DC Servo Motor

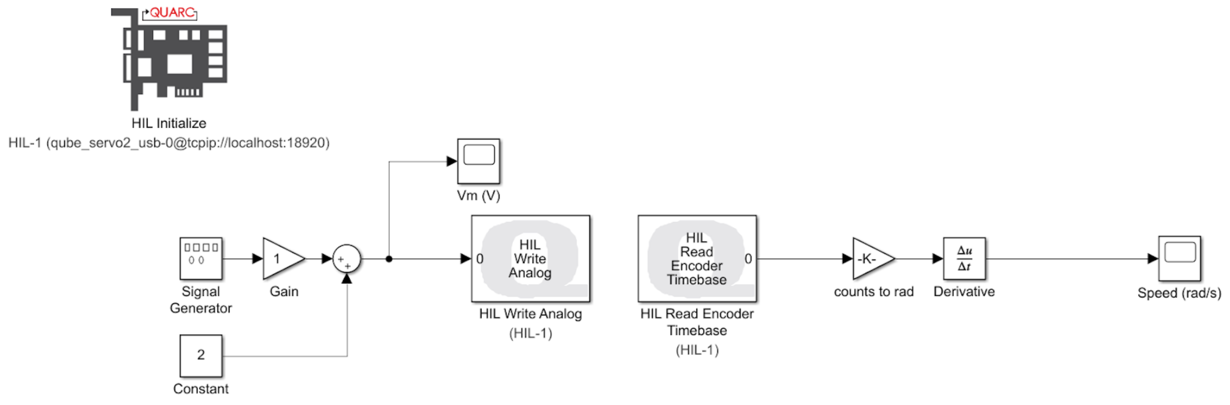
McMaster University
EE3EY4 Electrical Systems Integration Project - Winter 2021

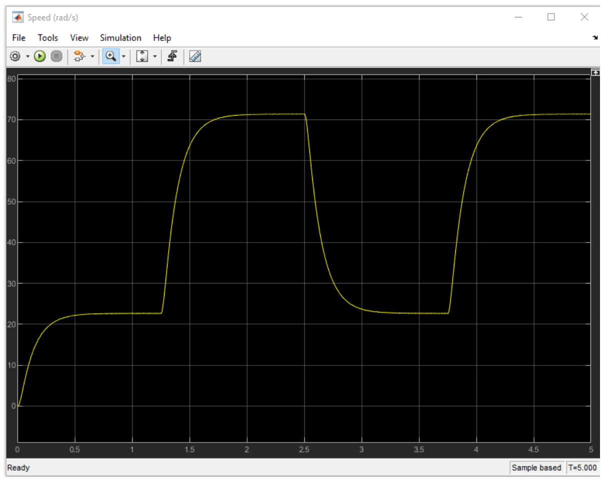
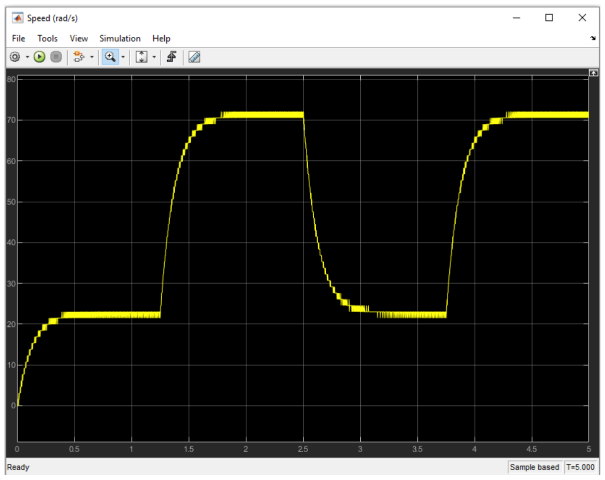
1. Filtering

(1.1)

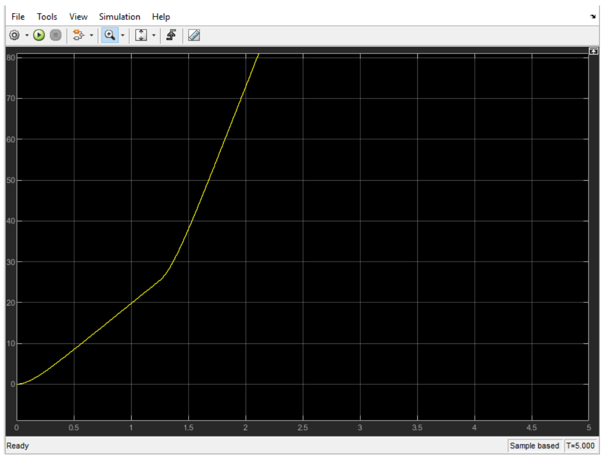
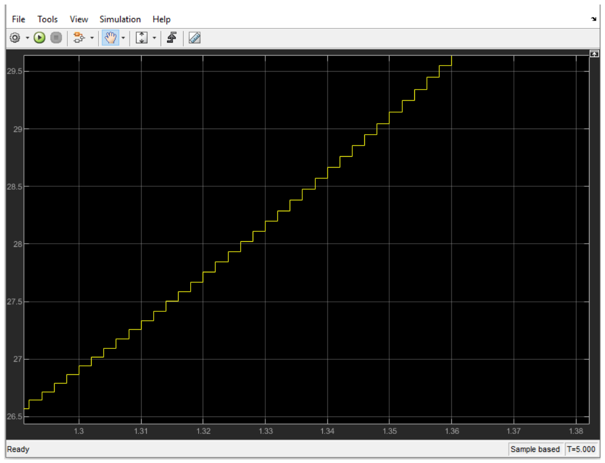
The encoder gain converts the encoder counts to radians. The encoder outputs 2048 counts/revolution in quadrature mode, that is, 512 lines/revolution. As such, to convert a count to radian, the count must be multiplied by 2π/(2048) or 2π/(512*4), which is the encoder gain.

(1.2)

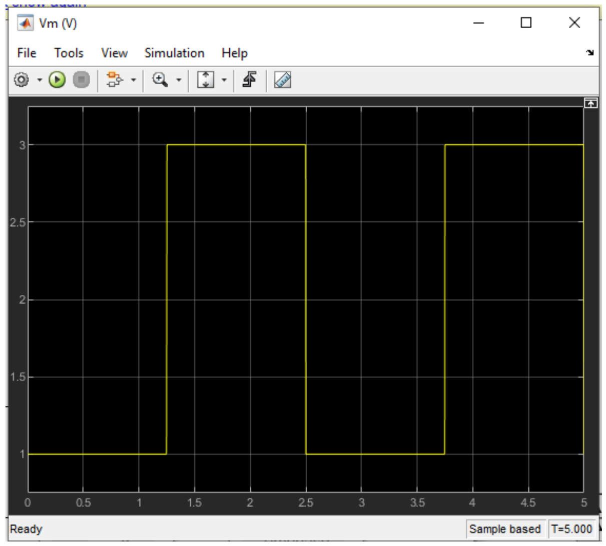
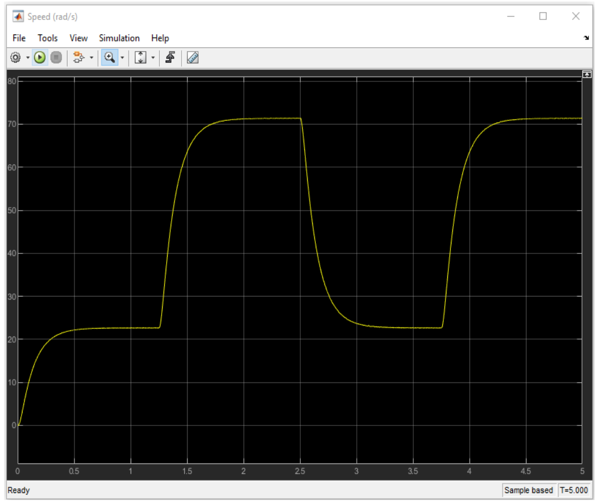


With Transfer Function (LPF):	Without Transfer Function (No filter):
	

After removing the LPF, there was a definite increase in noise, and the output appears to be very choppy. This is due to the quantization effect of the encoder since it converts signals to discrete signals. We can clearly see the quantization effect if we look at the position before applying the derivative to get angular velocity. This effect is amplified after applying the derivative, which we can see above when we measure the Speed without the transfer function. This happens because the derivative of a vertical line gives a very large value while a horizontal line gives a small value producing the sinusoidal noises we see.

Position	Magnified Position to show Quantization Error
	

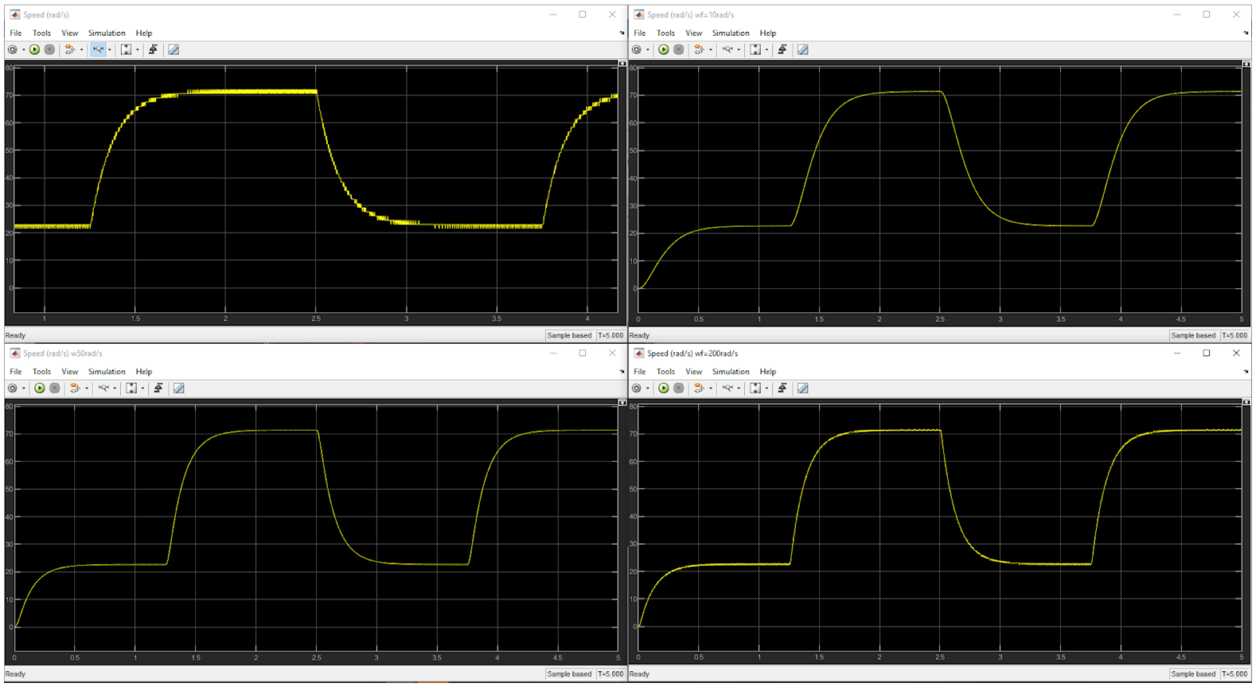
(1.3)

Motor Voltage	Speed filtered encoder-based speed response
	

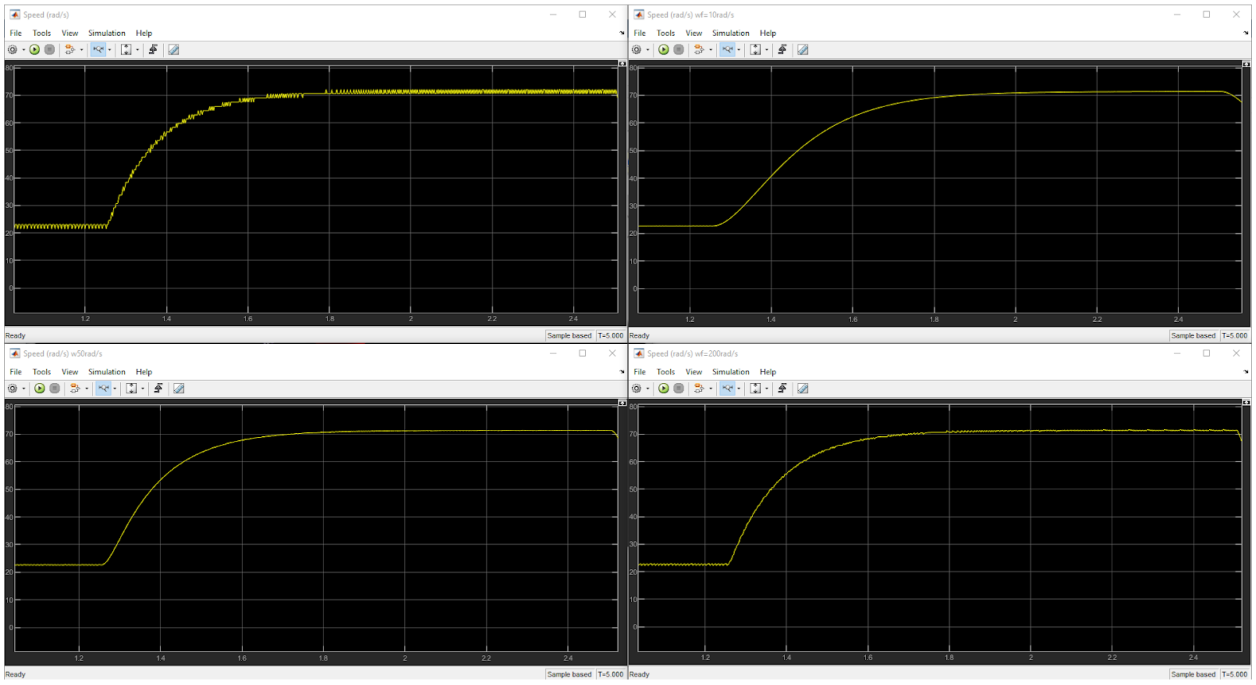
The transfer function definitely filters out the noises produced by the encoder, providing a clearer and smoother signal. This happens because it acts as a Low Pass Filter which dilutes the high-frequency response to the system.
The cut-off frequency of the low-pass filter, based on the transfer function, is 50 rad/s. To convert it to Hz, divide 50 rad/s by 2π, which gives 25/π Hz or approx. 7.96 Hz.

(1.4)

Speed of motors with and without LPF at different cutoff frequencies:



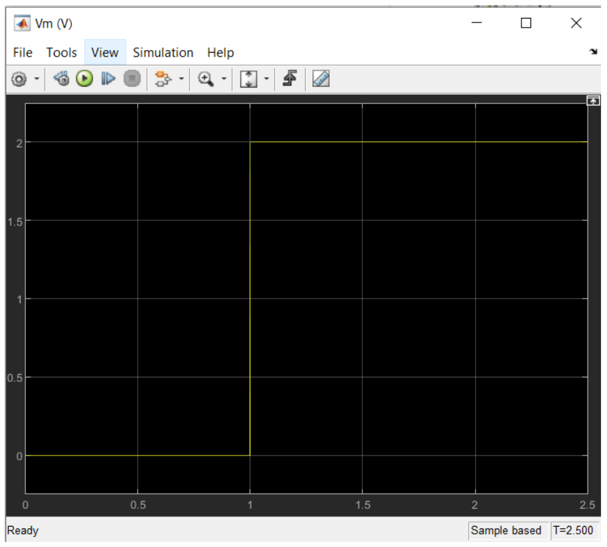
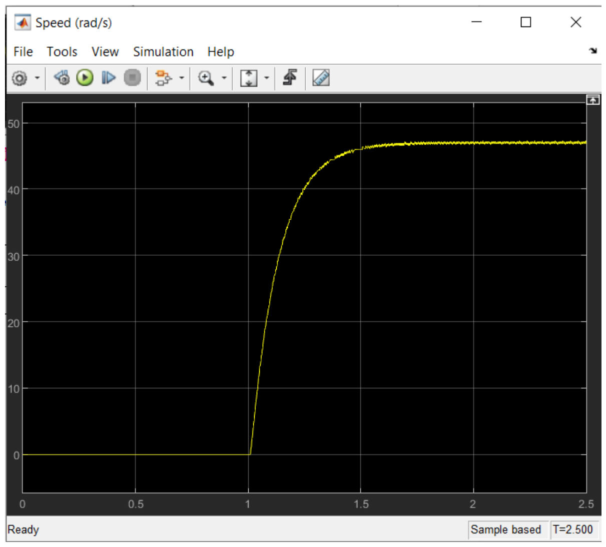
Graphs zoomed in:



When we add a cutoff frequency, for example, 50rad/s, the signal becomes smoother and the noise in the signal diminishes. However, when making the signal too small, the filter starts to remove much more frequencies than needed, thereby causing it to cut off actual/useful data of the signal. In this case, making the cutoff frequency 10rad/s delays the time it takes for the signal to reach around 70rad/s. This means that decreasing the cutoff frequency increases the signal's quality, but decreasing it too much will start cutting off or delaying the signal. Increasing the cutoff frequency only allows more frequencies to pass through. It also approaches the original signal's shape more, but at a certain point, it just increases the noise because it could not filter all of it. In this case, when setting the cutoff frequency to 200rad/s, the signal looks like the 50rad/s but with a lot more noise. What this tells us is that increasing the cutoff frequency will allow the signal to be closer to its appropriate shape, however, by increasing it too much, the noise will increase. Finally, what our results tell us is that our current cutoff frequency of 50rad/s is just right.

2. Step Response Modeling

(2.1)

Applied voltage	Measured speed response
	

(2.2)

$$K = \frac{\Delta y}{\Delta u} \qquad \Delta y = y_{ss} - y_0 \text{ and } \Delta u = u_{max} - u_{min}$$

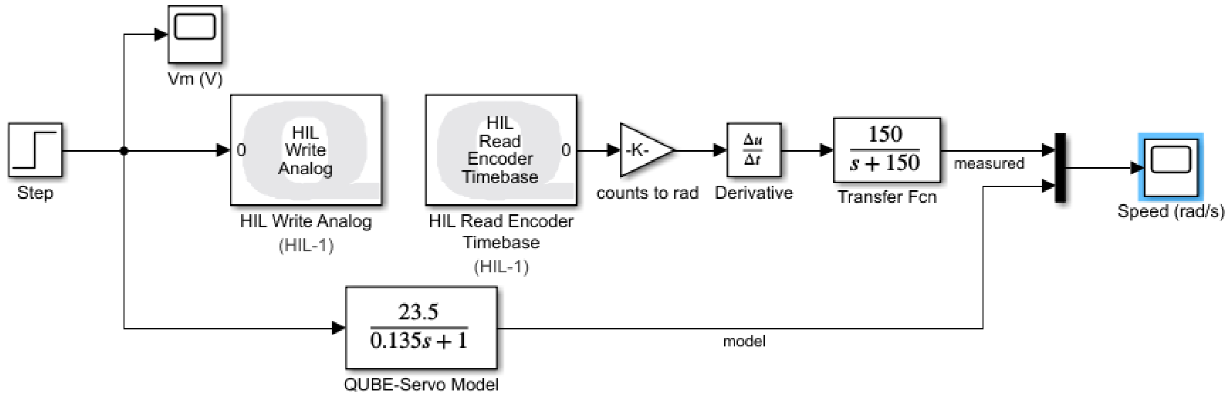
From the plots, the measured speed response reaches 47 rad/s from 0 rad/s and u_{max} is 2, with u_{min} equaling 0.

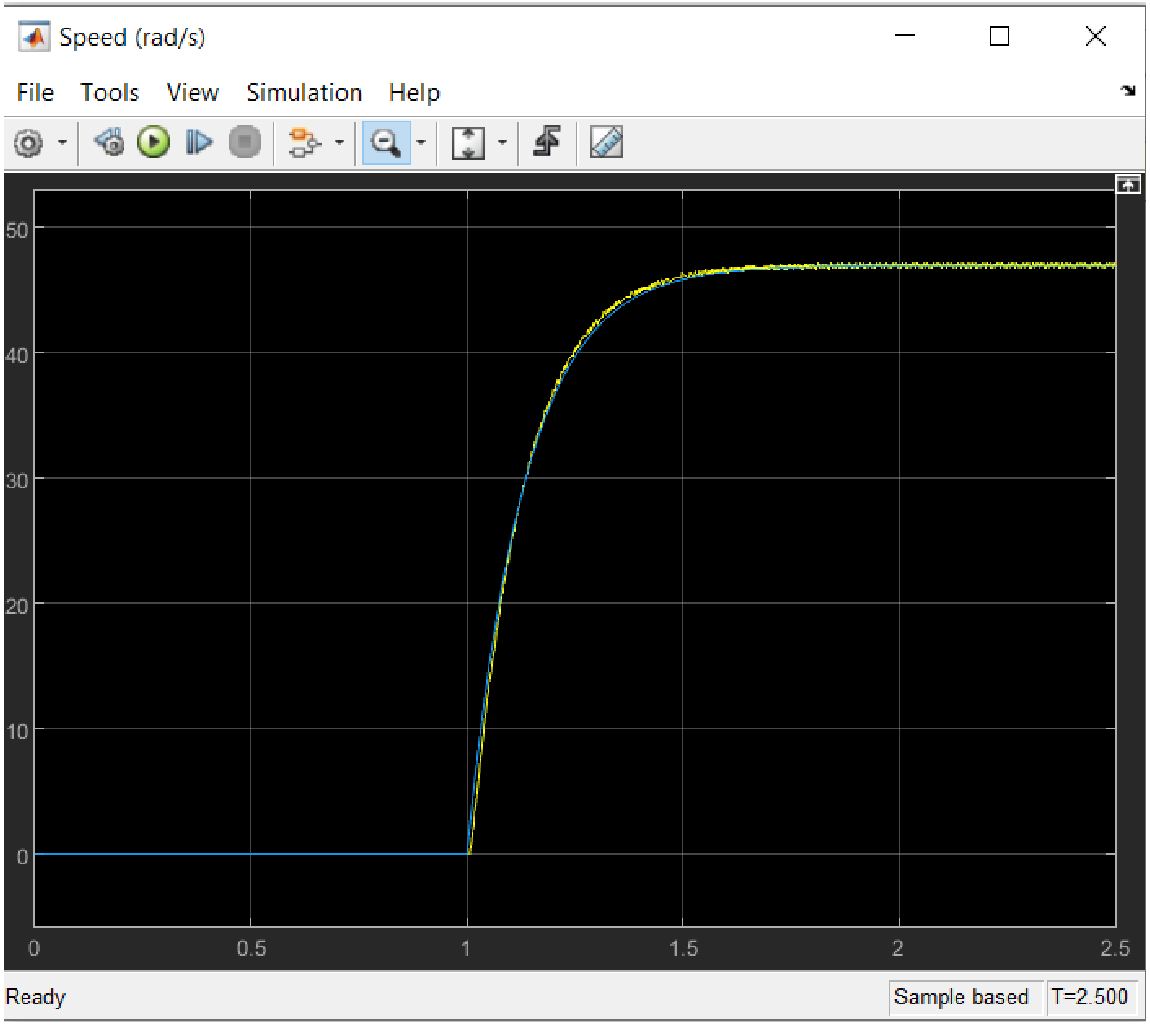
Steady-state gain, **K = Δy / Δu = 47/2 = 23.5 rad/(V s)**

Time response, τ, is the time for the system to respond to the application of a step input to reach ≈63.2% of its steady-state response.

τ = t_1 - t_2 = 1.1350 -1 = 0.1350s

(2.3)
Checking the values of K and τ using the model:





From the above plot, it can be seen that the voltage to speed transfer function is similar to the actual transfer function with some simple discrepancies. The derived transfer function responds faster in the beginning but takes longer to reach saturation. This could be because K and τ values are close approximations of the actual values.

3. Block Diagram Modeling

(3.1)

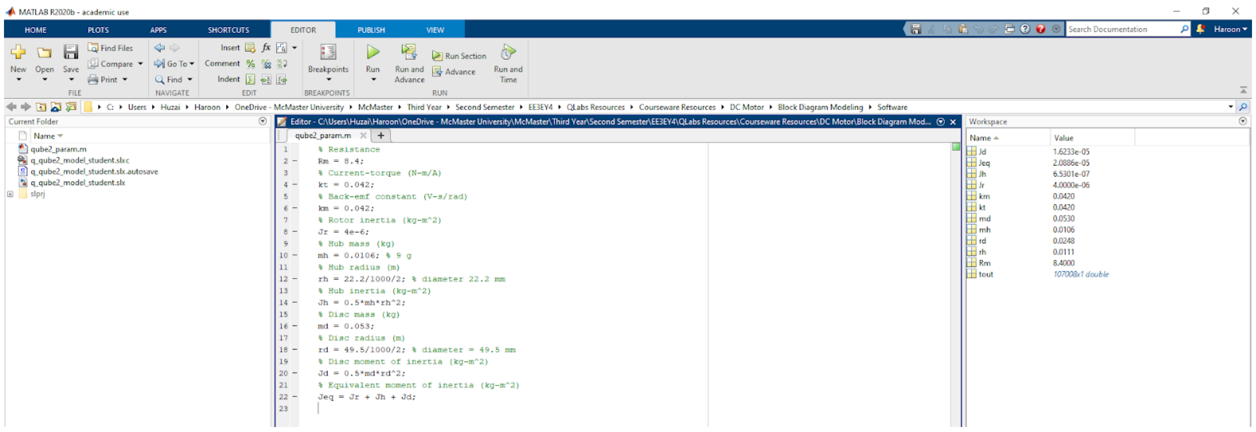
$$J_{eq} = J_m + J_h + J_d, \; J_d = \tfrac{1}{2} m_d {r_d}^2$$

$$J_{eq} = J_m + J_h + \tfrac{1}{2} m_d {r_d}^2$$

$$J_{eq} = 4.0 \times 10^{-6} kg \cdot m^2 + 0.6 \times 10^{-6} kg \cdot m^2 + \tfrac{1}{2} \cdot 0.053 kg \cdot (0.0248 m)^2$$

$$J_{eq} = 2.09 \times 10^{-5} kg \cdot m^2$$

(3.2)

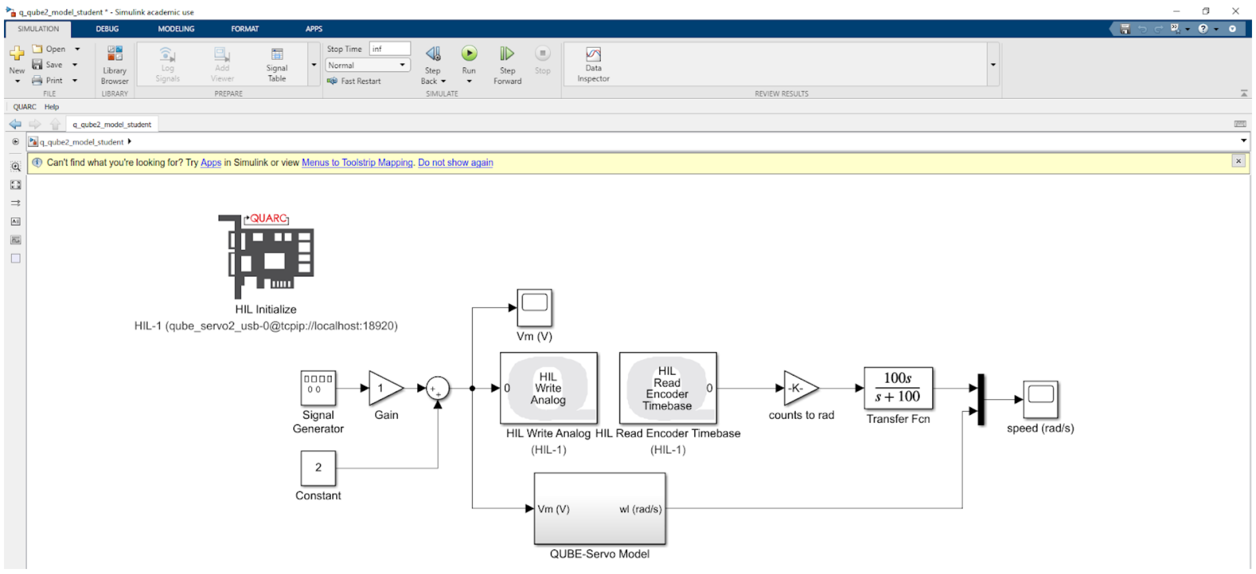


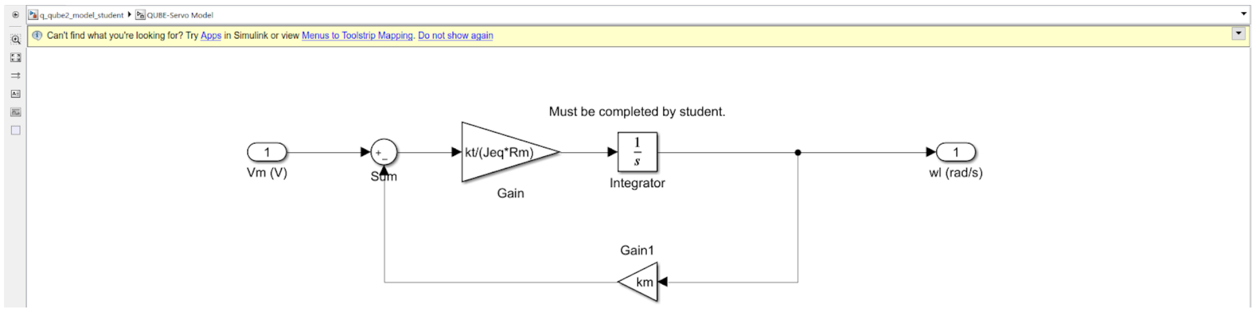
qube2_param.m

```
% Resistance
Rm = 8.4;
% Current-torque (N-m/A)
kt = 0.042;
% Back-emf constant (V-s/rad)
km = 0.042;
% Rotor inertia (kg-m^2)
Jr = 4e-6;
% Hub mass (kg)
mh = 0.0106; % 9 g
% Hub radius (m)
rh = 22.2/1000/2; % diameter 22.2 mm
% Hub inertia (kg-m^2)
Jh = 0.5*mh*rh^2;
% Disc mass (kg)
md = 0.053;
% Disc radius (m)
rd = 49.5/1000/2; % diameter = 49.5 mm
% Disc moment of inertia (kg-m^2)
Jd = 0.5*md*rd^2;
% Equivalent moment of inertia (kg-m^2)
Jeq = Jr + Jh + Jd;
```

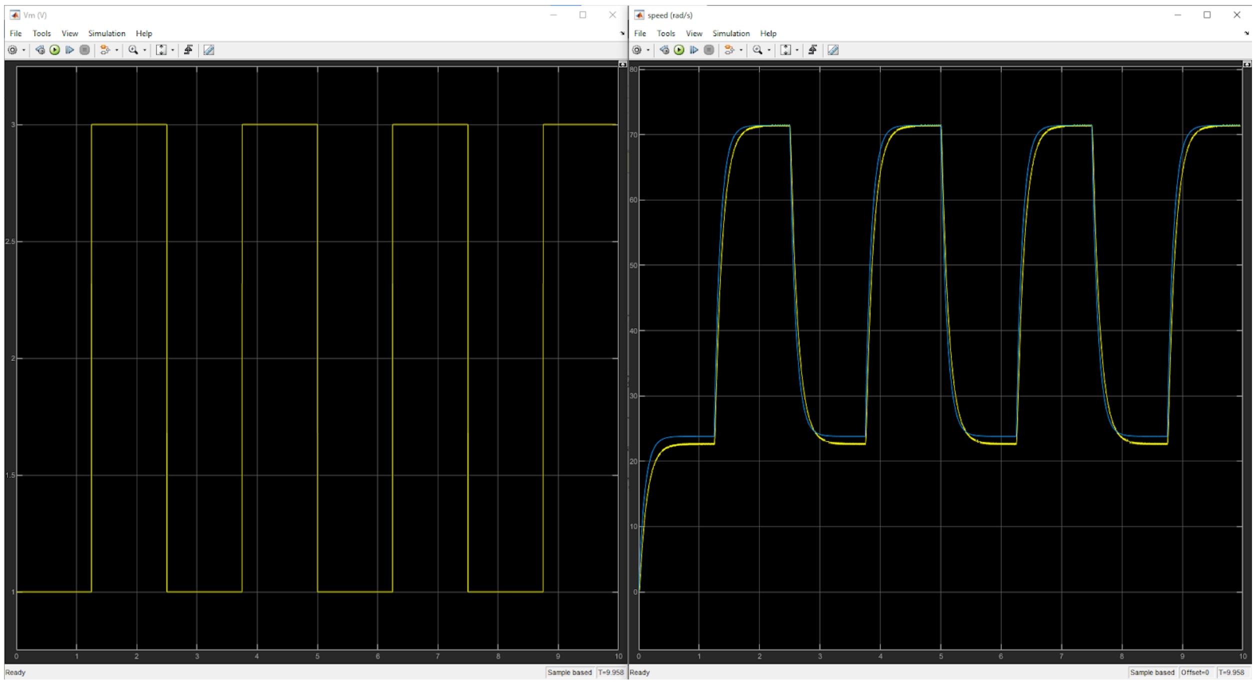
Workspace

Name	Value
Jd	1.6233e-05
Jeq	2.0886e-05
Jh	6.5301e-07
Jr	4.0000e-06
km	0.0420
kt	0.0420
md	0.0530
mh	0.0106
rd	0.0248
rh	0.0111
Rm	8.4000
tout	*107008x1 double*




q_qube2_model_student ▸ QUBE-Servo Model
Can't find what you're looking for? Try Apps in Simulink or view Menus to Toolstrip Mapping. Do not show again
Must be completed by student.
1
Vm (V)
Sum
kt/(Jeq*Rm)
Gain
1/s
Integrator
1
wl (rad/s)
Gain1
km

(3.3)



The model represents the QUBE-Servo 2 fairly well. It does have a faster response/less of a delay than the QUBE-Servo 2 and a higher driving force i.e. a higher stabilizing speed at 1v, but these changes are minute and the model still captures the behavior of the motor.

(3.4)
A reason for these differences could be that the parameters we were given may not be the only things that actually affect the motor. For one thing, the equations we use do not take the friction into account nor the Stribeck effect. A reason why the actual motor has a slower response and a lower speed at lower voltages can be attributed to friction, the delay is being caused by static friction and the lower speed at low voltages is being caused by kinetic friction.

(3.5)

$$(1)\ J_{eq}\overline{\omega}_m(t) = \tau_m(t)$$

$$(2)\ \tau_m(t) = k_t i_m(t)$$

$$(3)\ i_m(t) = \frac{v_m(t) - k_m\omega_m(t)}{R_m}$$

Sub (2) into (1)

$$(4)\ J_{eq}\overline{\omega}_m(t) = k_t i_m(t)$$

Sub (3) into (4)

$$J_{eq}\overline{\omega}_m(t) = k_t \frac{v_m(t)-k_m\omega_m(t)}{R_m}$$

$$\frac{J_{eq}R_m}{k_t}\overline{\omega}_m(t) = v_m(t) - k_m\omega_m(t)$$

$$v_m(t) = k_m\omega_m(t) + \frac{J_{eq}R_m}{k_t}\ \overline{\omega}_m(t)$$

$\overline{\omega}_m(t) = \frac{k_t}{J_{eq}R_m}(v_m(t) - k_m\omega_m(t))$ //use this for modeling the block diagram for part (3.2)

(3.6)

Laplace Transform

$$V_m(s) = k_m\Omega_m(s) + \frac{J_{eq}R_m}{k_t} s\ \Omega_m(s) - \frac{J_{eq}R_m}{k_t}\ \Omega_m(0),\ zero\ I.C.$$

$$V_m(s) = k_m\Omega_m(s) + \frac{J_{eq}R_m}{k_t} s\ \Omega_m(s)$$

$$V_m(s) = (k_m + \frac{J_{eq}R_m}{k_t} s)\ \Omega_m(s)$$

$$\Omega_m(s) = \frac{V_m(s)}{\frac{J_{eq}R_m}{k_t}s+k_m}$$

$$\Omega_m(s) = \frac{V_m(s)}{\frac{J_{eq}R_m}{k_t}s + k_m}$$

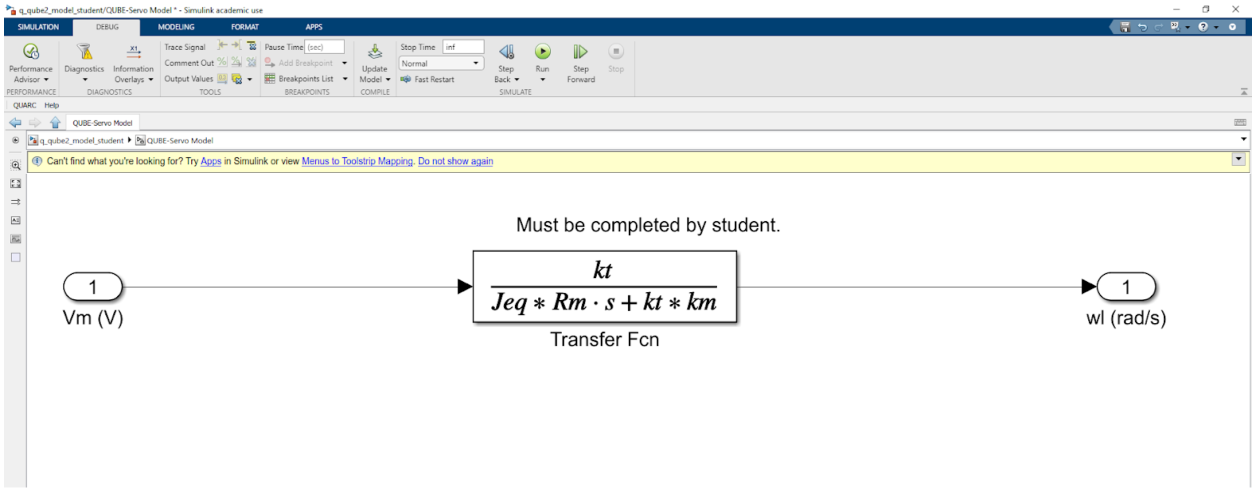
$$\Omega(s) = \frac{k_t}{J_{eq}R_m s+k_t k_m} V_m(s)$$

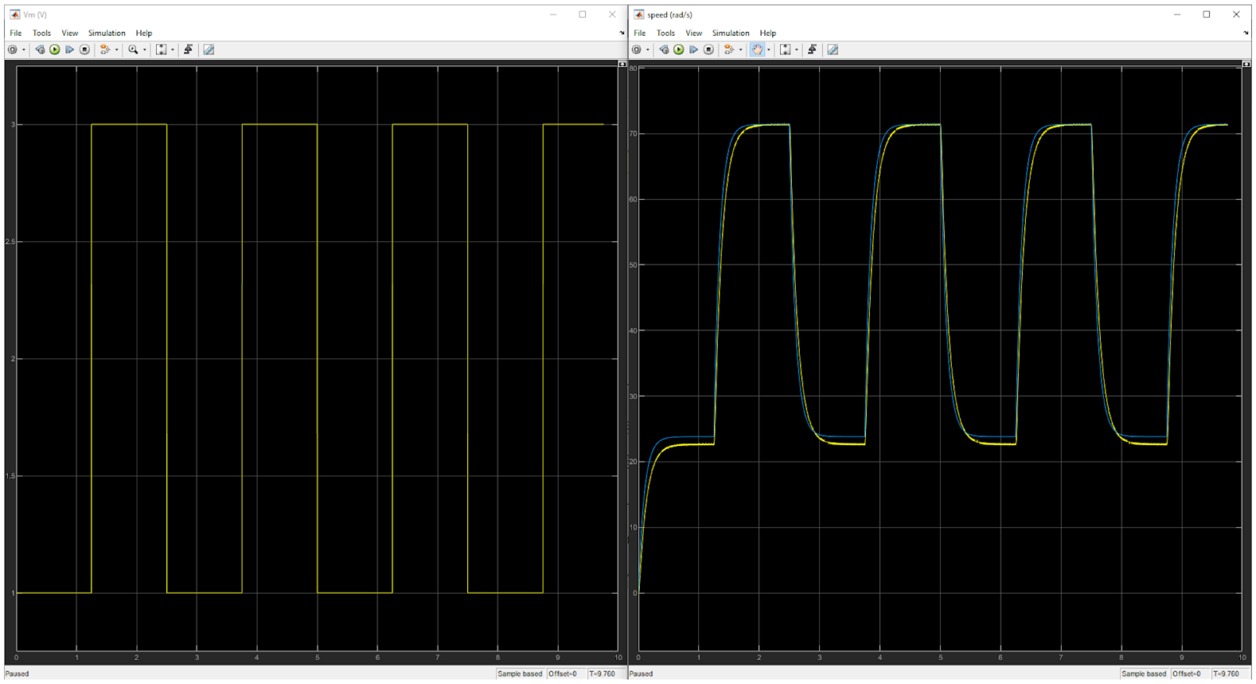
$$\frac{\Omega_m(s)}{V_m(s)} = \frac{1/k_m}{\frac{J_{eq}R_m}{k_t k_m}s+1} \simeq \frac{23.8}{0.1s+1}$$

Step Response Modelling Transfer function: $\frac{23.5}{0.135s+1}$

The transfer function we get for our model using parameters is very close to the one we got when using step response modeling. The difference we get while using parameters of the motor is that the K is slightly larger and τ is a little bit smaller. This will result in this model being slightly larger vertically and slightly faster.

(3.7)


Must be completed by student.
$\frac{kt}{Jeq * Rm \cdot s + kt * km}$
Transfer Fcn
Vm (V)
wl (rad/s)



First, this model looks exactly identical to our model in (3.3). This makes sense because we were trying to make a transfer function that would model the same way as the block diagram model, which we can see that we succeeded. Now we said that this model should be faster and larger vertically than the step response model, which is much closer to the actual results, so because of that, we can assume that this model is faster and larger vertically than the actual angular speed of the motor. In this case, the model is definitely faster, however, the stabilizing speed at 1V i.e., u_{min}, is slightly higher, which in turn would mean that k is lower. This was unexpected but after looking into this we realized that the assumption that we made was incorrect since the step response model may not be as accurate as we expected for different voltages.

4. Parameter Estimation

(4.1)

The equation for motor current, i_m(t) is:

$i_m(t) \; = \; \frac{v_m(t) - k_m\omega(t)}{R_m}$, where ω*(t)* = 0 because the motor shaft is not rotating.

The reason for locking the shaft is to be able to ignore the k_mω(t) term which depends on the speed of the shaft's rotation. This would simplify the expression.

Rearranging for R_m gives:

$R_m = \frac{v_m}{i_m(t)}$, where v_m is a constant value

(4.2)

The equation for steady-state motor current, i_m(t) is:

$i_m(t) = \frac{v_m(t) - k_m\omega(t)}{R_m}$, the motor inductance was ignored for steady-state operation.

Rearranging for k_m gives:

$k_m(t) = \frac{v_m - i_m(t) \cdot R_m}{w_m(t)}$, where v_m is a constant value.

(4.3)

To estimate R_m, a constant voltage of 5V was applied while keeping the motor shaft stationary. This resulted in a current of 0.5177A.

Since the voltage and current is not time-varying in this case,

$R_m = \frac{5}{0.5177} = 9.658\Omega$

After repeating this same step for all the voltages below we get the following resistances.

Applied voltage:V_m (V)	Measured Current: I_m (A)	Resistance: R_m (Ω)
-5	-0.5178	9.656
-4	-0.4143	9.655
-3	-0.3107	9.655
-2	-0.2072	9.652
-1	-0.1036	9.652
1	0.1035	9.662
2	0.2071	9.657
3	0.3106	9.659

4	0.4142	9.657
5	0.5177	9.658

The average value of R_m = 9.6563Ω
The value of R_m from motor specification sheet = 8.4Ω
The discrepancy in the R_m could be due to some unknown parameters or internal systems. One of these unknown parameters is definitely friction. Like we mentioned before in (3.4) the friction has a definite roll that the parameters do not include. The friction of the shaft would also increase the power consumed, thereby also increasing the current which we could see if we calculated it with our parameters.

(4.4)
To estimate k_m, a constant voltage is applied, in this case 5V, while keeping the motor shaft stationary. This resulted in a current of 0.01557A and a rotational speed of 120.1rad/s. The resistance value R_m is already known - 8.4Ω
Since the voltage, current, and rotational speed are not time-varying, in this case;

$$k_m(t) = \frac{5-(0.1557)(8.4)}{120.1} = 0.0405\ V \cdot s/rad$$

After repeating this same step for all the voltages below we get the following.

Applied voltage: V_m (V)	Measured Speed: w_m (rad/s)	Measured Current: I_m (A)	Back-emf: k_m (V-s/rad)
-5	-120.1	-0.01567	0.0405
-4	-95.74	-0.01404	0.0405
-3	-71.38	-0.01231	0.0406
-2	-47.01	-0.01068	0.0406
-1	-22.64	-0.008954	0.0408
1	22.64	0.008852	0.0409
2	47.01	0.01058	0.0407
3	71.38	0.01221	0.0406
4	95.74	0.01394	0.0406
5	120.1	0.01557	0.0405

The average value of k_m = 0.0406 V/(rad/s)
The value of k_m from motor specification sheet = 0.042 V/(rad/s)
The discrepancy in the k_m value could be due to some unknown parameters or internal systems.

5. State Space Modeling

(5.1)

From (3.5)

$$v_m(t) = k_m\omega_m(t) + \frac{J_{eq}R_m}{k_t}\,\overline{\omega}_m(t)$$

$$\therefore\ v_m(t) = k_m\theta'_m(t) + \frac{J_{eq}R_m}{k_t}\,\theta''_m(t)$$

Rearrange for (5.2)

$$\frac{J_{eq}R_m}{k_t}\,\theta''_m(t) = v_m(t) - k_m\theta'_m(t)$$

$$\theta''_m(t) = \frac{k_t}{J_{eq}R_m}v_m(t) - \frac{k_tk_m}{J_{eq}R_m}\theta'_m(t)$$

(5.2)

n=2, m=2, r=1

$$\bar{x}(t) = Ax(t) + Bu(t) \qquad \bar{x}(nx1) = A(nxn)\bullet x(nx1) + B(nxr)\bullet u(rx1)$$

$$y(t) = Cx(t) + Du(t) \qquad y(mx1) = C(mxn)\bullet x(nx1) + D(mxr)\bullet u(rx1)$$

$$u(t) = v_m(t)$$

$$x_1 = \theta_m(t),\ x_2 = \overline{\theta}_m(t)$$

$$(1)\ \bar{x}_1 = x_2$$

$$(2)\ \bar{x}_2 = \frac{k_t}{J_{eq}R_m}v_m(t) - \frac{k_tk_m}{J_{eq}R_m}x_2$$

$$\bar{x}_2(t) = Ax(t) + Bu(t)$$

Therefore,

$$A = [0,\ 1; 0,\ \frac{-k_tk_m}{J_{eq}R_m}] \qquad B = [0; \frac{k_t}{J_{eq}R_m}]$$

$$(1)\ x(t) = [\theta_m(t); \omega_m(t)]$$

$$(2)\ y(t) = [\theta_m(t); \omega_m(t)]$$

$$y(t) = Cx(t) + Du(t)$$

Therefore,

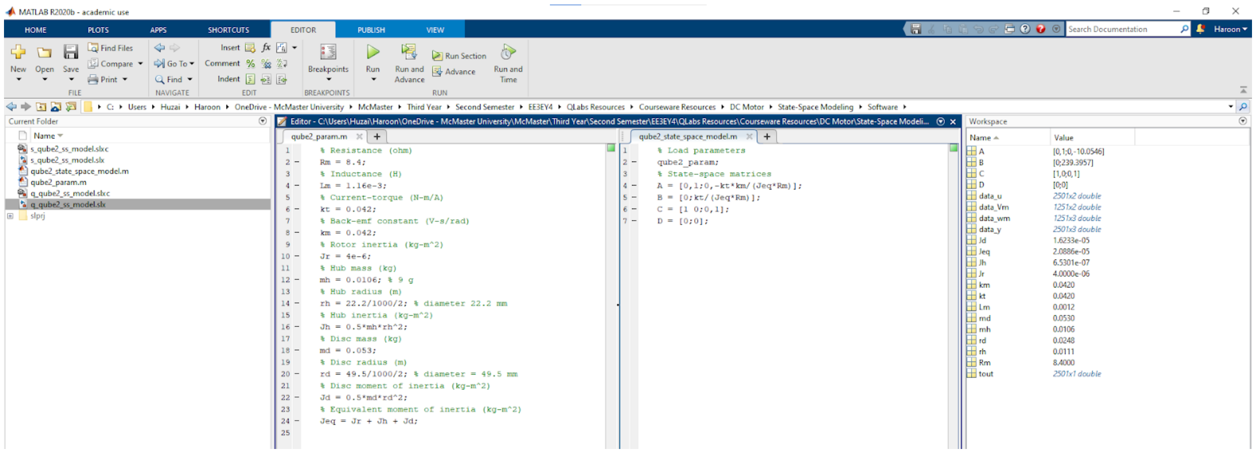
$$C = [1,\ 0; 0,\ 1] \qquad D = [0; 0]$$

This means the system is measuring motor position and speed and Vm(t) does not directly affect the output. Finally, the state space equations would be this:

$$\bar{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

(5.3)



qube2_param.m

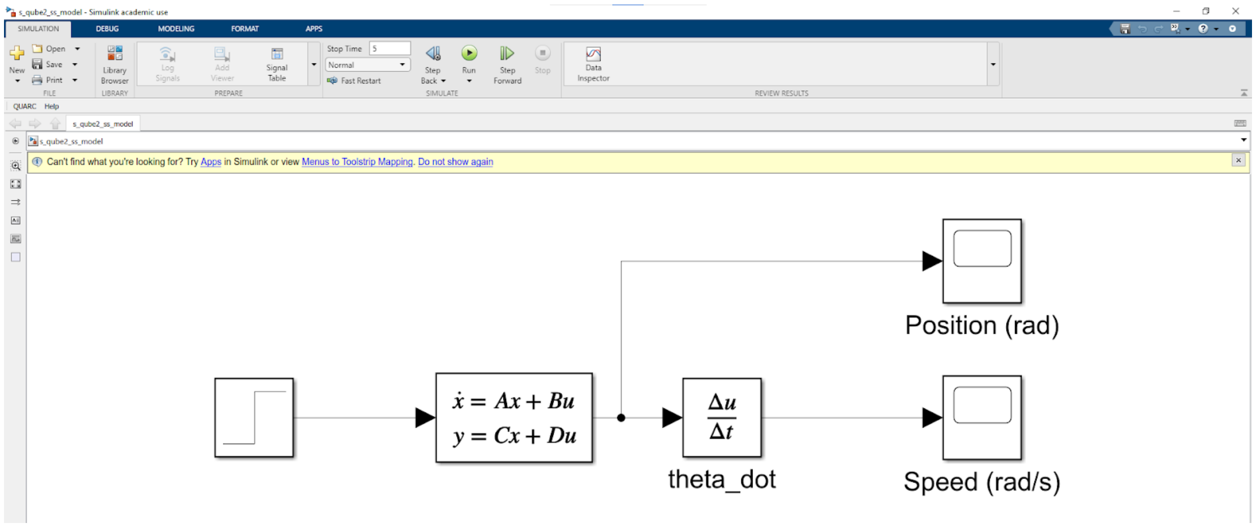
```
% Resistance (ohm)
Rm = 8.4;
% Inductance (H)
Lm = 1.16e-3;
% Current-torque (N-m/A)
kt = 0.042;
% Back-emf constant (V-s/rad)
km = 0.042;
% Rotor inertia (kg-m^2)
Jr = 4e-6;
% Hub mass (kg)
mh = 0.0106; % 9 g
% Hub radius (m)
rh = 22.2/1000/2; % diameter 22.2 mm
% Hub inertia (kg-m^2)
Jh = 0.5*mh*rh^2;
% Disc mass (kg)
md = 0.053;
% Disc radius (m)
rd = 49.5/1000/2; % diameter = 49.5 mm
% Disc moment of inertia (kg-m^2)
Jd = 0.5*md*rd^2;
% Equivalent moment of inertia (kg-m^2)
Jeq = Jr + Jh + Jd;
```

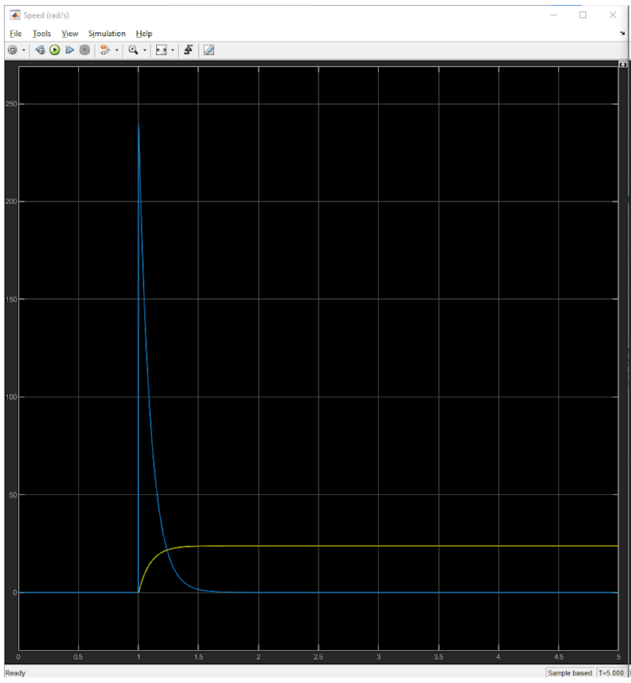
qube2_state_space_model.m

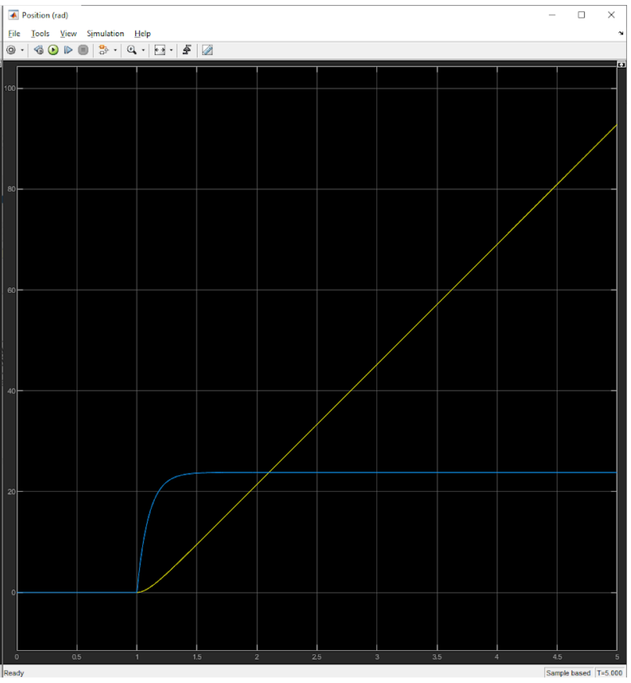
```
% Load parameters
qube2_param;
% State-space matrices
A = [0,1;0,-kt*km/(Jeq*Rm)];
B = [0;kt/(Jeq*Rm)];
C = [1 0;0,1];
D = [0;0];
```

Workspace

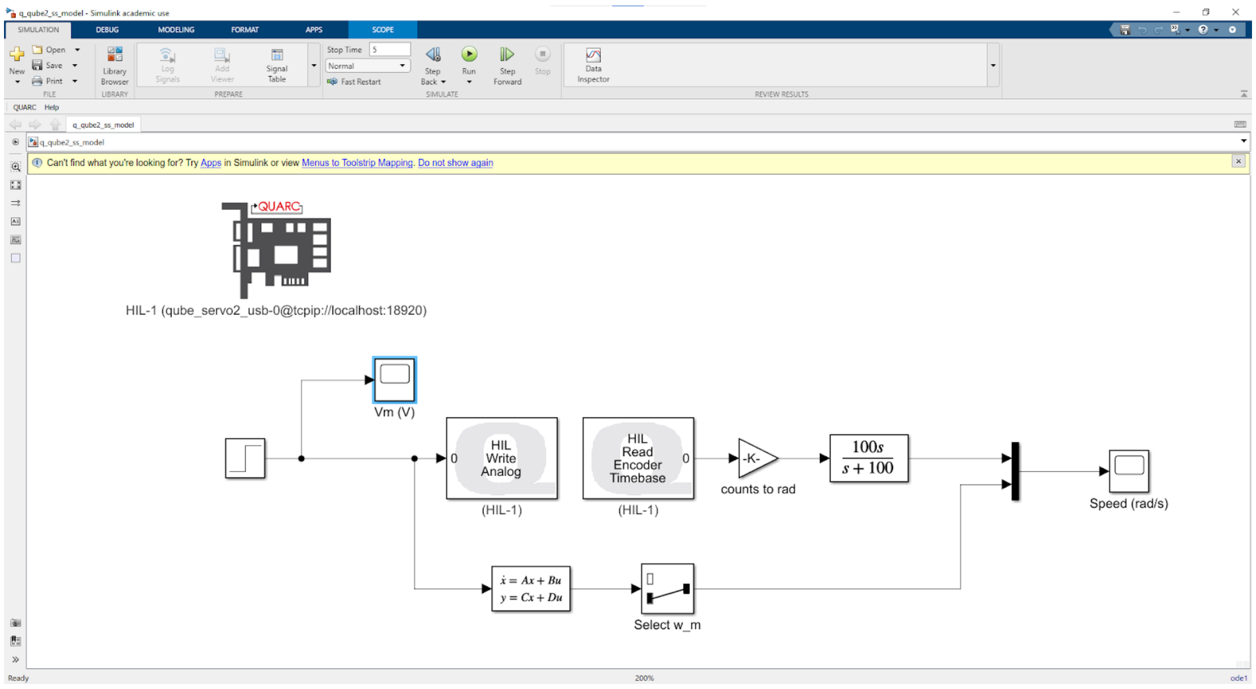
Name	Value
A	[0,1;0,-10.0546]
B	[0;239.3957]
C	[1,0;0,1]
D	[0;0]
data_u	2501x2 double
data_Vm	1251x2 double
data_wm	1251x3 double
data_y	2501x3 double
Jd	1.6233e-05
Jeq	2.0886e-05
Jh	6.5301e-07
Jr	4.0000e-06
km	0.0420
kt	0.0420
Lm	0.0012
md	0.0530
mh	0.0106
rd	0.0248
rh	0.0111
Rm	8.4000
tout	2501x1 double

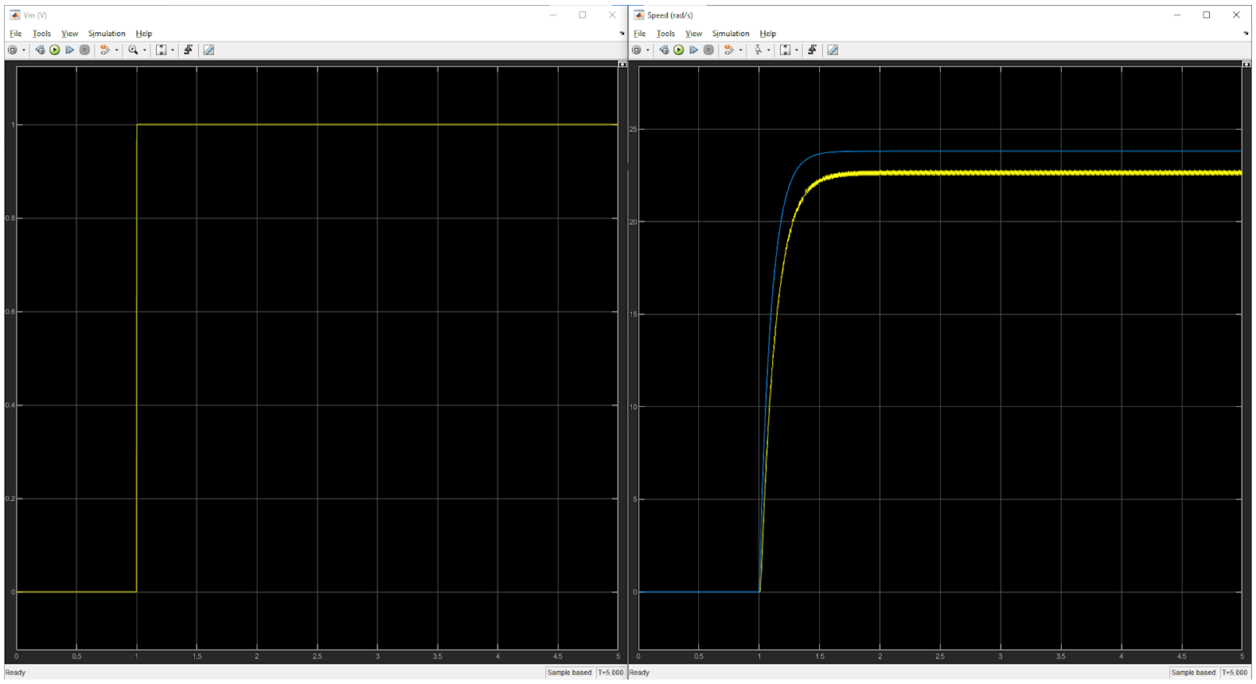

s_qube2_ss_model - Simulink academic use
Stop Time 5
Normal
Can't find what you're looking for? Try Apps in Simulink or view Menus to Toolstrip Mapping. Do not show again
$\dot{x} = Ax + Bu$
$y = Cx + Du$
$\frac{\Delta u}{\Delta t}$
theta_dot
Position (rad)
Speed (rad/s)



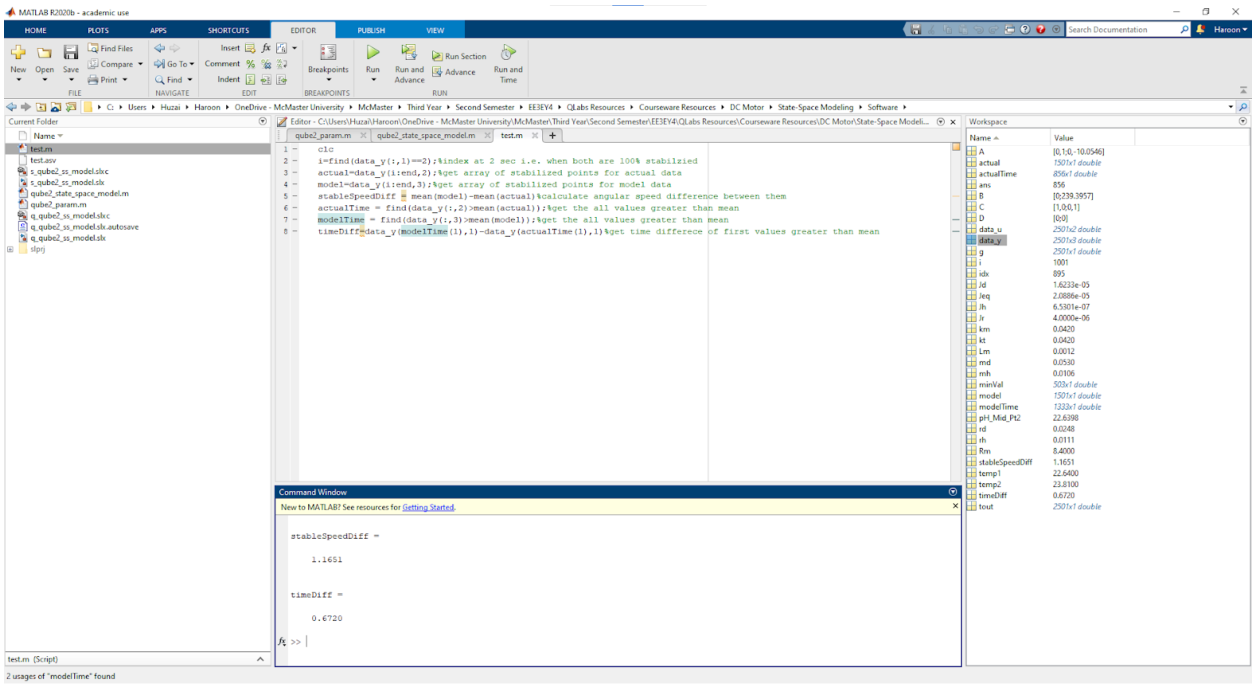


(5.4)


q_qube2_ss_model - Simulink academic use
QUARC
HIL-1 (qube_servo2_usb-0@tcpip://localhost:18920)
Vm (V)
HIL Write Analog
(HIL-1)
HIL Read Encoder Timebase
(HIL-1)
-K-
counts to rad
$\frac{100s}{s+100}$
Speed (rad/s)
$\dot{x} = Ax + Bu$
$y = Cx + Du$
Select w_m
Can't find what you're looking for? Try Apps in Simulink or view Menus to Toolstrip Mapping. Do not show again
Ready
200%
ode1



In the end the model is a decent representation of the motor but would have been much better if friction was accounted for. Next time it may be better to add some sort of artificial load to our model so that it could better match the actual result. The model is on average 1.2rad/s faster and 0.67s quicker than the actual when they are both stable. I got these numbers by using Matlab, which can be seen below. If being off by this much is not a problem then this model is a great stand-in.



Editor - C:\Users\Huzai\Haroon\OneDrive - McMaster University\McMaster\Third Year\Second Semester\EE3EY4\QLabs Resources\Courseware Resources\DC Motor\State-Space Modeli...

test.m

```matlab
clc
i=find(data_y(:,1)==2);%index at 2 sec i.e. when both are 100% stabilzied
actual=data_y(i:end,2);%get array of stabilized points for actual data
model=data_y(i:end,3);%get array of stabilized points for model data
stableSpeedDiff = mean(model)-mean(actual)%calculate angular speed difference between them
actualTime = find(data_y(:,2)>mean(actual));%get the all values greater than mean
modelTime = find(data_y(:,3)>mean(model));%get the all values greater than mean
timeDiff=data_y(modelTime(1),1)-data_y(actualTime(1),1)%get time differece of first values greater than mean
```

Command Window

```
stableSpeedDiff =

    1.1651


timeDiff =

    0.6720

>>
```

Workspace

Name	Value
A	[0,1;0,-10.0546]
actual	1501x1 double
actualTime	856x1 double
ans	856
B	[0;239.3957]
C	[1,0;0,1]
D	[0;0]
data_u	2501x2 double
data_y	2501x3 double
g	2501x1 double
i	1001
idx	895
Jd	1.6233e-05
Jeq	2.0886e-05
Jh	6.5301e-07
Jr	4.0000e-06
km	0.0420
kt	0.0420
Lm	0.0012
md	0.0530
mh	0.0106
minVal	503x1 double
model	1501x1 double
modelTime	1333x1 double
pH_Mid_Pt2	22.6398
rd	0.0248
rh	0.0111
Rm	8.4000
stableSpeedDiff	1.1651
temp1	22.6400
temp2	23.8100
timeDiff	0.6720
tout	2501x1 double