Embedded Control and Mechatronics (ECE-456)

Farzad Pourboghrat

Arduino Platform for PMDC Motor Modeling & Control using MATLAB/Simulink, Arduino Target Blockset, & Other Mathworks Tools

User Guide

Prepared by: Arjun Shekar Sadahalli

Embedded Control Systems Lab

Dept. Of Electrical & Computer Engineering

SIU Southern Illinois University
CARBONDALE

Contents

Agenda

- Data Acquisition:
  Input and output (I/O) data of the DC Motor is captured on the host PC using the Arduino Simulink blockset.

- System Identification / Modeling:
  System Identification Toolbox (SID) used to obtain the transfer function (Time domain) of PMDC motor platform using the captured I/O data as the input to the SID toolbox.

- Parameter Estimation:
  Simscape DC Motor parameters are tuned to match the DC motor on the platform, using the captured I/O data.

- Controller Design:
  Tune the parameters of the PID controller to control the speed of the Simscape DC motor model in the closed-loop structure.

- Rapid Prototyping:
  Build/load the tuned PID controller into the Arduino platform, capture the I/O data, and verify the speed response of the PMDC motor.

Objective

The objective of this experiment is to use MATLAB/Simulink tools for modeling and control design of embedded mechatronic systems. We consider a Permanent Magnet DC Motor (PMDC) interfaced with Arduino on a custom made circuit board. The goal is to identify the transfer function of the PMDC Motor, estimate its parameters and design/implement control to regulate the speed of the PMDC motor.

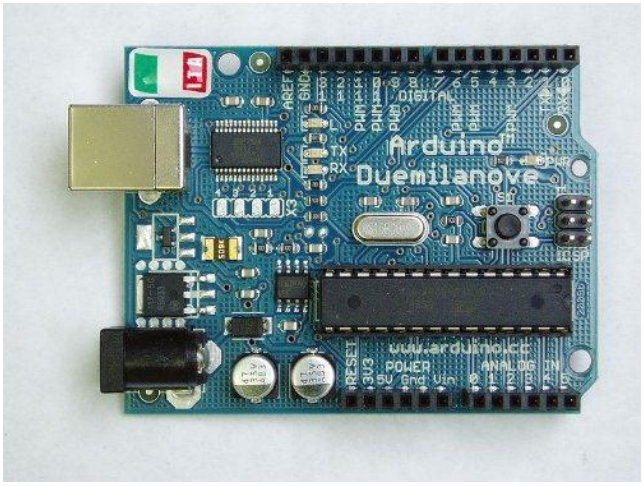
Hardware and Software Requirements

- Hardware
    1. Permanent Magnet DC Motor with an integrated tacho-generator.

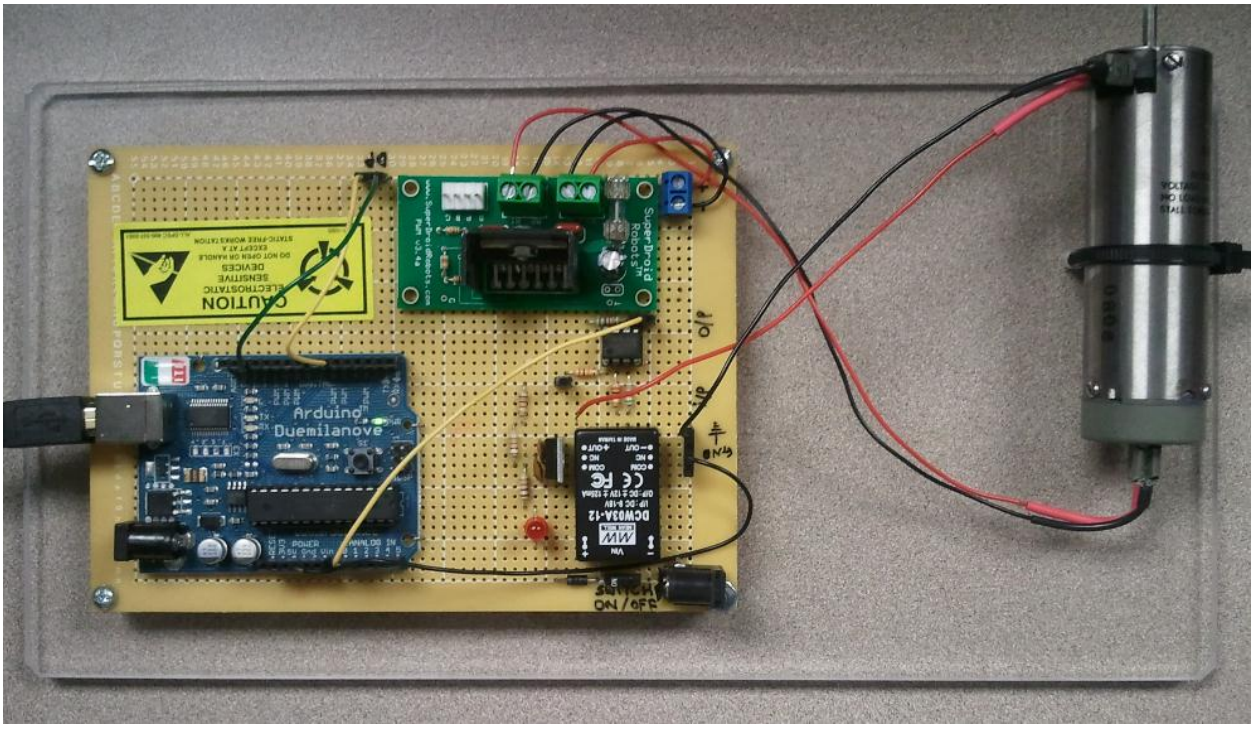
Specifications:

- Speed: +/- 4000 rpm @ 12V
- Inductance: 7mH
- Resistance: 10 Ω
- Torque: 3.45 Oz-in/amp
- Tacho: 1.9V / 1000 rpm

2. Arduino Duemilanove microcontroller development board.



3. Custom Circuit Board



Specifications:

- Arduino Duemilanove
- Super Droid H-Bridge Motor Driver (LMD 18200T)
- Op-Amp Level Shifter
- DC-DC Converter module (DCW03A-12) (+12 – 0 – -12)
- PMDC Motor

- Software

1. MATLAB (R2010a or later)
2. Simulink
3. *Arduino Target Toolbox*
4. System Identification Toolbox
5. Simulink design optimization Toolbox
6. Simulink Control Design Toolbox
7. Real-Time Workshop
8. Real-Time Workshop Embedded Coder
9. Instrument Control Toolbox
10. State Flow and State Flow code

Things to Check / do before Start

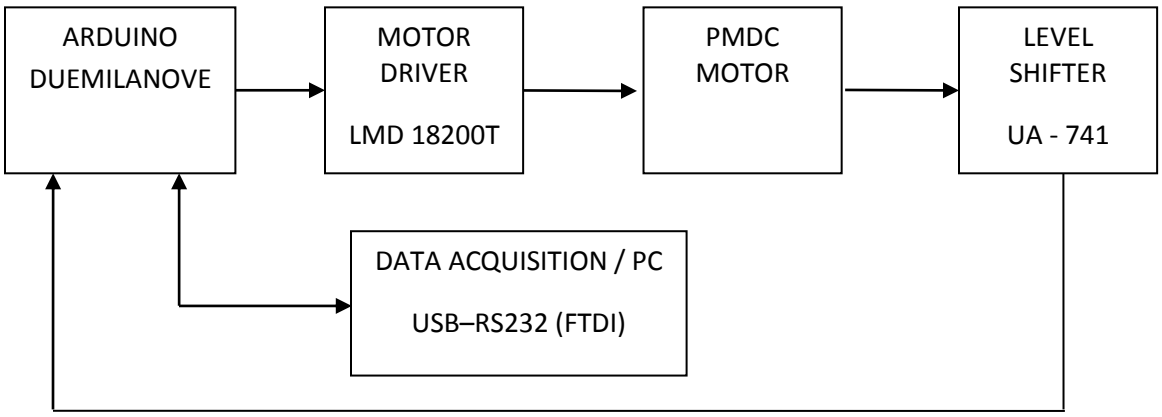
- Check to see if Arduino Target toolbox is installed in Simulink:
  -Open Simulink, and to the left you should find 'Arduino Target'.
- Check if you have a DC Adapter, USB Cable, Arduino work platform on the workbench.
- Open the folder on desktop\ECE456\Arduino and find 6 Simulink '.mdl' files and a '.mat' matlab file. Copy the contents.
- Create **your own folder** inside 'ECE456_Students' on the desktop and paste the copied contents. These are the files you would be working with.

Note: If any of the above is missing, contact the T.A. or lab personnel.
Please do not alter the files or change contents in the original directory.

Caution

- No food or drink allowed near the setup.
- Please exercise caution when working with the setup to avoid any static discharges.
- Please read the manual carefully before proceeding with the experiment.
- If you have any questions, please do not hesitate to ask. Authorized lab personnel would be present at all times during your experimentation.

Block Diagram

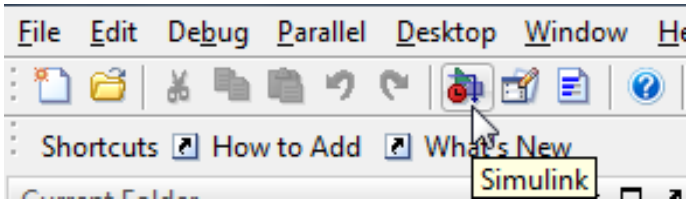


Procedure for SYSTEM IDENTIFICATION

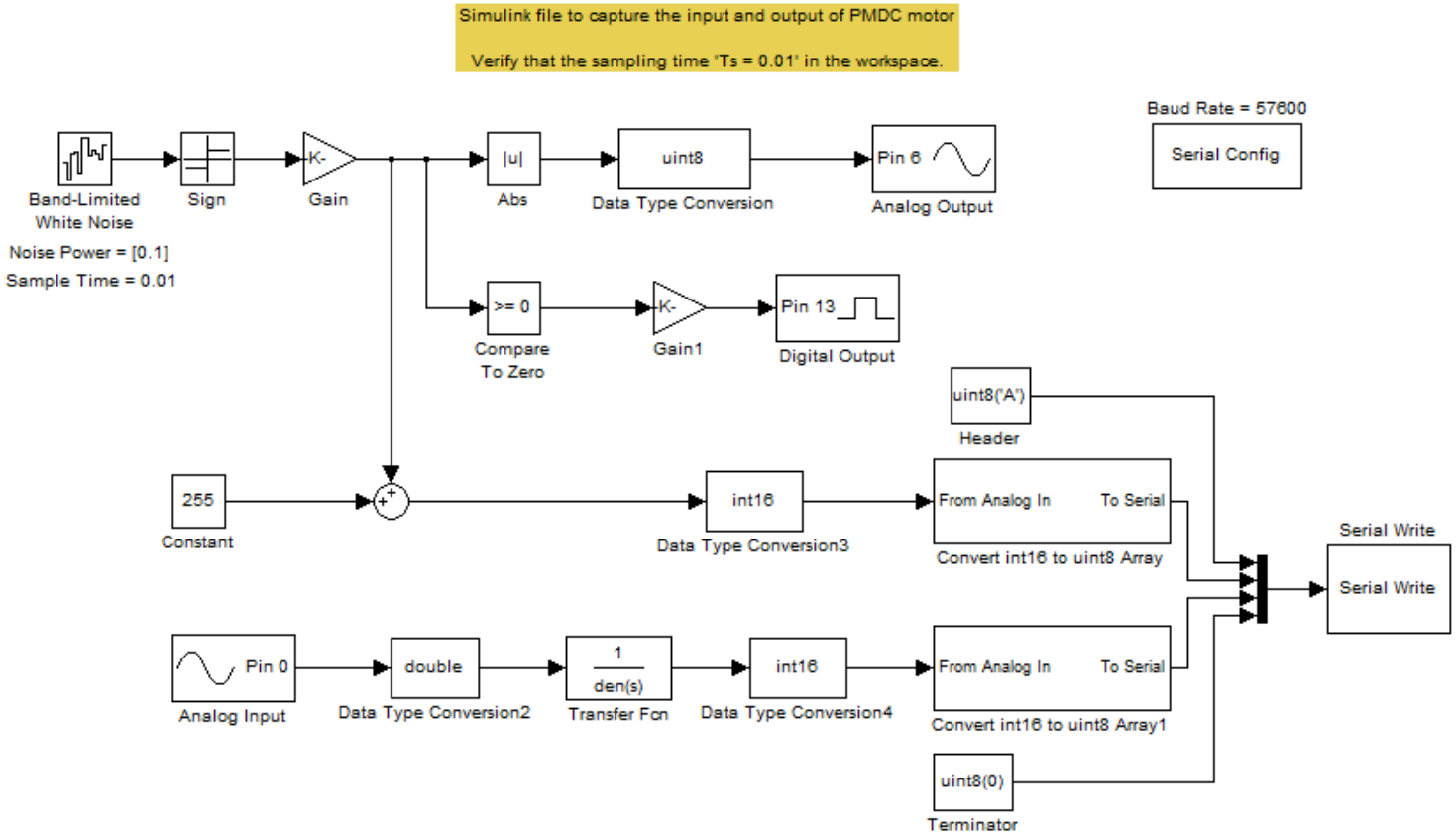
1) Connect the 'Arduino' board to the USB port of the PC. The FTDI drivers should automatically configure and assign a COM port. (*Admin rights required on the computer to complete the driver installation when the board is connected the first time*).

2) Start MATLAB R2010a and make your folder the current working directory in MATLAB.

3) Open 'Simulink' either by typing 'simulink' in the command window or by clicking on the Simulink icon present on the toolbar.

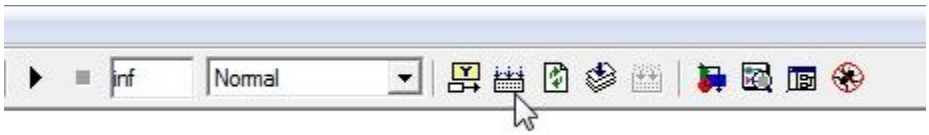


4) Open the file 'Mot_Ident.mdl'.



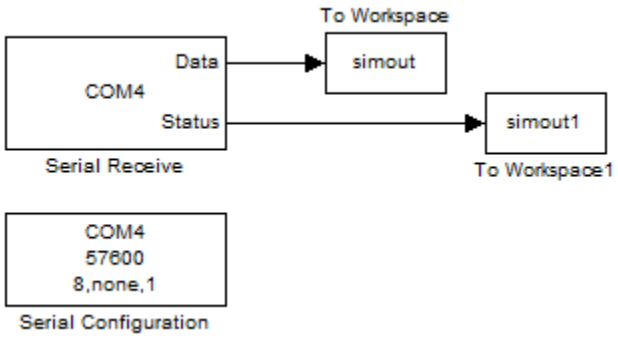
5) Check and note down the COM port number from the command window.

6) Build the 'Mot_Ident.mdl' file into the target. Use either 'Ctrl + B' or click on the icon 'Incremental Build' present on the toolbar as shown below.



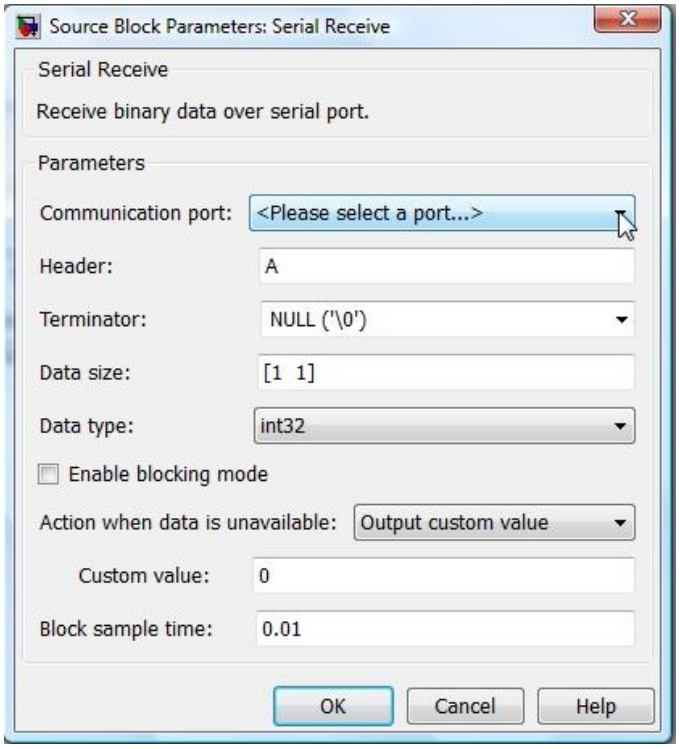
This successfully programs the target with the simulink file 'Mot_Ident.mdl'. In the process, observe the 'Tx' and 'Rx' LED's flashing on the Arduino board and also a 'successful build' message confirms the build procedure to the target. If there is an error during the build procedure, note the error message and call the T.A.

7) Open the file 'Mot_Ident_host.mdl' and check the COM port number.



8) Change the COM port number of the serial receive block if the system has assigned a different COM port at the beginning. This can be done by double clicking on the 'Serial Receive' block and changing the COM port from the drop down as shown in the figure (next page).

9) The system then queries the user for the addition of a 'serial configuration' block with the current COM port number. Click 'Yes'. Double click on the newly added 'Serial Configuration' block and change the 'Baud Rate' from 9600 to 57600. Delete the older 'serial configuration' block.



Attention: If your COM port number does not show up in the drop down, then STOP your procedure and restart MATLAB and Simulink and follow steps from start.

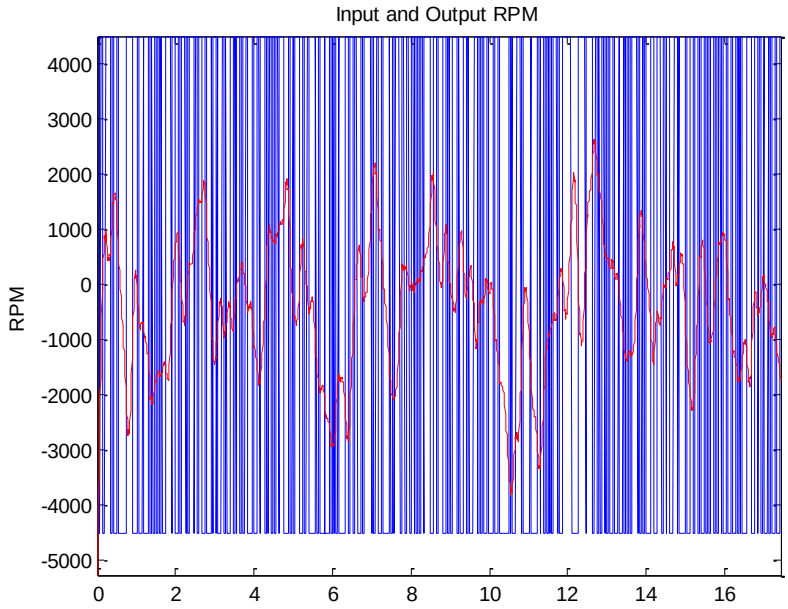
Also, check the Block sample time, Data type, Header, Terminator and Data size. They should exactly match the above parameters.

10) Power the board by connecting the DC adapter and switch 'ON' the DC Motor platform. You should see the DC motor running.

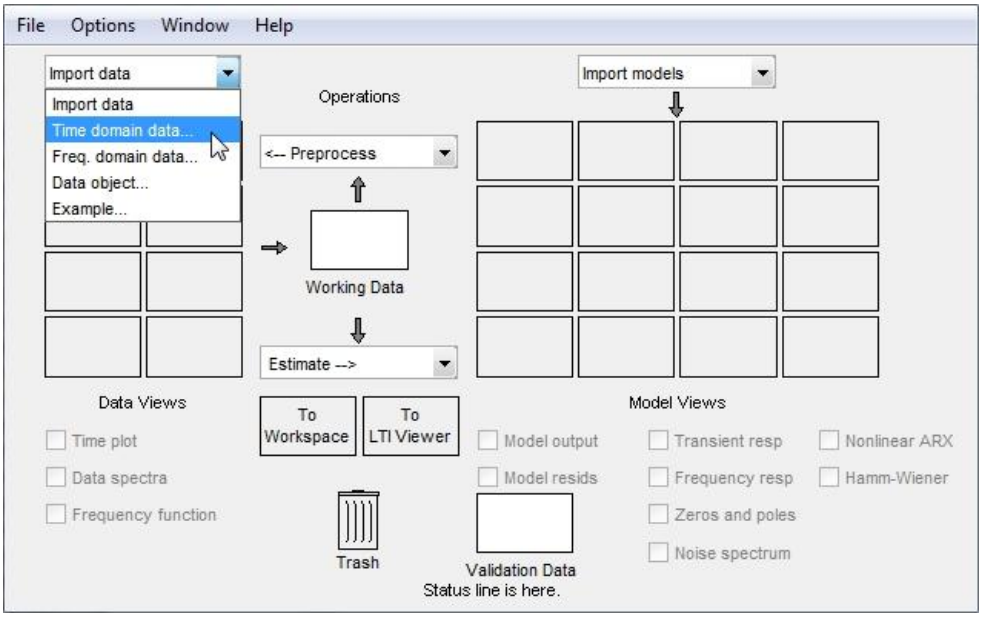
11) Start the data capture by pressing the 'Start Simulation' in 'Mot_Ident_host.mdl' as shown in the figure below.

12) Run the simulation for about 10 seconds and stop. This collects input and the output data from the DC Motor and stores them in the workspace in the variable name 'simout' and 'simout1'. Save all 5 variables present in the workspace into a single '.mat' file. (*example: Original_data.mat')*

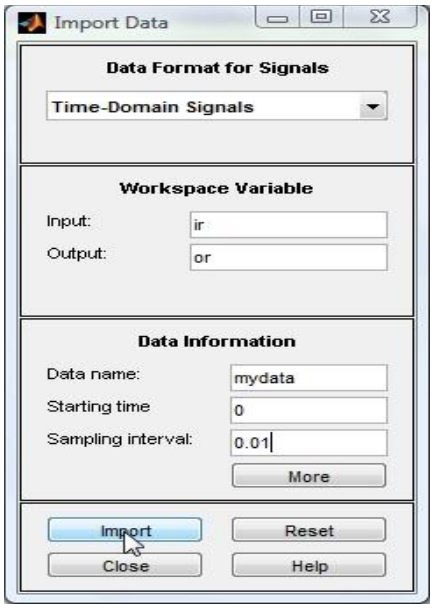
13) To extract the data, open the program ‘Extract_data.m’ and execute the ‘Without Controller’ cell by using ‘Ctrl+Enter’ and then observe the plot.



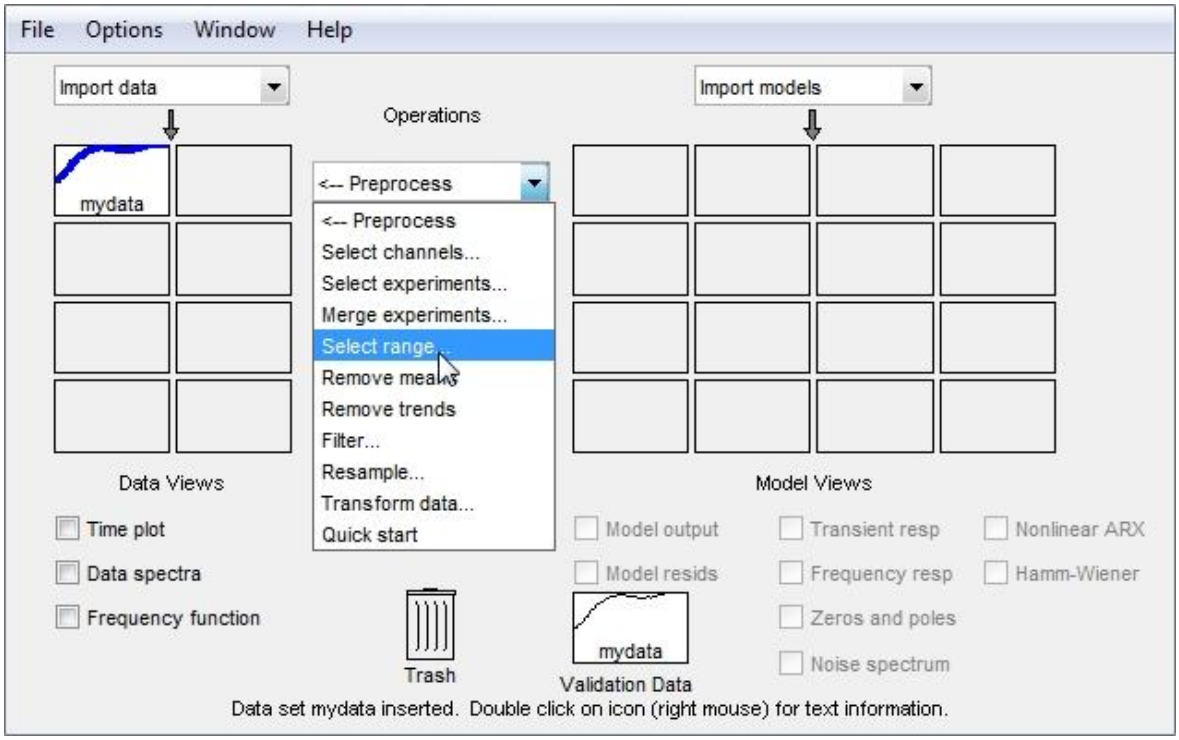
14) System Identification toolbox is used to find the time domain model of the DC motor. This toolbox needs the knowledge of input, output, and sample time. Toolbox is opened by typing ‘ident’ in the command window. Choose the ‘Time domain data’ menu from the drop down as shown. Enter the input ‘ir’ and output data ‘or’ variables in the Import data window.

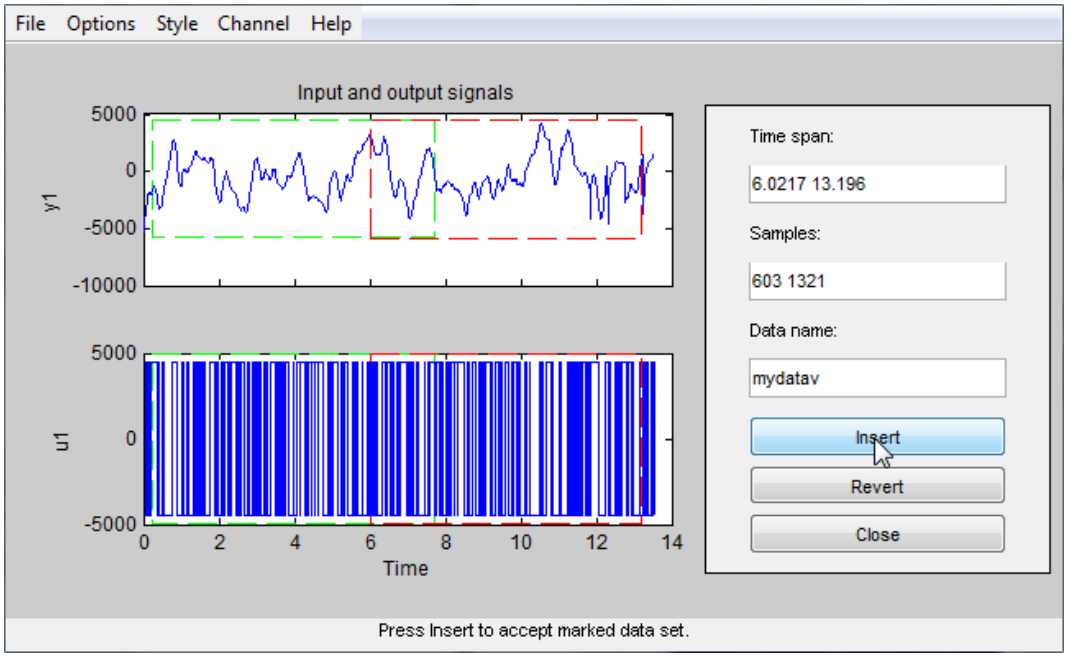


15) Variables of interest ‘ir’ and ‘or’ correspond to the Input and Output RPM’s of the DC Motor. They should be in the workspace. If not, please run the ‘Extract_Data.m’ program and then check these variables.



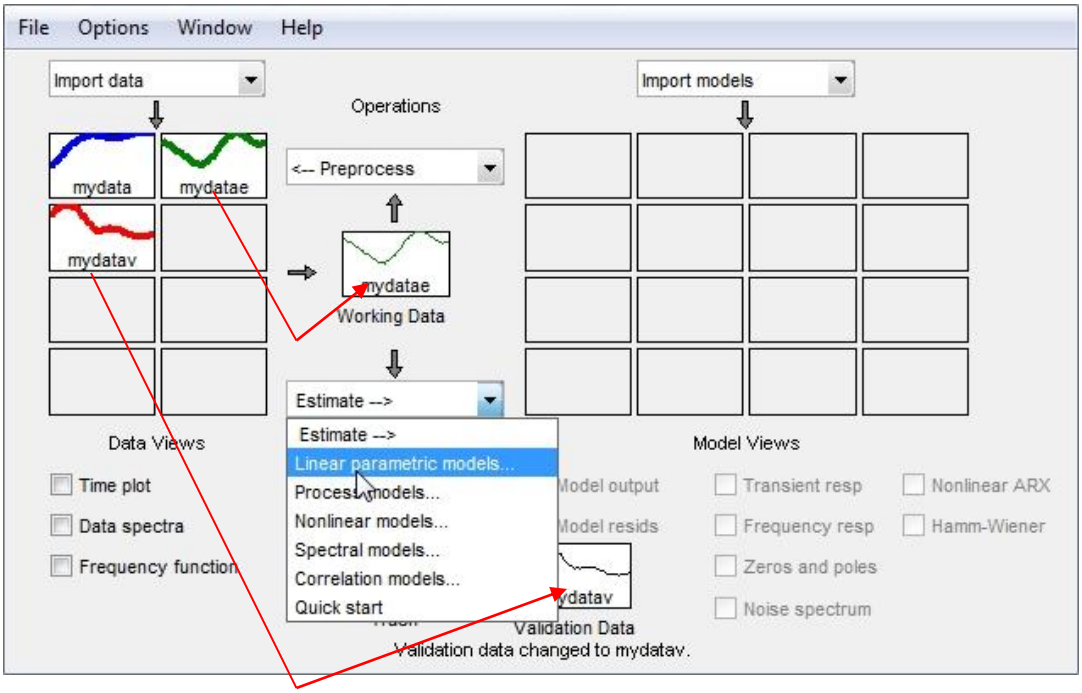
16) From the ‘Operations’ menu, select the range for ‘Working data’ and ‘Evaluation data’, as shown.



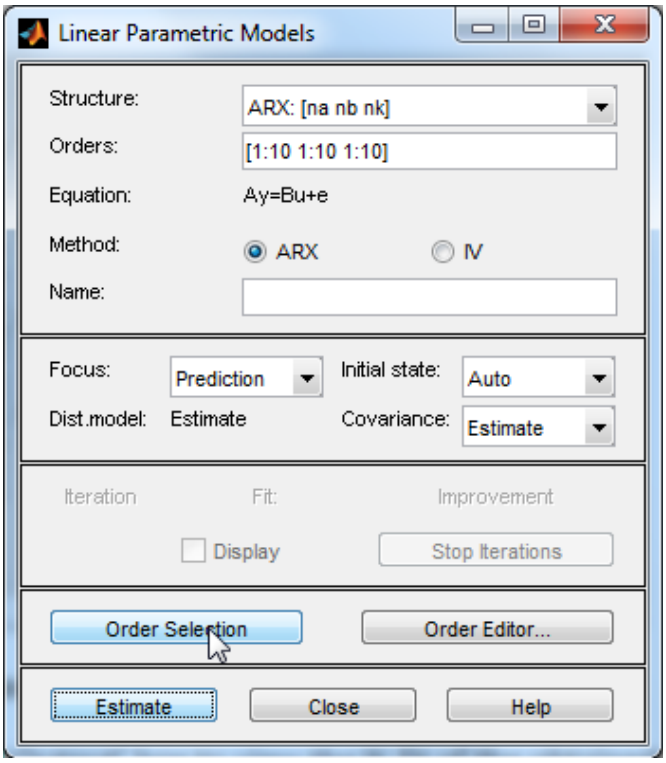


Once selected, Estimation data and Validation data will appear in the 'Data Views' pane.

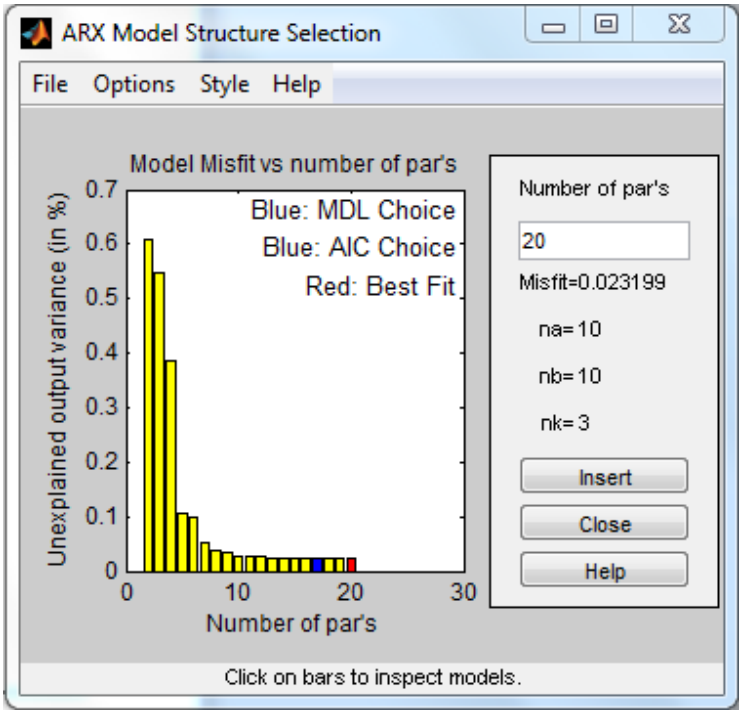
17) If your dataname is 'MyData' then estimation data will be 'MyDatae' and validation data will be 'MyDatav'. Drag the 'Estimation data' into the working data space and 'Validation data' into the validation data space as shown. 'Linear Parametric models' is chosen to estimate the system.



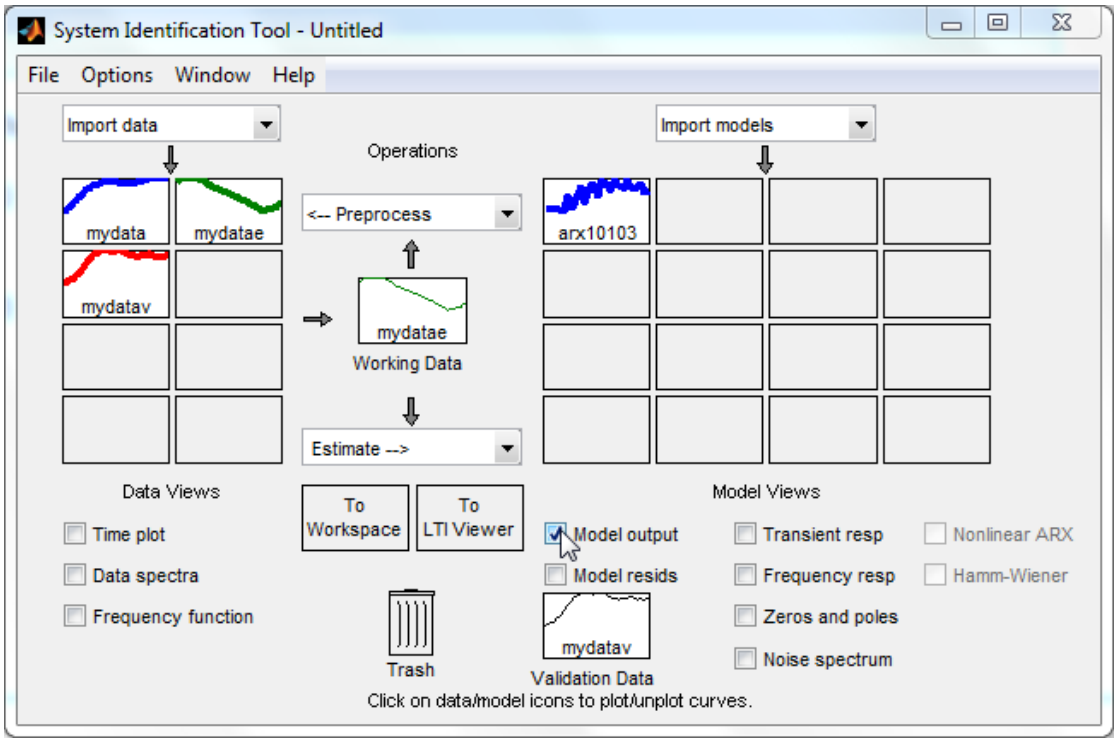
18) Choose ‘Order selection’ as criteria and click ‘Estimate’.



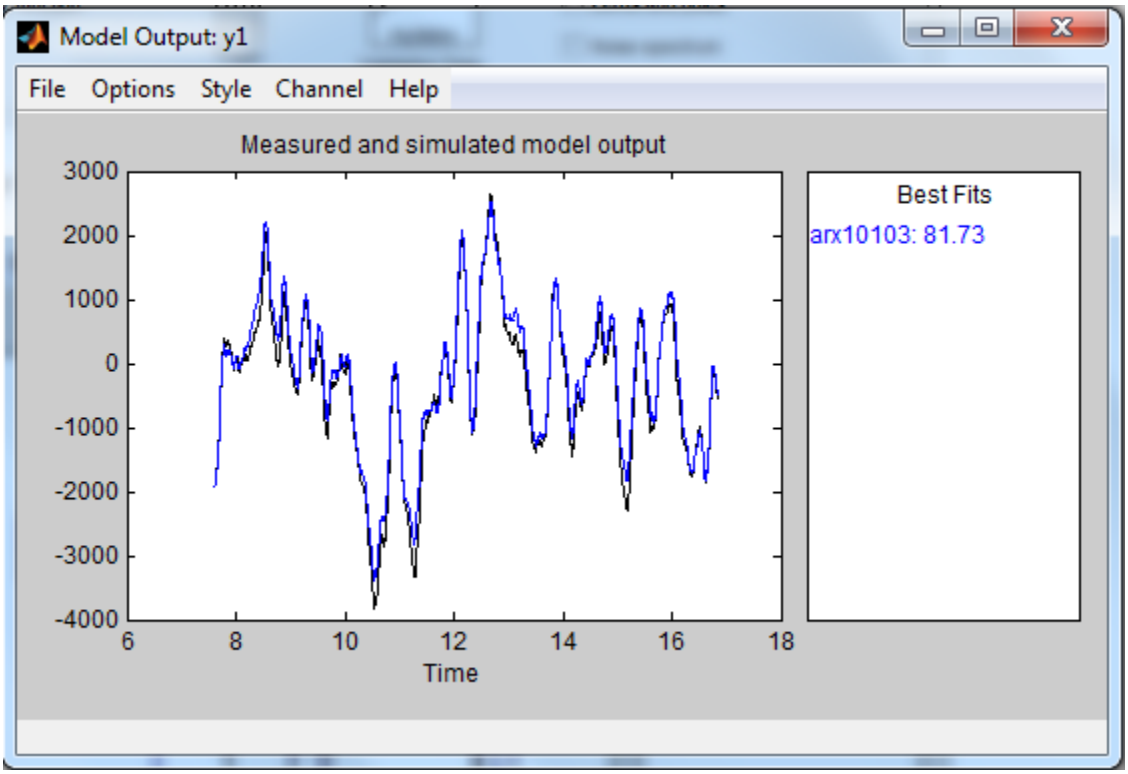
19) Make a selection from the bars and click on ‘Insert’ to import the model into the ‘Model views’ pane. Choose a smaller bar for a good estimate.



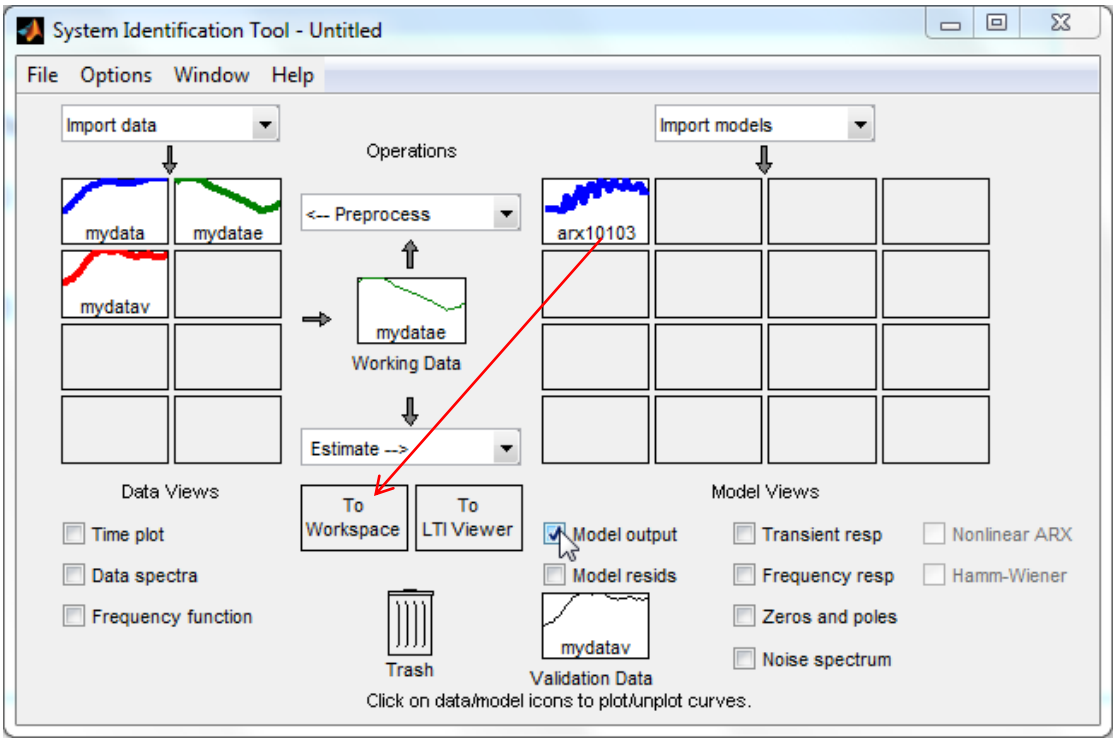
20) Check the ‘Model Output’ box to view the % fit of the obtained ‘ARX’ model.



The higher is the % fit, the better is the estimation of the system under test.



21) From the ‘Model Views’, drag and drop the ‘arxXXXX’ model into the ‘To Workspace’ block to bring the model into the workspace.



22) Find the transfer function and convert the transfer function to continuous time from discrete time.

Tf1=tf(arxXXXX,’m’);

Orig_tf=d2c(Tf1,’Ts’,0.01);

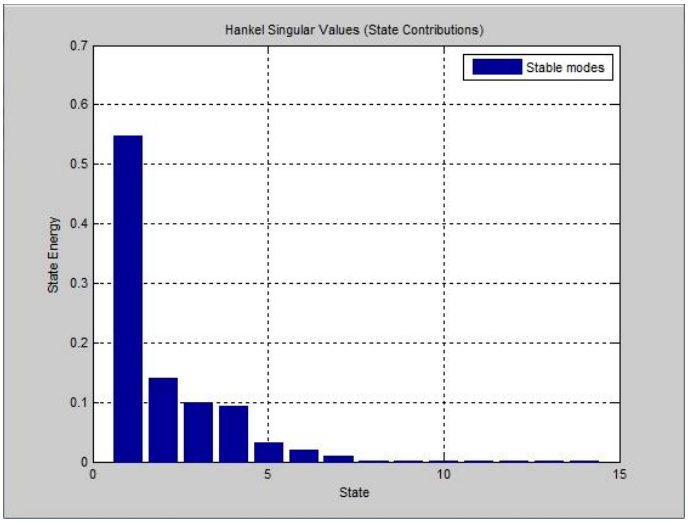
Hankel Singular Value decomposition (HSVD) is used to obtain a reduced order model of the system. HSVD determines the significant modes of the system and provides the state energy of all the states in the system. This information can be used to determine the lowest possible order the system can be reduced.

Tf1=tf(arxXXXX,’m’);

Orig_tf=d2c(Tf1,’Ts’,0.01);

hsvd(Orig_tf)

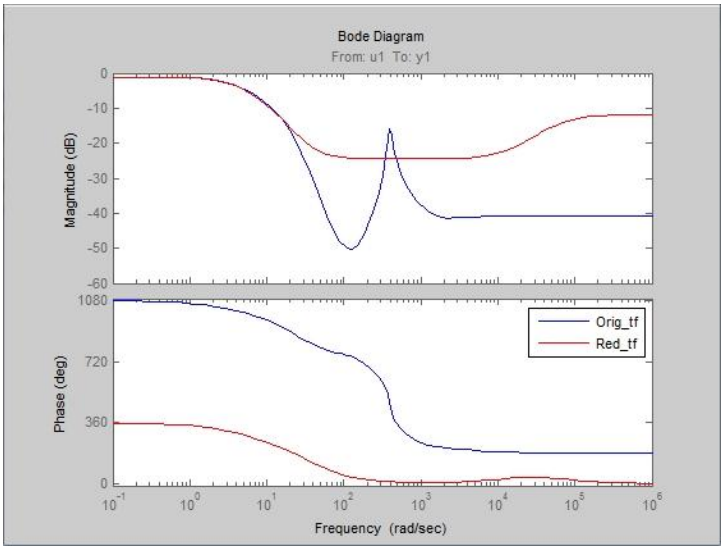
23) In the HSVD plot, the X-axis gives the 'order' of the system, and the Y-axis is gives the 'state energy'. The longer the bars, the more effect the corresponding order will have on the system. Type the commands below to perform these operations and also analyze the obtained figures.



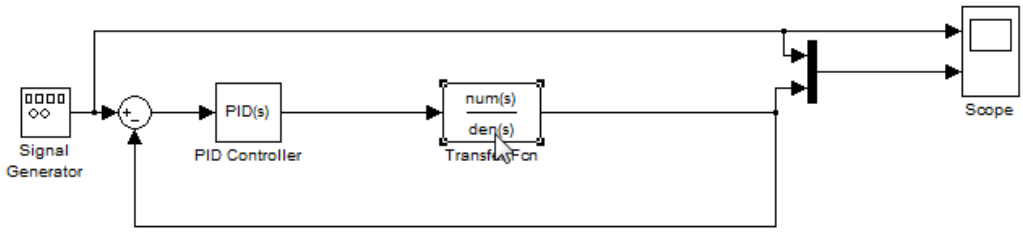
From the above HSVD plot, it is clear that most of the system's state energy is in its first four states. Hence, a 4^{th} or 3^{rd} order state equation can provide an acceptable model for the system. The balanced-order model reduction algorithm can be used to find a system model of specified order, as

Red_tf=balred(Orig_tf,3);

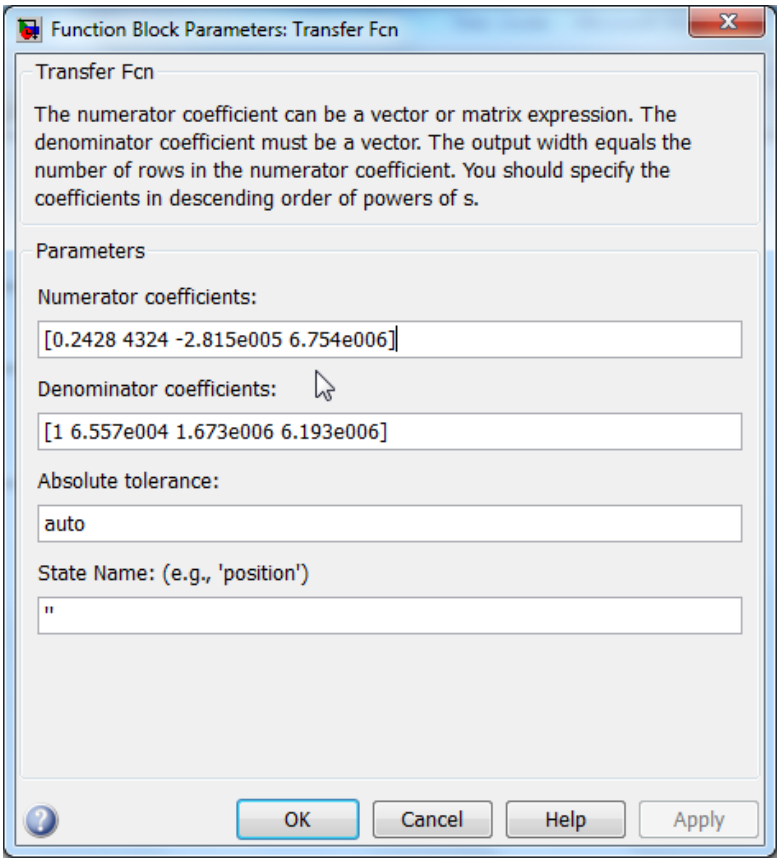
Bode plots of both the transfer functions are compared.



24) With the model obtained, a PID controller is designed to control the speed of the DC Motor. Open the Simulink file 'DC_Motor_Model_with_Control.mdl'.

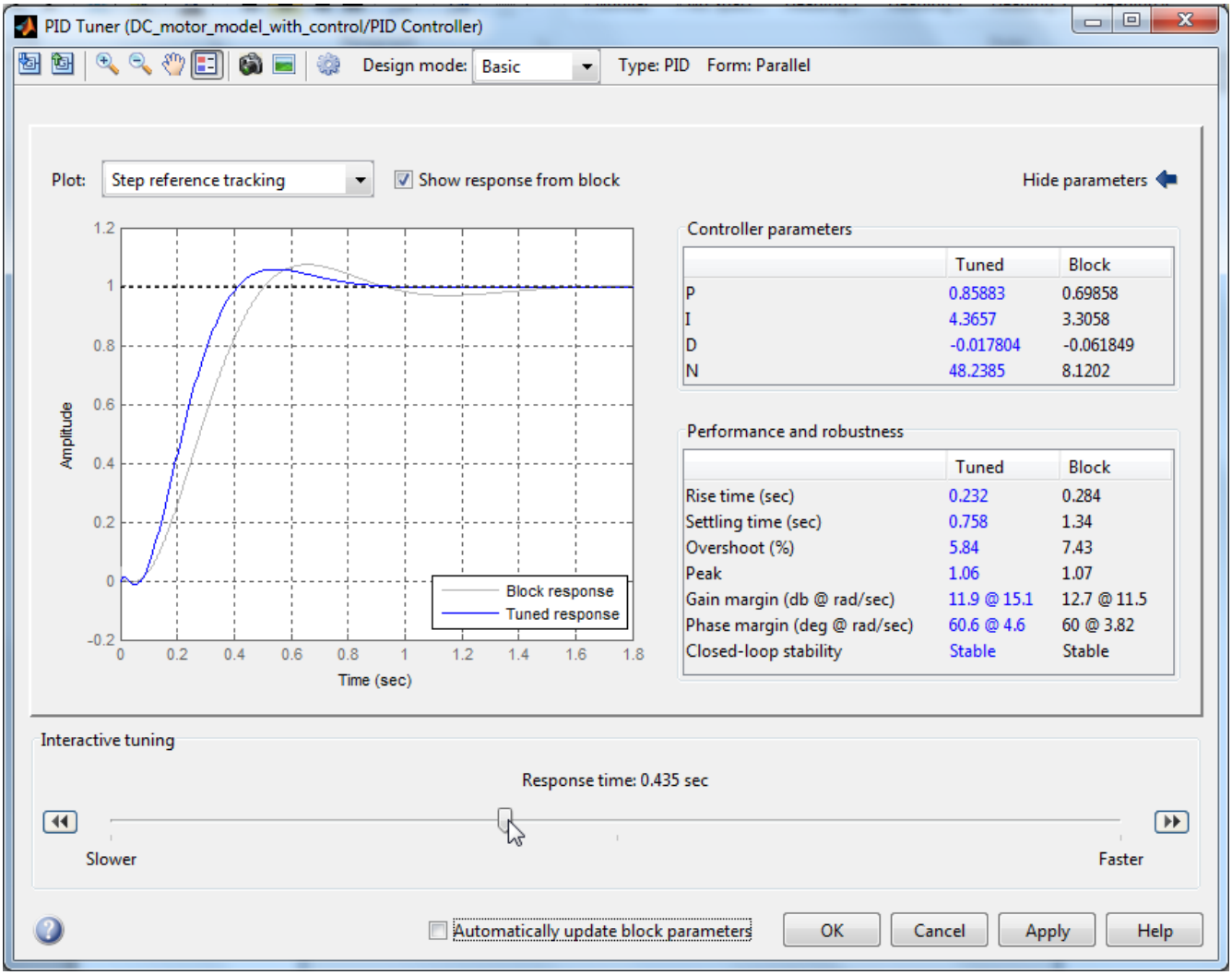


25) Double click on the 'Transfer Fcn' block and enter the numerator and denominator coefficients of the reduced order transfer function.

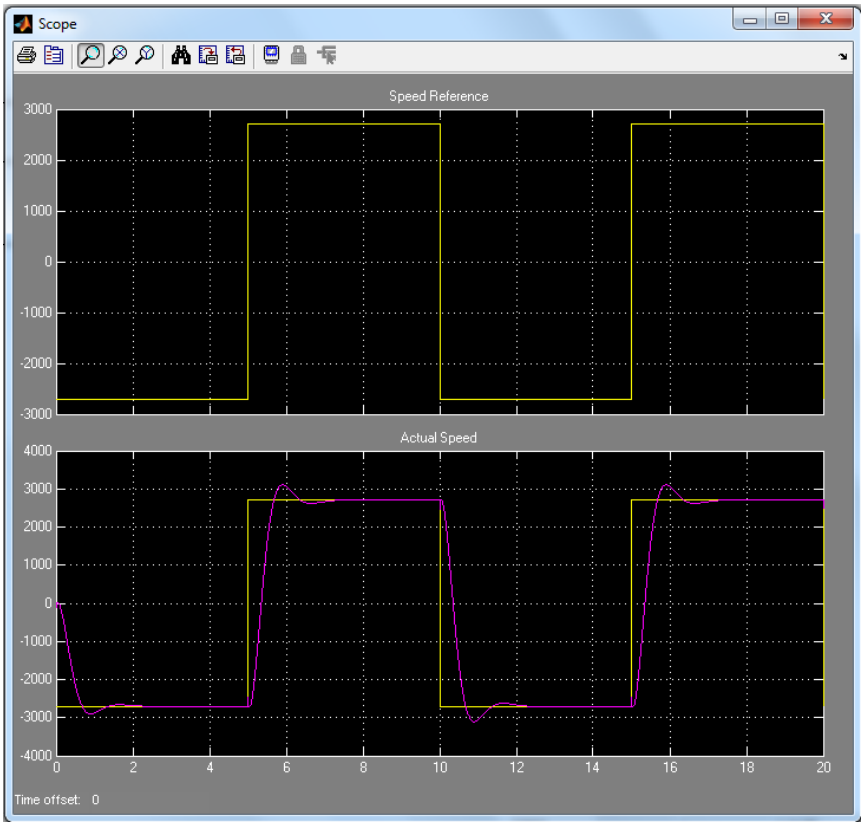


26) The PID controller is tuned to get the desired response from the DC Motor. To accomplish this, double click on the PID controller block in the current Simulink model, and click on 'Tune'. This launches the interactive PID tuning GUI to tune

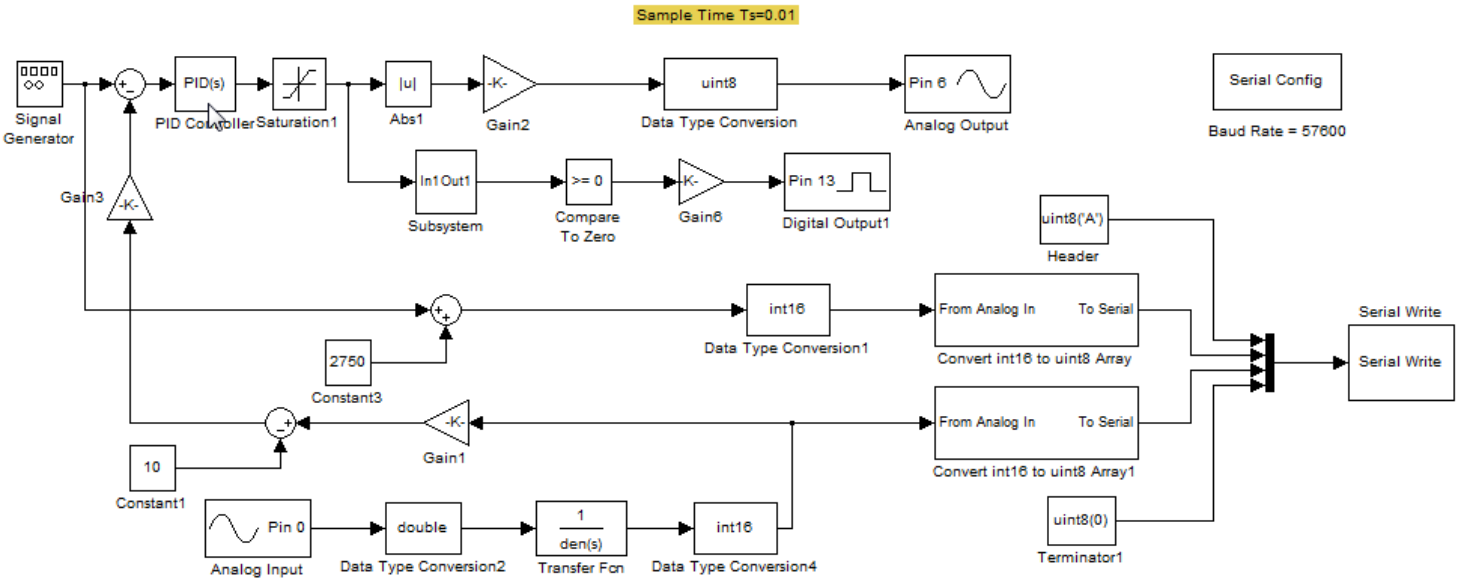
the PID controller to the desired performance parameters. The parameters are tuned by moving the slider as shown below. The tuned parameters are updated by checking the 'Automatically update block parameters'.



27) With the current tuned PID parameters, the response of the obtained DC motor model is checked by simulating the model and observing the output plot. The simulated output is shown (next page). The PID controller is tuned again if the output of the simulation does not satisfy the desired requirements. Repeat steps 26 and 27. On meeting the desired specification, copy the PID controller block from 'DC_Motor_Model_with_Control.mdl'.



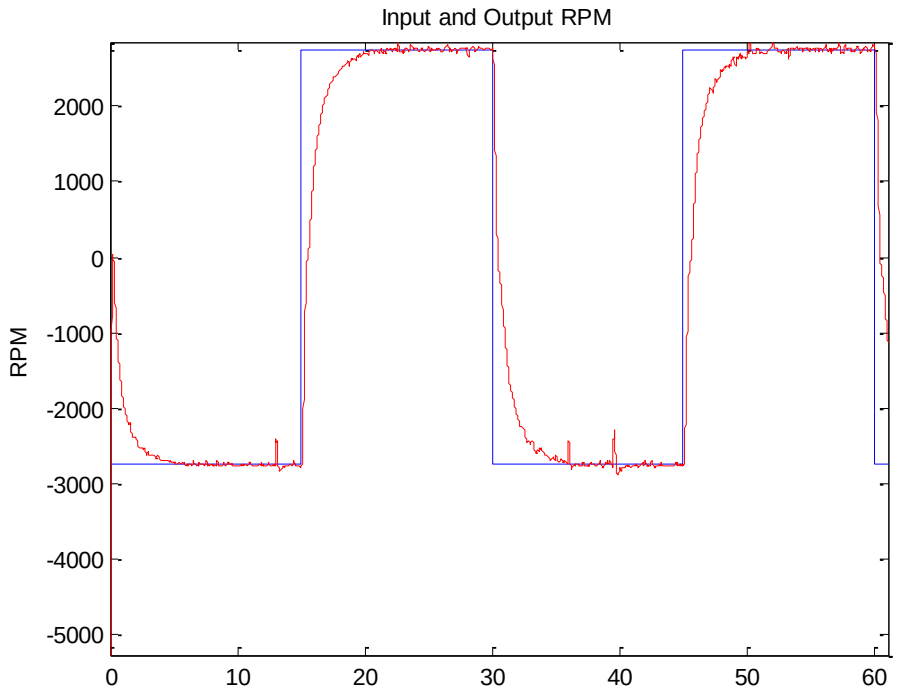
28) This tuned PID controller is implemented in the target to verify the response of the actual DC motor. Open the Simulink file 'Mot_with_Control.mdl'. Paste the copied PID controller in place of the existing PID controller block in the 'Mot_with_Control.mdl' Simulink model.



29) ‘Ctrl + B’ is used to build this model into the target and checked for the actual DC motor response.

30) Repeat steps 7-12 for capturing the data into MATLAB workspace. Capture the data for 4 cycles when the DC motor is running.

31) To extract the data, open the program ‘Extract_data.m’ and execute the ‘With Controller’ cell by using ‘Ctrl+Enter’ and then observe the plot.



32) This completes the DC Motor system identification with control design using the System Identification toolbox. The plot above successfully demonstrates the capability to successfully implement a control of a DC Motor using the existing MATLAB/Simulink tools on the Arduino platform.

Alternatively, the parameters of a Simulink/Simscape model of the DC Motor can be estimated and adjusted to match the captured I/O data of the actual DC Motor. Later, a controller can be designed to control the Simulink/Simscape DC Motor which can be implemented to target to check the performance of the actual DC Motor.

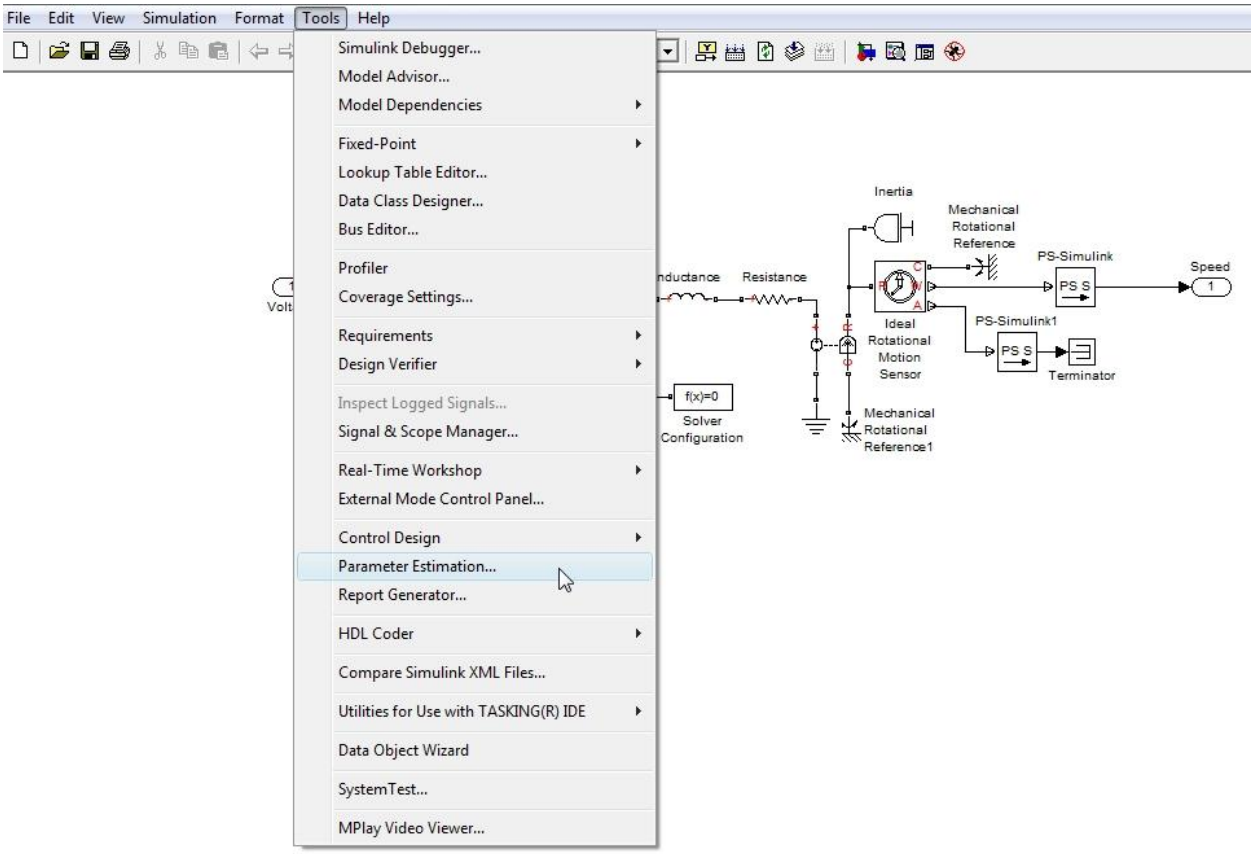
Procedure for PARAMETER ESTIMATION

1) Clear the workspace and load *'Original_data.mat'*. Confirm the 5 variables present in the workspace: Ts, comports, simout, simout1 and tout. Open 'Extract_Data.m' and execute the 'Without_Controller' cell.

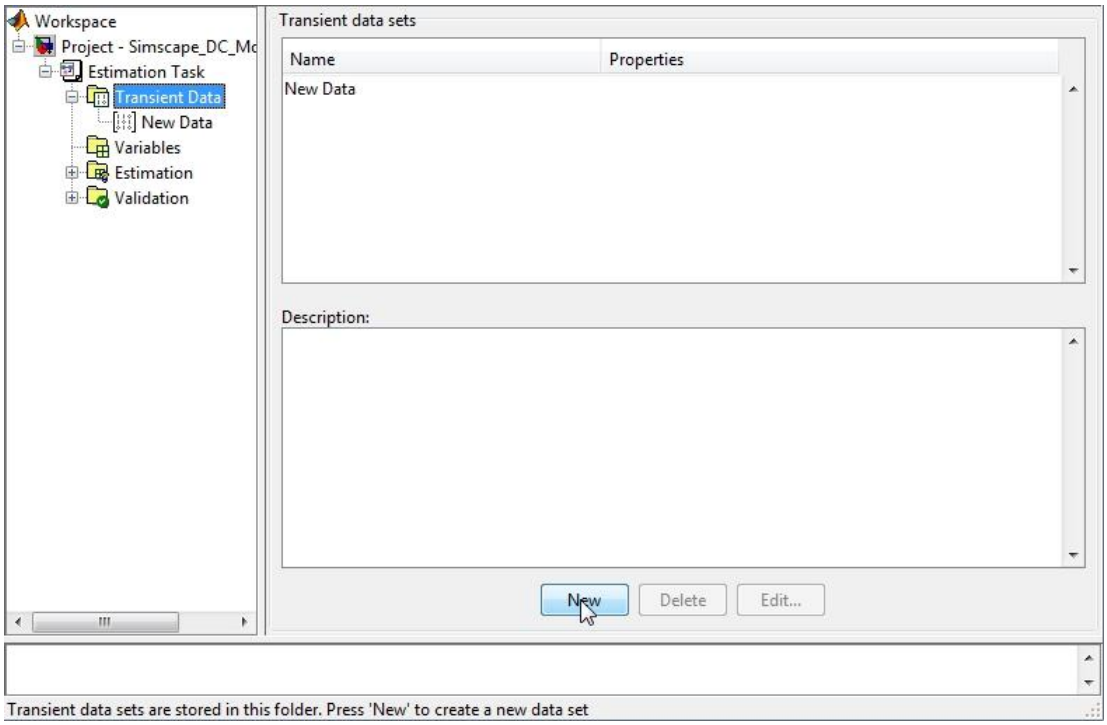
2) Open the Simulink file 'Simscape_DC_Motor_Model.mdl'. Double click on the subsystem to see the Simscape model of the DC Motor.

3) Check the initial values of the L, R, I, K, k(1:6) parameters of the DC motor in the workspace before you proceed further. If you don't find these variables, please contact the T.A.

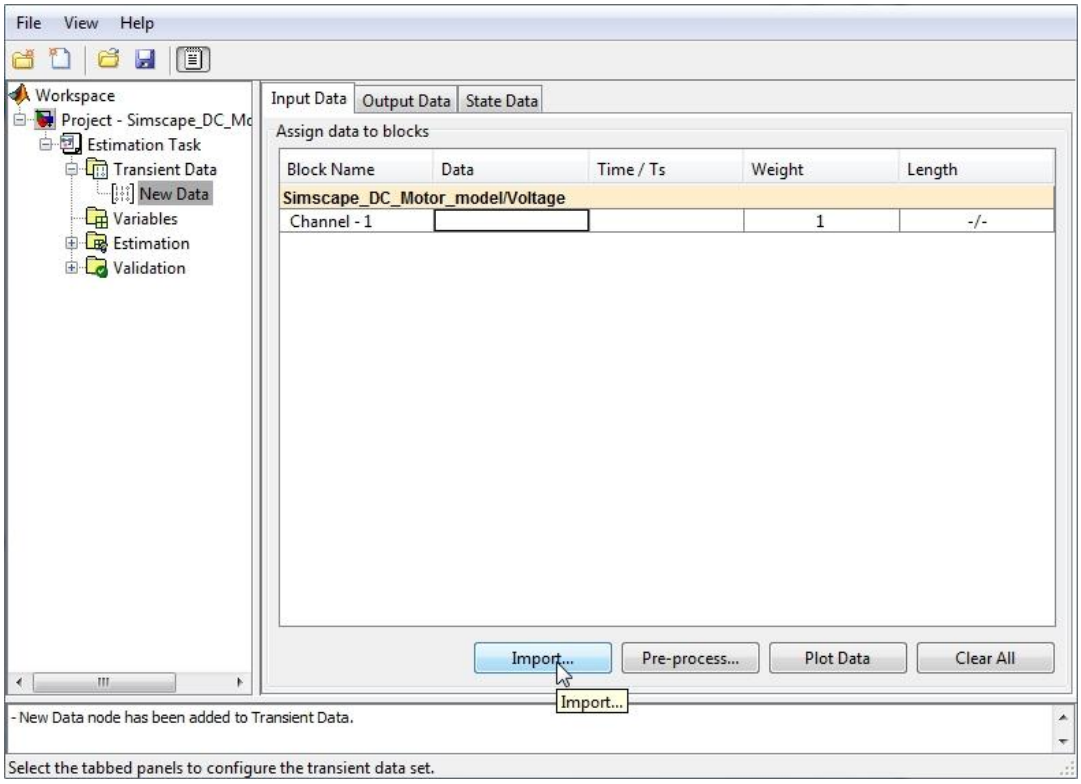
4) From the 'Tools' menu, open the 'Parameter Estimation' dialog.



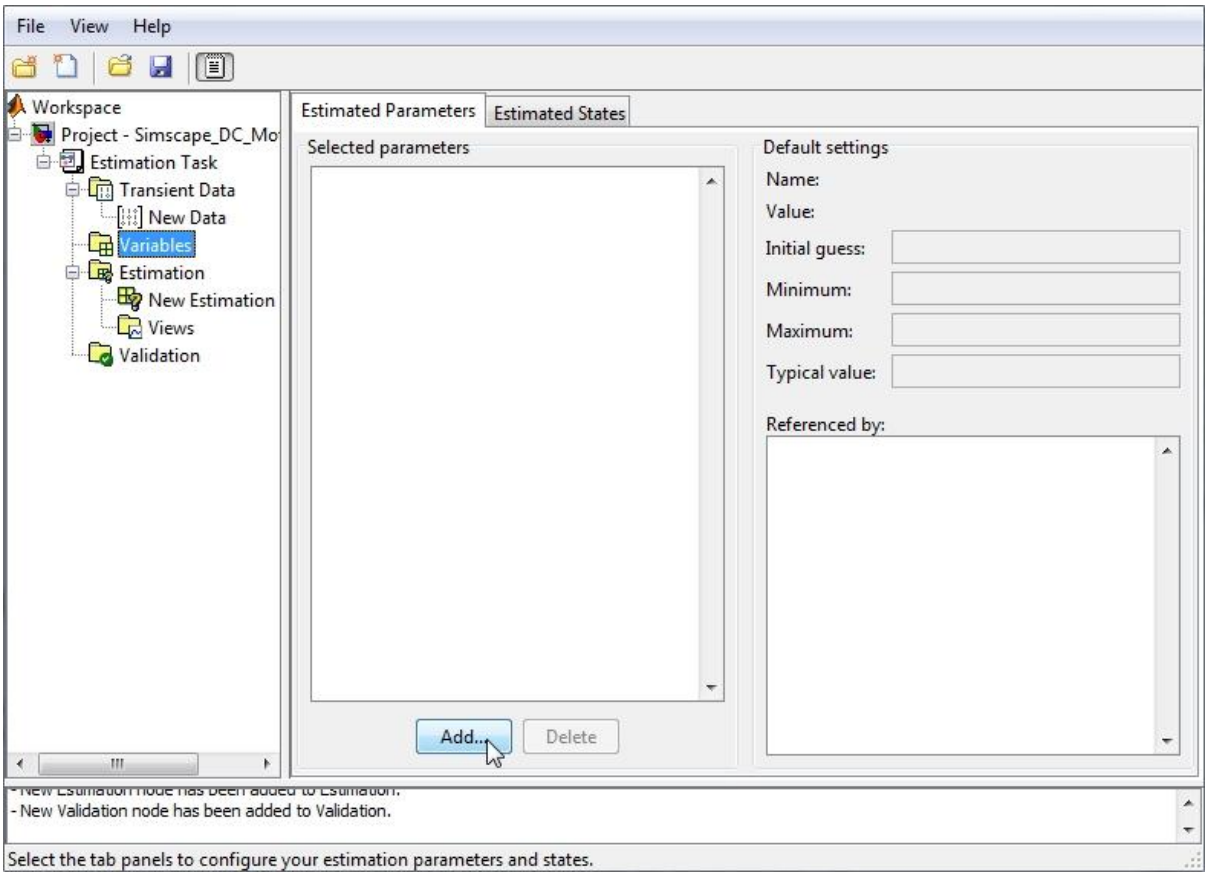
5) Click on ‘Transient Data’ and click on ‘New’ to create ‘New Data’ as shown.

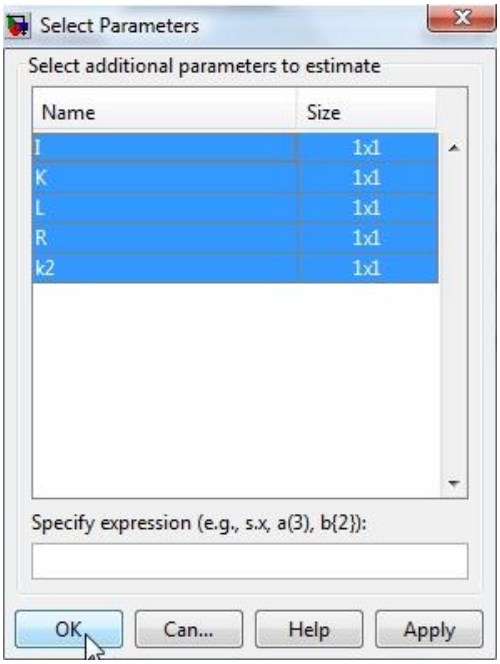


6) Click on ‘New Data’ and Import the Input data ‘*ir*’ and Output data ‘*or*’ from the workspace. *‘t1’* is the time vector.



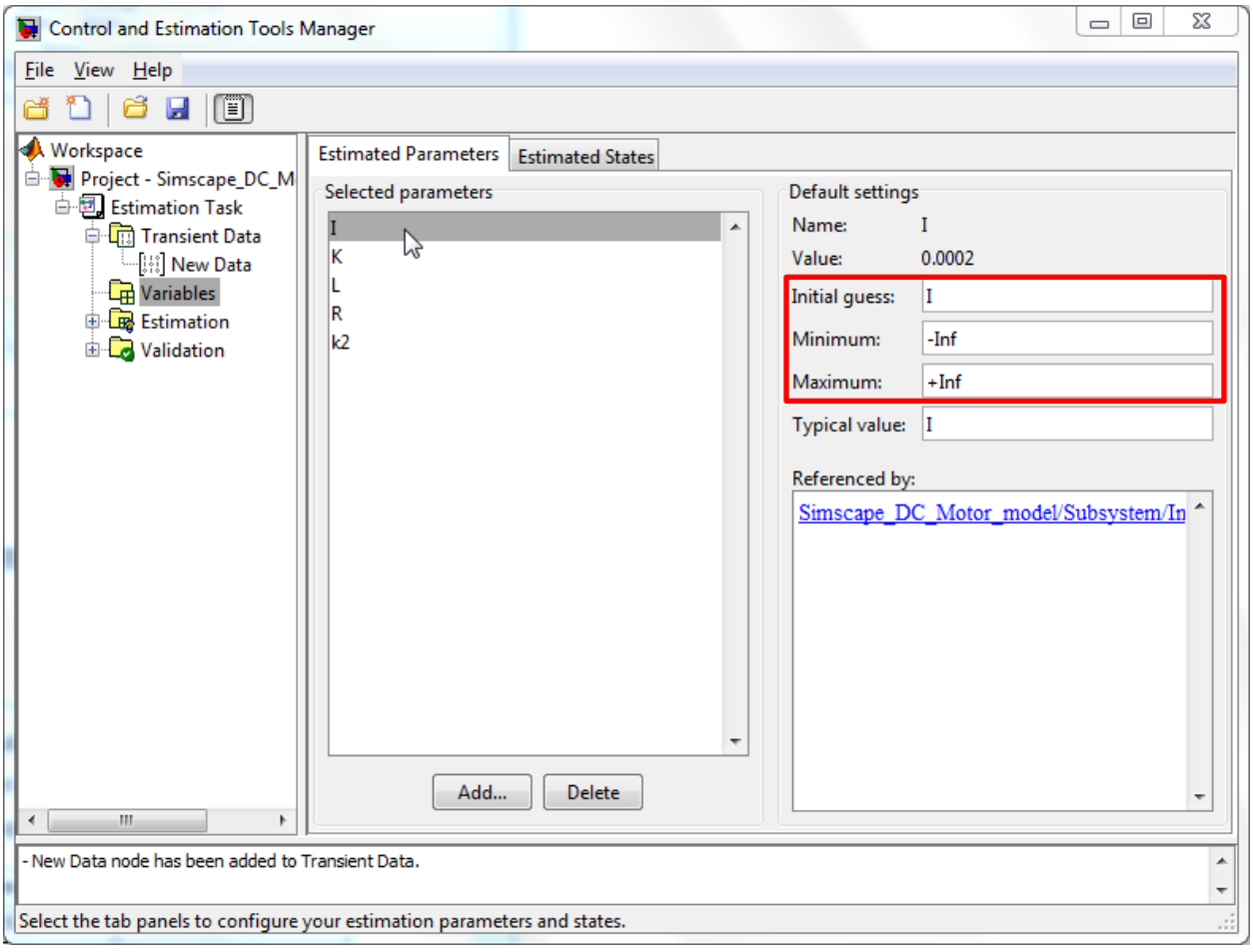
7) Under the ‘Estimation Task’ menu, select ‘Variables’ and then click on ‘Add’ to select all the parameters to be estimated.



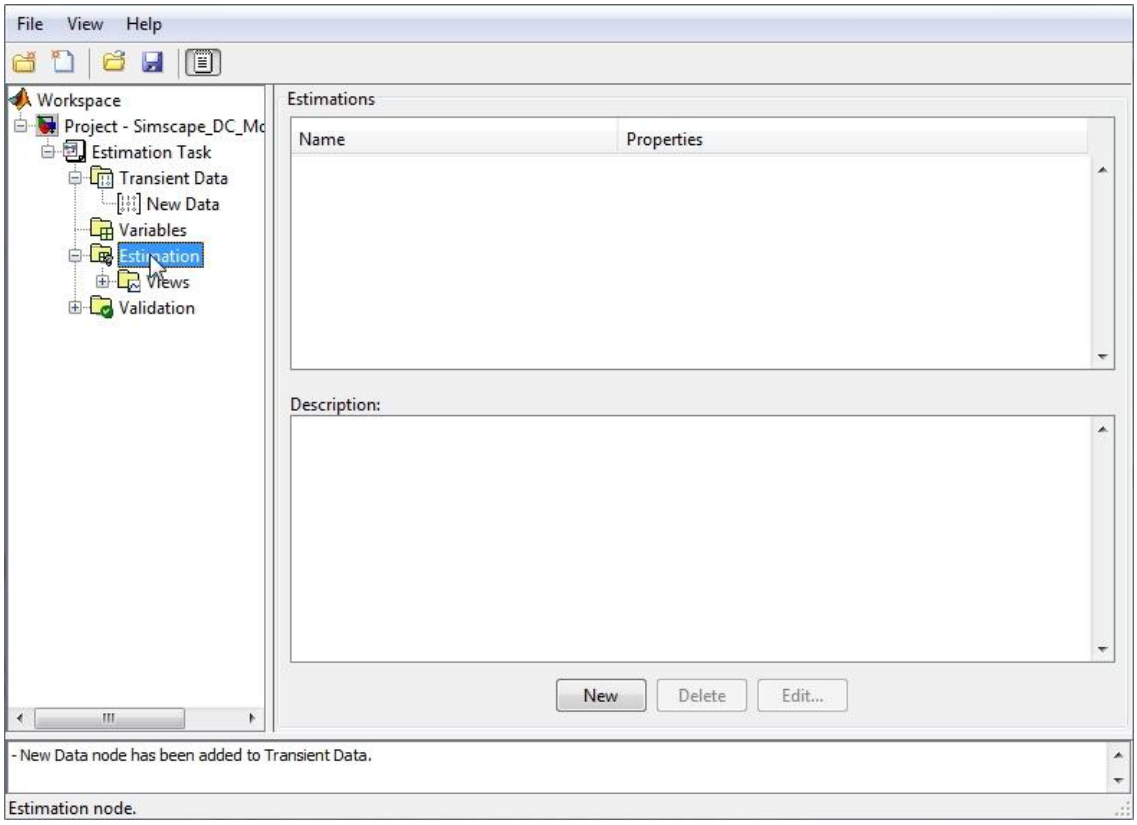


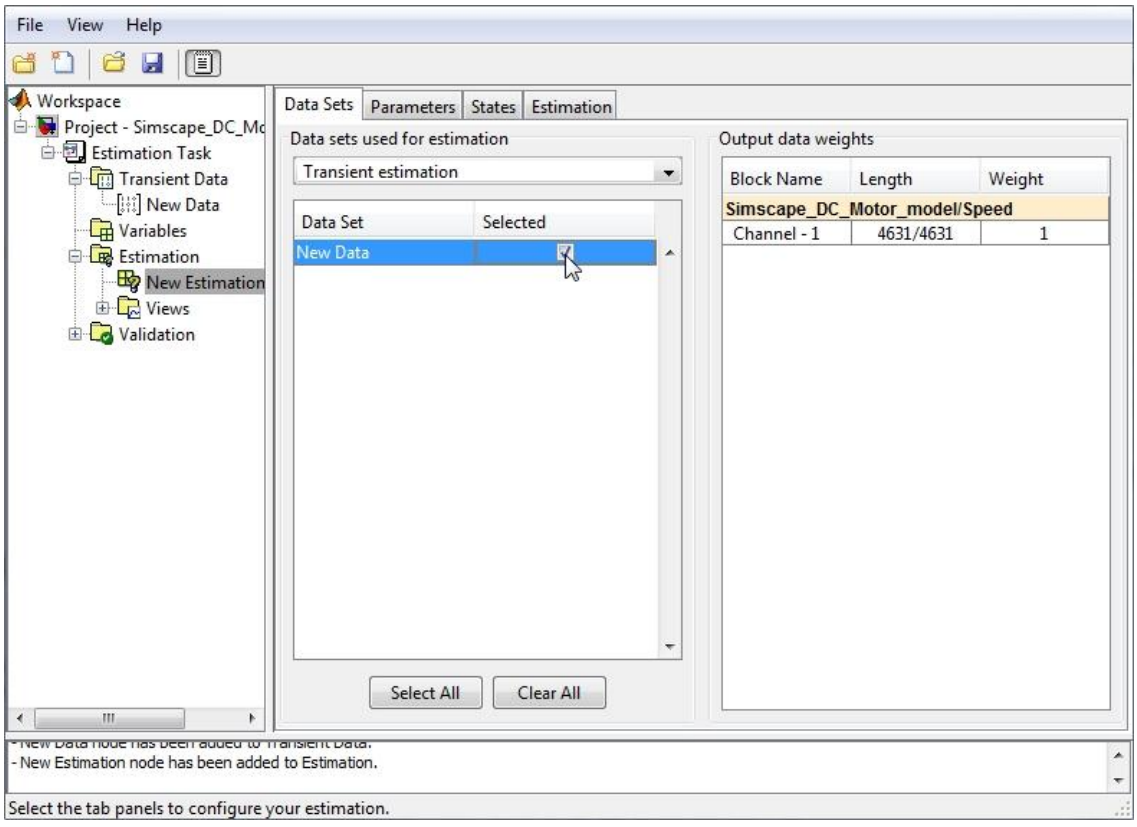
8) Specify the initial guess, minimum, and maximum values for all the chosen parameters. From the 'Selected Parameters' pane, choose the variable and edit its values in the highlighted window as shown. The values are specified in the table below:

Parameter	Initial guess	Minimum Value	Maximum Value
I	0.0002 Kgm^2	0.0001	0.001
K	12/4000 V/rpm	0.0002	0.003
L	7 mH	0	400
R	10 Ω	0	400
K2	12/4000 V/rpm	0.0002	0.003

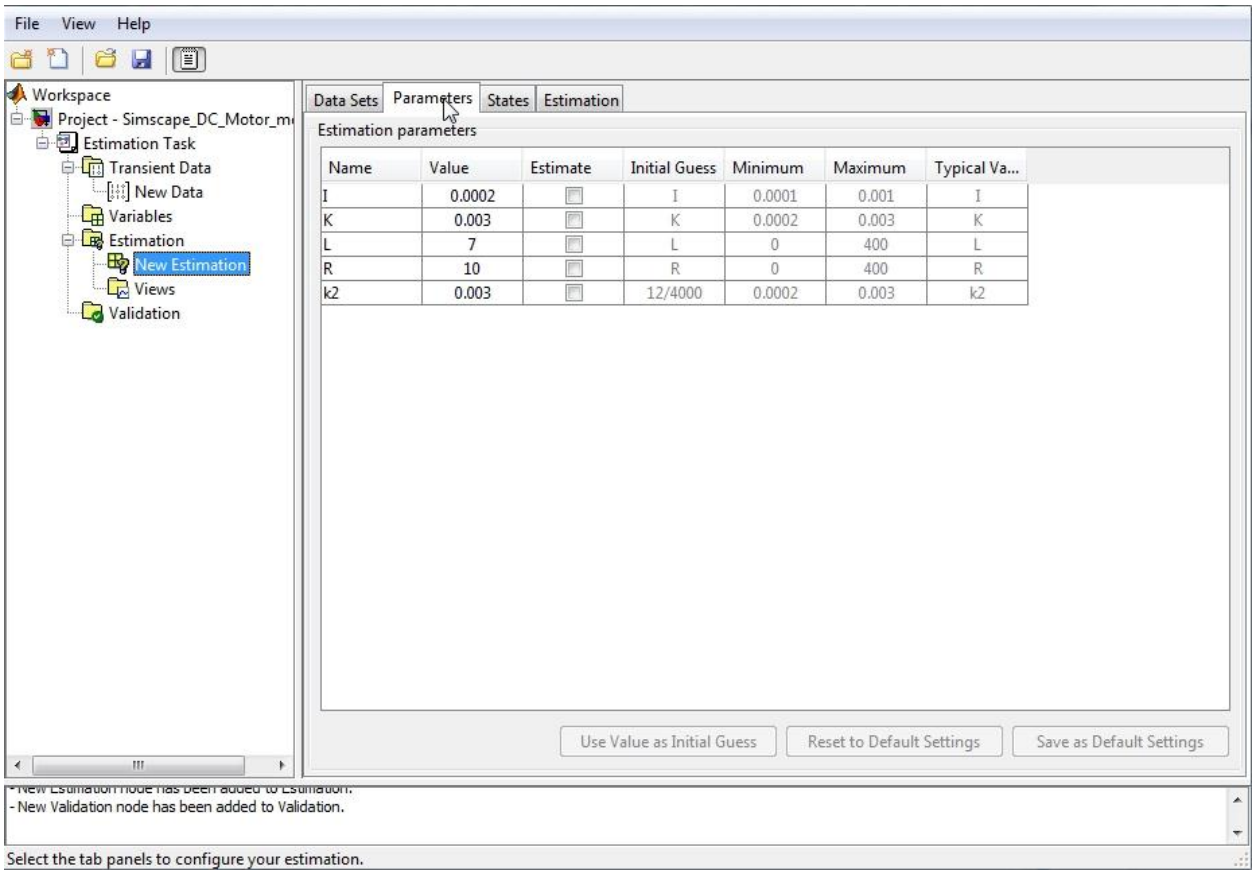


9) Click on 'Estimation' to add a 'New Estimation' environment. Click on 'New Estimation' to select the data set from the 'Data sets' tab as shown in the figures (next page).

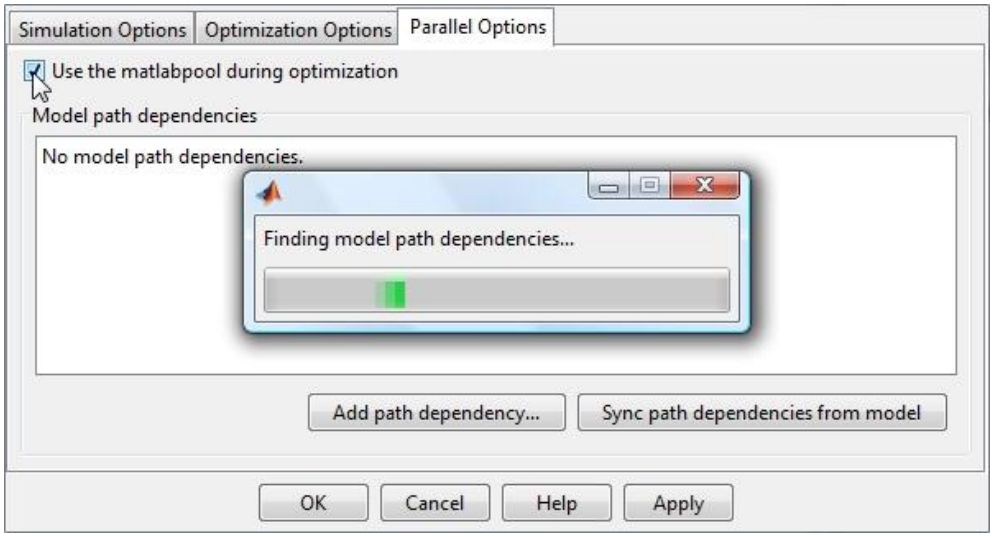

File View Help
Workspace
Project - Simscape_DC_Mo
Estimation Task
Transient Data
New Data
Variables
Estimation
Views
Validation
Estimations
Name
Properties
Description:
New
Delete
Edit...
- New Data node has been added to Transient Data.
Estimation node.


File View Help
Workspace
Project - Simscape_DC_Mo
Estimation Task
Transient Data
New Data
Variables
Estimation
New Estimation
Views
Validation
Data Sets
Parameters
States
Estimation
Data sets used for estimation
Transient estimation
Data Set
Selected
New Data
Select All
Clear All
Output data weights
Block Name
Length
Weight
Simscape_DC_Motor_model/Speed
Channel - 1
4631/4631
1
- New Estimation node has been added to Estimation.
Select the tab panels to configure your estimation.

10) Using the ‘Parameters’ tab, select and check all the variables for estimated.

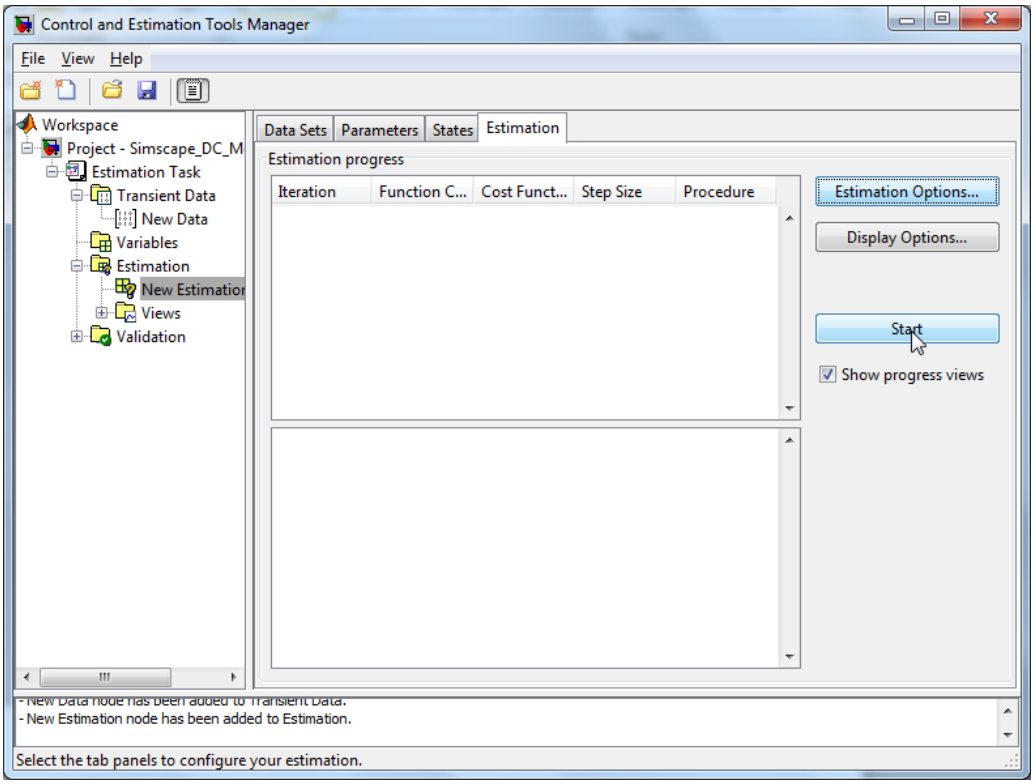


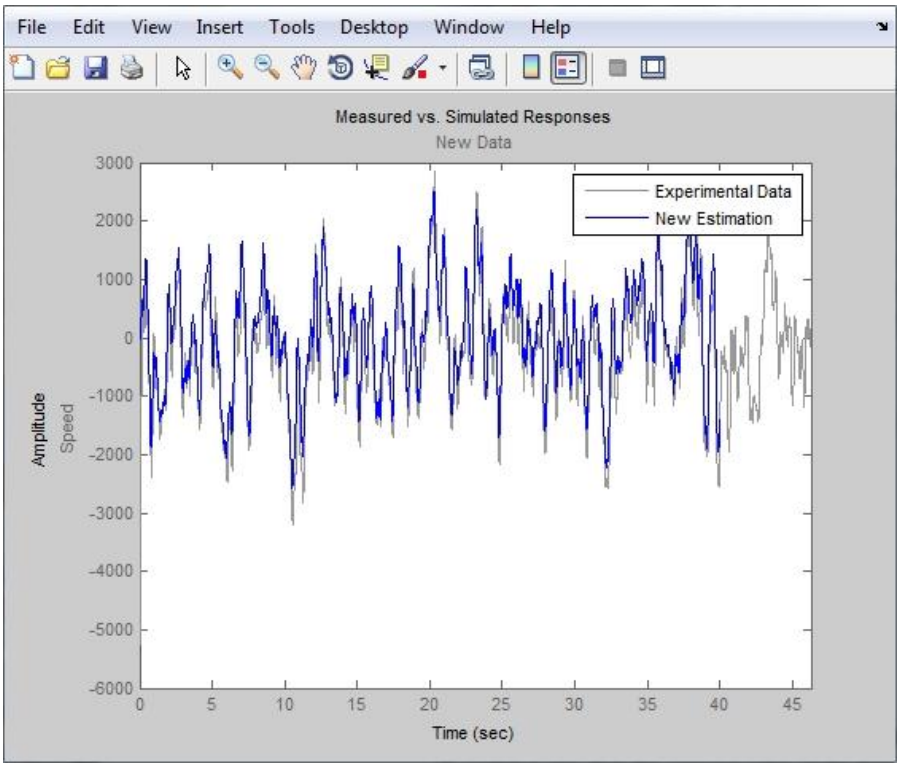
11) From the ‘Estimation’ tab, click on ‘Estimation Options’, and under the ‘Parallel Options’ tab, select the ‘Use the matlabpool during optimization’ and click OK.

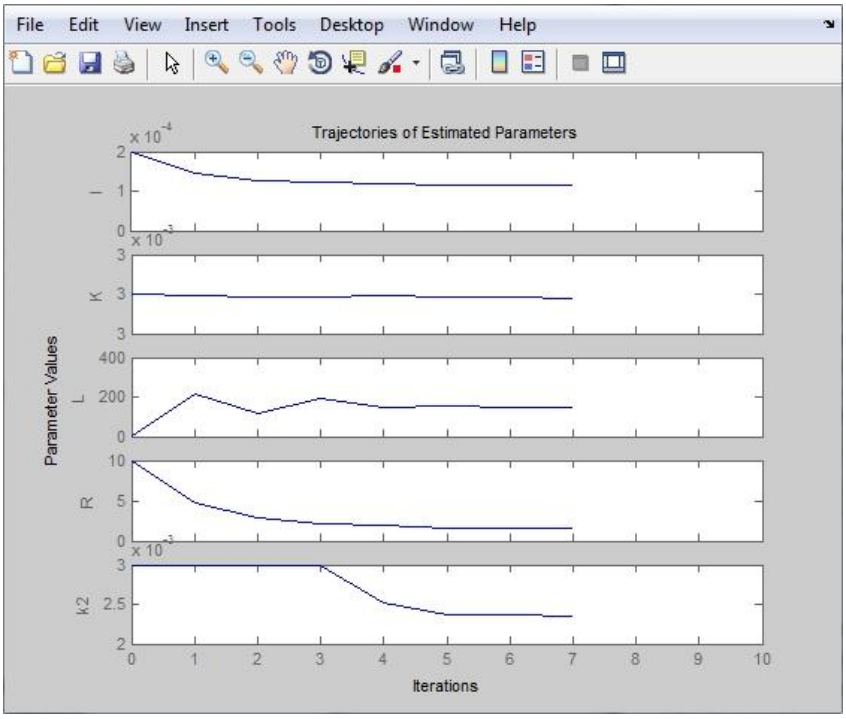


12) In the MATLAB command window, type ‘matlabpool’.

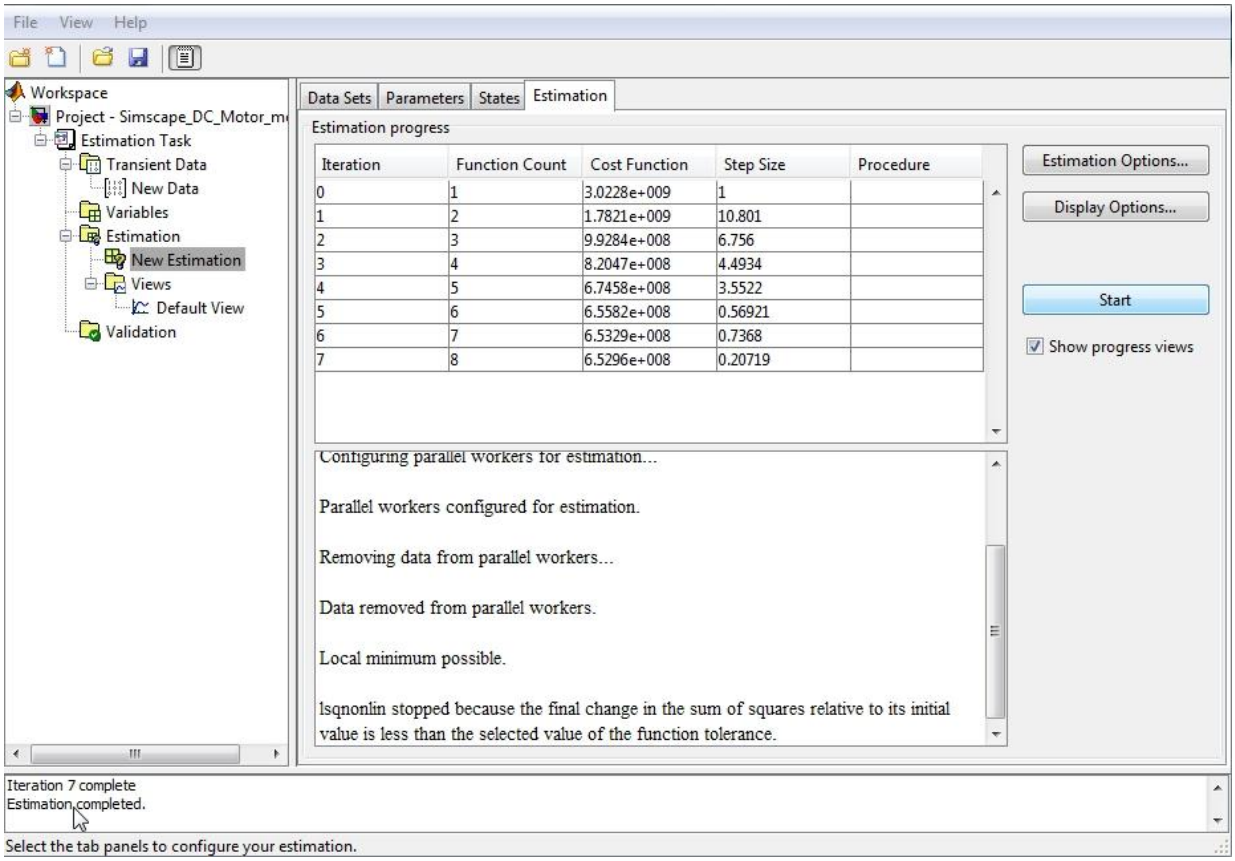
13) Under the ‘Estimation’ tab, select ‘Show Progress views’ and click on ‘Start’ to start the simulation and observe the plots.







14) Check if the estimation is completed as shown below.



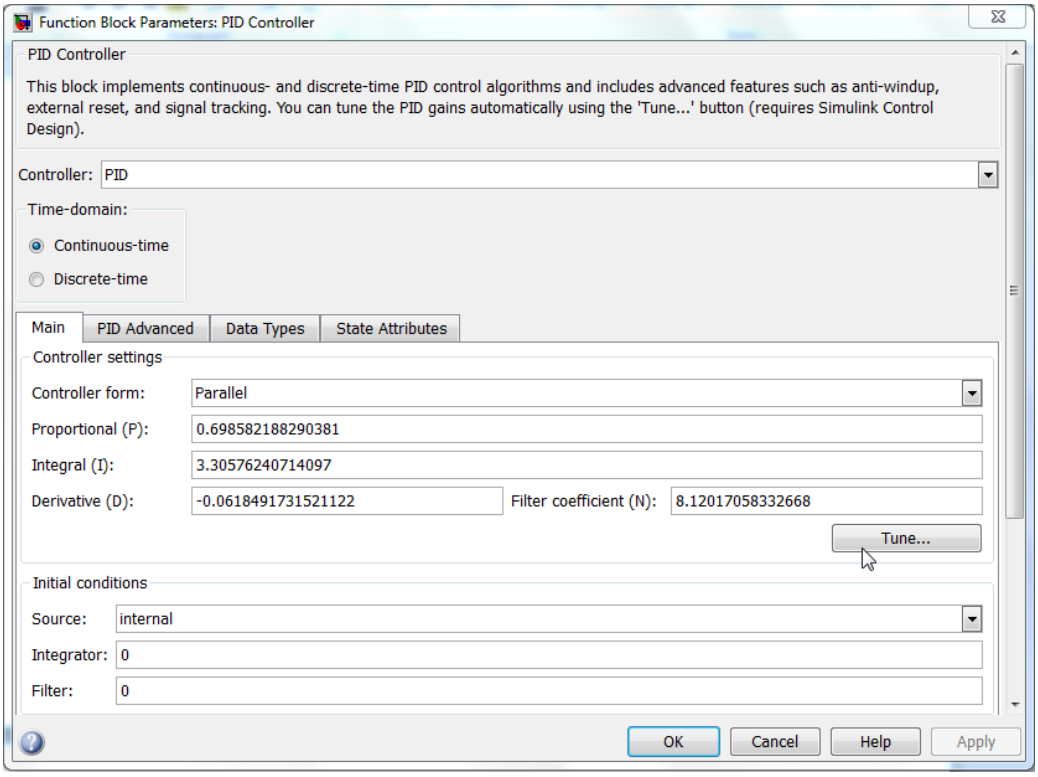
Procedure for CONTROL DESIGN

At this point, the parameters of the Simulink/Simscape DC motor are tuned using the captured I/O data to match the original DC motor under test. Copy the Simulink/Simscape model of the DC Motor subsystem

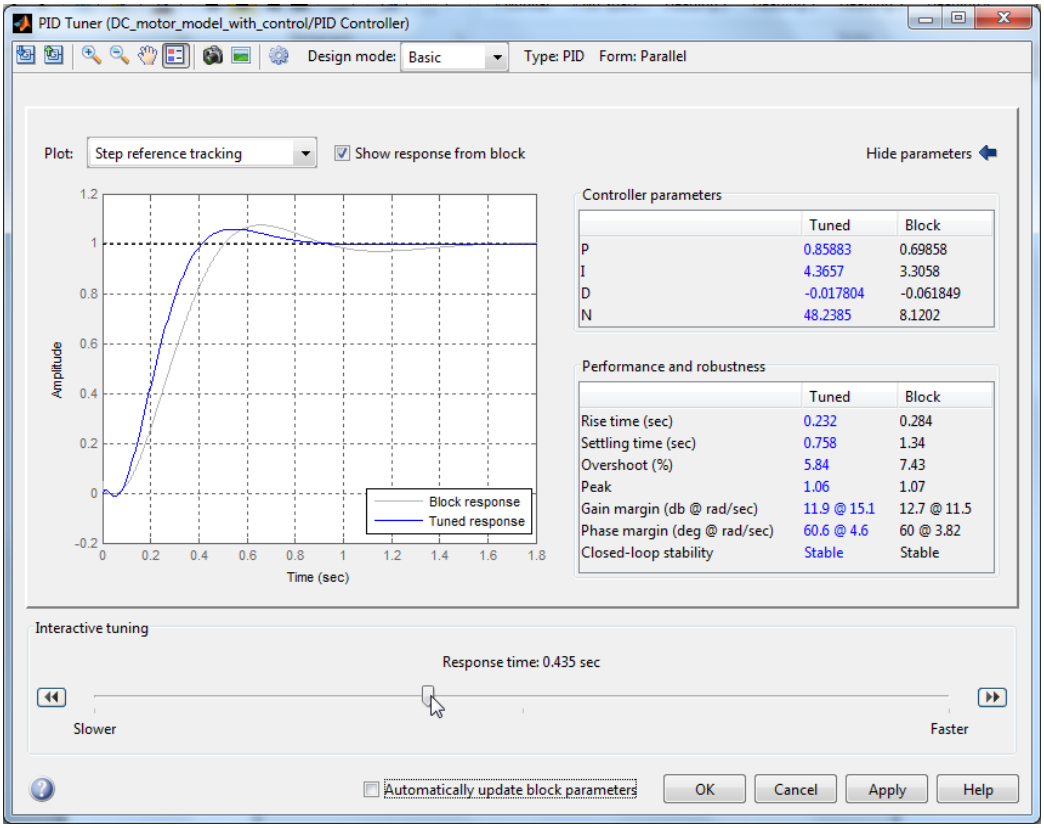
In this section, we design a PID controller to control the speed of the DC motor with satisfactory transient and steady-state characteristics. It should be noted that there are many control structures that could be considered for the speed control of the motor. Here, the PID control is chosen because it has a simple structure and it can be auto-tuned in MATLAB.

15) Open the Simulink file 'Simscape_DC_motor_model_with_control.mdl' and paste the copied DC Motor subsystem.

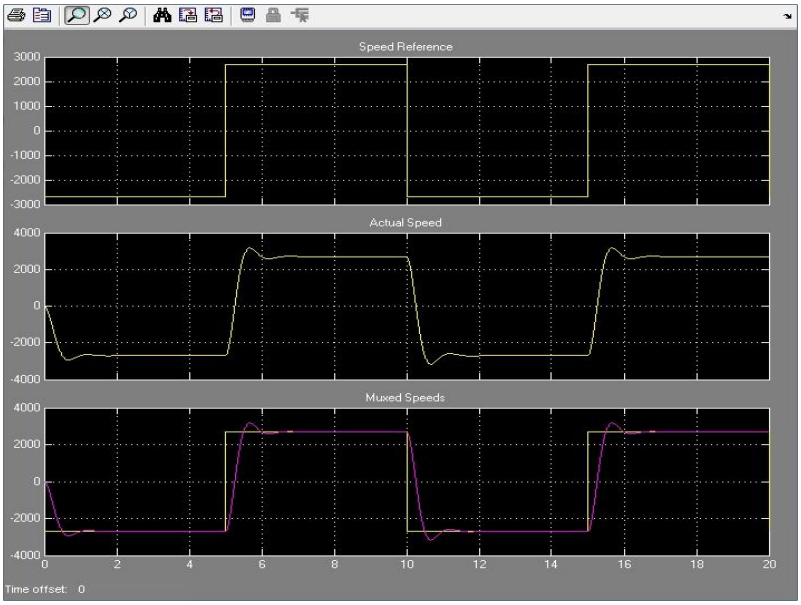
16) Double click on the PID Controller block to tune the PID parameters. Click 'Tune'.



17) Using the 'Interactive Tuning' sliding bar, you may further tune the PID controller to meet the desired specifications. Check the 'Automatically update block parameters'.



18) Run the simulation with the new PID gains and observe the output of the Simulink/Simscape model of the DC motor in the closed-loop structure.

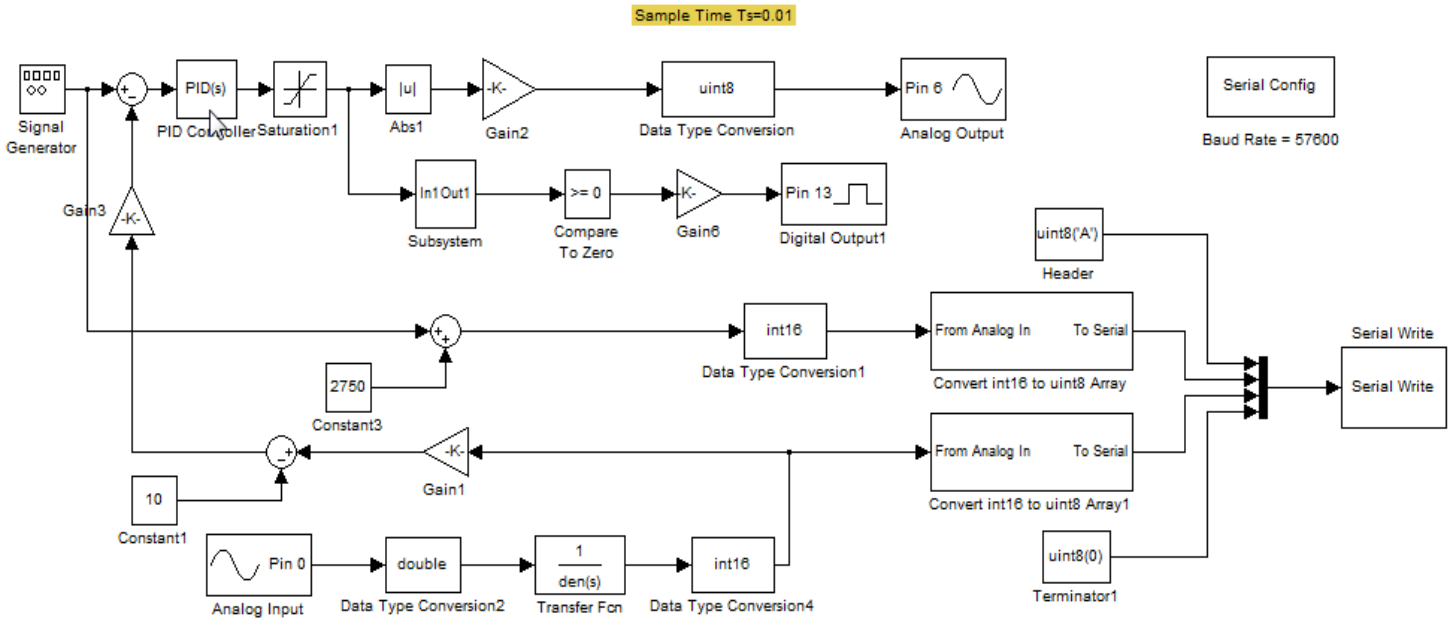


19) We have now successfully designed a PID control for the Simulink/Simscape model of the DC motor, which corresponds to the DC motor on the platform. Copy the PID Controller block.

Procedure for CONTROL IMPLEMENTATION and RAPID PROTOTYPING

The tuned PID controller is implemented in the Arduino target to verify the speed response of the DC motor in the hardware platform.

20) Open the simulink file 'Mot_with_Control.mdl' and paste the PID controller block into this model. Build and load this model into 'Arduino' using 'Ctrl+B'.

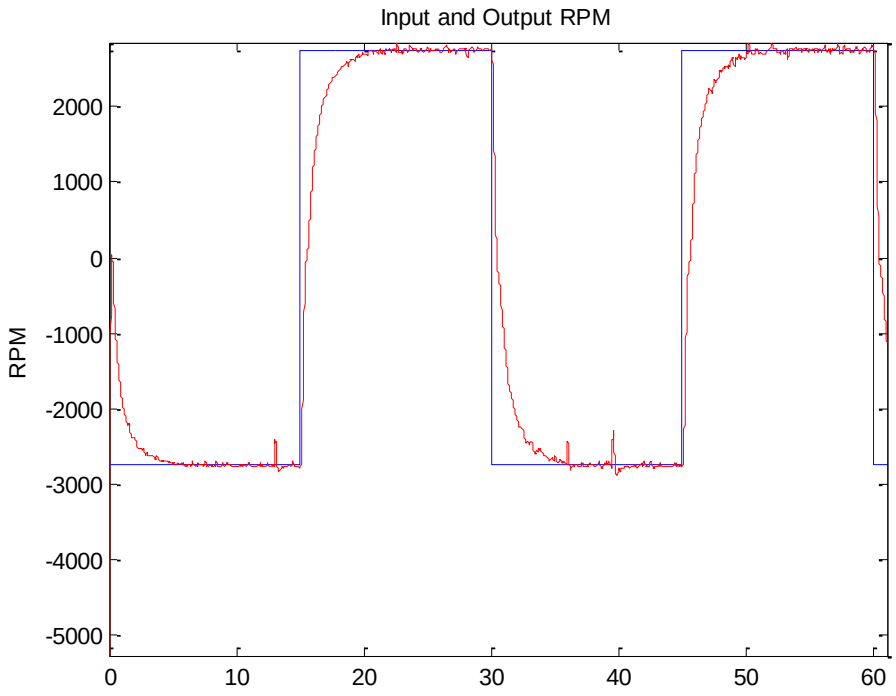


21) Clear the variables 'simout' and 'simout1' in the workspace.

22) Use steps 7 – 11 from the 'System Identification procedure' to capture the data into workspace.

23) Run the simulation to capture at least 4 cycles of the DC Motor data. We have 'simout' and 'simout1' in the workspace containing the new data of the controlled DC motor.

24) Open the MATLAB file 'Extract_data.m' and execute the 'With Controller' cell by pressing the 'Ctrl+Enter'. Plot and observe the output waveforms and verify if the speed response of the DC Motor in the hardware is satisfactory.



25) Show the waveforms to the T.A.

This concludes the PID control design, implementation, and rapid-prototyping part of the experiment for the DC Motor platform using ‘Arduino’.

Arduino Target for Simulink Installation procedure

Attention: Please note that this installation 'readme' file is valid and applicable only for 32-bit Operating systems running MATLAB 2010a or above with Instrument Control Toolbox, Real-Time Workshop and Real-Time Workshop with Embedded Coder. You need admin privileges.

1) Create a folder in C:\ called Arduino and download 3 folders from internet. Place/Download into the folder on C:\Arduino. (Need to log-in into your Mathworks account. If you don't have, create one for free).

a. Arduino Target for SIMULINK from Mathworks:
http://www.mathworks.com/academia/arduino-software/arduino-simulink.html
Extract the zipped folder into a folder named 'Arduino_Target'.

b. Arduino Interface for MATLAB:
http://www.mathworks.com/academia/arduino-software/arduino-matlab.html#
Extract the zipped folder into a folder named 'arduino_ml'.

c. Arduino Integrated development environment: (Many versions are available. Recommended would be the version 'arduino-0022'. This downloaded folder includes the USB (FTDI) virtual COM port drivers required for Windows XP).
http://arduino.cc/en/Main/Software

2) Start MATLAB and change the current working directory to C:\Arduino.

3) Connect the Arduino board. The operating system (Windows Vista and Windows 7) automatically configures the FTDI drivers and successfully assigns a COM port. You would be notified of the COM port number assigned. Note down the COM port #.

4) Run the executable file 'Arduino' form 'arduino-0018' folder and under tools select your COM port. At this juncture, your Arduino board is ready to be programmed in command line. You can run your program and compile into the board. For further instructions, please open the 'Readme' in the 'arduino_ml' folder.

5) Arduino Simulink Installation:

a. Check the current directory. Change it to C:\Arduino\Arduino_Target.

b. In the Matlab command window, type:

```
>>addpath(fullfile(pwd,'arduino'),fullfile(pwd,'blocks'),fullfile(pwd,'demos'))
>>savepath
```

If you encounter an error after savepath, then your operating system is not allowing any changes to be made to the MATLAB file 'pathdef.m'. To overcome this problem, right click on the MATLAB R2010a folder, click on properties, select security, click edit and then check the box 'Full Control'. Try 'savepath' again. If no success, contact the administrator.

c. Type:

```
>> sl_refresh_customizations
>> arduino.Prefs.setArduinoPath('c:\Arduino\arduino-0018')
>> arduino.Prefs.setMcu('atmega328p') % or atmega168
>> comPorts=arduino.Prefs.searchForComPort
>> arduino.Prefs.setComPort(comPorts{1});
```

d. Select the compiler for Matlab. Type:

```
>> mex -setup
```

The compilers would be listed, choose the Matlab Lcc win32 compiler.

e. Type:

```
>> demo_arduino_blink
```

Once the file opens, build the file. Use Ctrl+B to build the file into the target. During the build procedure, the TX and RX led's on the Arduino board blink when the hex file is downloading into the target.