NEW TOOL FOR SYSTEM IDENTIFICATION AND PREDICTIVE CONTROL BASED ON NEURAL NETWORKS USING LABVIEW SOFTWARE

Hamid ALSHAREEFI[1], Ciprian LUPU[2], Hayder ALBOUDKHAL[3], Laith ISMAIL[4]

In this paper, a new tool for system identification and model predictive control (MPC) has been developed. The mathematical approximation of the model identification was derived using the neural network theory. The emphasis of this paper is to employ a multi-layer recurrent neural network with three layers to match an autoregressive-moving average (ARMA) model in the identification stage to provide the mathematical model for the model predictive controller in the control stage. The tool was designed using LabVIEW software and the MathScript facility. The tool can be used to identify and control simulated or real systems using the stored input/output data of the system.

Keywords: system identification, neural network, ARMA, MPC, LabVIEW, MathScript

1. Introduction

An artificial neural network (ANN) is part of a computing system proposed to simulate the method by which the human brain processes and analyzes data. It is the basis of artificial intelligence (AI) and is used to solve issues that would be difficult or impossible for humans to solve by classical standards. ANN has the property of self-learning, which allows improving their performance as more data is collected. By mapping input-output data, ANN is utilized as a black-box model to detect unknown models. An artificial neural network in identification and control applications has been the subject of numerous books and papers [1] [2] [3] [4]. The goal of this work is to develop a tool that includes two main parts. The identification part, where converting the ANN model into a transfer function

[1] Ph.D. Student, Dept. of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: hamedgep@yahoo.com
2 Professor, Dept. of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: ciprian.lupu@acse.pub.ro
3 Engineer, Ministry of Electricity, Iraq, e-mail: haider1982malik@gmail.com
4 Ph.D. Student, Dept. of Computer Technology Engineering, AL-TURATH University College, Baghdad, Iraq, e-mail: laith.sabaa@turath.edu.iq

model, where will reveal more details for the dynamic behavior of the physical system.

The second part is to use the result mathematical model of the dynamic system in the design of the model predictive control (MPC) controller. The tool was designed using LabVIEW software and the MathScript facility. The major contribution of this paper is to clarify the clear and strong relation between the transfer function parameters and the artificial neural network weights, to develop a software tool that can estimate the mathematical model for an unknown system with two structures, transfer function and state-space using LabVIEW software. The identified model which approximated in this tool according to the input-output data set was used to design a model predictive controller. The tool utilized to estimate the mathematical model for any real or simulated system according to the set of input-output data.

2. System identification

The process of determining the model of a dynamic system from input and output data through minimizing an error cost function between the model output and the real system's output is known as system identification.

The Auto-Regressive Moving-Average (ARMA) models represent the linear regression models. The linear structure of ARMA processes also lead to a substantial simplification of linear prediction which utilize difference equations to link the model output to current inputs from previous outputs and previous inputs. A general formula of a discrete-time ARMA model is illustrated in (1) [5] [6].

$$y(k) = a_1 y(k-1) + \cdots + a_n y(k-n) + b_0 x(k) + \cdots + b_m x(k-m) \quad (1)$$

Also can be represented as:

$$y(k) = \sum_{i=0}^{m} b_i x(k-i) + \sum_{j=1}^{n} a_j y(k-j) \quad (2)$$

Where $y(k)$ and $x(k)$ represent the variables of the model's output and input at sample k, a_j and b_i are the parameters of the model. By using the z-transform, the transfer function can be represented as in (3).

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \cdots b_m z^{-m}}{1 - a_1 z^{-1} + \cdots a_n z^{-n}} \quad (3)$$

As a vector form, equation (2) can be represented as $y(k) = \boldsymbol{\theta}^T \boldsymbol{\Phi}(k)$, where $\boldsymbol{\Phi}(k) = [y(k\text{-}1), \ldots, y(k\text{-}n), x(k), \ldots, x(k\text{-}m)]$ is the vector of the measurement, and $\boldsymbol{\theta}^T = [a_1, \ldots, a_n, b_0, \ldots, b_m]$ is the vector of the parameters. The concept of parameter system identification is to estimate the parameters a_j, b_i, to find the identified parameter vector $\hat{\theta}^T = [\hat{a}_1, \ldots, \hat{a}_n, \hat{b}_0, \ldots, \hat{b}_m]$, which is used to compute the predicted output $\hat{y}(k) = \hat{\boldsymbol{\theta}}^T \boldsymbol{\Phi}(k)$. The main problem is to approximate the parameters that minimize the error between the required output $y(k)$ and the predicted output $\hat{y}(k)$. As shown in Fig. 1.

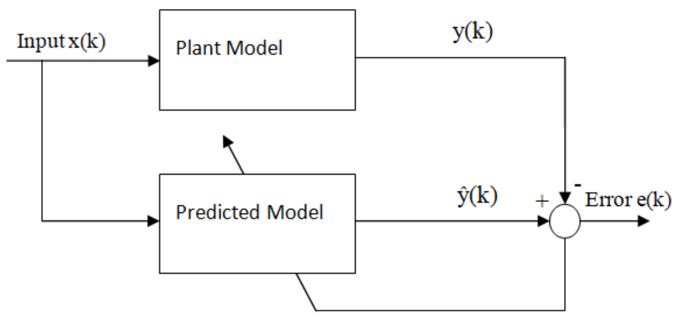


Fig. 1. General block structure of system identification

Most real systems are nonlinear in their behavior, and it demands a nonlinear modeling method. However, because of nonlinear input-output relationships in dynamic systems have so many structural options, identifying possible types of these systems with suitable structural models is relatively difficult. Identification of such models required sophisticated methods like Volterra series and Hammerstein models [7]. Or to update the ARMA structure to involve nonlinear elements and produce a nonlinear ARMA structure to be NARMA as in (4).

$$y(k) = \sum_{i=0}^{m} b_i x(k-i) + \sum_{j=1}^{n} a_j y(k-j) + \sum_{i=0}^{m}\sum_{j=0}^{n} b_{ij} x(k-i)x(k-j) + \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} y(k-i)y(k-j) + \sum_{i=0}^{m}\sum_{j=1}^{n} c_{ij} x(k-i)y(k-j) \tag{4}$$

This structure added new parameters (a_{ij}, b_{ij}, and c_{ij}) which produce more drawbacks in estimating the approximate order for each summation part. Furthermore, this structure doesn't follow the common transfer function in equation (3). These difficulties make neural networks a good solution for the modeling of nonlinear systems.

3. Neural network architecture

A neural network structure consists of multiple layers each layer can have many neurons. These neurons are information-processing units that represent the basis of neural networks. Two major classes of a neural network are distinct in the literature of neural networks. Feedforward network, where the information flows in a forward direction (from input to output). Recurrent network, which allows the information for feedback [8]. In special cases where the entire neural network's input represents the delayed network's inputs and output, this type of network can be considered a dynamic network as shown in Fig. 2.

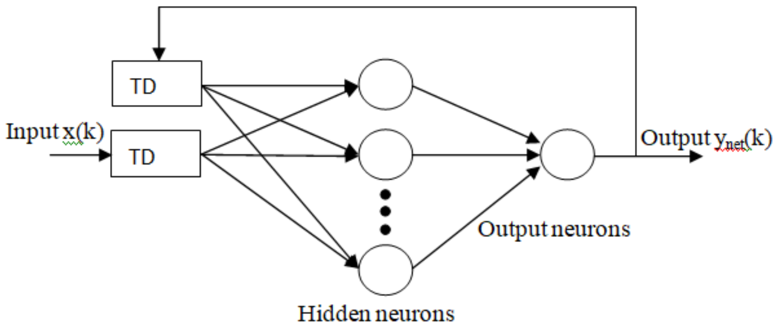


Fig. 2. General structure for a Dynamic Neural network

It's also feasible to employ delayed plant output instead of network output in feedback signals; in this scenario, the network's output will be a function of previous outputs and inputs, like in (5).

$$y_{net}(k) = f(x(k), x(k-1), \dots, y(k-1), y(k-2), \dots) \quad (5)$$

The input samples multiplied by their relative weights, as shown in Fig. 3, for example, w_{ih} links the input variables $x(k\text{-}i)$ with the hidden neuron h, and v_{jh} links the output variables $y(k\text{-}j)$ with the same hidden neuron. The nodes in the hidden layer produce the node outputs $N_1(k)$ ... $N_h(k)$ which they multiplied by the output weights $wout_1$ ... $wout_H$ to produce the output of the network $y_{net}(k)$.

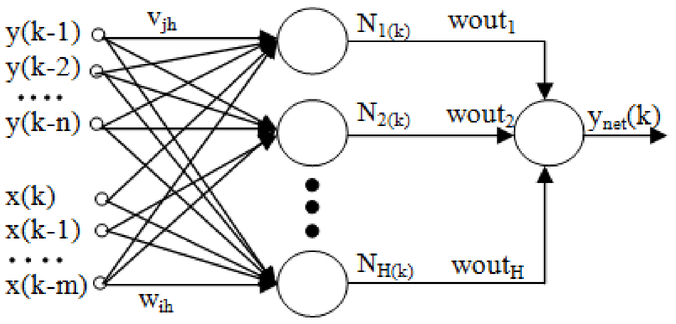


Fig. 3. Neural network structure used for system identification.

In the Fig. 3 architecture, the biases were omitted to avoid introducing new inputs to the network which may add new difficulties for model identification theory. The mathematical representation of the network will be as follows:

$$y_{net}(k) = g[\textstyle\sum_{h=1}^{H} N_h(k) wout_h] \quad (6)$$

$$N_h(k) = f[\textstyle\sum_{i=0}^{m} w_{ih} x(k-i) wout_h + \sum_{j=1}^{n} v_{jh} y(k-j)] \quad (7)$$

Where *g* and *f* represent the activation functions (transfer functions) of the output and hidden nodes respectively. Sigmoid, tangent sigmoid or other activation functions could be used.
Any identification problem demand to optimize a cost function that represents the error between the system output *y*(*k*) and the network output $y_{net}(k)$. The most commonly utilized cost function is the Sum Square Error (SSE) as in (8).

$$SSE = \sum_{k=1}^{k}(e(k))^2 = \sum_{k=1}^{k}(y_{net}(k) - y(k))^2 \tag{8}$$

Many approaches can be used to train the network by adjusting the optimal weights, such as gradient descent as in (9).

$$W_{new} = W_{old} - \alpha\frac{dJ}{dW} \tag{9}$$

Where α is the learning rate, *J* is the cost function which refers to the error between the network output and the system output.

The focus of this paper is on using converged network weights for designing a tool to mimic the original system's transfer function, rather than on weight convergence. As a result, there will be no extensive discussion of how to train the network weights. This has been well-documented in [1]. Linear and nonlinear systems have been effectively identified using neural networks, because the neural network has high approximation capabilities and adaptable characteristics.

4. Neural network to transfer function approximation

For the transfer function approximation, an ARMA model has been used to match the neural network architecture. The most commonly used activation function in a neural network is the hyper tangent which gives a flat and continuous output between +1 and -1, as well as it is simple to differentiate. As a result, the hidden node output from equation (7) becomes:

$$\boldsymbol{N_h(k) = \frac{1-e^{-\alpha net_{h(k)}}}{1+e^{-\alpha net_{h(k)}}}} \tag{10}$$

$$\boldsymbol{net_h(k) = \sum_{i=0}^{m} w_{ih}x(k-i) + \sum_{j=1}^{n} v_{jh}y(k-j)} \tag{11}$$

For the approximation of equation (10) it is possible to use Taylor expansion about the ‘0’ point, as in (12)

$$\boldsymbol{N_h(k) = \frac{1-e^0}{1+e^0} + \frac{2e^0}{(1+e^0)^2}(net_h(k)-0) + \frac{-2e^0(1-e^0)}{(1+e^0)^3}(net_h(k)-0)^2 + \cdots} \tag{12}$$

For simplicity and avoiding the nonlinearities the only first two terms are used, and equation (12) will be

$$\boldsymbol{N_h(k) = 0.5\sum_{i=0}^{m} w_{ih}x(k-i) + 0.5\sum_{j=1}^{n} v_{jh}y(k-j)} \tag{13}$$

The output neuron is

$$ynet(k) = \sum_{h=1}^{H}(wout_h N_h(k)) \qquad (14)$$

By substituting (13) in (14) results

$$ynet(k) = 0.5\sum_{h=1}^{H}\left[wout_h\left[\sum_{i=0}^{m} w_{ih}x(k-i) + \sum_{j=1}^{n} v_{jh}y(k-j)\right]\right] \qquad (15)$$

By comparing (15) with (2), the parameters of ARMA model can be approximated as in (16) and (17)

$$\hat{a}_j = 0.5\left[\sum_{h=1}^{H}(wout_h v_{jh})\right] \qquad (16)$$

$$\hat{b}_i = 0.5\left[\sum_{h=1}^{H}(wout_h w_{ih})\right] \qquad (17)$$

Equations (16) and (17) confirm the mathematical relation between the ARMA model parameters and neural network weights, where finally the neural network used for the parametric system identification and transfer function approximation. Note that, in the case of using the sigmoid activation function the parameters $\hat{a}$ and $\hat{b}$ will be multiplied by 0.25 instead of 0.5, and there an offset factor equal to $(0.5\sum_{h=1}^{H} wout_h)$ will be added to the final predicted output, brief explanation in [9]. In general, the advantage of using neural networks in systems identification rather than traditional methods, due to its capability to model nonlinear dynamic system according to the nonlinear mapping of its structure, as well as, its capability to adapts and training in online mode which make it very good choice for identifying dynamic systems.

5. Model predictive control (MPC) algorithm

MPC is considered one of the advanced methods of process control, which is considered a type of open-loop optimal control method. MPC refers to a type of computer control algorithm that uses a linear system model to predict the future output response of a real system is primarily based on a problem of optimization at each time sample, k. The major purpose of this optimization issue is to find a new control input sequence while also taking into account the system output and input constraints [10] [11] [12]. The structure of an MPC algorithm is shown in Fig. 4, which is organized into three essential parts:

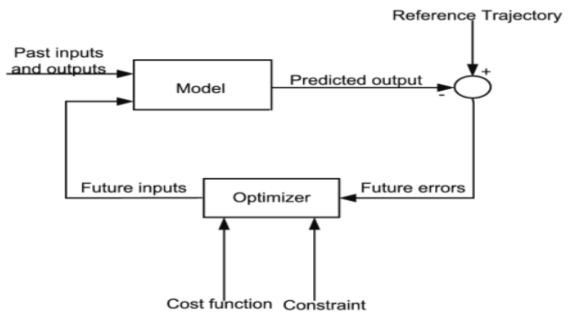


Fig. 4. Basic structure of MPC

5.1. Model

The fundamental challenge with MPC algorithm design is that it requires a system model, i.e., a model that specifies the system's input to output relationship. It is possible to utilize a linear or nonlinear mathematical model generated from physical rules or empirical data. In MPC it is supposed that the model is a discrete state-space model as in (18).

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{18}$$

Where x_k system state, y_k system output, x_{k+1} is the predicted state, B is the matrix of the system input, A is the matrix of the system, C is the matrix of the system output, D is the feedforward matrix, (D is zero matrix in a case where the system model does not have direct feedthrough).

5.2. Cost function

The MPC main concept is that it computes a vector of future control in such a way that a cost function is minimized inside the control and prediction horizons Nc, Np, respectively where (Nc ≤ Np), as seen in Fig. 5.

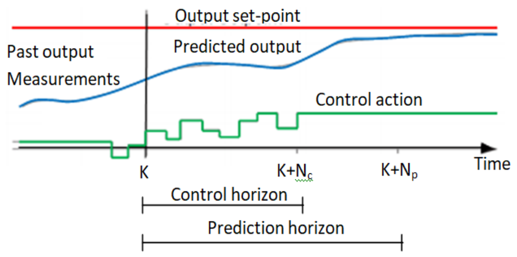


Fig. 5. Control and prediction horizon

For generic MIMO systems, the cost function frequently utilized in MPC is as shown in (19) (linear-quadratic function).

$$\boldsymbol{J} = \sum_{K=0}^{Np} (\hat{y} - \boldsymbol{r})^{T} \boldsymbol{Q} (\hat{y} - \boldsymbol{r}) + \sum_{K=0}^{Nc} \Delta \boldsymbol{u}^{T} \boldsymbol{R} \Delta \boldsymbol{u} \tag{19}$$

Where *Nc* is control horizon, *Np* is prediction horizon, $\hat{y}$ is predicted system output, *r* is set-point, *R* is control weight matrix, *Q* is output error weight matrix, Δu is predicted change in control value as in (20).

$$\Delta u_k = u_k - u_{k-1} \tag{20}$$

Where for SISO systems as in (21)

$$\boldsymbol{J} = \sum_{K=0}^{Np} \boldsymbol{q} (\hat{y} - \boldsymbol{r})^2 + \sum_{K=0}^{Np} \boldsymbol{r} \Delta \boldsymbol{u}^2 \tag{21}$$

So the basic problem is to solve: $\frac{\partial J}{\partial u} = 0$

We obtain the future optimal control input by solving this optimization issue.

5.3. Constraints

Most of the practical systems have constraints, like actuator limits, as well as safety restrictions like pressure and temperature. Furthermore, we have performance constraints and limitations such as overshoot. Normally, MPC defines the following constraints:

a) *Constraints in the outputs:*

$$y_{min} > y > y_{max}$$

b) *Constraints in the input and rate change of input:*

$$\Delta u_{min} > \Delta u > \Delta u_{max}$$

$$\boldsymbol{u}_{min} > \boldsymbol{u} > \boldsymbol{u}_{max}$$

When determining future controls, the MPC algorithm takes all of these limitations into considerations.

6. Software design using LabVIEW software

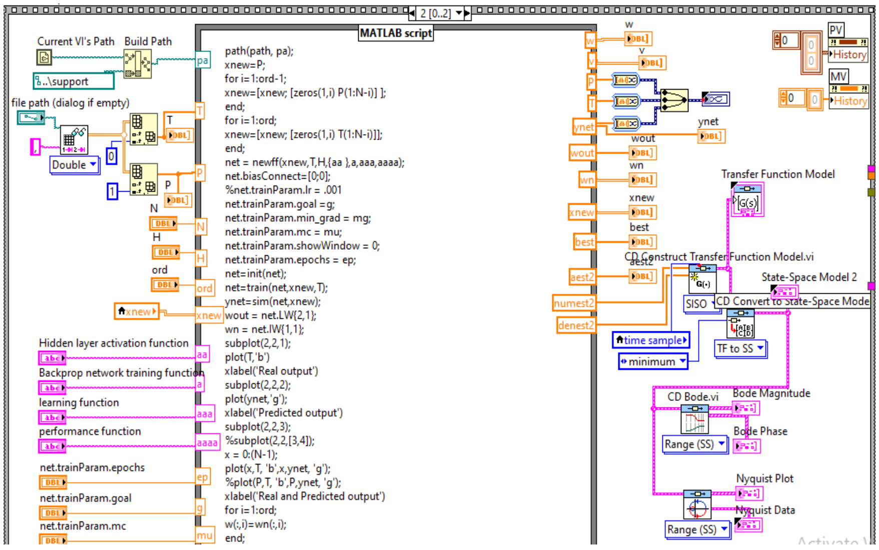
The LabVIEW design of the tool includes the of the Block Diagram part and the Front Panel Page part

6.1. Block Diagram design

The Block Diagram designed with two main control tab the first control tab for the identification algorithm and the second for the MPC controller

a) Identification algorithm

In this part, the algorithm of the identification has been written using the MathScript facility in LabVIEW according to the mathematical analysis mentioned in section 3. The architecture of ANN and the optimal hidden and output neuron’s weight have been found using the MATLAB functions “newff()” and “train()” as shown in Fig. 6.



```
path(path, pa);
xnew=P;
for i=1:ord-1;
xnew=[xnew; [zeros(1,i) P(1:N-i)] ];
end;
for i=1:ord;
xnew=[xnew; [zeros(1,i) T(1:N-i)]];
end;
net = newff(xnew,T,H,{aa },a,aaa,aaaa);
net.biasConnect=[0;0];
%net.trainParam.lr = .001
net.trainParam.goal =g;
net.trainParam.min_grad = mg;
net.trainParam.mc = mu;
net.trainParam.showWindow = 0;
net.trainParam.epochs = ep;
net=init(net);
net=train(net,xnew,T);
ynet=sim(net,xnew);
wout = net.LW{2,1};
wn = net.IW{1,1};
subplot(2,2,1);
plot(T,'b')
xlabel('Real output')
subplot(2,2,2);
plot(ynet,'g');
xlabel('Predicted output')
subplot(2,2,3);
%subplot(2,2,[3,4]);
x = 0:(N-1);
plot(x,T, 'b',x,ynet, 'g');
%plot(P,T, 'b',P,ynet, 'g');
xlabel('Real and Predicted output')
for i=1:ord;
w(:,i)=wn(:,i);
end;
```

Fig. 6. Block diagram design of the neural network system identification tool

The nominator and denominator of the discrete transfer function that result from the MathScript code, used to construct the transfer function of the system by using the LabVIEW Vis “CD Transfer Function Model.vi”. As it mentioned before the MPC algorithm relies on state-space representing, for that, a VIs “CD Convert to State-Space Model.vi” is used to convert the transfer function model to state-space model as shown in Fig (4).

b) MPC part

For the design of the MPC controller, a selector switch has been added for the purpose of selecting the model that will be adopted in the MPC controller’s calculations. There are two types of models that are selected by the selector. The first model is the model which constructed manually through using the Vis “CD Construct Special TF Model.vi” which is converted from continuous to discrete by using the Vis “CD Convert Continuous to Discret.vi”, and then convert the transfer function to state-space, the second model is the identified model which derived from the identification algorithm in MathScript. As shown in Fig. 7.

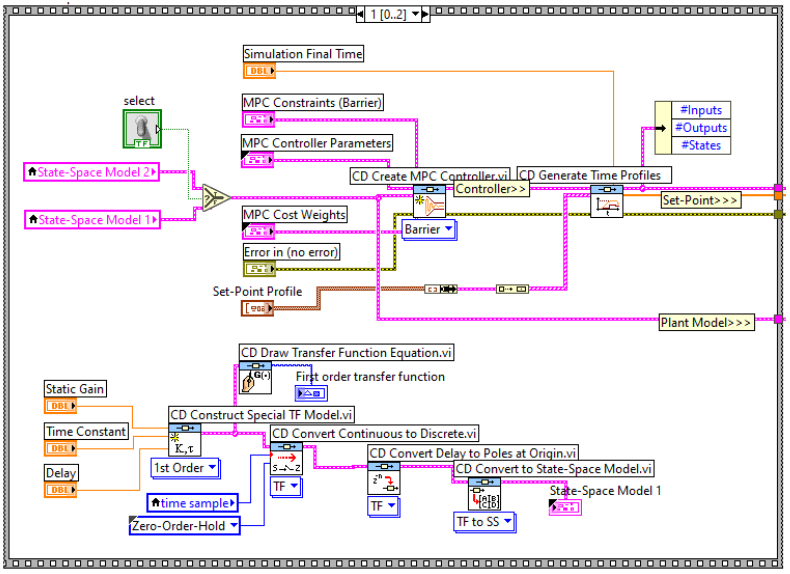

1 [0..2]
Simulation Final Time
select
MPC Constraints (Barrier)
MPC Controller Parameters
State-Space Model 2
State-Space Model 1
CD Create MPC Controller.vi
CD Generate Time Profiles
Controller>>
#Inputs
#Outputs
#States
Set-Point>>>
MPC Cost Weights
Barrier
Error in (no error)
Set-Point Profile
Plant Model>>>
CD Draw Transfer Function Equation.vi
First order transfer function
Static Gain
CD Construct Special TF Model.vi
Time Constant
CD Convert Continuous to Discrete.vi
1st Order
CD Convert Delay to Poles at Origin.vi
Delay
CD Convert to State-Space Model.vi
TF
State-Space Model 1
time sample
TF
TF to SS
Zero-Order-Hold

Fig. 7. Model constructing Block diagram design for MPC

Finally, implementing the MPC controller through using the Vis “CD Implement MPC Controller.vi” as shown in Fig. 8.

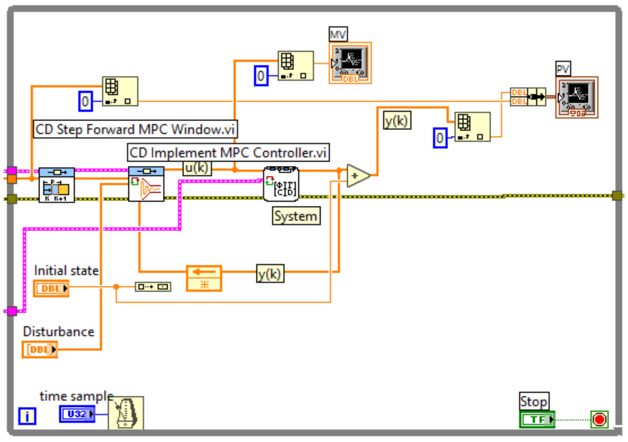


Fig. 8. MPC Block diagram design

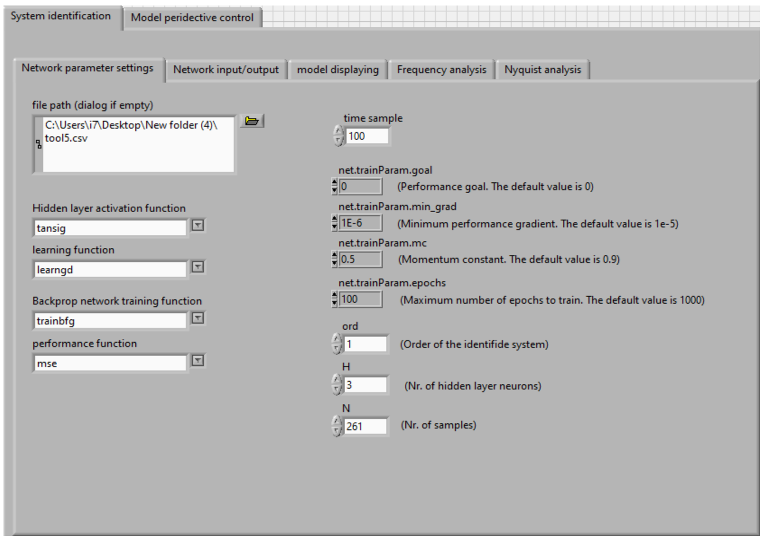
6.2. Front Panel design

The front panel is designed from two main control tabs, one for the identification and the other for the MPC controller. Each one from these two main control tabs include sub-control tabs

a) Identification part

The page of identification algorithm include the following control tabs

- Network parameter settings. On this page, the data input/output of the real or simulated system is imported through the ‘file path’ window. On the same menu page, it is possible to adjust the order of the identified system and the number of hidden neurons of the network. Moreover, it is possible to adjust all the Neural network settings and training options, like the activation function “tansig or sigmoid”. Training function, which can be any of the backpropagation training functions like “trainbfg, trainlm, trainrp and traingd”. Learning function, which can be “learngd or learngdm”. Performance function, which can be “mse or msereg” [13] [14] [15]. All these functions are MATLAB functions and written using MathScript facility in LabVIEW as shown Fig. 9.



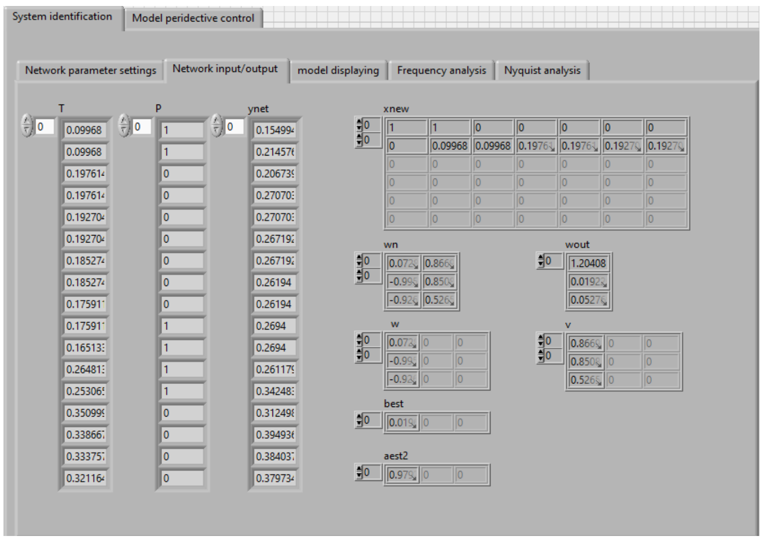
file path (dialog if empty)
C:\Users\i7\Desktop\New folder (4)\tool5.csv

Hidden layer activation function: tansig
learning function: learngd
Backprop network training function: trainbfg
performance function: mse

time sample: 100
net.trainParam.goal: 0 (Performance goal. The default value is 0)
net.trainParam.min_grad: 1E-6 (Minimum performance gradient. The default value is 1e-5)
net.trainParam.mc: 0.5 (Momentum constant. The default value is 0.9)
net.trainParam.epochs: 100 (Maximum number of epochs to train. The default value is 1000)
ord: 1 (Order of the identifide system)
H: 3 (Nr. of hidden layer neurons)
N: 261 (Nr. of samples)

Fig. 9. Network parameter settings page

- Network input/output. This page shows the vectors of input and output data and the optimized weights as shown in Fig. 10.



T	P	ynet
0.09968	1	0.15499
0.09968	1	0.21457
0.19761	0	0.20673
0.19761	0	0.27070
0.19270	0	0.27070
0.19270	0	0.26719
0.18527	0	0.26719
0.18527	0	0.26194
0.17591	0	0.26194
0.17591	1	0.2694
0.16513	1	0.2694
0.26481	1	0.26117
0.25306	1	0.34248
0.35099	0	0.31249
0.33866	0	0.39493
0.33375	0	0.38403
0.32116	0	0.37973

xnew

1	1	0	0	0	0	0
0	0.09968	0.09968	0.1976	0.1976	0.1927	0.1927
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

wn

0.072	0.866
-0.99	0.850
-0.92	0.526

wout

1.20408
0.0192
0.0527

w

0.072	0	0
-0.99	0	0
-0.92	0	0

v

0.866	0	0
0.850	0	0
0.526	0	0

best

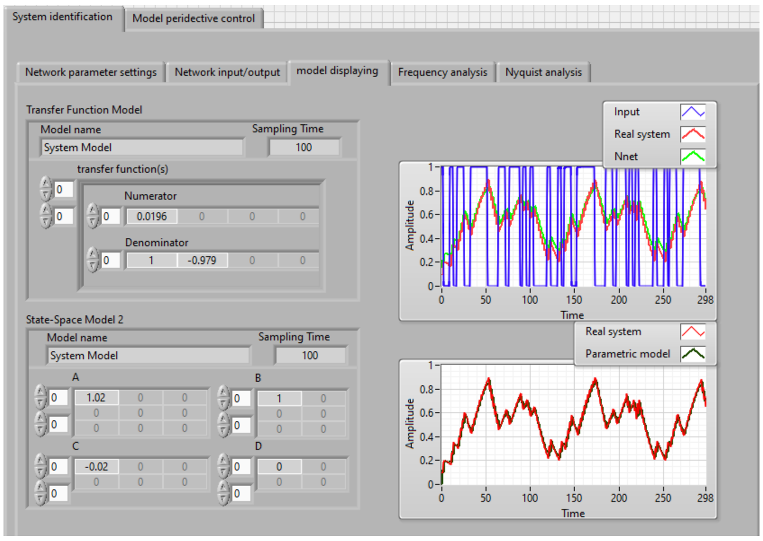
0.01	0	0

aest2

0.97	0	0

Fig. 10. Network input/output page

- Model displaying. This page shows the mathematical model result of the identified system in transfer function and state-space representing, as well as the chart of the input/output data curves for the real system and the neural network output. The other chart shows the output response of the real system and the parametric identified model as shown in Fig. 11.



Transfer Function Model

Model name: System Model
Sampling Time: 100

transfer function(s)

Numerator: 0.0196 0 0 0

Denominator: 1 -0.979 0 0

State-Space Model 2

Model name: System Model
Sampling Time: 100

A

1.02	0	0
0	0	0
0	0	0

B

1	0
0	0
0	0

C

-0.02	0	0
0	0	0

D

0	0
0	0

Fig. 11. Model displaying page

- Frequency analysis. This page shows the bode plot of the frequency response for the identified model, displaying the Bode magnitude plot and Bode Phase plot for the identified system model as shown in Fig. 12.

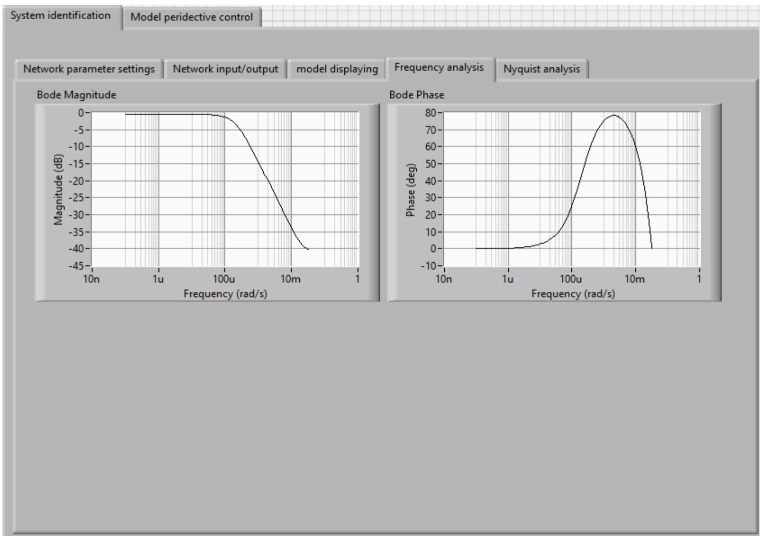
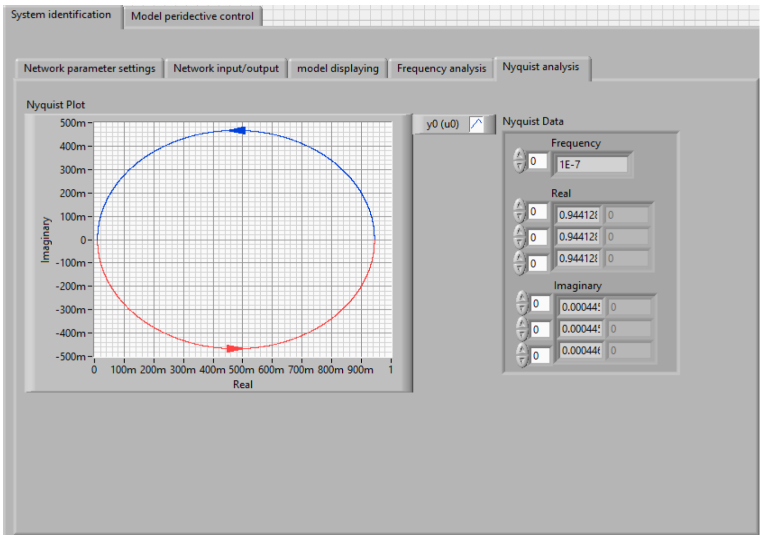


Fig. 12. Frequency analysis page

- Nyquist analysis. This page shows the Nyquist plot and data for the identified system model as shown in Fig. 13.



Nyquist Data

Frequency

0	1E-7

Real

0	0.94412	0
0	0.94412	0
0	0.94412	0

Imaginary

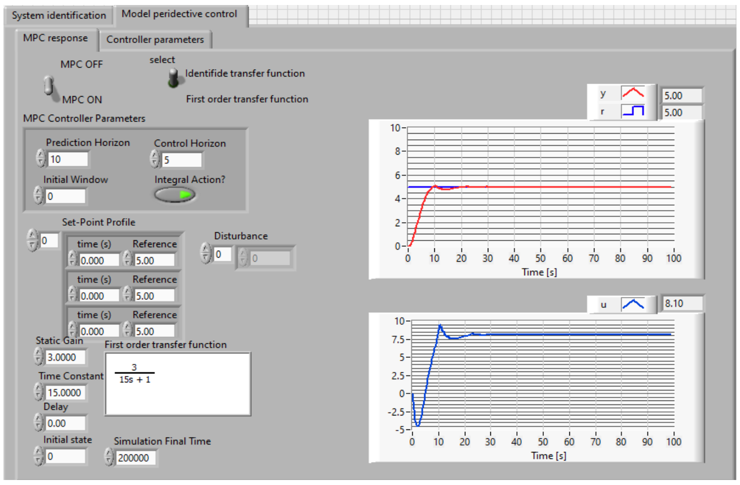
0	0.00044	0
0	0.00044	0
0	0.00044	0

Fig. 13. Nyquist analysis page

b) MPC part

The page of identification algorithm includes the following control tabs

- MPC response
  This page includes the data entry of the prediction and control horizon, selector to select between the model-based from the MPC algorithm, either from the manually setting transfer function or from the identified model transfer function, in addition to the chart to display the output response of the system output with MPC control and the control input action as shown in Fig. 14



MPC OFF / MPC ON

select: Identifide transfer function / First order transfer function

MPC Controller Parameters

Prediction Horizon: 10
Control Horizon: 5
Initial Window: 0
Integral Action?

Set-Point Profile: 0

time (s)	Reference
0.000	5.00
0.000	5.00
0.000	5.00

Disturbance: 0 / 0

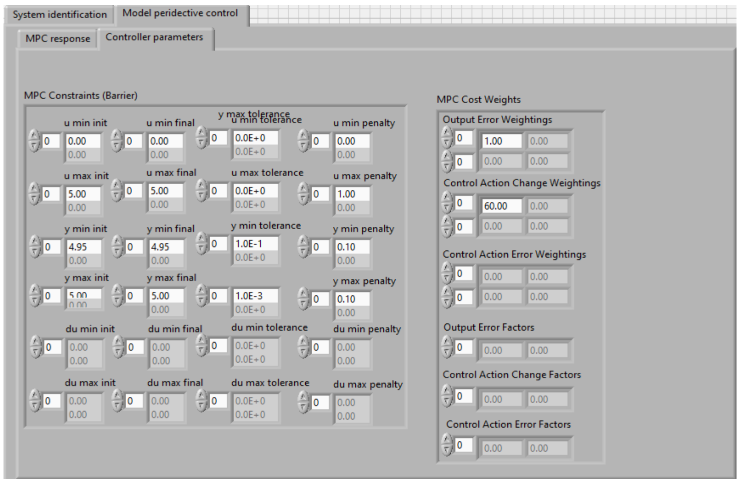
Static Gain: 3.0000
Time Constant: 15.0000
Delay: 0.00
Initial state: 0

First order transfer function: $\frac{3}{15s+1}$

Simulation Final Time: 200000

Fig. 14. Front Panel page of MPC response

- Controller parameters
  This page includes the setting of the MPC constraints (Barrier) and the controller cost weights parameters as shown in Fig. 15.



MPC Constraints (Barrier)

u min init: 0 / 0.00 / 0.00	u min final: 0 / 0.00 / 0.00	y max tolerance / u min tolerance: 0 / 0.0E+0 / 0.0E+0	u min penalty: 0 / 0.00 / 0.00
u max init: 0 / 5.00 / 0.00	u max final: 0 / 5.00 / 0.00	u max tolerance: 0 / 0.0E+0 / 0.0E+0	u max penalty: 0 / 1.00 / 0.00
y min init: 0 / 4.95 / 0.00	y min final: 0 / 4.95 / 0.00	y min tolerance: 0 / 1.0E-1 / 0.0E+0	y min penalty: 0 / 0.10 / 0.00
y max init: 0 / 5.00 / 0.00	y max final: 0 / 5.00 / 0.00	0 / 1.0E-3 / 0.0E+0	y max penalty: 0 / 0.10 / 0.00
du min init: 0 / 0.00 / 0.00	du min final: 0 / 0.00 / 0.00	du min tolerance: 0 / 0.0E+0 / 0.0E+0	du min penalty: 0 / 0.00 / 0.00
du max init: 0 / 0.00 / 0.00	du max final: 0 / 0.00 / 0.00	du max tolerance: 0 / 0.0E+0 / 0.0E+0	du max penalty: 0 / 0.00 / 0.00

MPC Cost Weights

Output Error Weightings: 0 / 1.00, 0.00; 0 / 0.00, 0.00
Control Action Change Weightings: 0 / 60.00, 0.00; 0 / 0.00, 0.00
Control Action Error Weightings: 0 / 0.00, 0.00; 0 / 0.00, 0.00
Output Error Factors: 0 / 0.00, 0.00
Control Action Change Factors: 0 / 0.00, 0.00
Control Action Error Factors: 0 / 0.00, 0.00

Fig. 15. Front Panel page of MPC parameters setting

5. Results

The neural network system identification tool was tested on the proposed second-order ARMA model as in (22). The proposed system is stimulated by a pseudo-random binary sequence (PRBS) input signal as illustrated in Fig. 16, which represents the LabVIEW design of the PRBS input generator and the proposed system. The input/output data is logged and imported by the tool as explained in Fig. 9.

$$y(k) = 0.099 * u(k-1) - 0.082 * u(k-2) + 1.85 * y(k-1) - 0.082 * y(k-2) \quad (22)$$

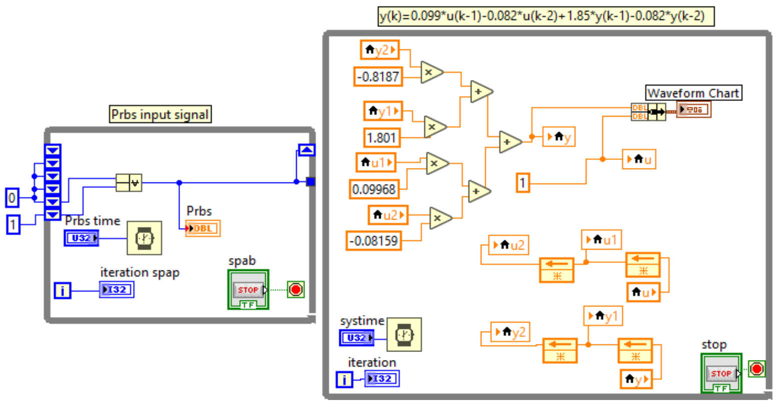


Fig. 16. PRBS signal generator and ARMA system model by LabVIEW

The test showed very good results in terms of estimation accuracy and the matching between the proposed system and the identified model. Fig. 17 shows the input signal which stimulated the supposed system and the output response of the original system with the neural network output.

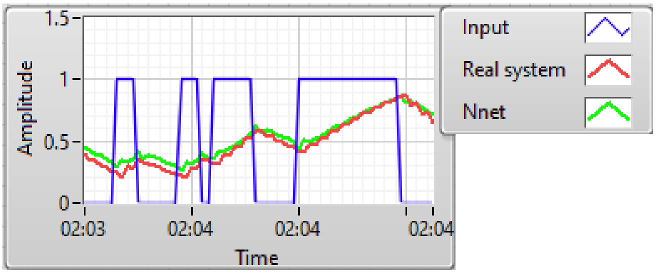


Fig. 17. Output response of the original system with neural network output

Fig. 18 illustrates the output of the supposed system with the parametric ARMA identified model which is illustrated in (23). It is clear from Fig. 17, Fig. 18, and equation (23), that the identified model, approximately matches the original system with high accuracy.

$$y(k) = 0.1028 * u(k-1) - 0.0845 * u(k-2) + 1.701 * y(k-1) - 0.7187 * y(k-2) \quad (23)$$

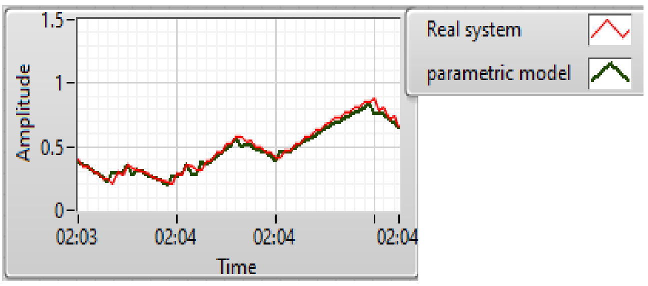


Fig. 18. Output response of the original system with an estimated parametric model

Another third-order ARMA model was proposed to show the capability of the neural network to model high order system as in (24).

$$y(k) = 0.0143 * u(k-1) - 0.0923 * u(k-2) + 0.1183 * u(k-3) + 0.998 * y(k-1) + 0.653 * y(k-2) + 0.4724 * y(k-3) \quad (24)$$

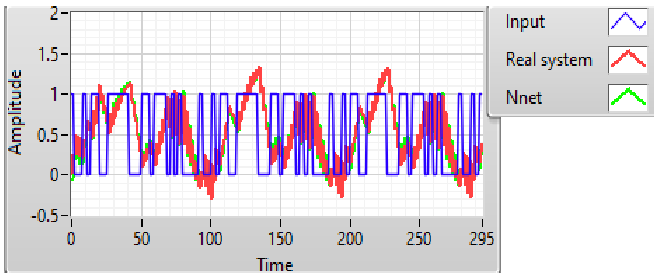


Fig. 19. Output response of the original system with neural network output

Fig. 19 shows the input signal which stimulated the supposed system and the output response of the original system with the neural network output.

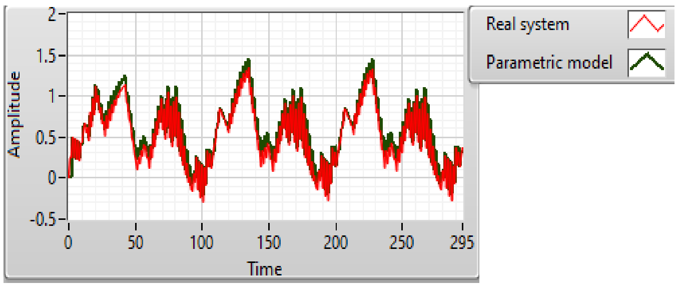


Fig. 20. Output response of the original system with an estimated parametric model

Fig. 20 illustrates the output of the proposed third-order system with the parametric ARMA identified model which is illustrated in (25). It is clear from Fig. 19, Fig. 20, and equation (25), that the identified model, approximately matches the original system with high accuracy.

$$\boldsymbol{y(k) = 0.00843 * u(k-1) - 0.0123 * u(k-2) + 0.203 * u(k-3) + 0.198 * y(k-1) + 0.951 * y(k-2) + 0.205 * y(k-3)} \tag{25}$$

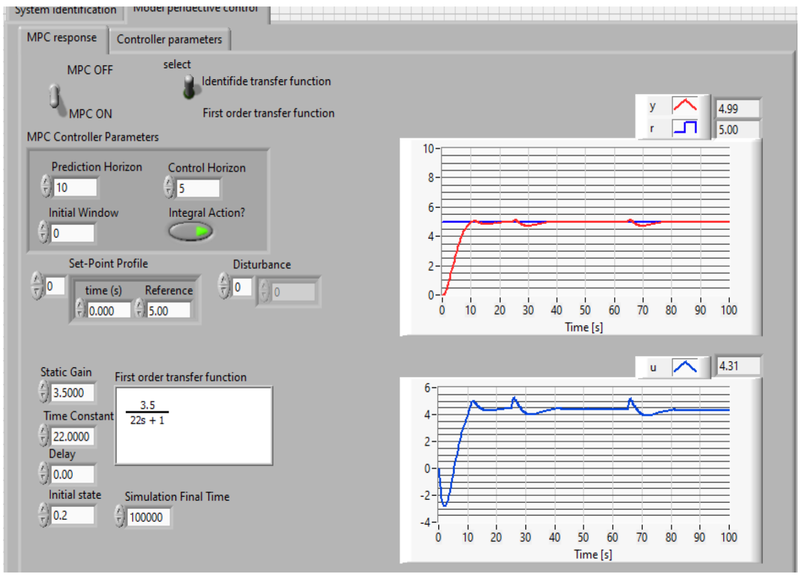
For the sake of result confirmation table 1 shows some statistical data regarding the results of the neural network model when the network structure and ARMA model were changed (for the first example). The model order was changed from 2 to 3 and the number of hidden neurons was changed from 3 to 4. These changes were applied to examine the proposed tool with different conditions and aspects to generate some statistical data. In all cases, a linear activation function was utilized in the network output layer, and the hyper tangent activation function was utilized to activate the neurons in the hidden layer

Table 1

Statistical data of the identified neural network model

Model order	No. of hidden neurons	MIN	MAX	Average	Variance
Second	3	0.000116	0.019676	-0.03789	0.002551627
	4	0.000354	0.023991	-0.03394	0.002479
Third	3	0.000354	0.023991	-0.00052	0.002479
	4	3.9E-05	0.012211	-0.0339	0.002765306

Regarding the control part, Fig. 21 shows the perfect effect of the designed MPC controller on the proposed system through the setpoint tracking and overshoot eliminating. To examine the robustness of the controller, a disturbance input was added to the system output in seconds 30 and 70, respectively, and it’s clear from Fig. 19, the small effect of the disturbances on the output response of the system was rejected rapidly through the high and fast response of the MPC controller



MPC Controller Parameters

Prediction Horizon 10
Control Horizon 5
Initial Window 0
Integral Action?

Set-Point Profile 0
time (s) 0.000
Reference 5.00
Disturbance 0 0

Static Gain 3.5000
Time Constant 22.0000
Delay 0.00
Initial state 0.2
Simulation Final Time 100000

First order transfer function

$$\frac{3.5}{22s + 1}$$

y 4.99
r 5.00
u 4.31

Fig. 21. Output response of the simulated system with MPC

6. Conclusions

A neural network as an important part of artificial intelligence theories has been utilized in a variety of practical applications. Researchers have had a lot of success using neural network models in system identification and control applications. These network models, on the other hand, hide the system's parametric information within their architecture. This paper showed a design of a software tool for solving the system identification problem using LabVIEW software, getting into consideration the advantage of MATLAB neural network functions through the MathScript facility, and using the identified model in the design of the MPC controller. This tool can transform the weights of the neurons in a multi-layer neural network to an estimated transfer function for any real or simulated system through the import of the input-output data of the system and can be used to discover important information about the estimated system with some reliable options like the order of the identified system and the number of the hidden neurons and some other training options. Testing the tool involved estimating an arbitrary second-order system and showing the comparison between the neural network output and the output of the proposed system with the output of the parametric estimated system. For this example, a table of some statistical data is illustrated regarding to the results of the neural network model when the network structure and ARMA model were changed. These changes were applied to examine the proposed tool with different conditions and aspects. A third-order system was examined to show the capability of the neural network to model high-order systems. The test showed very good results in terms of the estimation accuracy and the matching between the proposed systems with the identified models, regarding to system identification problem. Regarding to the control problem, the MPC controller which implemented by the VIs LabVIEW functions showed a perfect setpoint tracking and high robustness due to the fast disturbance rejecting. The tool can be developed in the future for more identification requirements and with more comprehensive features to achieve some functionality requirements of any simulated or real-time identification and control application.

R E F E R E N C E S

[1]. S. Haykin, Neural Networks and Learning Machines 3rd edition. Prentice Hall 2008.

[2]. B. Haznedar, A. Kalinli, "Training ANFIS structure using simulated annealing algorithm for dynamic systems identification, Neurocomputing," 2018, Pages 66-74, https://doi.org/10.1016/j.neucom.2018.04.006.

[3]. O.Isaac Abiodun, A. Jantan, A. Omolara, K. Dada, N. Mohamed, H.a Arshad, "State-of-the-art in artificial neural network applications: A survey," Volume 4, Issue 11, 2018, ISSN 2405-8440, https://doi.org/10.1016/j.heliyon.2018.e00938.

[4]. H. Demuth, M. Beale. And M. Hagan. Neural Network Toolbox 6 User’s Guide Mathworks 2009.

[5]. Landau ID, Zito G., Digital control systems. London: Springer; 2006.

[6]. WANG, Chen, et al. Auto-regressive moving average parameter estimation for 1/f process under colored Gaussian noise background. Journal of Algorithms & Computational Technology, 2019, 13: 1748302619867439.

[7]. G. Franklin, J. Powel and M. Workman, Digital Control of Dynamic Systems. 3rd edition Ellis-Kagle Press 1998.

[8]. SN. Kumpati. P. Kannan,“Identification an control of dynamical systems using neural networks”, IEEE transactions on neural networks. VOL. I. NO. I. March 1990.

[9]. T. A. Tutunji “Approximating transfer functions using neural network weights,”2009 4th International IEEE/EMBS Conference on Neural Engineering, 04 2009.

[10]. M. Norambuena, J. Rodriguez, Z. Zhang, F. Wang, C. Garcia and R. Kennel, "A Very Simple Strategy for High-Quality Performance of AC Machines Using Model Predictive Control," in IEEE Transactions on Power Electronics, vol. 34, no. 1, pp. 794-800, Jan. 2019, doi: 10.1109/TPEL.2018.2812833.

[11]. G. Marafioti,“Enhanced Model Predictive Control: Dual Control Approach and State Estimation Issues,” Doctoral theses at NTNU, 2010:235, ISBN 978-82-471-2462-8, ISSN 1503-8181.

[12]. Rawlings, J.B., Mayne, D.Q., and Diehl, M.M,“Model Predictive control:Theory, Computation, and Design, Second Edition” (2017). Nob Hill Publishing.

[13]. Hans-petter Halvorsen, “Tutorial on Model Predictive Control in LabVIEW”, Department of Electrical Engineering, Information Technology and Cybernetics, Telemark University College, 2011.

[14]. K. Deepak, K. R. Sharma and T. Ananthan, "Model Predictive Control for rotary inverted pendulum using LabVIEW", IOP Conference Series: Materials Science and Engineering, vol. 577, no. 1, pp. 012113, 2019

[15]. Hans-Petter Halvorsen, (2011) LabVIEW Math Script. Tutorial. Telemark University College, Norway.

[16]. MATLAB, www.mathworks.com.

[17]. ***National Instruments, https://www.ni.com/pdf/manuals/371057g.pdf.