Neural Motor Control

Péter Odry, Gábor Kávai

Polytechnical Engineering College, Marka Oreskovica 16, Subotica, Serbia and Montenegro
odry@vts.su.ac.yu

Abstract: The control of the rotation speed of serial universal motors is very complicated when done using traditional control techniques, as it requires a very complex mathematical model. Using neural network eliminates the need for mathematical modeling and allows easy realization of a solution. The universal serial motor is driven with a PWM signal and rpm control is realized with neural network. We compared neural with P(I(D)) realizations.

1 Background Information

The control of the rotation speed of serial universal motors is very complicated when done using traditional control techniques, as it requires a very complex mathematical model. **[3]** Using Neural network eliminates the need for mathematical modeling and allows easy realization of a solution. Our other system realization is heat pump, also very interesant control systems are in Neural realizations **[4]**.

The Neural Network consist of junctions which connected with **links**, also known as **processing units**. For each junction ordered one numer, that's name is weight. The weights are tools for the long-distance information storing in the neural network, the learning process occur with the appropriate modification of weights. We modify the weights, so that the networks input/output behaviour be in consonance with the enviroment, which provide the input datas.

The calculation algorithm consist of two basic step:

1 Calculation the output of network, with inputs and weights

2 Modification of weights with learning algorithm

1.1 Calculation the Output of Network

Each input is multiplyed by its W_i weight, and we summarize the products. This value is the input of the activation function which gives the output:

$$Y = f\left(\sum X_i \cdot W_i\right) \tag{1}$$

The ***f( )*** *activation* functions type is dependin on application, determined by the problem that needed to be solved.

We used linear activation function for the easier computation. The serial universal motor has different property depending on direction of rotation(designed for one-way rotation). We used different weights for one and another direction. Its look, like that we have two neural network for both directions.

The controller, which developed by us, use such neurons, one layer's outputs is the inputs for the next layer.

The network is constructed from layers. This layers contains neurons, and individual layer's neurons connected with front and behind layer's neurons. The connections is always one-way directed.

The first layer sends the the informations for the next layer, not modifies the input datas.

The second layer's task is the data processing. It summarize and modificate the input datas. One network can contain one or more hidden layers, in our case, we have 5 neuron in the hidden layer.

The third layers each and every neuron is an output, which number is optional. Its function is identical with the hidden layers neurons. In our case one output it is.

1.2 Modification the Weights with the Learning Algorithm

Be the activation function linear:

$$f(x) = x \tag{1}$$

The final backpropagated error in the hidden layer:

$$\delta_j(n) = \delta_i(n) \cdot W_{ij}(n) \cdot X_k'(n) = e(n) \cdot W_{ij}(n) \tag{2}$$

2 Hardware Description

An MSP430F149 device is used in this motor control demo application. It can easily be substituted by any other MSP430 device with required hardware elements available **[1], [2]**. The MSP430 is sourced by an 8-MHz crystal to provide a high-resolution clock source that is used for both PWM generation and speed measurement.

The universal serial motor is driven with a PWM signal that is generated using a Timer_B capture/compare block operated in compare mode. The MSP430 output

signal is then delivered to the to the actual motor driver output stage (TPIC0108B), this driver stage is supplied by 9V.

To provide feedback for the control loop, an optical encoder is used. With this, we can determinate the direction of rotation, by the side of current speed. In one rotation 45 impulse come from the encoder. **The actual speed determination in this way happen: with Comparator_A we request an interrupt, if rising edge occurs on CA0.**

Then we increment the value of a variable, which contained a number of impluses, which has came up to the present. At the same time, we save the information from the direction of the rotation speed.

We do it for a 25ms, where Timer_A give the possibility, and we save and reset the number of impulses, which has came up to the present, and we decide, that in which direction has the shaft rotated. This measurig method not claim different averaging, by this means the systems reaction time not decrease, as well as we obatain processing time and the speed sampling time remain constant.

3 Software Description

After setting up the MSP430 clock system, the peripherals are configured.

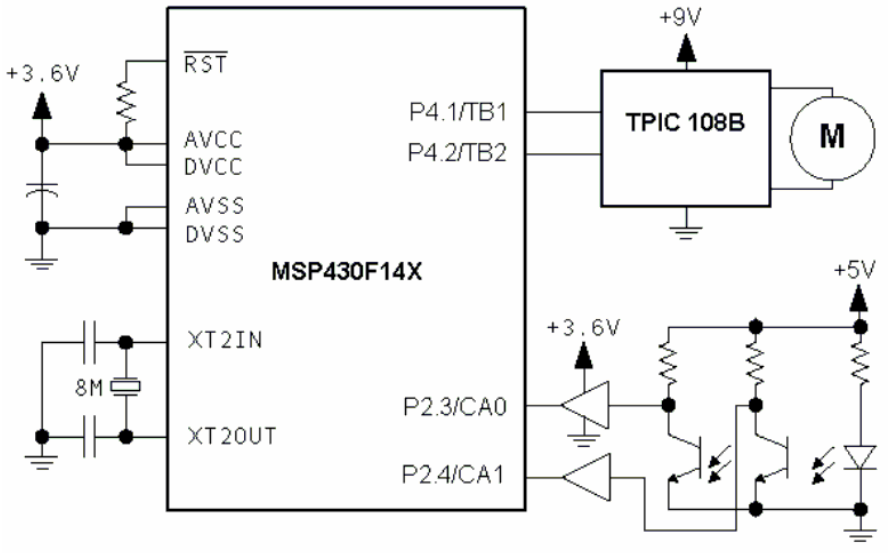

Figure 1
Realised hardware

The program consist of 3 general component, these are: **Comparator_A interrupt**, **a Timer_B interrupt** and the **main()** function. The Comparator_A interrupt is used to count the impulses from the encoder, and for storing the informations of directions of rotation. With Timer_B, we provide the time base for speed measuring and the PWM signal, what serve for a motor driving and its frequency is 2kHz. In the Timer_B interrupt service rutine we calculate the value of speed of rotation in [rps].

In the **main()** function we calculate the output of neural network and the modofication value of weights(learning).

We do this with the following functions:

signed int hneuron_lin(unsigned char n,unsigned char m) - for calculating the output of one neuron in the hidden layer.

signed int oneuron_lin(unsigned char n,unsigned char m) - for calculating the output of one neuron in the output layer.

void nnetwork(void) - make a loop from the two previous functions and gives the neural network's output.

void backprop(void) - modofies the value of weights(learning) with the backpropagation algorithm

4 The Recording the Graphs

We recorded the graphs with different outputs of controller. In the next figure is shown the simplifiest case, when the neural network's output is connected with gain to the motor driving circuit.

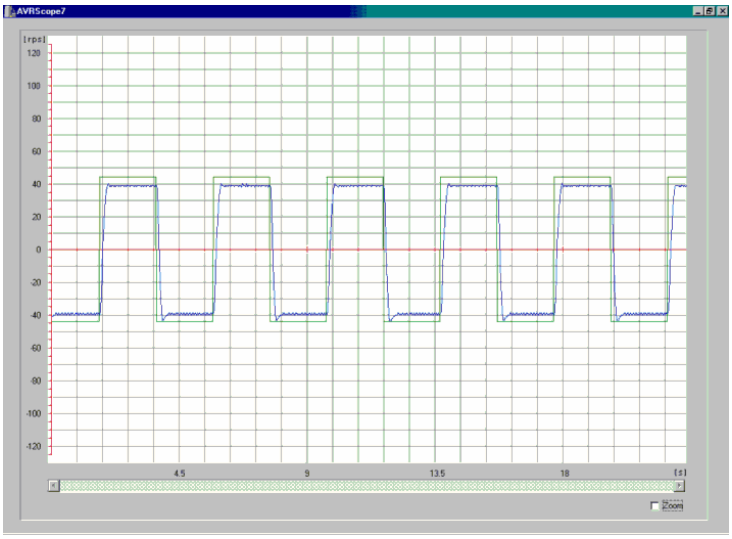

Figure 2

The rps diagram for a simple Neuro controller

Attributes: like a P controller, constant error is remain, this is constant in the full merasuring range, small oscillation arise.

In the next figure **P and Neuro** conroller combination is visible:

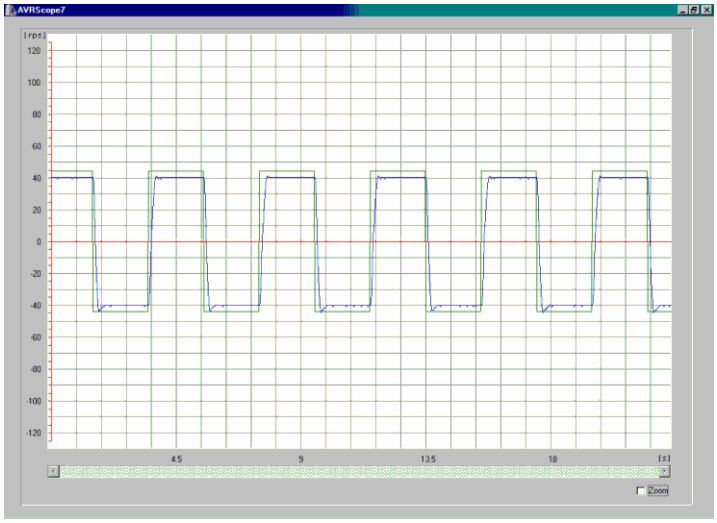

Figure 3

The rps diagram for P and Neuro controller

Attributes: the constant error remains, on the other hand stabler holds the referent value.

In the next figure we **P and I** effects are used to the output of Neuro controller:

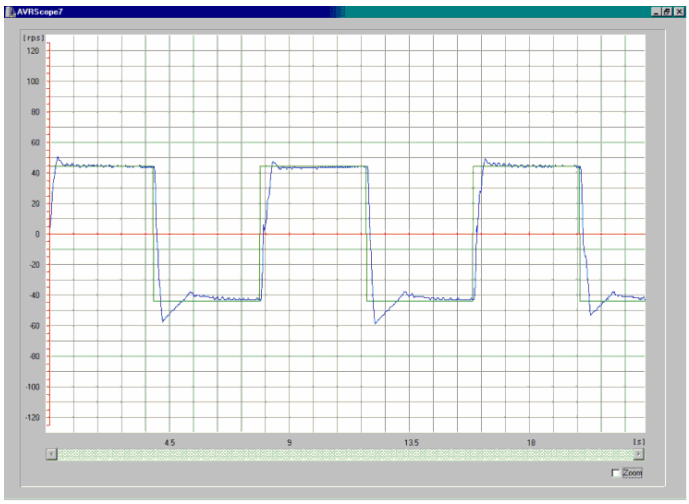

Figure 4
The rps diagram for PI and Neuro controller

Attributes: the constant error disappear, on the other hand the motor clatter, when track down to the accurate value, this can not be seen in the in the figure.

In the next figure, can be seen the optimal solution, when we summarize the Neuro controller output values. Here the Neuro controller gives one differential value in its output.

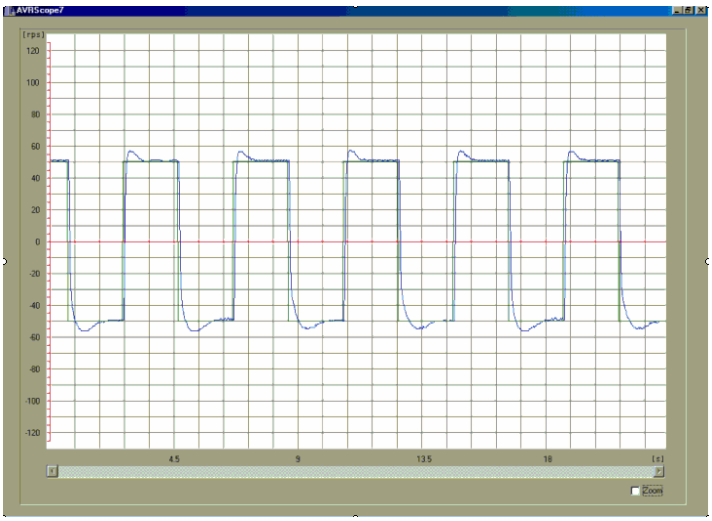

Figure 5
The rps diagram for I and Neuro controller

Attributes: The constant error disapperars, on the other hand bigger overshoot occurs. After transition periods he holds the referent value more stabler.

Conclusion

On the **Figure 3** we can see the Neural system's output combined with P controller, the improvement is preceptible in the constant error and it more stabler holds the referfent value.

On the **Figure 4** we can see the Neural system's output combined with PI controller, the constant error disappear, on the other hand the motor clatter, when track down to the accurate value.

On the **Figure 5** we can see the Neural system's output combined with I controller, the constant error disapperared, on the other hand bigger overshoot occurs. After transition periods he holds the referent value more stabler.

The Neural systems important property is besides the adaptation capability and smaller oscillations in overshoots, compared with the traditional *P(I(D))*.

References

[1] MSP430x13x, MSP430x14x Mixed Signal Microcontroller Data Sheet (SLAS272)

[2] MSP430x1xx Family User's Guide (SLAU049)

[3] **Odry Péter et. al.:'**Fuzzy Logic Motor Control with MSP430x14x', Application Report, Texas Instruments, (SLAA 235)**,** 2005 february

[4] **Nyers J.:** Comperision of a traditional and a microprocessor-controlled heat pump control strategy, international conference, Sweinfurt Germany may 2001