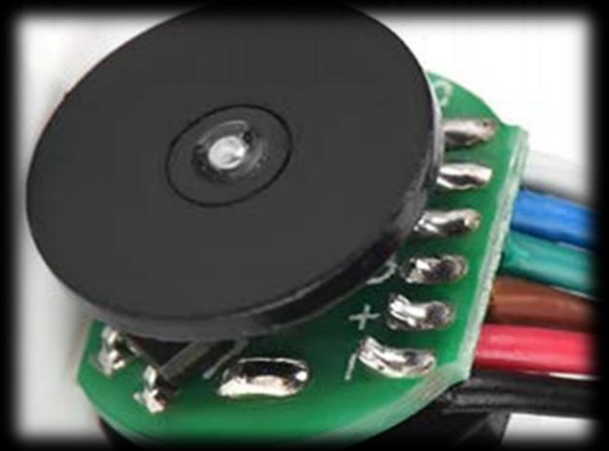
Contents

Least Squares for Parameter Estimation of a DC Motor



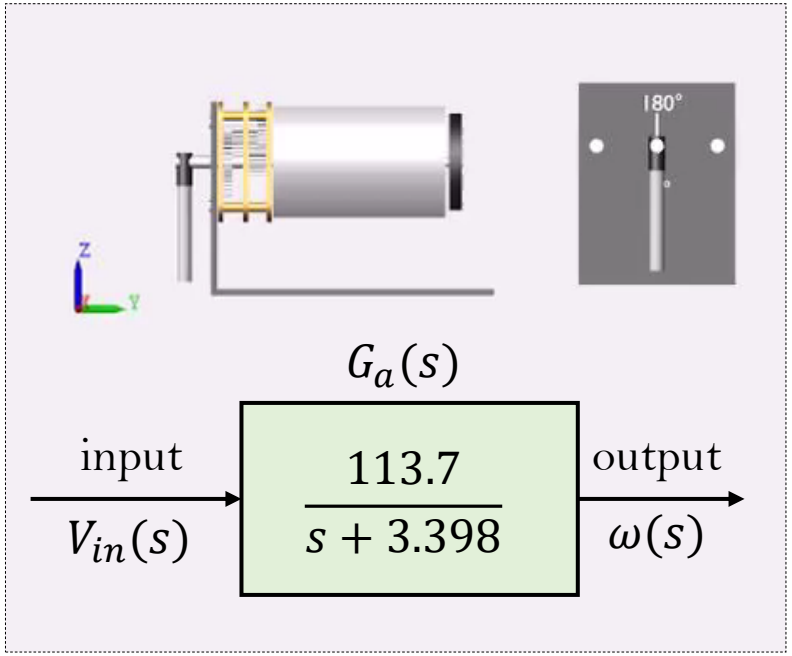
Key Learning Points

After this Lecture, you will be able to understand the following:

1. How to capture input and output data from an DC motor with an encoder using an Arduino Uno
2. How to use least squares to develop a mathematical model based on known and measured input and output data

4.1 Introduction

- The DC motor will be subject to a voltage step input, with the revolutions per minute (RPM) output measured using a hall effect sensor
- Based on the known/measured input-output data, a black-box model is to be developed (i.e., not considering the underlying physics of the motor) – with the model then used to develop the controller



- Encoder A phase
- Encoder B phase
- Encoder ground
- Encoder power
- Motor power
- Motor ground

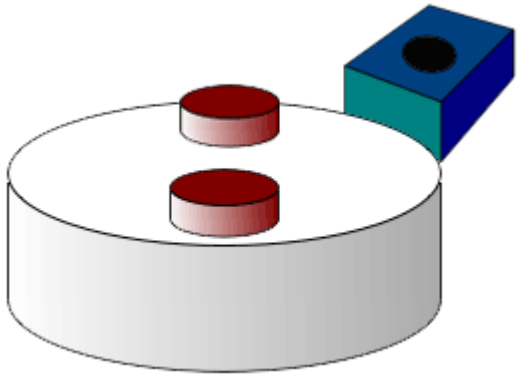
$$G_a(s)$$

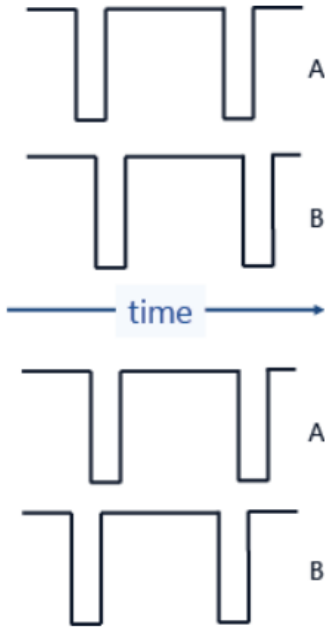
input $V(s)$ → $\frac{K}{(Ls+R)(Js+b)+K^2}$ → output $\omega(s)$

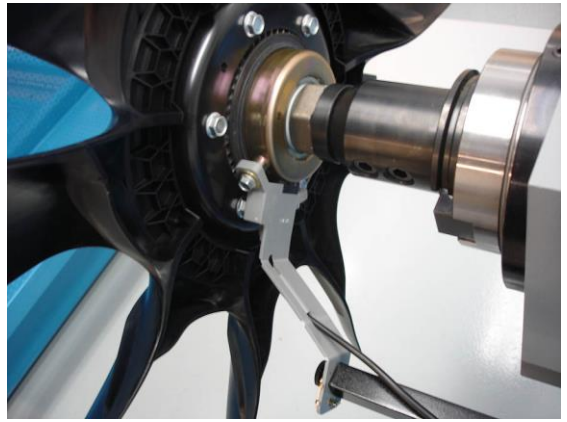
4.1 Introduction

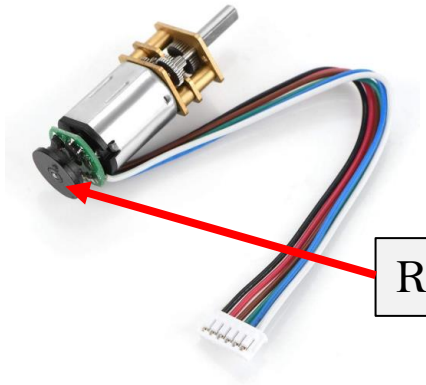
- A Hall effect sensor detects a magnetic field
- The output voltage of a Hall sensor is directly proportional to the strength of the field. It is named for the American physicist Edwin Hall.
- By detecting the passing of a magnet, the Hall effect sensor generates a pulse-train output

- Hall effects sensors are commonly used to time the speed of wheels and shafts, such as for internal combustion engine ignition timing, tachometers and anti-lock braking systems
- Hall effect sensors are used in brushless DC electric motors to detect the position
- Employing an offset Hall effect sensor can generate a quadrature output for determining the direction of the motion





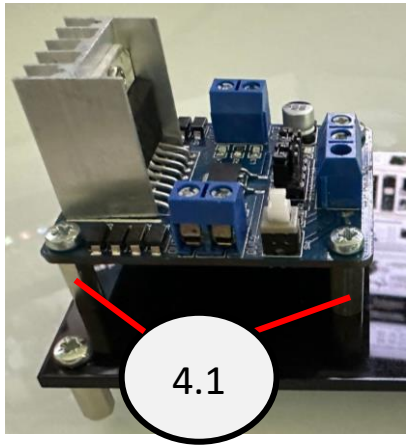


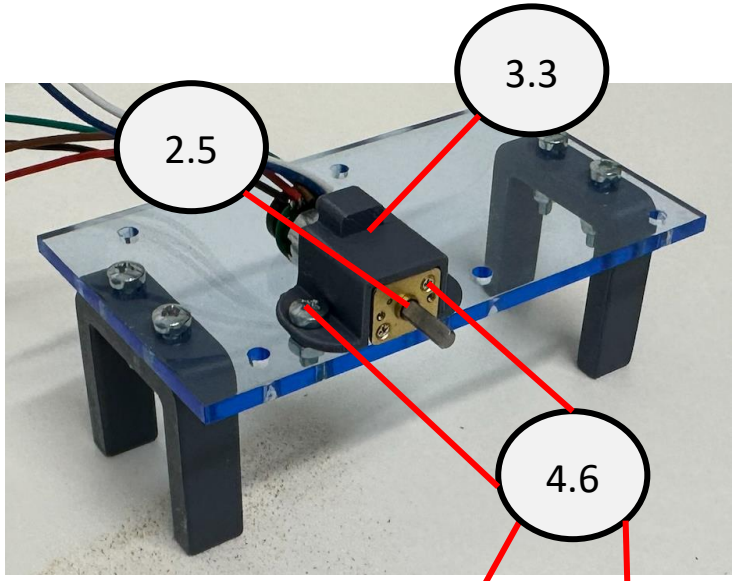


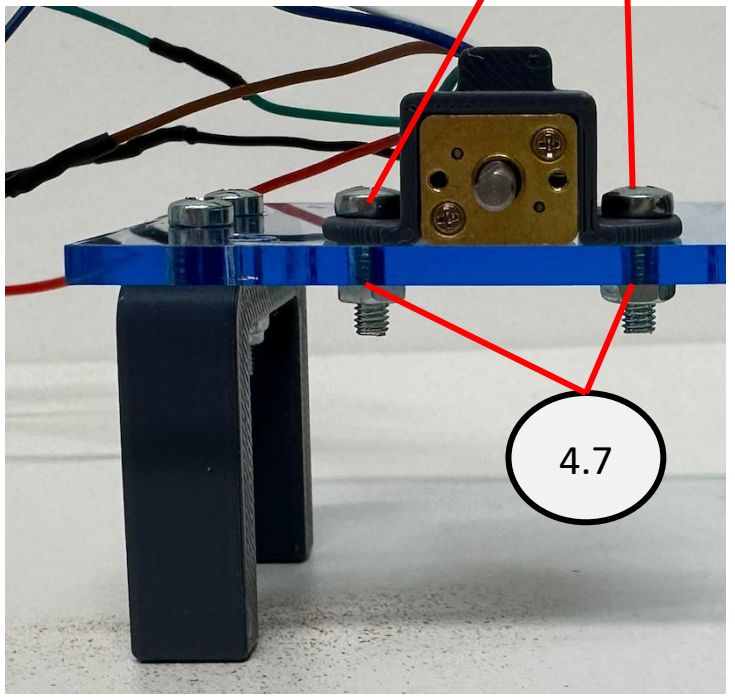
Resolution of 12 impulses per revolution

4.1 Introduction

- Set-up the CLB rig as illustrated to the right (the same set-up as used in CLB-3.3), using the following components:
  - 2.3: L298N dual H-bridge motor driver
  - 2.5: Brushed geared DC motor with encoder
  - 3.3: DC motor mount
  - 4.1: 4 x hex threaded spacer, 12mm, M3
  - 4.4: 8 x bolt, 6mm, M3
  - 4.6: 2 x bolt, 16mm, M3
  - 4.7: 2 x nuts, M3



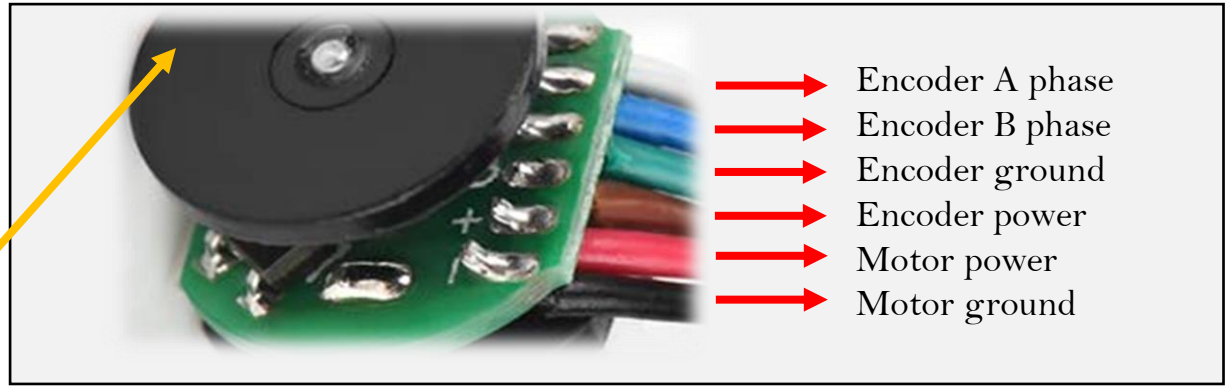


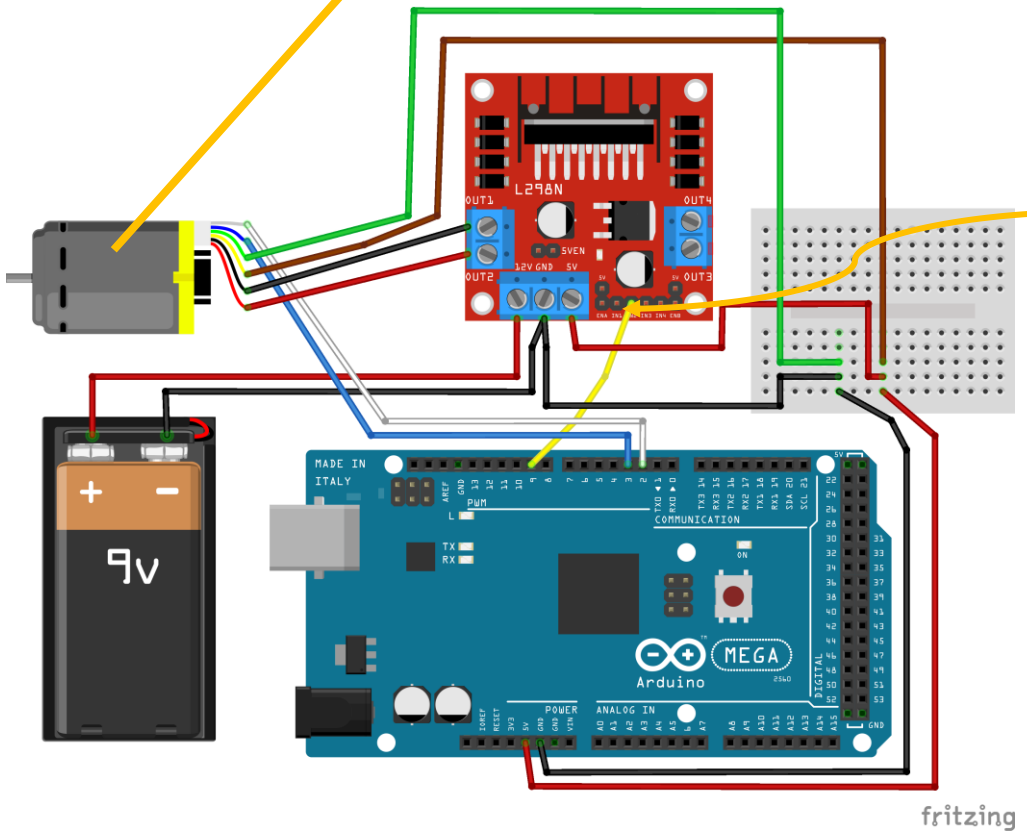


4.2 Hardware

- The task involves connecting a H-bridge and DC motor with an encoder to an Arduino
- Required hardware for the exercise:
  i. Supported Arduino Mega 2560 board
  ii. USB cable
  iii. H-bridge
  iv. 6V DC motor with encoder
  v. 9V battery
  vi. 9V power jack
  vii. Various wires
- Note that this initial part is the same as DC motor with an encoder (i.e., CLB-3.3)

Circuit diagram given here varies ever so slightly to the motor we are using

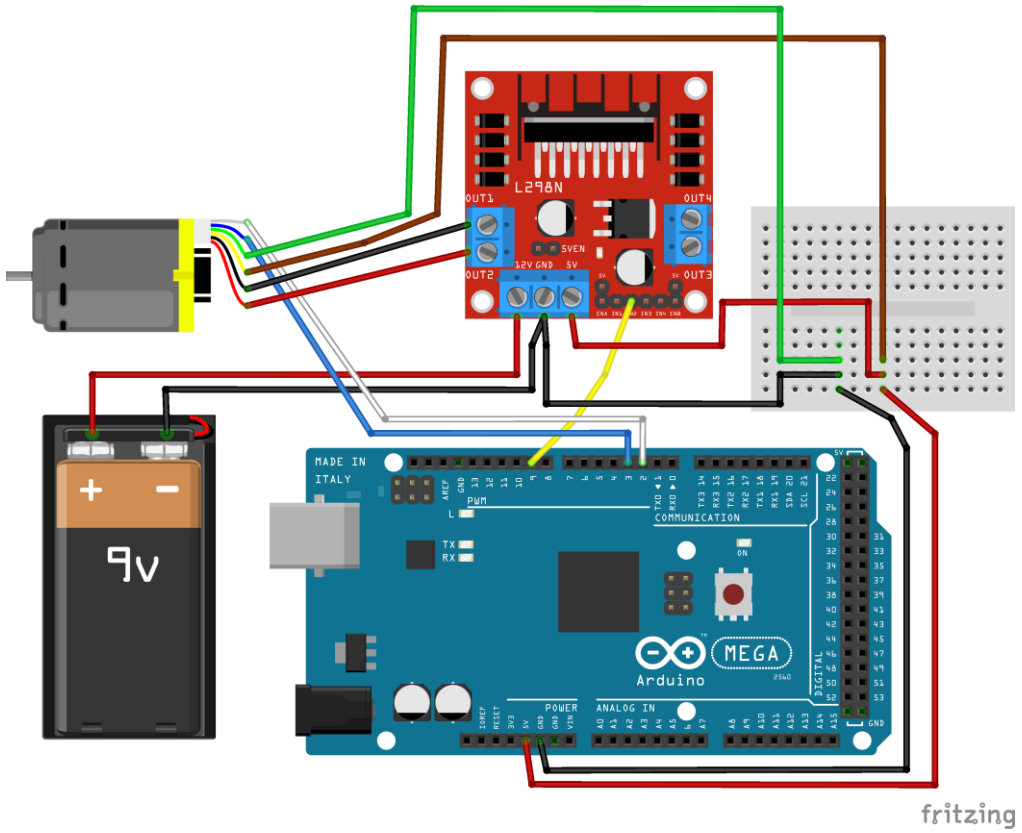




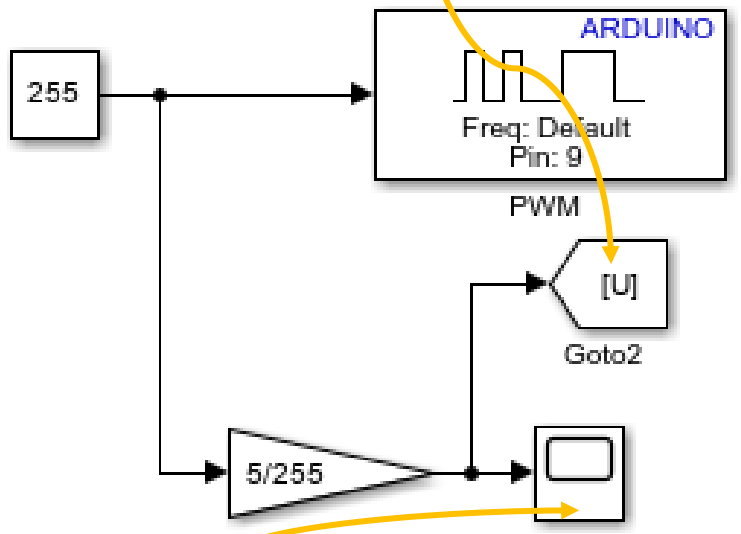
Note that 'In1' and 'In2' appear in a different position on the H-bridge to the one being used here

4.3 Algorithm Design

- A 5 Volt step input is applied to the DC motor



'Goto' blocks are used to capture the input voltage step data (and on the next Slide to capture the speed output data)



Within the 'Scope' block 'Configuration Properties', click on 'Logging' and select 'Log data to workspace' and 'Save format' to 'Structure with time'

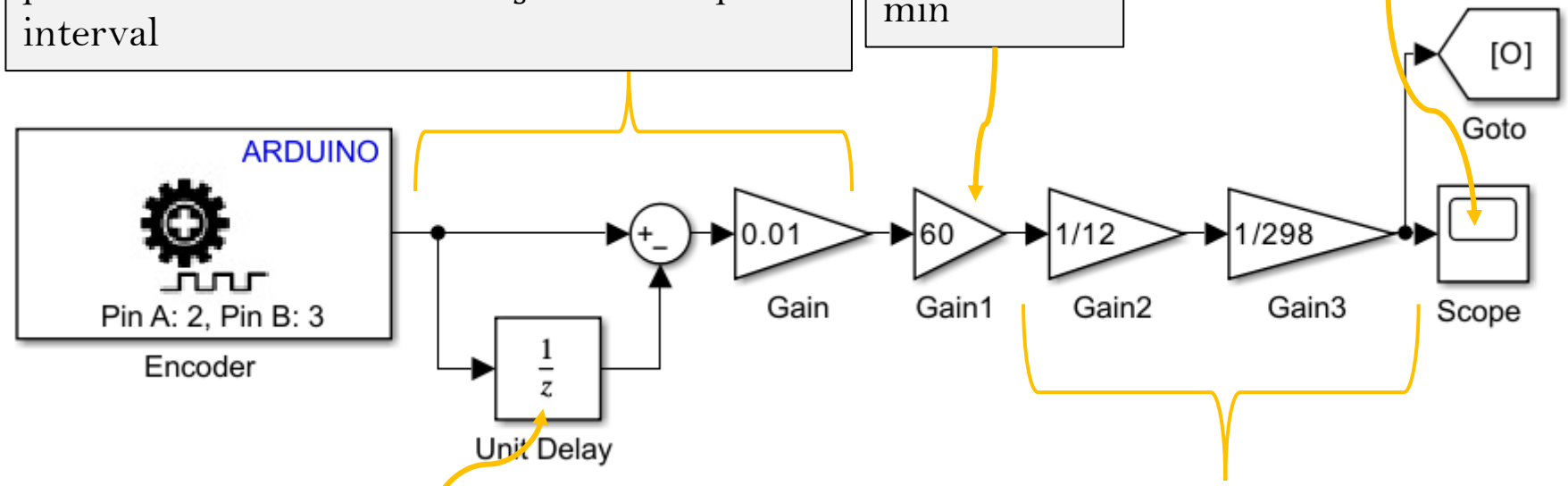
4.3 Algorithm Design

- The encoder on the DC motor is now used to estimate the motor speed based on the encoder counts
- The encoder counts indicate the motors position, hence the motors speed (rate of change of the position) over a specified sample interval can be determined

The change in position (speed) of the DC motor is determined using $\frac{y_k - y_{k-1}}{T_s}$ - the Backward Euler approximation (this converts the signal from counts/second to revolutions/second), where y_k is the current position measurement, y_{k-1} is the previous position measurement and T_s is the sample interval

A gain block is used to convert the signal from revolutions/sec to revolutions/min

Within the 'Scope' block 'Configuration Properties', click on 'Logging' and select 'Log data to workspace' and 'Save format' to 'Structure with time'



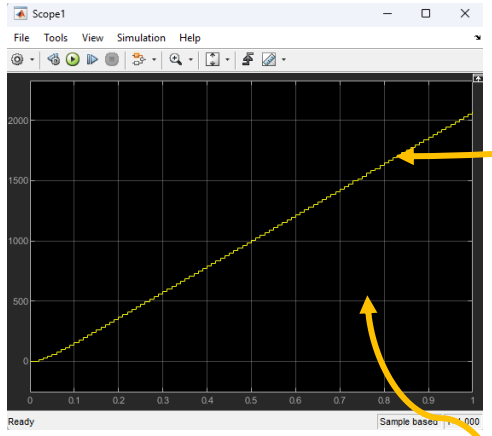
Since the past values of a sampled signal exist, it is normal to use z^{-1}, where z^{-1} means shift to $k-1$, with this given by the following:

$$z^{-1}y_k = y_{k-1}$$

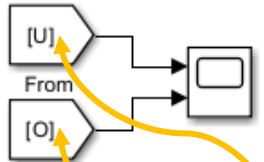
A gain block is used to account for the gear ratio (i.e., 1:298) and the encoders 12 impulses per revolution

4.3 Algorithm Design

- Before running the algorithm design on the Arduino, ensure the following are set:
  i. Run time of 1 second
  ii. Sample interval of 0.01 second

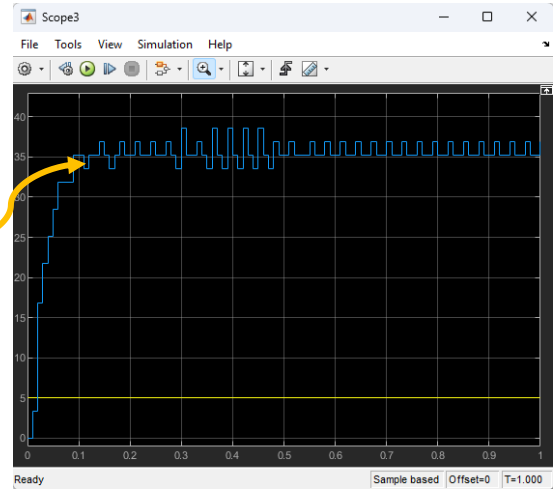


The data observed from the scope illustrates counts/second from the DC motor encoder, with this increasing when the DC motor is rotating in the clockwise direction



The three gain terms from the previous slide are 'lumped' together

From observation of the output data (revolutions per minute), the response takes the form of a 1st order transfer function



'From' blocks are used to capture the input voltage step data and the speed output data

4.4 Least Squares

- Using the known/measured input/output data, Least squares is used to develop a mathematical model based on initial observations, i.e., the response being that of a 1st order transfer function
- The following code will 'pick up' the data from the two scopes and put this into vector form

```
clc; close all;
%% Load measured input/output data

y = ScopeData.signals.values;
y = reshape(y,length(y),1);
t = ScopeData.time;
u = ScopeData2.signals.values;
u = reshape(u,length(u),1);

%% Linear least squares to estimate discrete-time model parameters
Phi=[-y(1:end-1) u(1:end-1)];
Y=y(2:end);
theta=inv(Phi'*Phi)*Phi'*Y

%% Simulate discrete-time difference model
N=length(y);
y_sim=zeros(N,1);
for k=2:1:N
y_sim(k)=[-y_sim(k-1) u(k-1)]*theta;
end
```

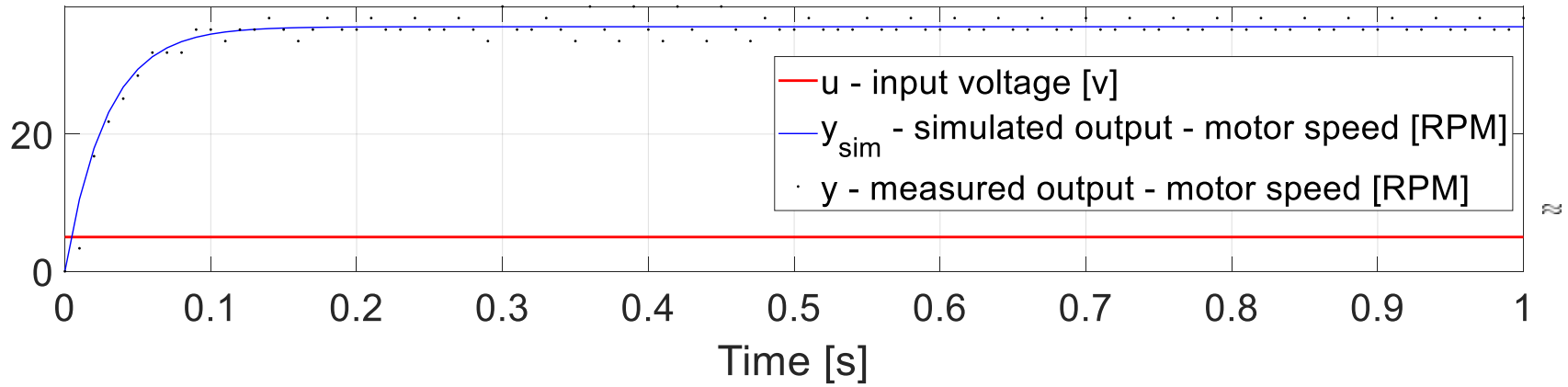
4.4 Least Squares

- The actual/physical output is plotted against the estimated model output
- The model fit is evaluated, comparing the measured and simulated output data

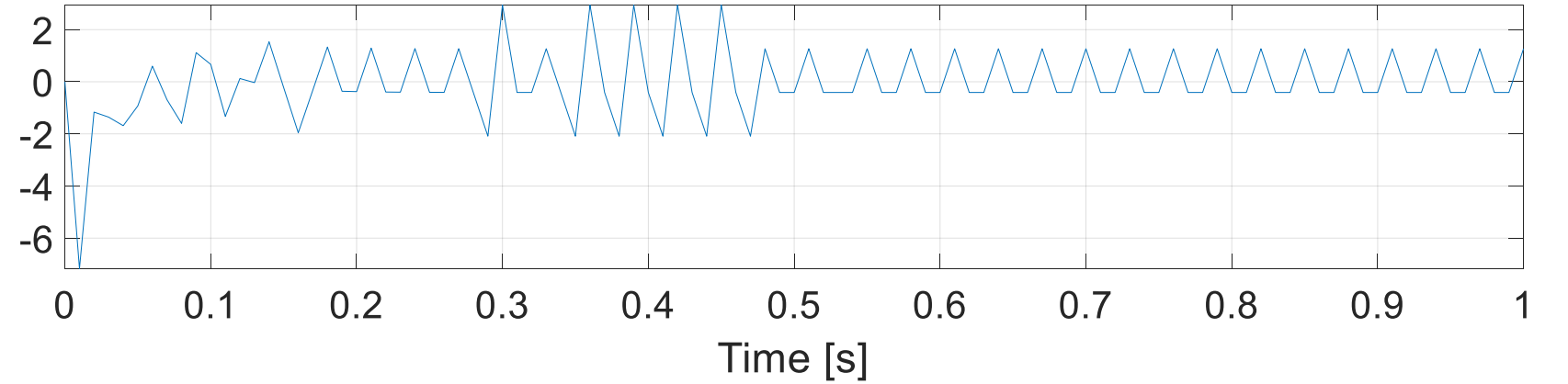
```
%% GRAPHICAL OUTPUTS (MEASURED AND ESTIMATED MODEL)
subplot(211)
hh=plot(t,u, t,y_sim, t,y);
grid on; axis tight;
legend('u - input voltage [v]','y_{sim} - simulated output - motor speed [RPM]','y - measured output - motor speed [RPM]');
xlabel('Time [s]')
set(gca,'FontSize',30);
set(hh(1),'LineWidth',2,'Color','red','LineStyle','-');
set(hh(2),'LineWidth',1,'Color','blue','LineStyle','-');
set(hh(3),'LineWidth',2,'Color','black','LineStyle','none','Marker','.');

%% EVALUATE MODEL FIT (Average error between measured and simulated output)
AverageError = mean(y-y_sim);
subplot(212)
plot(t,y-y_sim);
grid on; axis tight;
title(['Average modelling error = ',num2str(AverageError,2),'[RPM]']);
xlabel('Time [s]')
set(gca,'FontSize',30);
```

4.4 Least Squares

- The graphical outputs are shown
- From visually inspection, it is clear the average modelling error is very small



Average modelling error = -0.03[RPM]



4.5 Transfer Function

- The time constant, τ, and system gain , K, for the 1st order transfer function can be captured using MATLAB, i.e.,

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1}$$

- Using the MATLAB script, the following is determined:

$$G(s) = \frac{\dot{\theta}(s)}{V(s)} = \frac{7.13}{0.029s + 1} \left[\frac{RPM}{V}\right]$$

The form here is given by:

$$\frac{b_0}{z - a_1}$$

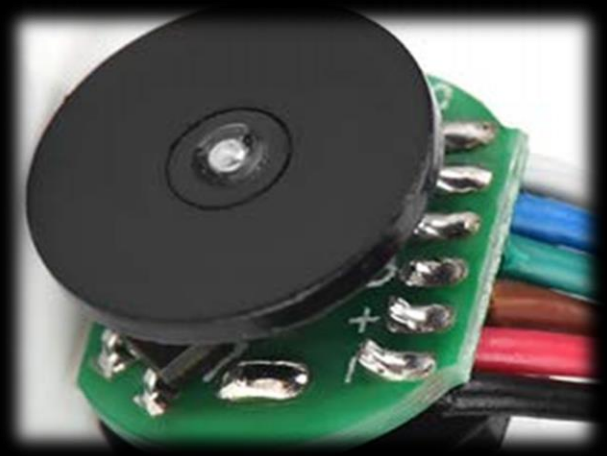
```
%% Transfer function, time constant and system gain
a1 = theta(1);
b0 = theta(2)
Gd = tf([b0],[1 a1],0.01) %discrete-time transfer function
G = d2c(Gd)

b = G.Denominator{1,1};
b = b(1,2);
Tau = 1/b % 1 time constant (63.2% of yss)
FiveTau = 5*Tau % 5 time constants (99% of yss)
a = G.Numerator{1,1};
K = a(1,2)/b %system gain
```

```
Gd =
   2.105
  ----------
  z - 0.7047

Sample time: 0.01 seconds
Discrete-time transfer function.
Model Properties

G =
   249.5
  ---------
  s + 34.99

Continuous-time transfer function.
Model Properties

Tau =

   0.0286


FiveTau =

   0.1429


K =

   7.1292
```

Contents

Least Squares for Parameter Estimation of a DC Motor

4.6 Summary



- The hardware required for capturing the input and output data from an DC motor with an encoder has been presented
- Least squares algorithm has then been used to develop discrete-time and continuous-time mathematical models of the DC motor
- The error of the modelling has been determined, i.e., difference between the DC motor output data and the DC motor mathematical model output