A Toolbox – based Solution to ARGESIM Comparison C18 'Neural Nets / Transfer Functions' with MATLAB

Aleš Belič, University of Ljubljana, Faculty of Electrical Engineering; ales.belic@fe.uni-lj.si

Simulator: MATLAB 5.3 (www.mathworks.com) with Neural networks toolbox, running on Debian Linux 3.0 was used to solve this comparison.

Task a: Identification with linear dynamical model: First identification with linear dynamical model was tried. MATLAB *arx* function was used and 2nd order discrete-time model was identified:

$$G(z) = \frac{0.5289z^2 - 0.5206z}{z^2 - 1.586z^1 + 0.5985}$$

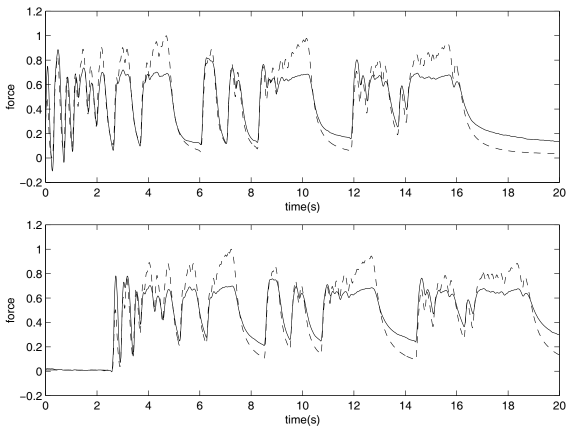

Figure 1: Simulated (solid line) in compare with measured force (broken line). Training data set – above, validation data set below.

Model simulation with respect to measured data is presented in Figure 1. As can be seen, linear second order model can describe the general system dynamics; however, the details are not matched.

Task b: Identification with linear dynamical model and the artificial neural network (ANN) in parallel. For this task, features of the Neural Network Toolbox were used.

In following MATLAB code, E represents the difference between real system's measurements T and linear model simulation y. Next, a network structure net is created with 7 neurons on the first layer and 1 neuron on the output layer, and is trained according to the system input P and target E. The network and the linear dynamical model are then simulated in parallel and the result of the hybrid system is shown in Fig. 2.

```
E = T-y';  net = ...
newff(minmax(P),[7,1],{'tansig','purelin'});
net1 = train(net,P,E);  y1 = sim(net1,P);
plot(t,y+y1',t,T,'--')
```

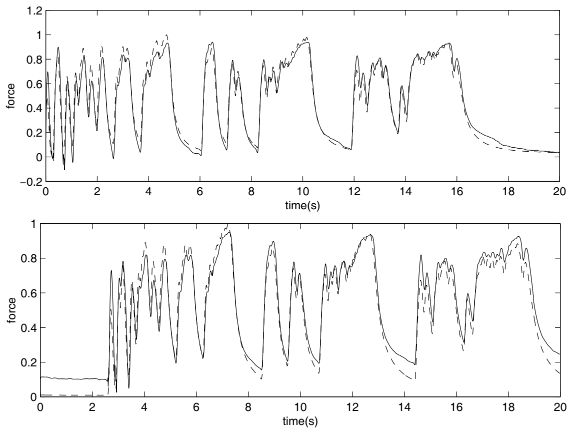

Figure 2: Simulation of the parallel structure of linear dynamical model and the ANN (solid line) in compare with measured force (broken line). Training data set – above, validation data set - below.

Task c: Identification with dynamical ANN. The following code shows again the use of the Neural Network Toolbox, especially for the ANN model training. The results are shown in Figure 3.

```
net = ...
newff([0 1],[10,1],{'tansig','purelin'});
net.layerconnect = [0 1;1 0];
net.layerweights{1,2}.delays = [1,2];
net.inputweights{1,1}.delays = [1,2];
net.trainparam.epochs = 50;
net.trainparam.show = 1;
net2 = train(net,con2seq(P),con2seq(T));
y2 = seq2con(sim(net2,con2seq(P)));
```

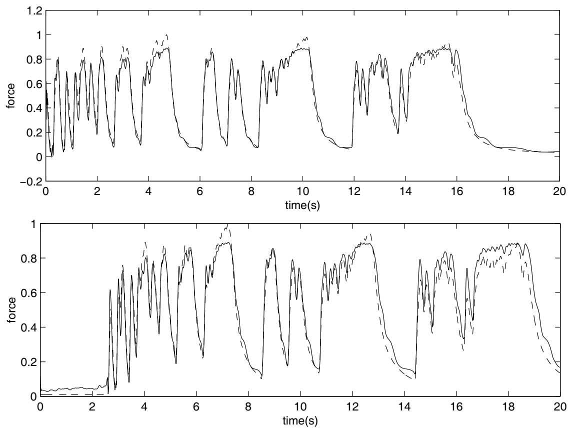

Figure 3 Simulation of the neural network model (solid line) in compare with measured force (broken line). Training data set – above, validation data set - below.

C17 Classification: Toolbox-based CACSD Approach
Simulator: MATLAB 5.3 (Linux) wit Neural Net TB