taken for training. Weights are initialized from input to hidden layer & hidden to output layer. After training is completed the remaining 200 data are taken for validation. In this case, the NN model obtained is called inverse NN model. The network is next validated on the remaining set of data to evaluate the model. After suitable training model is obtained then the network is validated using the remaining data. This inverse model after training and validation is taken for control. Here the inverse model itself acts as the controller.

7. Implementation and Results

In this investigation a controller is designed using the direct inverse control and internal model control for a TRMS. Matlab/Simulink is used to implement the plant model, inverse model and overall control scheme. The reference signal and the output responses of the system for direct inverse control and internal model control are shown in Figure 8 and 9 respectively.

Comparing Figures 8 and 9, it can be clearly demonstrated that the output response of the system using internal model control is better than that using direct inverse control. As observed, the output responses for both cases have initial disturbances but in case of internal model control, it's much less than the direct inverse control. Moreover, the important to note that the reference input and output response of Figure 9 are overlapped on each other that clearly reveals that the output of the system can follow the reference input and finally settles to the desired position that results zero steady-state error. So, it is obvious the internal model controller stabilize the system better.

To investigate further, the unit step response of the plant using internal model control is recorded and shown in Figure 10. Here the simulation time is set to 10 sec. So from the Figure 10, we can see that even though there is an overshoot in transient response, the output finally settles and completely follow the reference input.

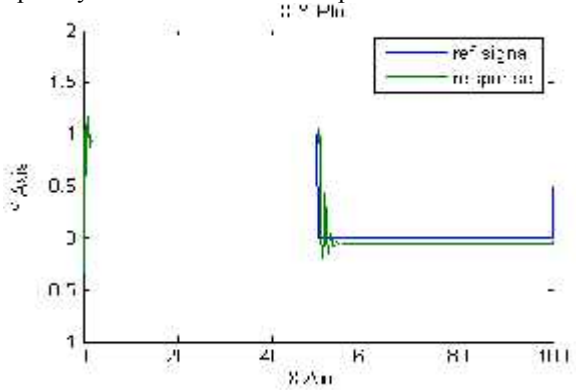

Fig. 8: Reference vs the system output for Direct Inverse Control(DIC)

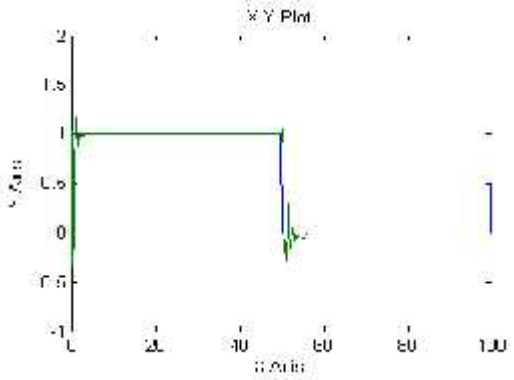

Fig. 9: Reference vs the system output for Internal Model Control(IMC)

Different time domain performance measures of output response due to internal model control are recorded as: settling time ≈0.7s, Rise time≈ 0.1s, Overshoot≈ 20 % and steady state error≈ 0. All these performance measures clearly show the effectiveness of the design controller.

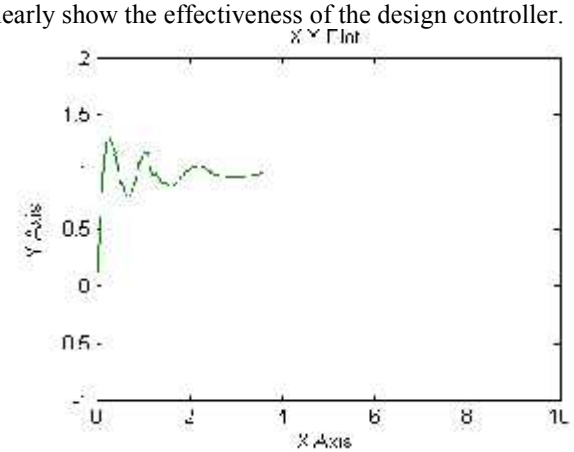

Fig. 10: Reference vs the system output for IMC(simulation time=10s)

8. Discussion and Conclusion

Supervised learning uses an existing controller or human feedback in training the neural network. In order to train the neural network to imitate an existing controller a vector of inputs and control targets from the controller must be collected. With supervised control, a neural network could be trained to imitate a robust controller. The robust controller can operate correctly, if the process operates around a certain point. And in case of direct inverse control, it is simple but this approach has some drawbacks.[11]

a) The learning procedure is not goal directed. As the model is not trained in the same situation in which it will be used after training, the structure of training is said not to be goal directed.

b) In situations where the mapping is not 1:1, an incorrect inverse can be obtained.

So the advantage of using direct inverse control (DIC) over supervised control is that inverse control does not require an existing controller in training

Two proposed control strategies; direct inverse control (DIC) and internal model control (IMC) have been implemented in this paper. Although both DIC and IMC strategies achieved interesting results, the performance of IMC is better as far as input tracking is concerned. Moreover, the output of the system settles to the desired value that yields a zero steady state error. On the other hand, in case of DIC, there is a constant steady state error that indicated the system settles to a different value rather than the desired level. Both the control strategies have an interesting side:

a) DIC is relatively simpler and easier to implement.

b) IMC forms a more robust control loop an thus presents better results from mean squared error point of view.

The neuro-controller operates similarly to the robust controller but can also adapt if any disturbance occurs in the system. If the inverse model is very accurate, the nonlinearities in the ANN will cancel out the nonlinearities in the process. The advantage of using inverse control over supervised control is that inverse control does not require an existing controller in training. The problems associated with direct inverse control such as plant-model mismatch, etc are reduced using IMC.

References

1. Feedback Instruments Ltd, Twin Rotor MIMO System Control Experiments, Manual: 33-949S Ed01 122006, Crowborough, East Sussex, UK, 2015.
2. Alam, M.S. and Tokhi, M.O., Modelling of a twin rotor system: a particle swarm optimisation approach, *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 221(3), pp. 353-374, 2007.
3. Afruz, J. and Alam, M.S., Non-linear Modelling of Twin Rotor MIMO System Using Particle Swarm Optimisation (PSO) Algorithm, *In Proc. of IEEE International Computer Symposium (ICS)*, Tainan, Taiwan. Dec 16-18, 2010.
4. Rahideh, Akbar, A. H. Bajodah, and M. Hasan Shaheed. "Real time adaptive nonlinear model inversion control of a twin rotor MIMO system using neural networks." *Engineering Applications of Artificial Intelligence* 25, no. 6 (2012): 1289-1297.
5. Sieberling, S., Q. P. Chu, and J. A. Mulder. "Robust flight control using incremental nonlinear dynamic inversion and angular acceleration prediction." *Journal of guidance, control, and dynamics* 33, no. 6 (2010): 1732-1742.
6. Haykin, Simon, and Neural Network. "A comprehensive foundation." *Neural Networks* 2, no. 2004 (2004).
7. Norgaard, M., O. Ravn, O., Poulsen, N.K., Hansen, L.K., *Neural Networks for Modelling and Control of Dynamic Systems: A Practitioner's Handbook*, ISBN 1-85233-227-1, Springer-Verlag London Limited 2000.
8. Kennedy, J., and Eberhart, R.C.. Particle swarm optimization, *Proc. of IEEE InternationalConference on Neural Networks (ICNN)*, vol.IV, pp.1942-1948, Perth, Australia, 1995.
9. Poli, Riccardo, James Kennedy, and Tim Blackwell. "Particle swarm optimization." *Swarm intelligence* 1, no. 1 (2007): 33-57.
10. Lazinica, A., *Particle Swarm Optimization*, (Edited), Published by In-Tech, Croatia, January, 2009.
11. Miller, W. Thomas, Paul J. Werbos, and Richard S. Sutton, eds. *Neural networks for control*. MIT press, 1995.
12. Shi, Y. and Eberhart, R.C., A modified particle swarm optimizer, *Proc. IEEE Int. Conf. Evolutionary Computation*, pp. 69-73, 1998.
13. MATLAB Reference Guide, The Math Works, Inc., 2015.