NEURO-CONTROLLER DESIGN USING GENETIC OPTIMIZED POLE PLACEMENT METHOD

CORNEL RENTEA
Department of Automation and Computer Science
University "Lucian Blaga" Sibiu
Str. E. Cioran, No. 4, Sibiu-550025
ROMANIA

***Abstract:* -** In this paper, in order to improve the training of a neural controller implemented using a direct inverse scheme we use pole placement design enhanced with the help of a genetic algorithm. We discuss this optimization of training a neural network controller for the Inverted Pendulum problem, considered an acknowledged benchmark in nonlinear system control.

Key-Words: - Pole placement, Genetic algorithm, Inverted pendulum, State-space representation, Neural network control

1 Introduction

Artificial Neural Networks (ANN) are an emerging technology, yet, in continuous dynamic behavior, much work has been done to attempt to generate a formal method to design a controller based on this technology. We have various methods for training such network, how can we improve them? One solution would be improving the feedback control system which models the system to be controlled.

Modern control theory gives us a number of analytical tools for finding feedback controls. The method we chose for optimization of a controller designed using pole placement method, is a genetic algorithm. There are many optimization algorithms but they have a hard time finding the optimal (global) solution in multi-parameter search space. A genetic algorithm is a parallel, global search technique that emulates natural genetic operations. Because it simultaneously evaluates many points in the parameter space, it is more likely to converge toward the global solution. We used a genetic algorithm that modifies parameters for the pole placement method of design for a feedback control system.

As an example of application for this method, we will develop a pole placement designed feedback control system that automatically stabilizes an inverted pendulum system while moving the cart to its commanded position. After that, we will use it a direct inverse architecture of a neuro-controller used for our problem.

We used Matlab implementation for all the needed components: the control system, the genetic algorithm and the neuro-controller [1].

2 Inverted Pendulum Problem

The inverted pendulum system is a typical benchmark for dynamic non-linear systems. The classic example consists of either a point mass at one of its end of an ideal rod or a rod without a point mass at one of its ends. In each of these two cases, the other end of the rod is attached, through a joint that can pivot in a plane, to a moving cart. The cart can move on the *x* axis, in the plane of motion of the pendulum's pivot point. The cart accelerates due to a force (F) applied to it. The acceleration of the cart induces a rotation in the pendulum's pivot. The pendulum can be made to balance at the top of its arc by controlling the acceleration of the cart.

The system we considered consists of a uniform distributed mass rod attached to the cart.

For our example we assume that:

▪ M – mass of the cart	0.5 kg
▪ m – mass of the pendulum	0.5 kg
▪ b – friction of the cart	0.1 N/m/sec
▪ l – length to pendulum	0.3 m
▪ center of mass	
▪ I – inertia of the pendulum	$0.006\ kg\cdot m^2$
▪ F – force applied to the cart	1 N
▪ x – cart position coordinate	
▪ θ – pendulum angle from the vertical	
▪ g – gravitational acceleration	$9.8\ N/m^2$

To linearize the equations regarding the pivot angle of the pendulum, we assume that the rod does not reach a value of the angle (θ) bigger than 0.05 radians from the vertical. By doing this, we have sinθ ≈ θ. Bellow we represent a schematic of an inverted pendulum:

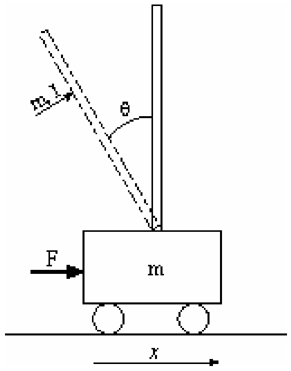

Fig.1 The inverted pendulum system.

The linearized system equations can be represented in state-space form:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Mml^2} & \frac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Mml^2} & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

This problem is especially interesting because without control, the system is unstable. This is a fourth order nonlinear system, which is linearized about the vertical equilibrium. In this example, the angle of the vertical pole is the controlled variable, and the horizontal force applied by the cart is the actuator input. The goal of the controller is to move the cart to its commanded position without causing the pendulum to tip over

3 Control design using pole placement method

The design is formulated in terms of obtaining a closed-loop system with specific pole locations. We build a controller for this system using pole placement design [2]. The controller generates a control signal that is going to be applied to the inverted pendulum in order to control the arm in a vertical position.

The schematic of a full-state feedback system is the following:

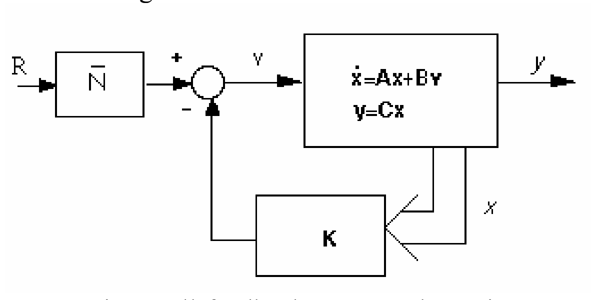

Fig.2 Full-feedback system schematic.

The characteristic polynomial for this closed-loop system is:

det(sI – (A – BK)).

In this problem R represents the commanded step input to the cart. The 4 states represent the position and velocity of the cart and the angle and angular velocity of the pendulum. The output y contains both the position of the cart and the angle of the pendulum. We want to design a controller so that when an step input is given to the system, the pendulum should be displaced, but eventually return to zero (i.e. the vertical) and the cart should move to it's new commanded position.

The C matrix is 2 by 4, because both the cart's position and the pendulum's position are part of the output. For the state-space design problem we will be controlling a multi-output system so we will be observing the cart's position from the first row of output and the pendulum's with the second row.

Since the matrices A and B*K are both 4 by 4 matrices, there will be 4 poles for our control system. By using full-state feedback we can place the poles anywhere we want. For example, the poles can be chosen as the eigenvalues of the A matrix. We know that the size of the real parts of our chosen poles have an effect on the rate at which the linearised system is fully damped and the imaginary components have an effect on the oscillatory behavior of the system.

Nevertheless, as these poles can be placed anywhere we want, one cannot guarantee the correction and optimality of the obtained control system for the considered problem. Thus, in order to improve the control system performance we used a genetic algorithm to help choosing these poles that help designing a control system and effectively and efficiently optimize it's performance.

4 The Genetic Algorithm

Choosing control parameters for the pole placement method can be done in various ways, but none of them gives the best solution. Thus, we can use a genetic algorithm to optimize these choices and try to find the best solution we can. Problems of control can be viewed as requiring the discovery of a controller or a control strategy that takes the state variables of a problem as its inputs and produces the values of the control variable(s) as its outputs. Genetic programming is well suited to difficult control problems where no exact solution is known and where an exact solution is not required.

The genetic algorithm is a probabilistic algorithm, which maintains a population of individuals. Each individual represents a potential solution to the problem at hand, and is implemented as some data structure. Each solution is evaluated to give some measure of *fitness*. Then selecting the more fit individuals forms new population. Some members of the new population recombine by means of "genetic" operators to form new solutions. There are unary transformations like mutations, which create new individuals by a small change in only one individual, and binary transformations, like crossovers, which create new individuals by mixing traits from the two parents. After some number of generations the search converges and is successful if the best individual represents the optimum solution [4].

In our genetic algorithm, we use real encoding. An individual is a vector of four numbers that define the four poles of the system described in the previous section. If we denote (Re1, Im1, Re2, Im2) one such individual, we can write the poles of the control system like:

(Re1 + i * Im1, Re1 – i * Im1,
Re2 + i * Im2, Re2 – i * Im2).

The testing data that we have chosen for calculations of the fitness value in the genetic algorithm include a 0.2 m step input for the cart and some design criteria as follows:

- Settling time for x and theta of less than 5 seconds;
- Rise time for x and theta of less than 1 second;
- Overshoot of theta less than 20 degrees (0.35 radians);
- Steady-state error within 2%.

After recording the results for the control system, we compute the fitness value as

$$f = 25 \cdot \left(rt_x^2 + rt_\theta^2 \right) + s_x^2 + s_\theta^2 + \frac{25}{0.1225} \cdot O_\theta ,$$

where f = fitness value for current individual;
rt_x, rt_θ = rise time for x and theta;
s_x, s_θ = settling time for x and theta;
O_θ = overshoot of theta.

Our genetic algorithm uses an elitist strategy, Monte Carlo method of selection, convex crossover and uniform mutation.

5 Neuro Controller architecture

The first step in constructing the architecture of the neural network controller is system identification for the inverted pendulum model. Then, a neuro-controller is then designed using neural network model for the pendulum.

For our implementation we used a MLP network with the Levenberg-Marquart backpropagation learning algorithm. The MLP network has three layers, with 4 units at its input layer, 3 units at the hidden layer and 1 unit at the output level. All activation functions are linear.

We use the direct-inverse architecture for the neuro-controller [1], [5]. In Fig.3 we show the graphical representation of the direct-inverse neuro-controller for our problem.

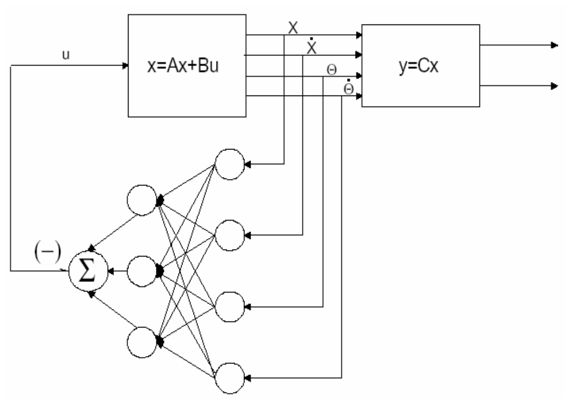

Fig.3 Direct inverse neuro-controller architecture

Were the inputs for the network are the state variables of our system: angle, angle derivative, car position and its derivative, as described in section 2, and the output is the control force acting on the inverted pendulum cart.

The input/output data pairs which were generated using the linearized model are used now in order to train the network.

The inverted pendulum neuro-controller is trained with a linear controller made via the pole placement design method first, and then with the genetic enhanced pole placement design. The input to the model receives signals from states of the plant and its output is a control signal directed towards the inverted pendulum.

5 Experimental results

The tests we ran had 85% probability of crossover and 10% or 15% probability of mutation.

We used different combinations of number of individuals in a generation and number of generations such as: 100, 200, 250 or 500 individuals in a generation that evolved during 100, 250, 500 or 1000 generations. We observed that our genetic algorithm tends to continuously improve the control system performance.

After the requested number of generations, the algorithm returns the best-fit individual.

The best individual from all the experiments was:

(-18.7617 -6.3227 -18.9177 -4.0390),

and had the fitness value of 4.8722.

In the training process of the neuro-controller we used the control system designed with simple and then enhanced pole placement method for generating the training data.

The network had 3 neurons in the hidden layer, 5000 data points generated for training and it was trained on 100 epochs. The neural network plant model is trained with static Levenberg-Marquardt backpropagation and is reasonably fast.

We represented the obtained performance of the network with the two training data sets (Fig.4 and Fig.5).

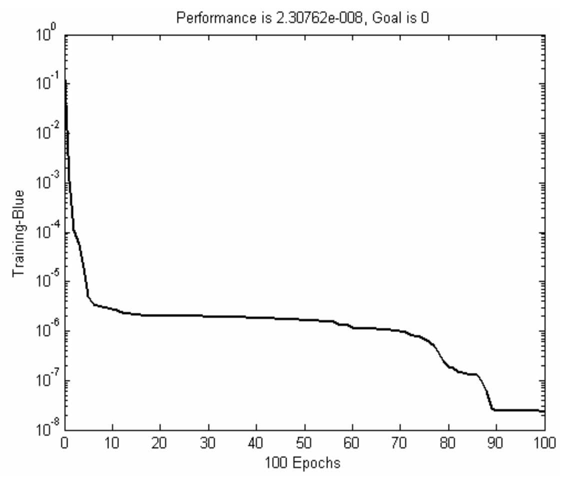

Fig.4 Performance of the network during the simple training process

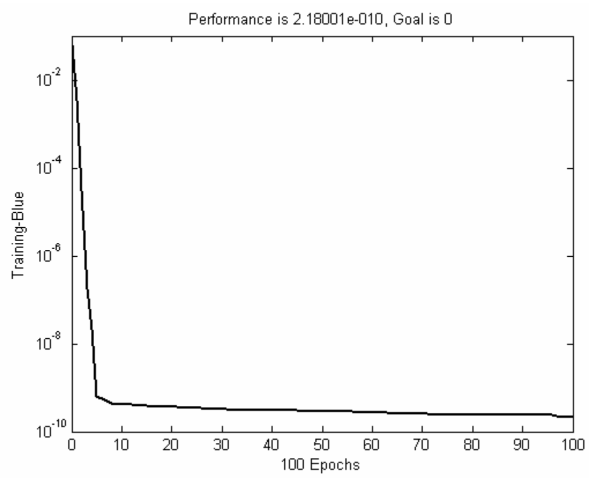

Fig.5 Performance of the network during the genetic enhanced training process

We can see that genetic enhanced data set is more efficient in the training process of the network. It achieves higher performance and in shorter time.

6 Conclusions

This work shows results from two neurocontrol systems for the inverted pendulum, one trained based on a pole placement design of the system and the other trained based on a genetic improved pole placement design. We have shown the enhanced performance achieved in the training process based on genetic improvement design of a part of the neurocontroller using a direct inverse scheme.

This optimization using a genetic algorithm can be successfully applied for the optimization of any non-deterministic design method for a control system, as long as there exists a mathematical model to describe the dynamics of the system to be controlled.

Furthermore, we could use the genetic algorithm to help improving the other parts of the neurocontroller , for example the most appropriate structure for the neural network, training method, and number of samples in a data set, etc.

References:

[1] Kuo, B. C. and Hanselman, D. C., 1994, *Matlab Tools for Control System Analysis and Design*, Prentice Hall, Englewood CliÆs, New Jersey

[2] Franklin, G. F., Powell, J. D., and Emani-Naeini, A., 1994, *Feedback Control of Dynamic Systems*, 3rd edn., Addison-Wesley, Reading, Massachusetts

[3] Koza, J. R., *Genetic Programming: On the programming of computers by means of natural*

selection, Cambridge, MA: MIT Press, 1992

[4] Hunt K.J., Sbarbaro-Hofer D., Zbikowski R.,Gawthrop P.J., *Neural Networks for Control Systems- A survey*, Automatica, Vol. 28, pp. 1083-1112, 1992.

[5] Åström K.J., Wittermark B., *Adaptive Control*, Addison Wesley, Reading, MA., 1989.