**NEURAL NETWORKS BASED CONTROLLER FOR DC MOTOR POSITION**

**Transfer Function**

J = 3.2284E-6; %moment of inertia

b = 3.5077E-6; %motor viscous friction constant

K = 0.0274; %Gain

R = 4; %electric resistance

L = 2.75E-6; %electric inductance

s = tf('s'); %define transfer function

p\_motor = K/(s\*((J\*s+b)\*(L\*s+R)+K\*K)) %define transfer function

P\_motor =

0.0274

-------------------------------------------

8.878e-12 s^3 + 1.291e-05 s^2 + 0.0007648 s

Continuous-time transfer function.

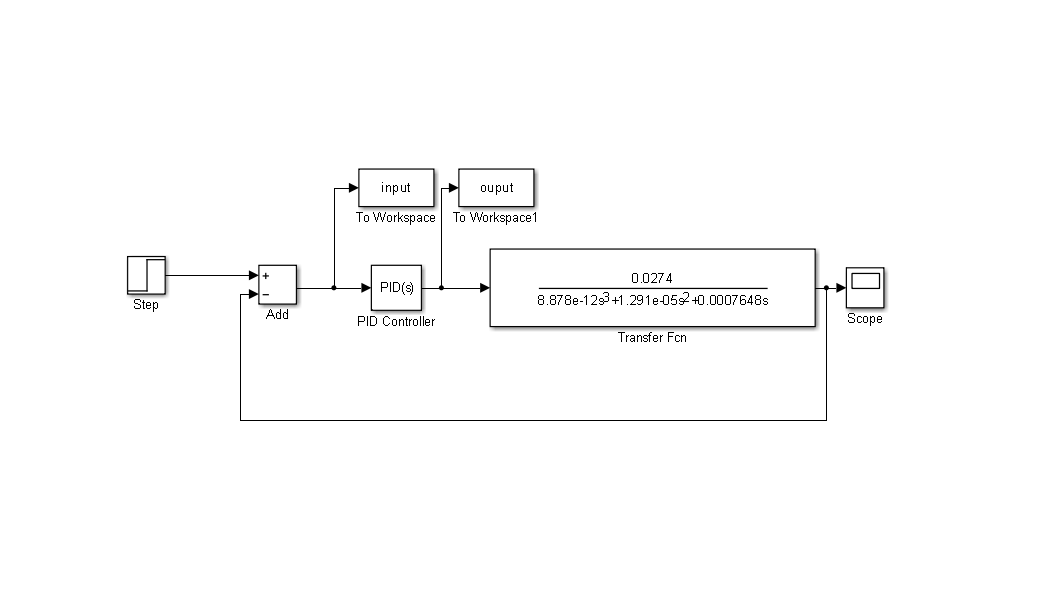


Figure : Simulink Model for Motor Position

**Simulink** **Model** is constructed as shown above. The PID Controller tunes the output to finally obtain peak overshoot of 9% as shown below.



Figure : PID tuned unit step response

After the data from **SIMULINK** is exported to **WORKSPACE**, we can convert the column arrays to row arrays as this is the way the the neural net would accept the input and target pairs.

Then type:

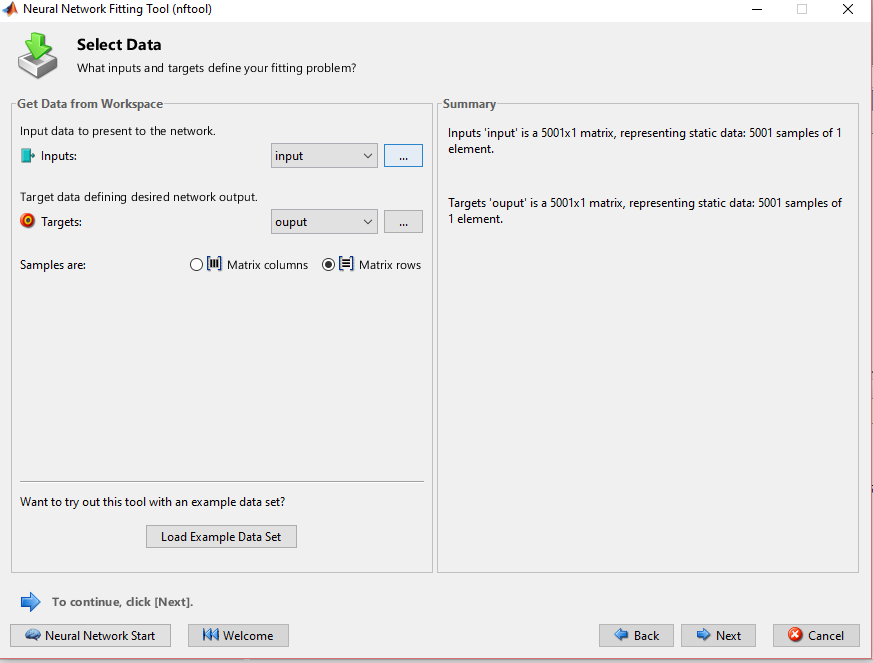
**>> nftool**

A GUI presented for Neural Network Fitting Tool.

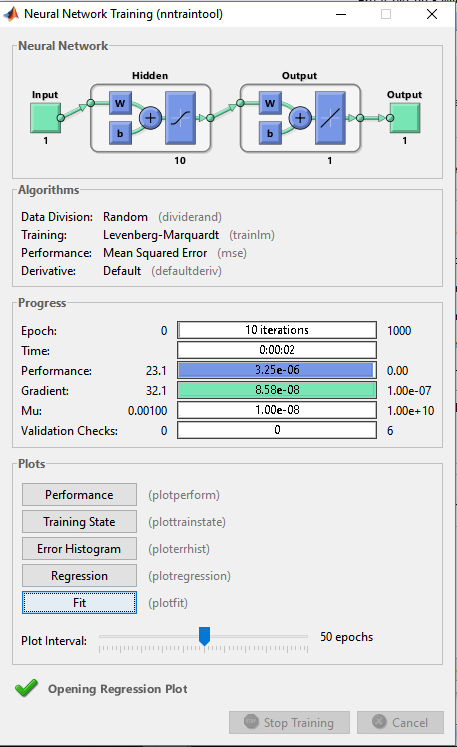
The network is TRAINED.

**>> nftool**

A GUI presented for Neural Network Fitting Tool as shown below. Inputs are named as “**INPUT**”. Outputs are named as “**OUTPUT**”. Samples are being “**Matrix** **Rows**”



The training parameters inputted to the neural network to build a PID controller using neural networks:



Checking the values of **PERFORMANCE** and **REGRESSION:**

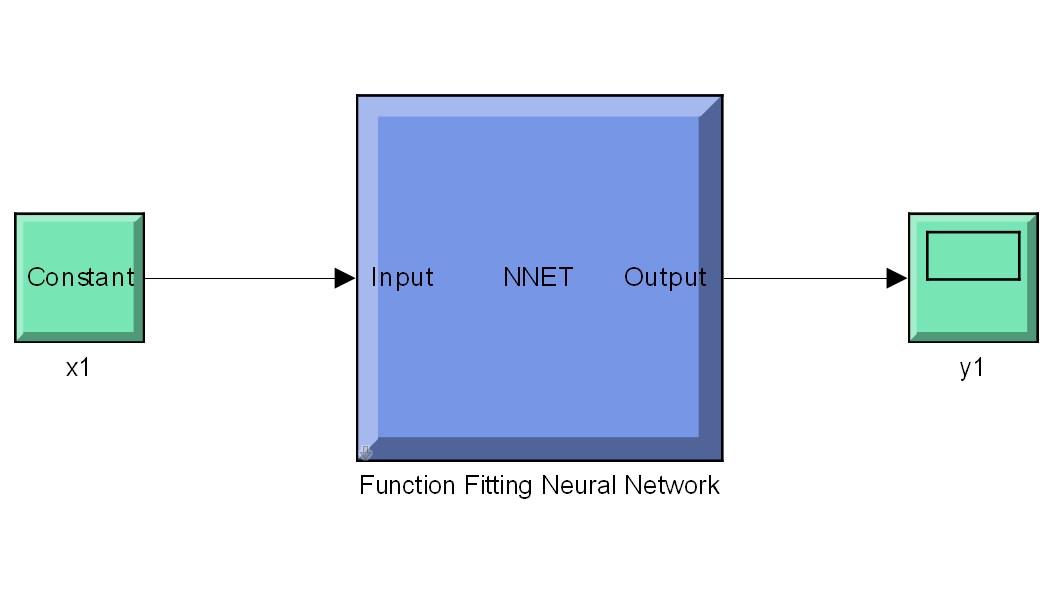
The Performance plot should be such that the **Mean Squared Error (MSE)** is minima between the **Train and (Validation and Test).** Below, it is seen that error is of the magnitude 10e**‑4**

****

The **R values** for Training set, Validation set, Test set, and overall. R should be as close to 1 as possible.

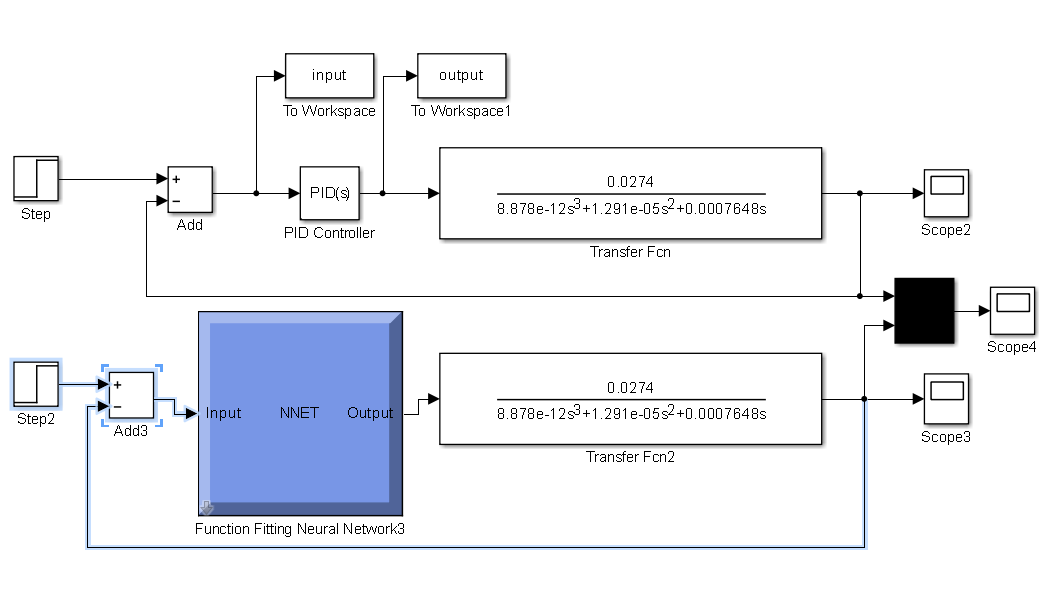
****

The neural network equivalent is:



Now that the neural network is trained completely, the next step is to generate SIMULINK DIAGRAM from the GUI itself. The scope colors appear as : Yellow, purple, blue.

Import the neural network model into the Simulink model:



figure();

plot(pid\_controller\_output3(:,1),pid\_controller\_output3(:,2),'r',pid\_controller\_output3(:,1),pid\_controller\_output3(:,4));

xlabel('time in seconds');

ylabel('unit step response');

legend('PID tuning','neural network tuning');

Step response obtained using PID Tuned system and Neural network tuned plant is obtained and it is concluded that the neural network performs exactly the same manner as the PID tuner.

