



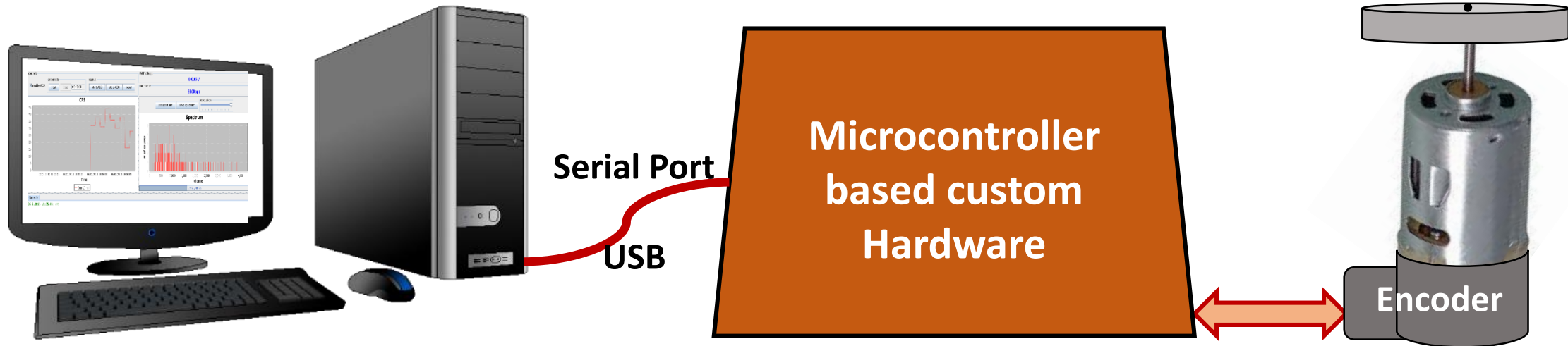
Indian Institute of Technology Bombay

Hardware Interfacing & Control

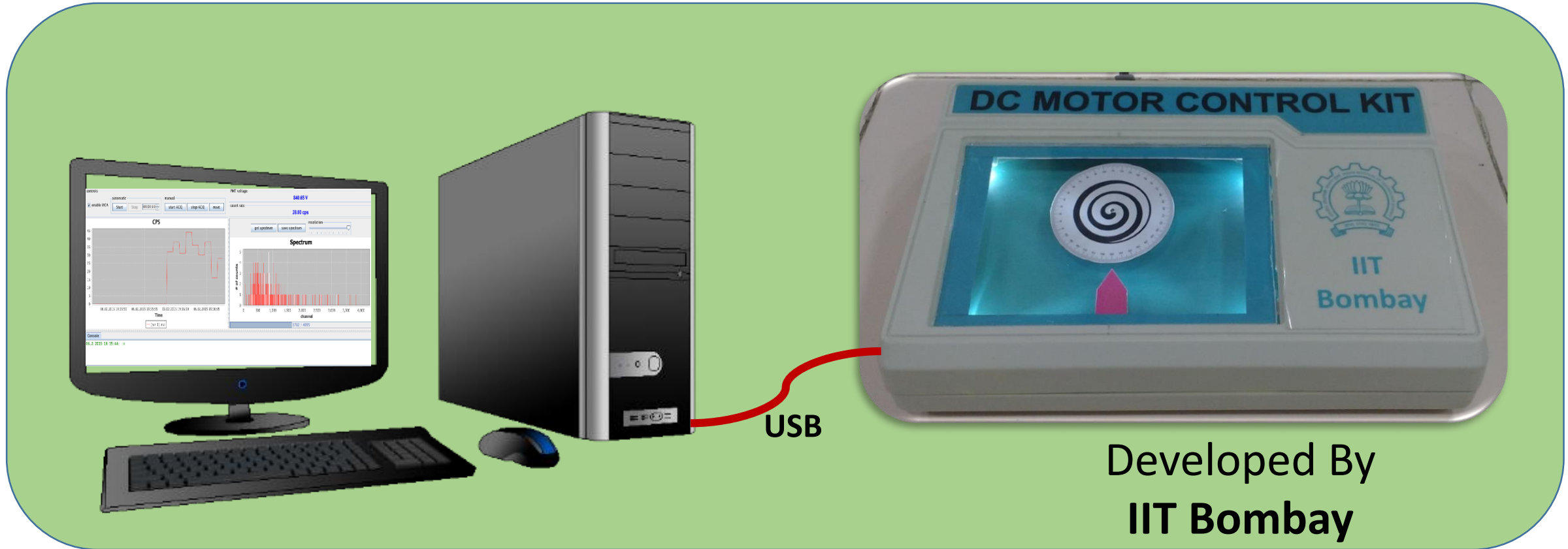
Prof. P. S. V. Nataraj

PC Based Interface

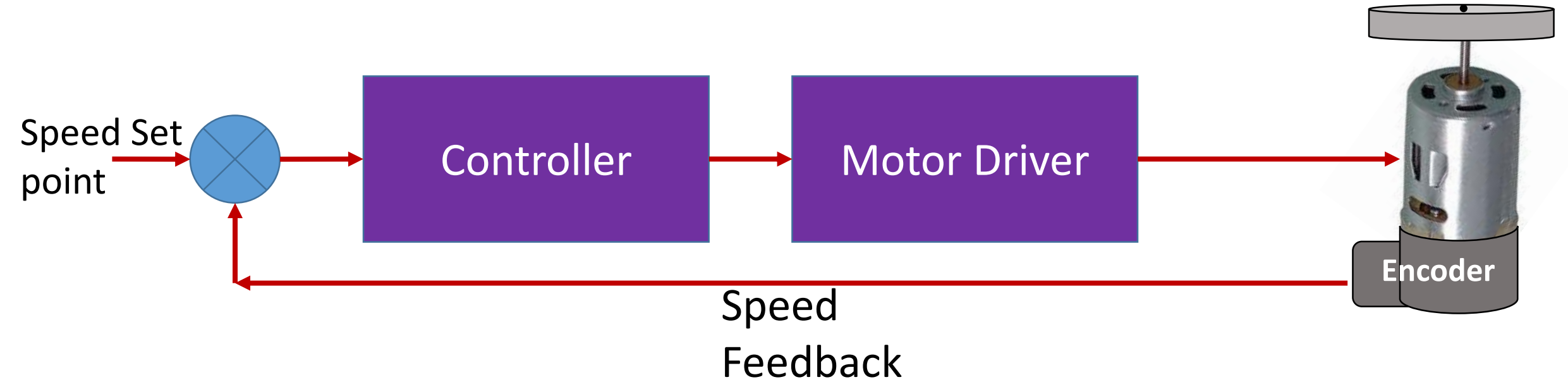
Using Custom Hardware interface



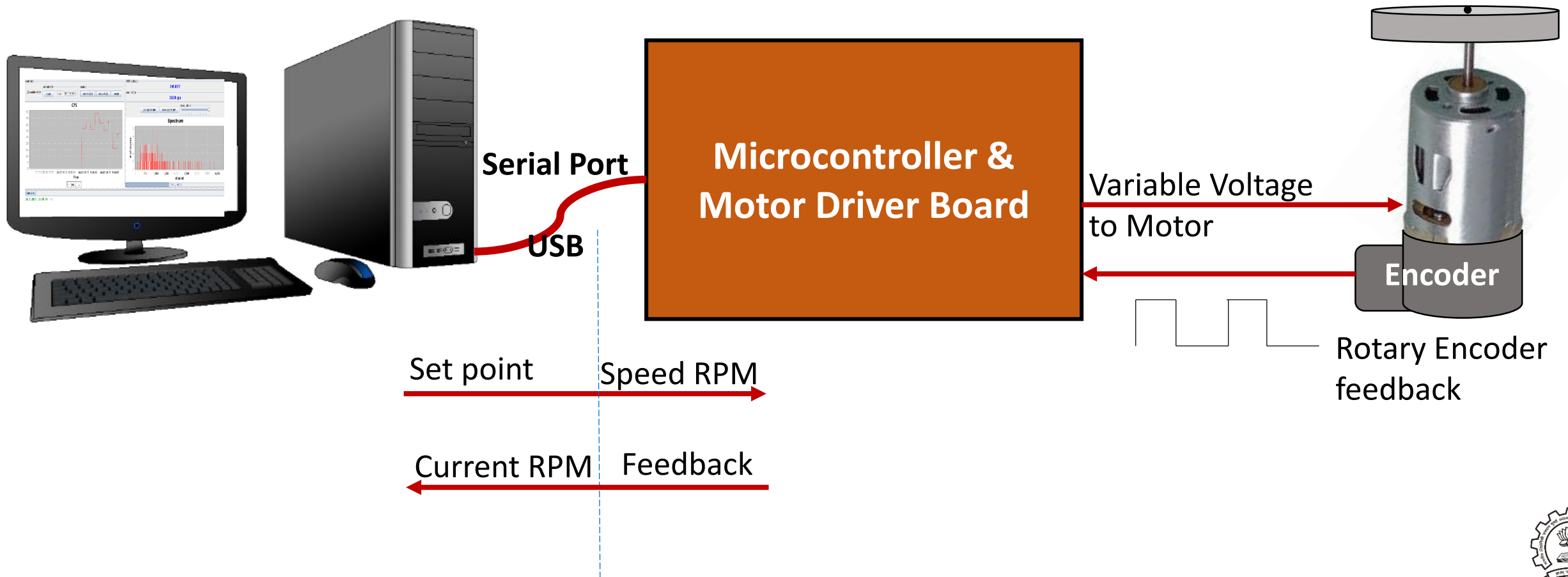
DC motor Control Kit



DC Motor Control



DC motor Control Kit



DC motor Control Kit



Training Modules

Modules for Speed
& Position control

System Identification

Traditional Control using P PI PID

Neural Networks

Deep Learning

MPC

Robust Control



DC motor Control Kit



Used for class room teaching over 4 years in IIT

USB Connectivity – Plug & Play

Creates real world plant experience on your desktop

Interactive frontend Software modules

Complete Lab Development Package for Academics

Open communication command set

Low Cost



Indian Institute of Technology Bombay

**DC Motor control using
MATLAB/SIMULINK**



IIT Bombay

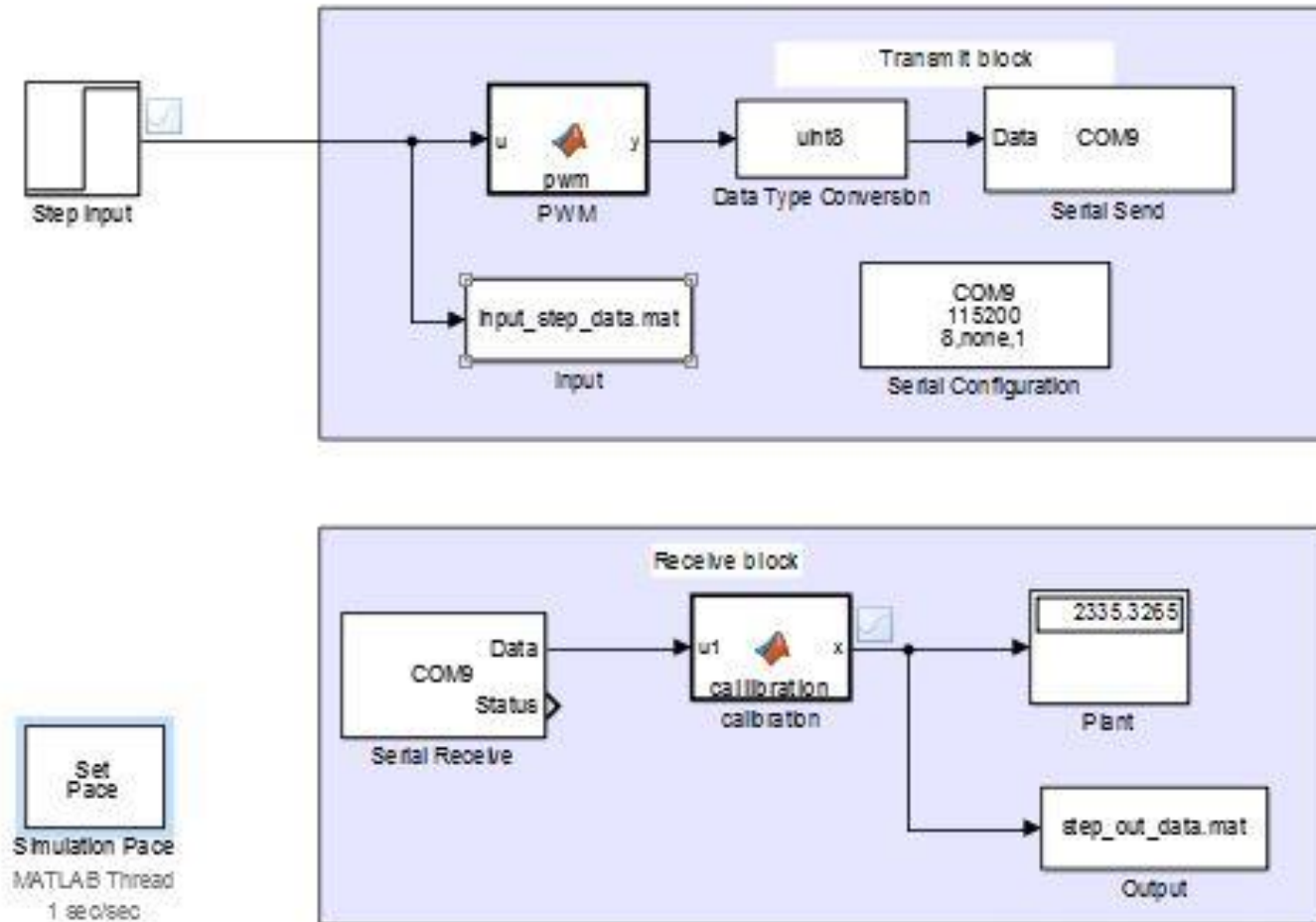
Outline

- Experiment No- 1
 - Validation of motor model for speed control
- Experiment No-2
 - PI Control Gains for Motor speed control



Experiment No-1

Validation of motor model for speed control



Procedure

- Set $t_s=0.015$
- Run the matlab simulink model
- To stop the motor press the reset button on the DC motor kit .



FOPTD

- The First-order Plus Time Delay (FOPTD) model is given by

$$G(s) = \frac{\Delta Y(s)}{\Delta U(s)} = \frac{K e^{-t_d s}}{\tau s + 1}$$

gain K , time constant τ and dead time t_d



Apply two-point method for system

- $t_{63.2}$ = Time required for the output to reach 63.2 % of the steady-state value
- $t_{28.3}$ = Time required for the output to reach 28.3 % of the steady-state value.
- $K = \frac{\text{Difference in two steady states of output}}{\text{Difference in two steady states of input}}$
- $\tau = 1.5(t_{63.2} - t_{28.3})$
- $t_d = t_{63.2} - \tau$



Sample values

- $t_{63.2} = 0.45$ sec
- $t_{28.3} = 0.33$ sec
- $\Delta u(t) = 20$ PWM units
- $\Delta y(t) = 359$ RPM
- Using the two-point method

$$K = 17.95$$

$$\tau = 0.18 \text{ sec}$$

$$L = 0.12 \text{ sec}$$



Transfer function

- $G(s) = \frac{\Delta Y(s)}{\Delta U(s)} = \frac{17.95 e^{-0.12s}}{0.18s+1}$

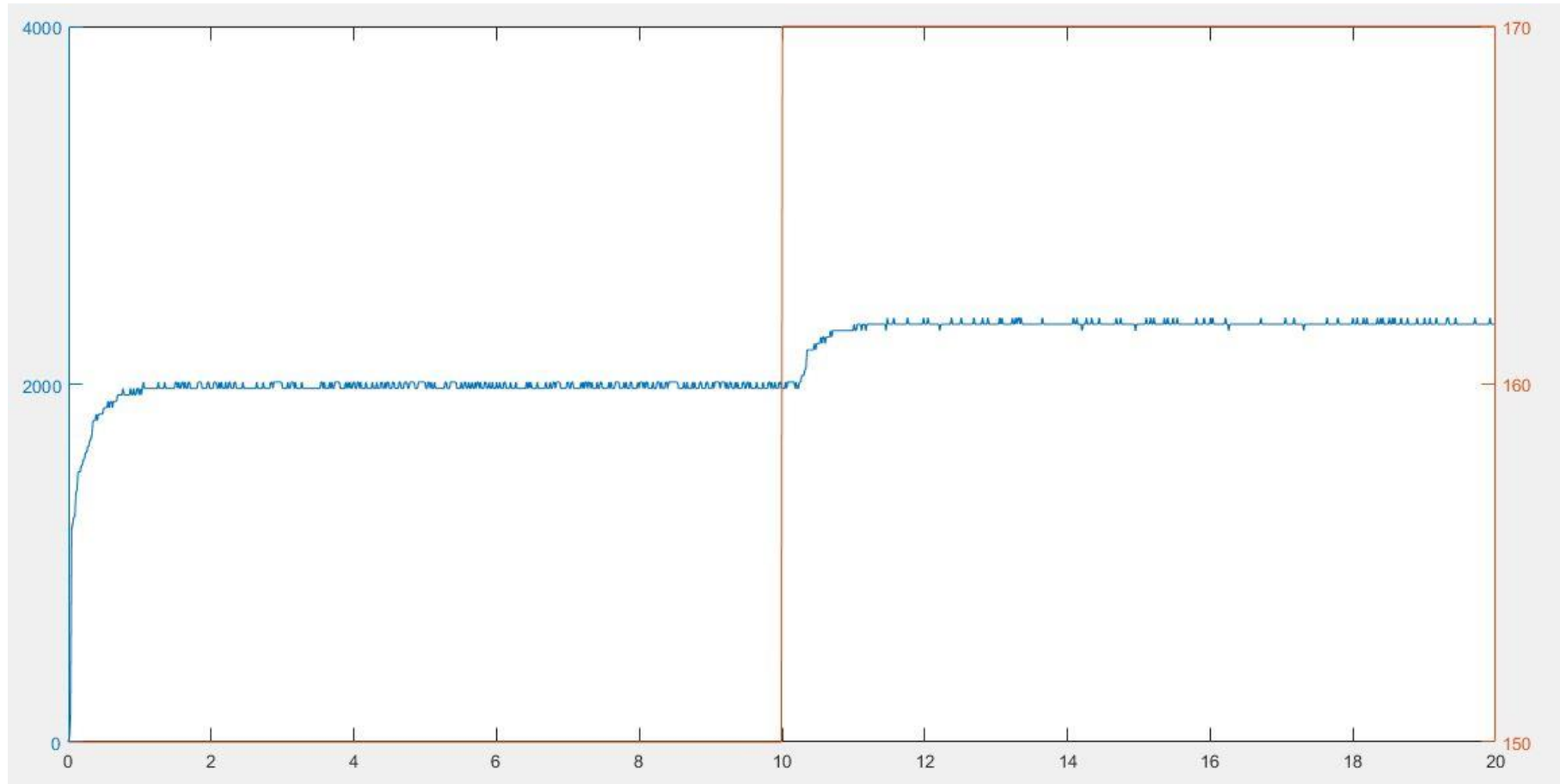


Output response

- In the open loop, the plant is brought to equilibrium by applying a step of 150 PWM units.
- The corresponding speed is around 2000 RPM
- After the motor speed settles, the PWM input is instantaneously changed to 170.
- As a result, the speed increases to around 2400 RPM.

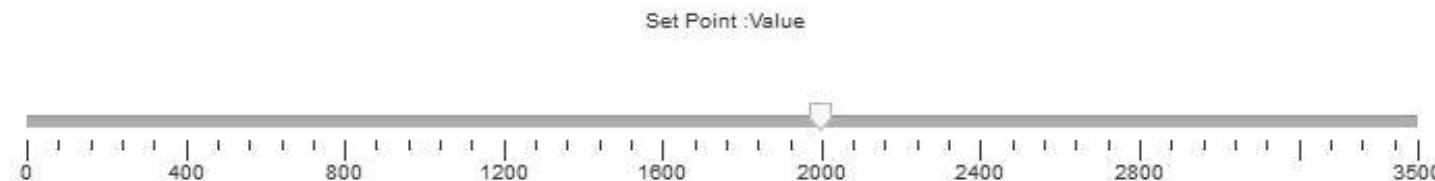
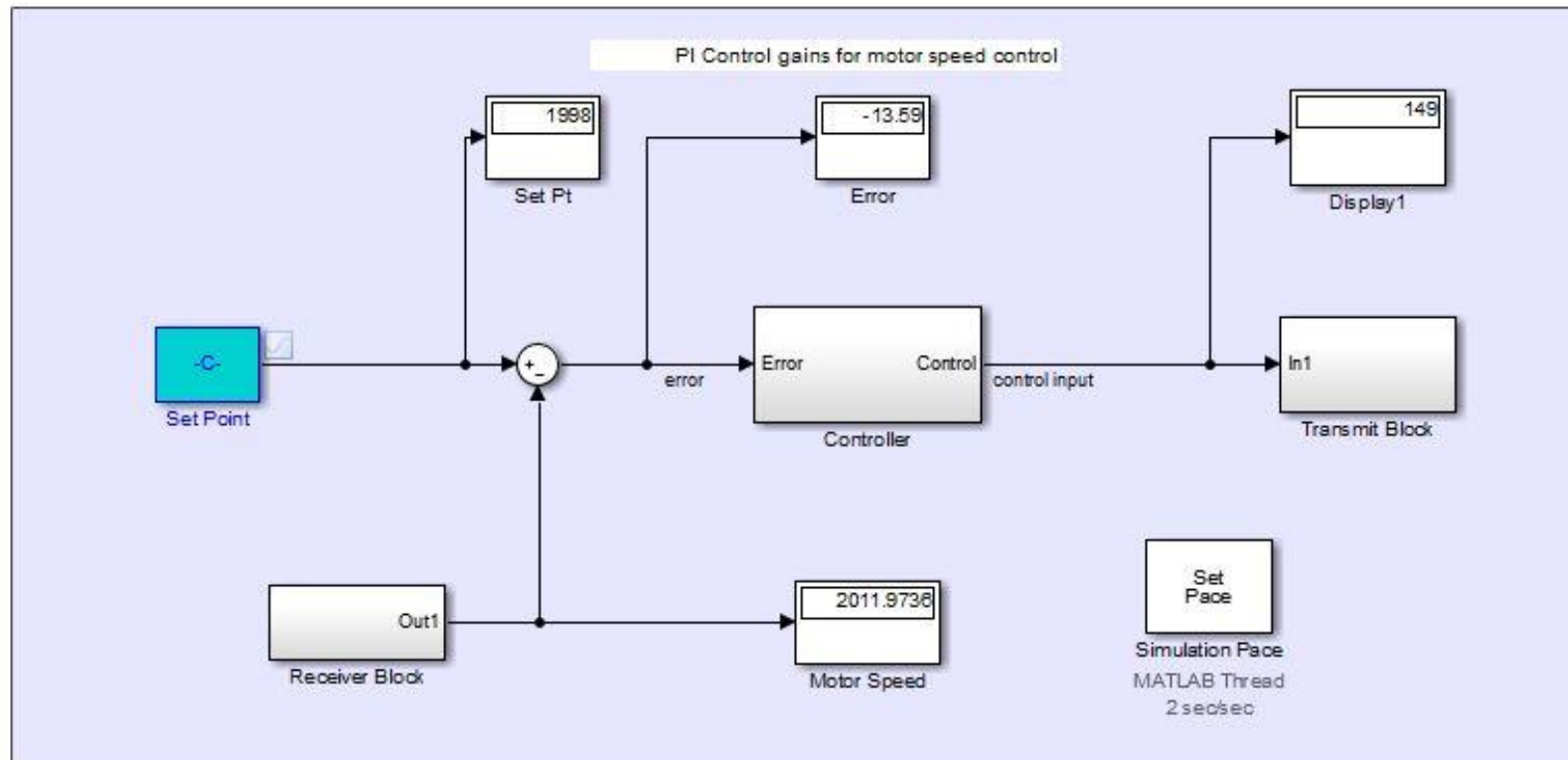


Output vs Input



Experiment No-2

PI Control Gains for Motor speed control



Ziegler-Nichols Rule for Tuning PID Controllers

Type of controller	K_c	T_i	T_d
P	$1/RL$	∞	0
PI	$0.9/RL$	$3L$	0
PID	$1.2/RL$	$2L$	$0.5L$



Calculations

Compute the controller parameters as follows:

$$K_c = \frac{0.9}{RL}$$

$$T_i = 3L$$

$$R = k/\tau$$

Sample values

$$K_c = 0.04102$$

$$T_i = 0.825 \text{ sec}$$

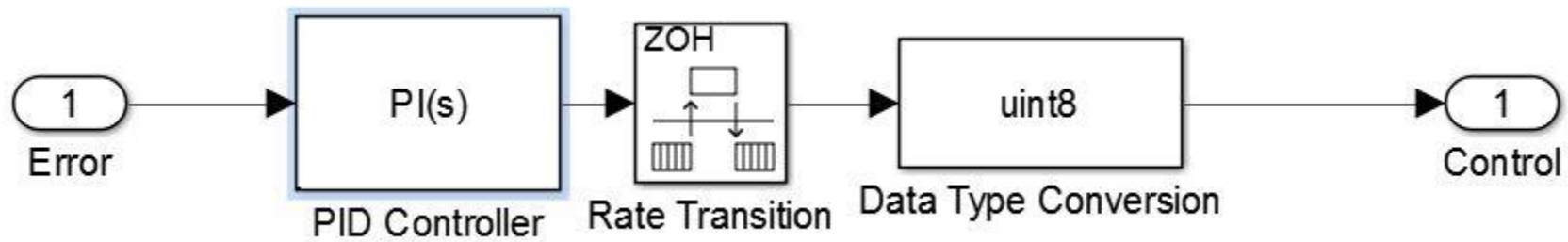


Procedure

- Double click on the controller block .
- Double click on PID Controller block.
- Enter the P and I values calculated using the **Ziegler-Nichols Rule** .



PID Block



PID block configuration

Block Parameters: PID Controller

PID Controller

This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune...' button (requires Simulink Control Design).

Controller: Form:

Time domain:

Continuous-time
 Discrete-time

Main | PID Advanced | Data Types | State Attributes

Controller parameters

Source: [Compensator formula](#)

Proportional (P):

Integral (I):

$$P \left(1 + I \frac{1}{s} \right)$$

Initial conditions

Source:

Integrator:

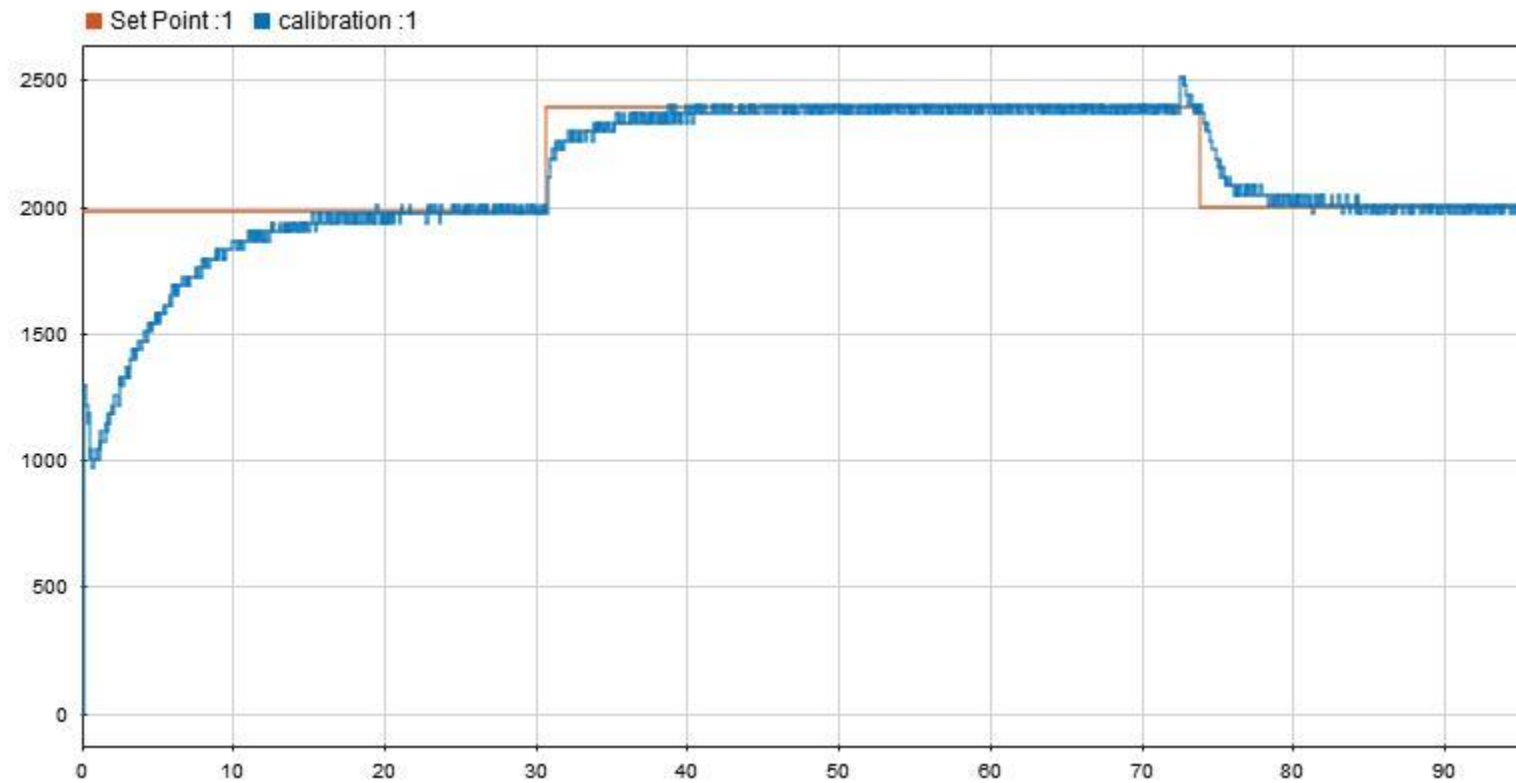


Output

- $K_c = 0.075$ and $T_i = 0.36$ sec.
- After the speed settles at 2000 RPM, a step of 400 RPM is applied.
- It is seen that the output follows the set point and the speed settles at 2400 RPM.
- Next, a negative step of 400 RPM is applied.
- It is clearly observed that motor speed decreases and settles at 2000 RPM.



Output response



Neural Network Model And Neural Network Controller for DC Motor

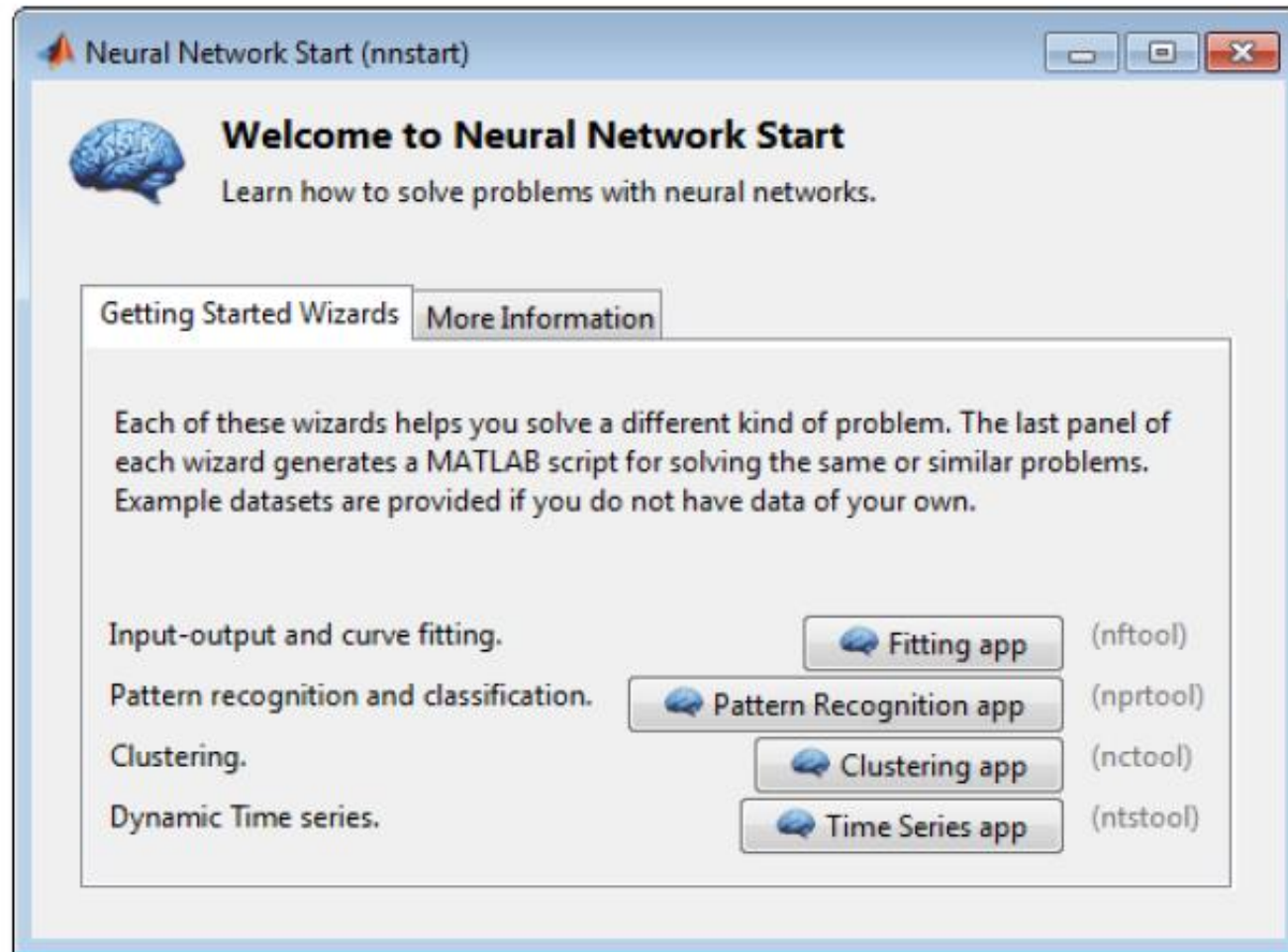
Prof.P.S.V.Nataraj

Systems and Control Engineering

IIT Bombay



1. How to use Neural Network tools



2. Network Fitting GUI

Neural Fitting (nftool)

Welcome to the Neural Fitting app.
Solve an input-output fitting problem with a two-layer feed-forward neural network.

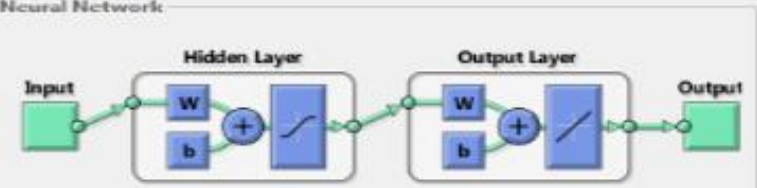
Introduction

In fitting problems, you want a neural network to map between a data set of numeric inputs and a set of numeric targets.

Examples of this type of problem include estimating house prices from such input variables as tax rate, pupil/teacher ratio in local schools and crime rate (`house_dataset`); estimating engine emission levels based on measurements of fuel consumption and speed (`engine_dataset`); or predicting a patient's bodyfat level based on body measurements (`bodyfat_dataset`).

The Neural Fitting app will help you select data, create and train a network, and evaluate its performance using mean square error and regression analysis.

Neural Network



A two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (`fitnet`), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.

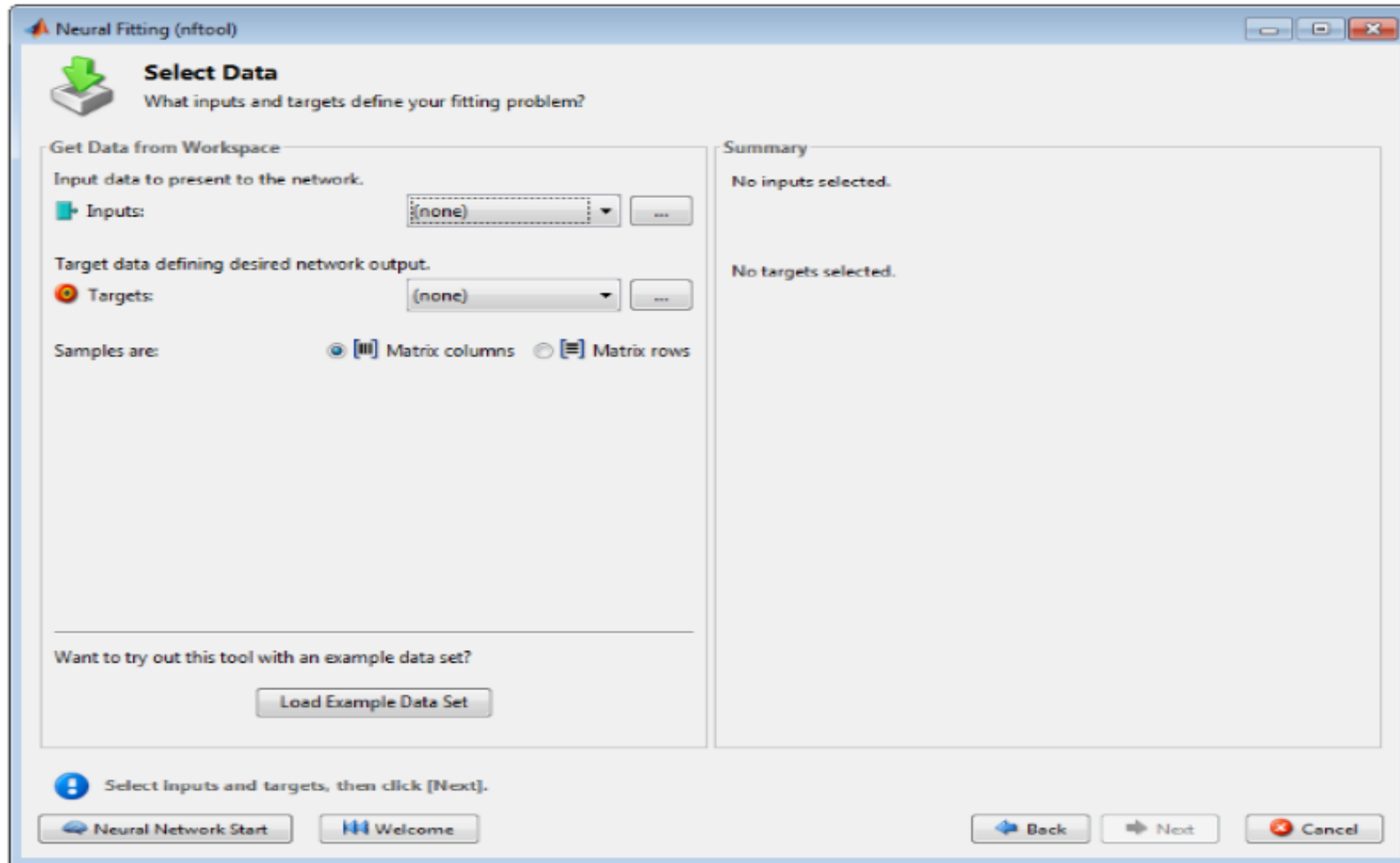
The network will be trained with Levenberg-Marquardt backpropagation algorithm (`trainlm`), unless there is not enough memory, in which case scaled conjugate gradient backpropagation (`trainscg`) will be used.

To continue, click [Next].

Neural Network Start Welcome Back Next Cancel



3.Data set selection



4. Network Architecture

The screenshot shows the 'Neural Fitting (nftool)' window. The title bar reads 'Neural Fitting (nftool)'. The main window is titled 'Network Architecture' and contains the following elements:

- Hidden Layer:** A section with the text 'Define a fitting neural network. (Fitnet)'. Below it, 'Number of Hidden Neurons:' is followed by a text box containing the number '10'. A 'Restore Defaults' button is located at the bottom of this section.
- Recommendation:** A text box containing the text: 'Return to this panel and change the number of neurons if the network does not perform well after training.'
- Neural Network:** A diagram showing the flow from an 'Input' (8 neurons) to a 'Hidden Layer' (10 neurons) to an 'Output Layer' (1 neuron) to an 'Output' (1 neuron). Each layer is represented by a box containing 'w' (weights) and 'b' (biases), followed by an addition sign (+) and a sigmoid-shaped activation function box.
- Navigation:** At the bottom, there are buttons for 'Neural Network Start', 'Welcome', 'Back', 'Next', and 'Cancel'. A blue arrow icon points to the 'Next' button. Text below the arrow says: 'Change settings if desired, then click [Next] to continue.'

8. Click Next.



5. Train Network

Neural Fitting (nftool)

Train Network

Train the network to fit the inputs and targets.

Train Network

Choose a training algorithm:

Levenberg-Marquardt

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

Train using Levenberg-Marquardt. (trainlm)

Train

Results

	Samples	MSE	R
Training:	348	-	-
Validation:	75	-	-
Testing:	75	-	-

Plot Fit **Plot Error Histogram**

Plot Regression

Notes

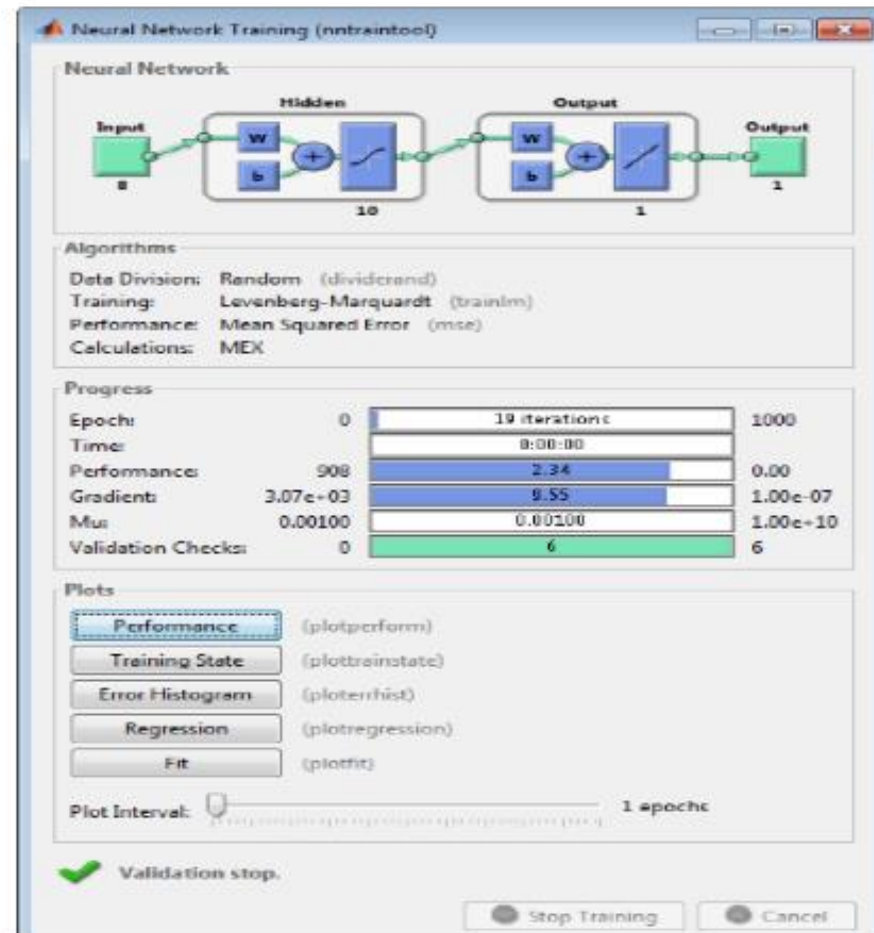
- Training multiple times will generate different results due to different initial conditions and sampling.
- Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.
- Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.

Train network, then click [Next].

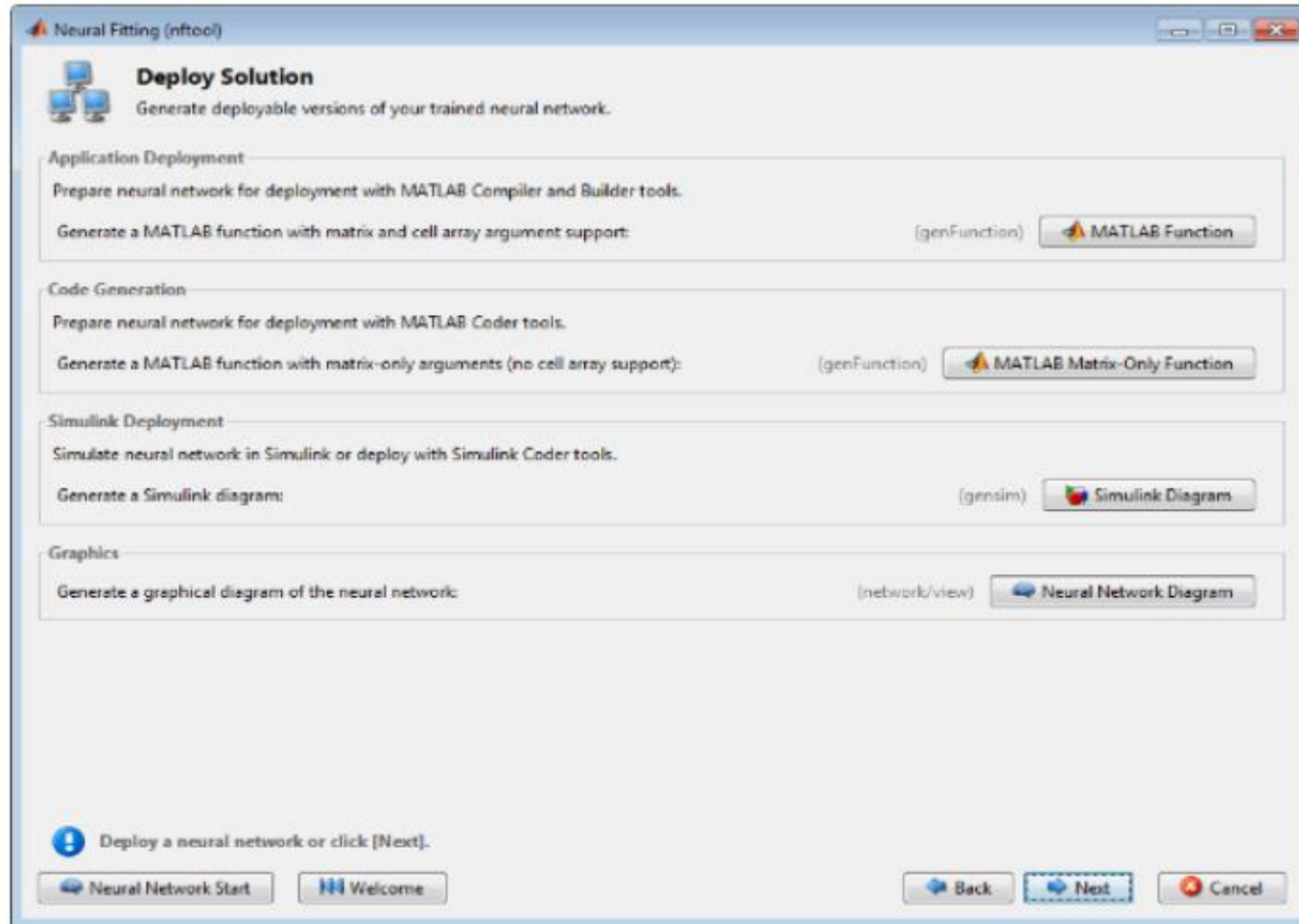
Neural Network Start Welcome Back Next Cancel



6. Training Results

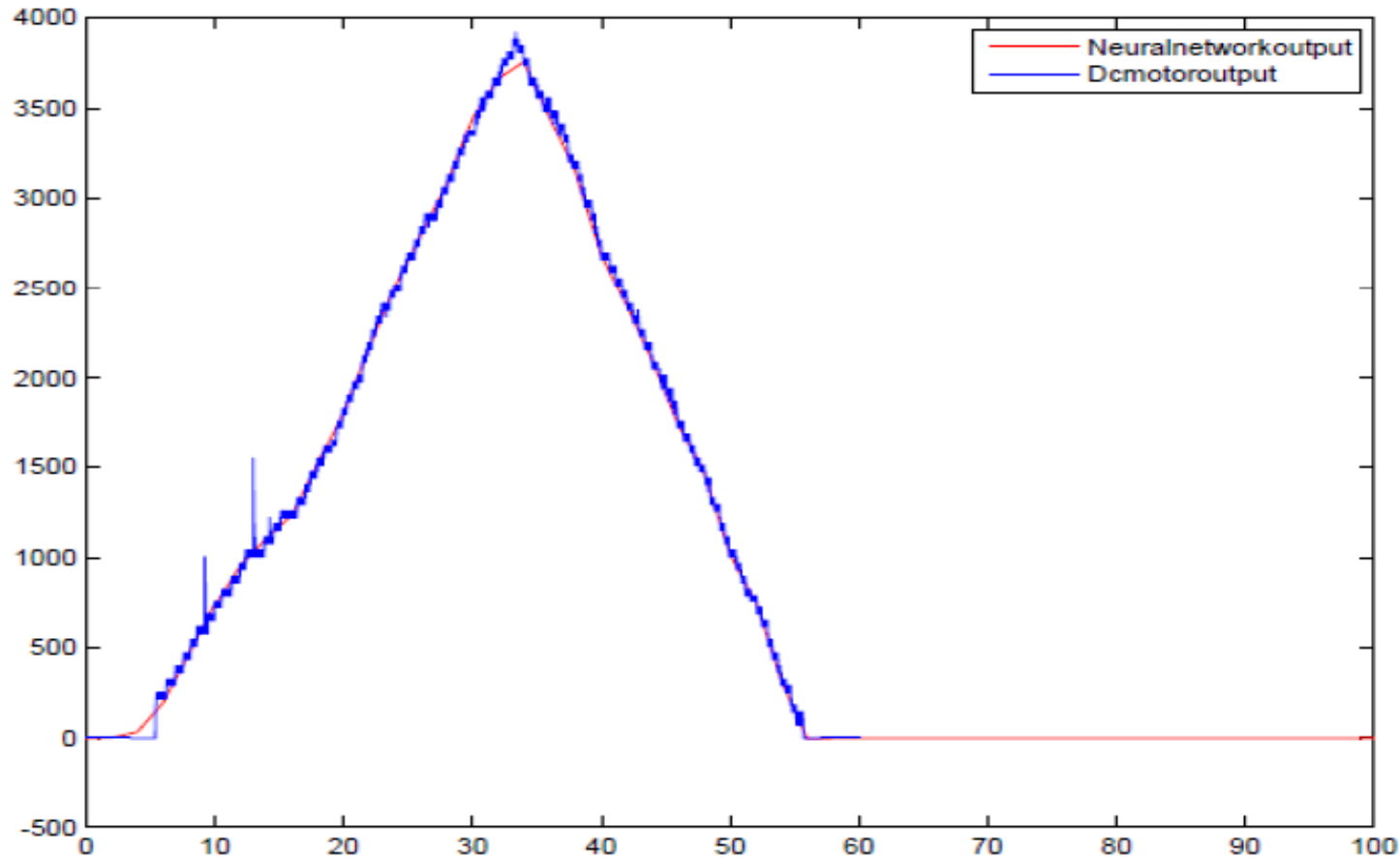


7. Deploy Solution

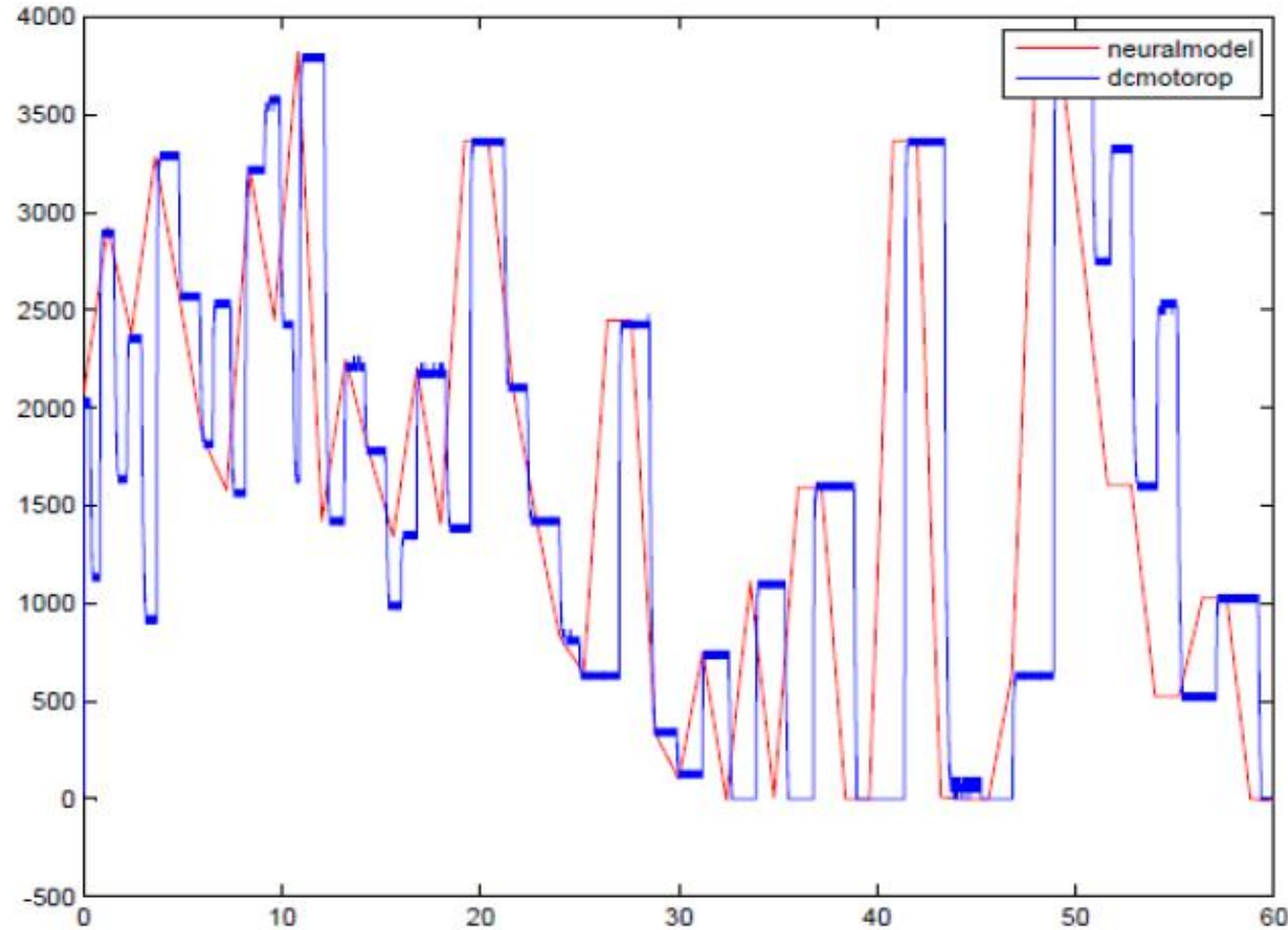


Comparison between DC Motor Model and Neural Network Model

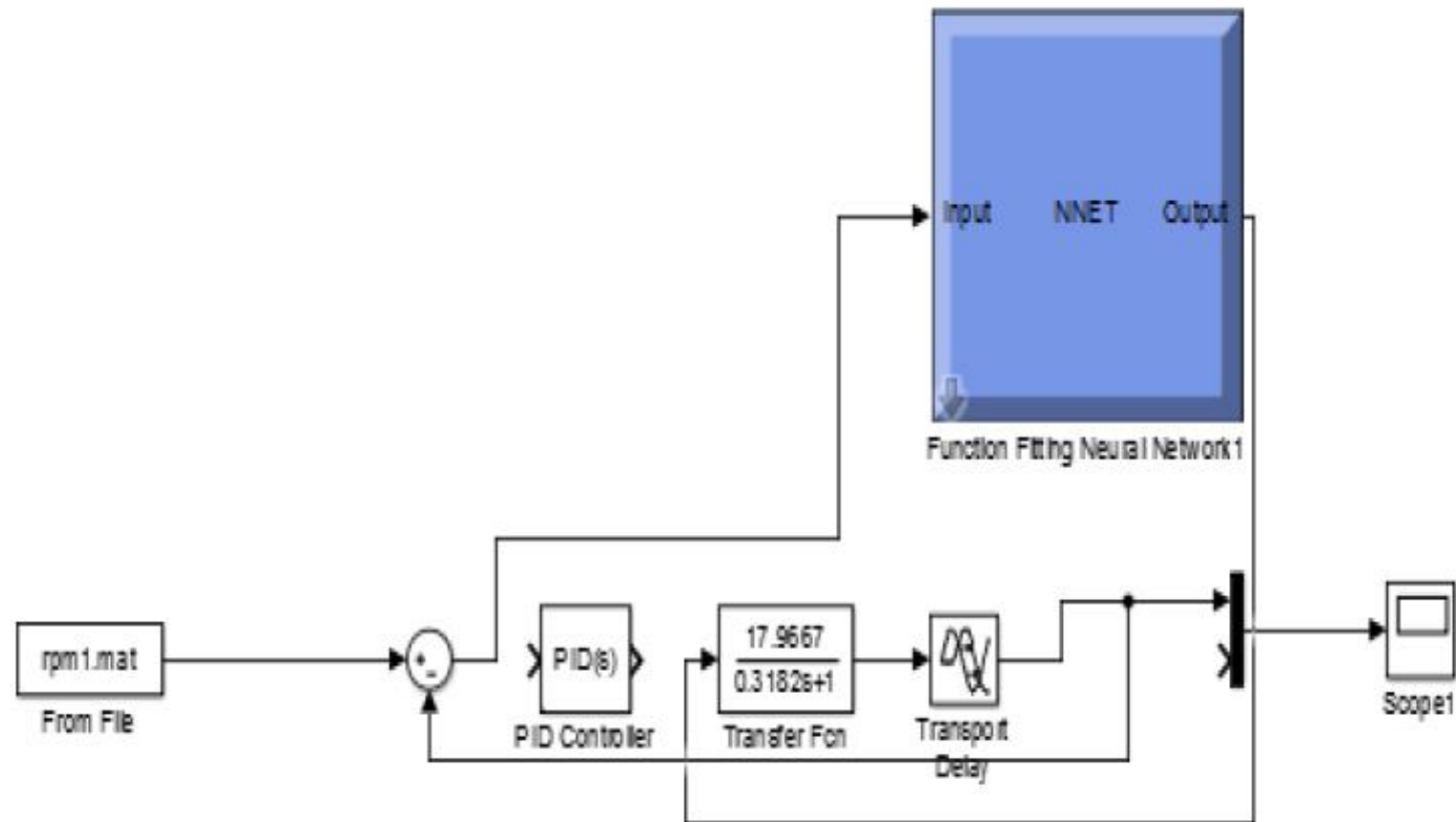
- Steady state output



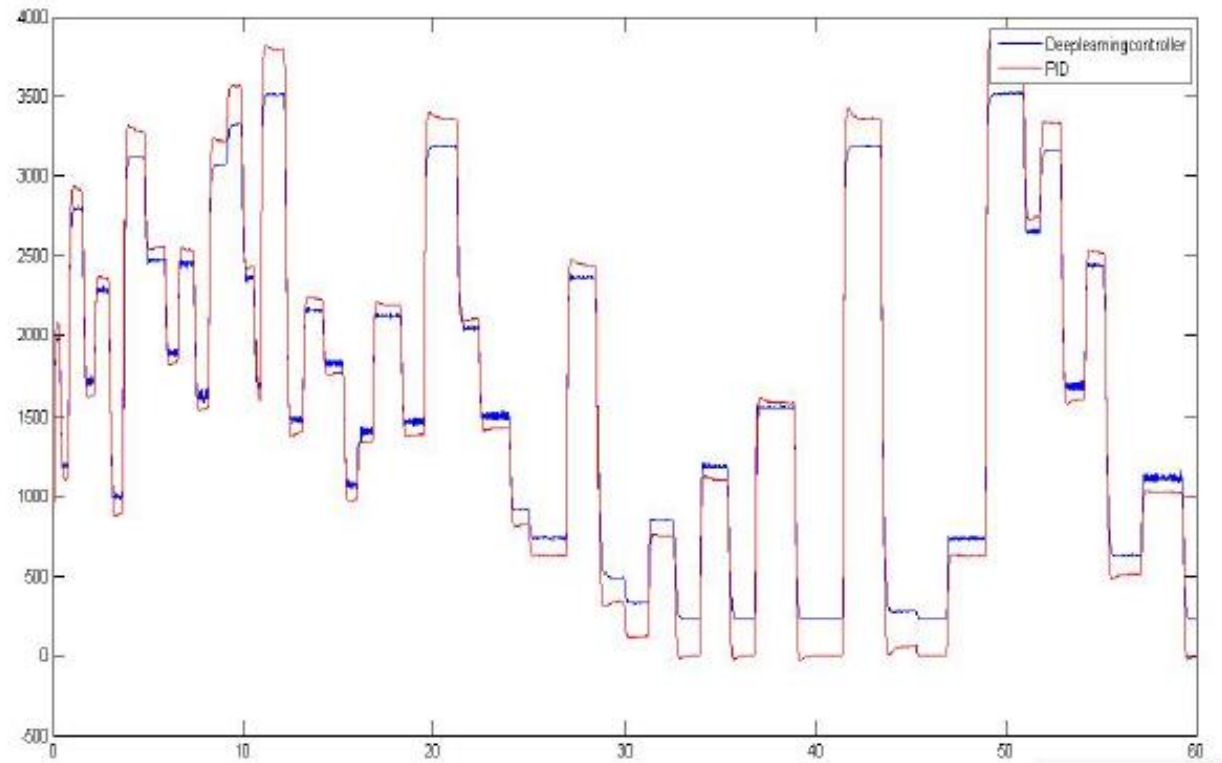
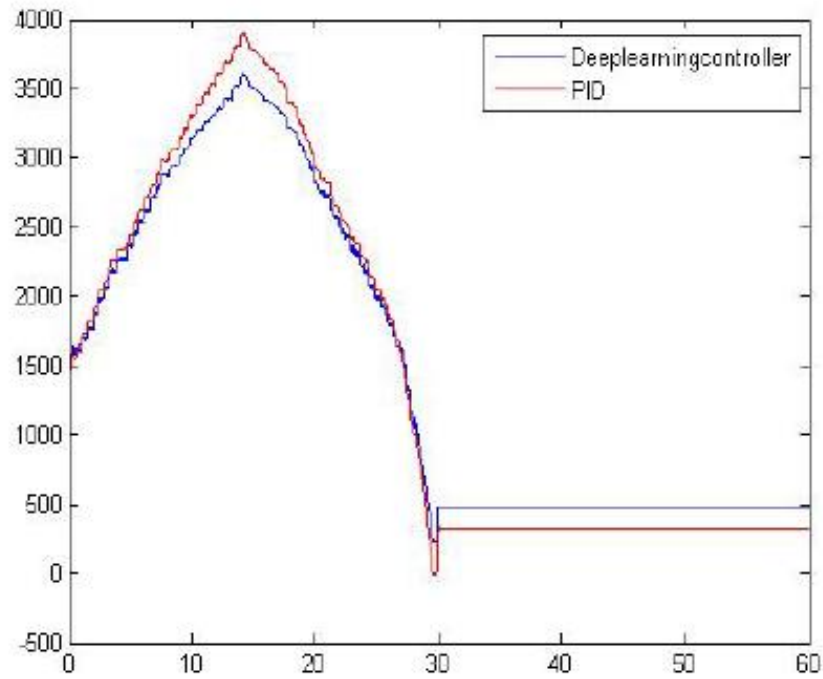
Random Reference Signal



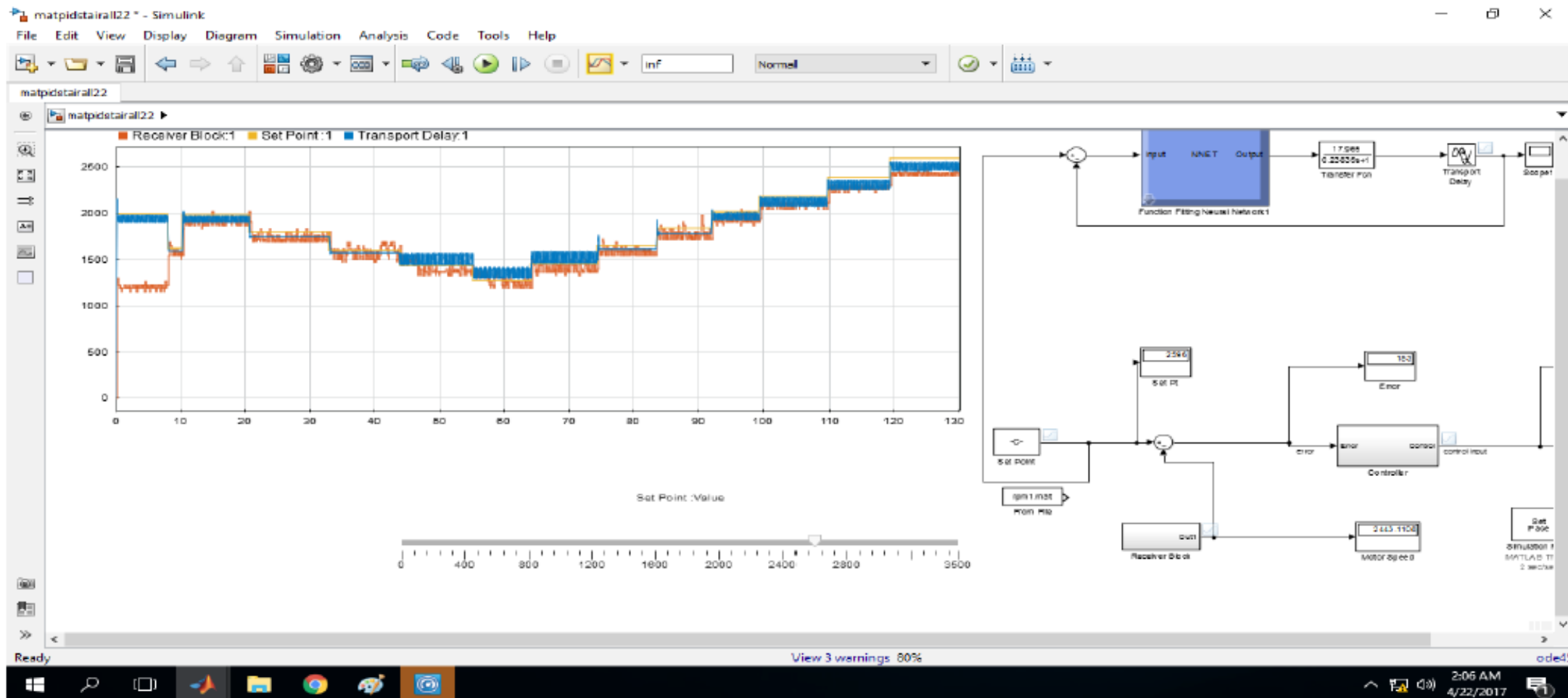
Neural Network Controller Replacing PID



Steady state and random reference signal tracking



Neural Network Controller with DC Motor



Contact

Prof. P. S. V. Nataraj
Systems & Control Engineering IIT Bombay
nataraj@sc.iitb.ac.in

Pramod Mhaske
Sr. Project Manager
pramod.mhk@gmail.com
Tel 022-25764884 , 9320201648

