Lab 6: Motor Simulation

Ryan St.Pierre (ras70)

November 9, 2017

1 Motor simulations

1.1 Step response

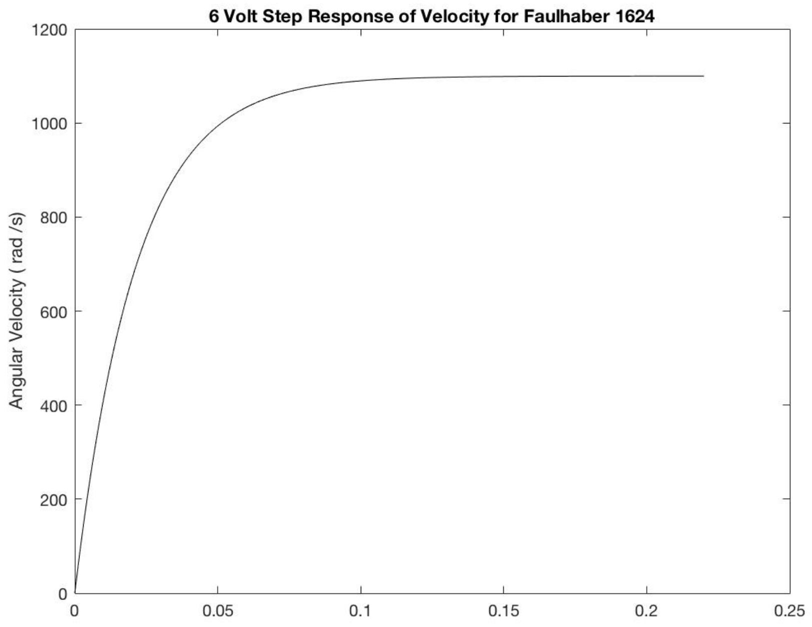

Figure 1: Step response graph for the Faulhaber 1624-T-006-S DC Micromotor

The step response for the Faulhaber 1624-T-006-S DC Micromotor in Fig. 1 does make sense. With the aid of Matlab, the transfer function of the motor was determined to be:

$$\frac{\Theta_m}{E_a} = G_{pos} = \frac{0.0053}{1.36\text{e-}11\ s^3 + 6.188\text{e-}07\ s^2 + 2.892\text{e-}05\ s} \tag{1}$$

The above transfer function relates angular position to input voltage, while Fig. 1 plots the response of angular velocity to input voltage. To get the transfer function for angular velocity to input voltage, a differentiation factor s needs to be added to Equation 1. Adding this factor results in:

$$\frac{\omega_m}{E_a} = G_{vel-1624} = \frac{3.9\text{e}8}{s^2 + 45500s + 2.13\text{e}06} \tag{2}$$

The denominator of $G_{vel-1624}$ has two real roots, since $45500^2 > 4 * 2.13\text{e}06$. Therefore, the system is an overdamped, second order system, which corresponds to the response in Fig. 1. As Fig. 1 shows, an overdamped, second order system has no overshoot in response to a step input.

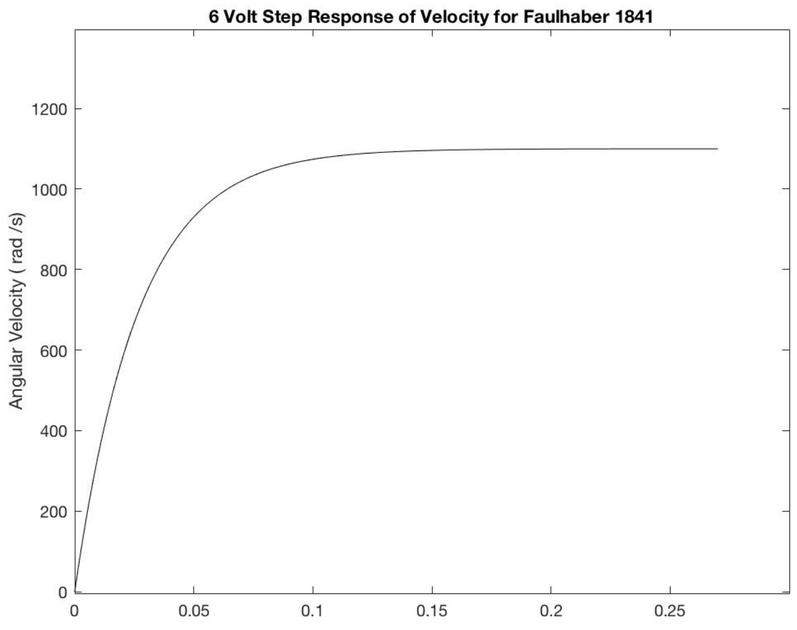

Figure 2: Step response graph for the Faulhaber 1841-T-006-S-01G DC Motor-Tach Combination

The step response shown in Fig. 2 for the Faulhaber 1841-T-006-S-01G DC Motor-Tach Combination also make sense. With the aid of Matlab, and again after applying a differentiating factor, the transfer function for velocity given input voltage for the 1841 is:

$$\frac{\omega_m}{E_a} = G_{vel-1841} = \frac{3.1\text{e}8}{s^2 + 45500s + 1.70\text{e}6} \tag{3}$$

The denominator of $G_{vel-1841}$ has two real roots, since $45500^2 > 4 * 1.70\text{e}06$. Therefore, the system for the 1841 is also an overdamped, second order system, which corresponds to the response in Fig. 2.

The table for the calculated time constants for the 1624 and 1841 are given below.

	1624	1841
Simulated time constant	21.4 ms	26.7 ms
Manufacturer's time constant	22 ms	27 ms
Percent error	2.73%	1.11%

Table 1: Simulated and nominal time constants

1.2 Pulse train response

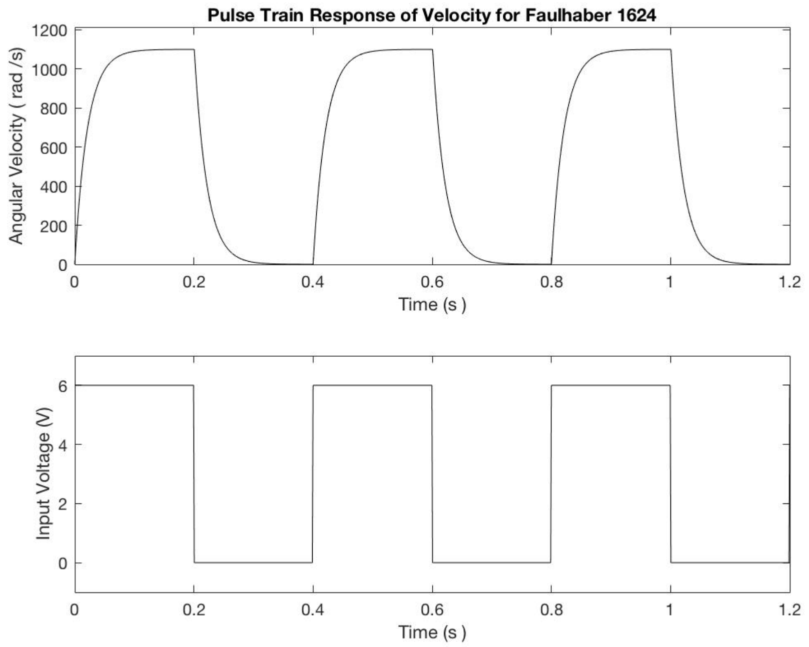

Figure 3: Pulse response graph for the Faulhaber 1624-T-006-S DC Micromotor with Duty Cycle=50%

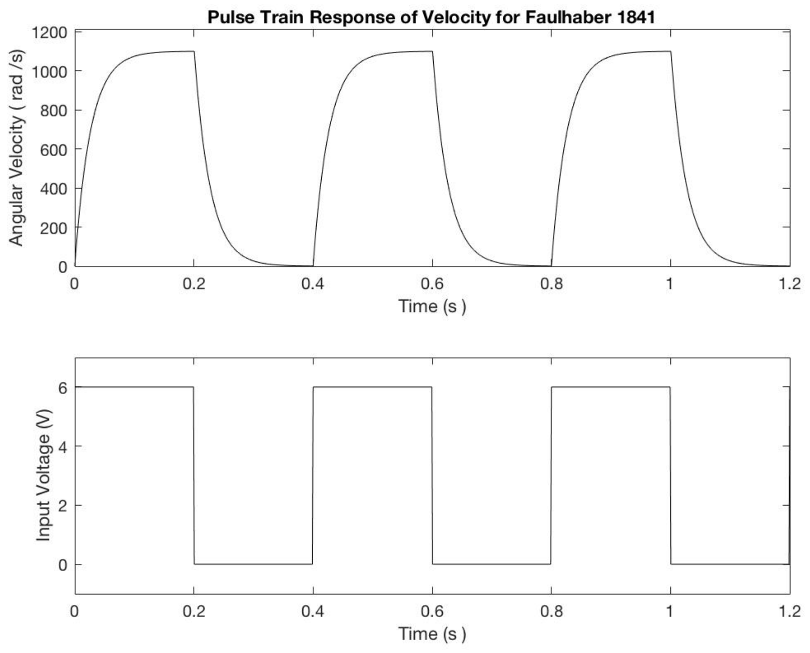

Figure 4: Pulse response graph for the Faulhaber 1841-T-006-S-01G DC Motor-Tach Combination with Duty Cycle=50%

The graphs of the pulse response for the 1624 and 1841 do make sense. As seen in Fig. 3 and Fig. 4, the input voltage is a series of alternating positive and negative steps. The angular response graphs have alternating positive and negative step responses, with each of these alternating responses following the same or negated version of the response found in **Section 1.1**.

At a duty cycle of 50% the motor speed does seem to settle before the input toggles on and off for the 1624 and 1841. At the default period length ($T = 0.4s$), The period of the pulse train is almost five times the time constants of both motors.

As the duty cycle is increased, the motor is on for a higher percentage of time. Thus, when the duty cycle is increased, the motor speed has a longer time to settle before the motor is switched off, but a shorter time to settle before the motor is turned on. When the duty cycle is 25% it appears the motor speed for the 1624 still settles before the motor turns on and off. However, at this duty cycle, there is not much time between the settling of the motor speed before the motor toggles off. At a duty cycle of 75% the speed of the 1624 motor settles in plenty of time before the motor is turned off, but does not settle before the motor is turned on.

Changing the duty cycle has similar results for the 1841. Again, the motor speed of the 1841 does not settle in time when the motor is switched on at a duty cycle if 75%. Additionally, since it takes the motor speed of the 1841 slightly longer to settle, the 1841 settles with less time to spare before the motor is turned off in comparison to the 1624, when the duty cycle is 25%.

The train period corresponds to how quickly the motor turns on and off. As the train period becomes smaller than the motor time constant, the motor speed does not settle as the motor switches. This results in the motor speed approaching a value that is proportional to the maximum motor speed times the duty cycle percentage. When this occurs, the motor speed oscillates around this value. The size of these oscillations is proportional to the size of the period. As the period is decreased these oscillations decrease. These oscillations are indiscernible for a period of T=0.0004 seconds.

2 Steady-state error

2.1 Symbolic system simplification

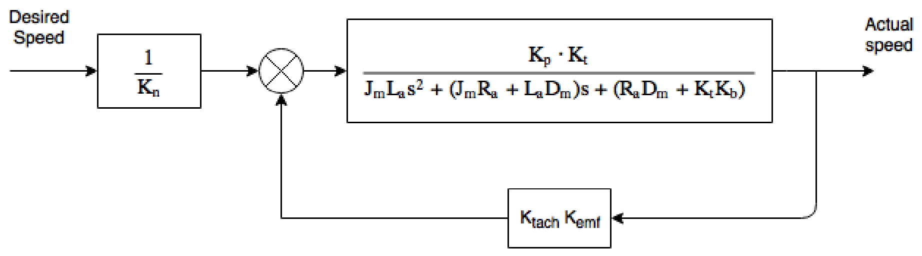
Desired Speed
$\frac{1}{K_n}$
$\frac{K_p \cdot K_t}{J_m L_a s^2 + (J_m R_a + L_a D_m)s + (R_a D_m + K_t K_b)}$
Actual speed
$K_{tach} K_{emf}$

Figure 5: Simplified block diagram for closed-loop system

(See attached Maple worksheet for closed-loop transfer functions.)

2.2 Closed-loop steady-state error

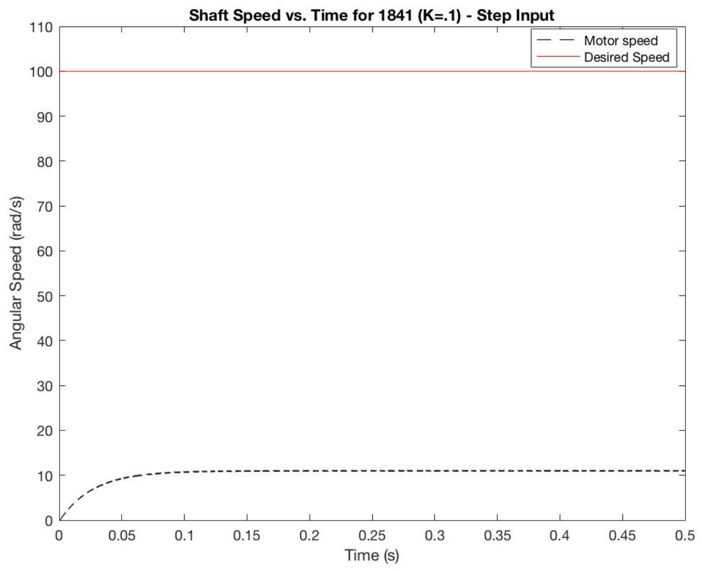

Figure 6: Simulated response of closed-loop system with gain of 0.1

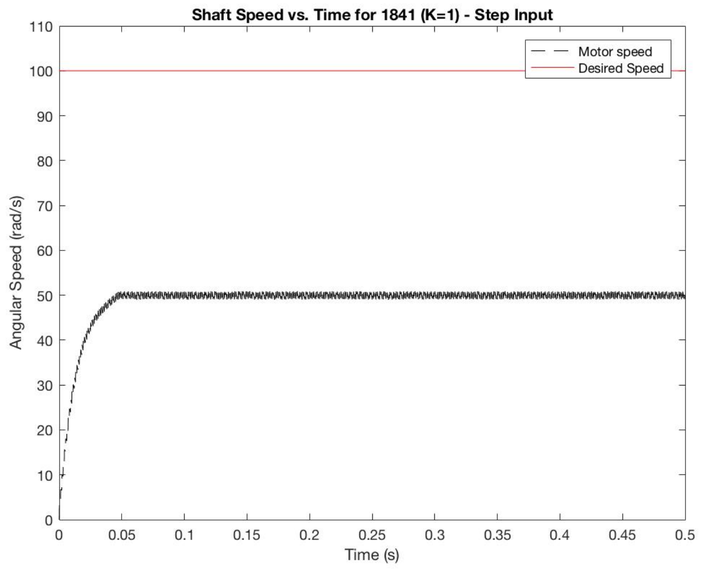

Figure 7: Simulated response of closed-loop system with gain of 1

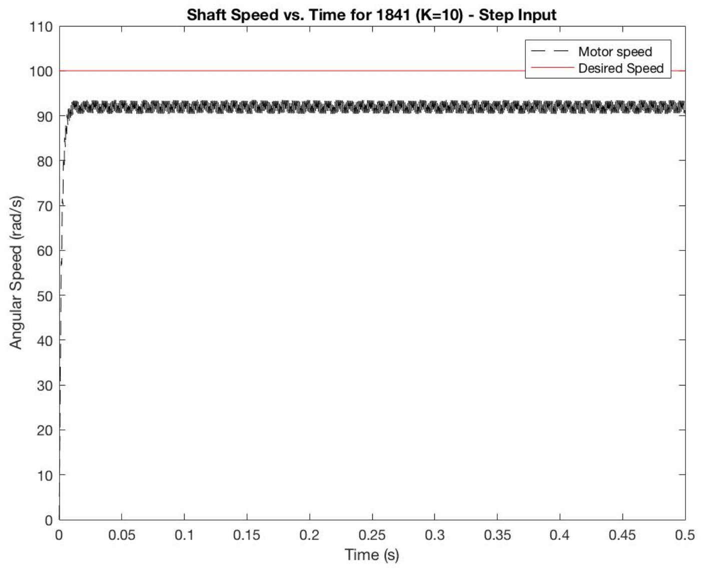

Figure 8: Simulated response of closed-loop system with gain of 10

The experimental e_{ss} values, as well as the K_p and theoretical e_{ss} values calculated from Maple, are given below.

Gain	K_p	theoretical e_{ss}	experimental e_{ss}
0.1	0.0972	0.911	0.892
1	0.968	0.508	0.509
10	9.260	0.0974	0.0955

Table 2: Steady-state error

The theoretical and experimental values for steady state error are given in Tab. 2. From this table, it is clear that increasing the gain increases the position constant K_p. Additionally, increasing the gain decreases steady state error.

These findings match the theory. The motor speed of the 1841 is modeled by a Type 0 system, as seen in **Section 1.1**. A Type 0 system has a steady state error equal to $\frac{1}{1+K_p}$ for a step input. Therefore, as K_p increases the steady state error decreases. This also aligns with intuition. As the gain of the system increases, the system is able to detect smaller error in desired and actual output.

2.3 Closed-loop steady-state error for a ramp input

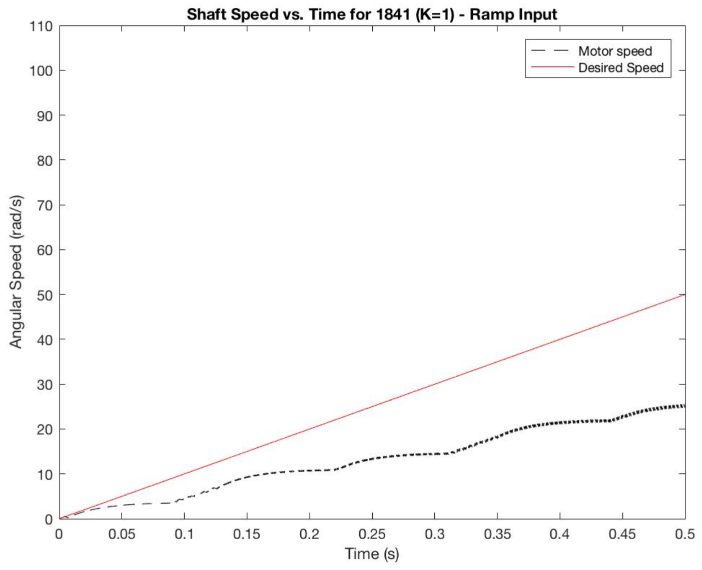

Figure 9: Simulated ramp response of closed-loop system with gain of 1

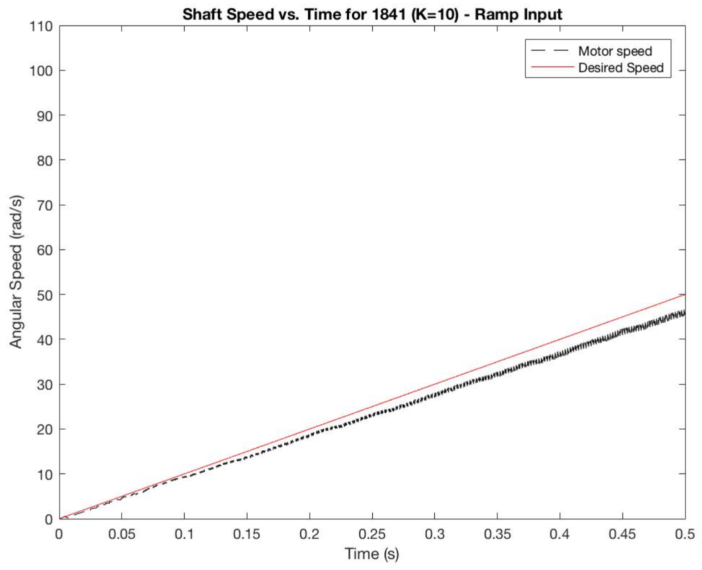

Figure 10: Simulated ramp response of closed-loop system with gain of 10

It appears from Fig. 10 that increasing the gain decreases the steady-state position error for a ramp. However, this statement is not an accurate representation of what is occurring. From Fig. 9 and Fig. 10 it is clear that the error is going to grow without bound as time increases. The output of the system and desired output are both increasing linear functions*, but their slopes are different, which will cause them to deviate as time approaches infinity.

This result aligns with theory. Type 0 systems are incapable of tracking a ramp input with finite steady state error.

*Note: the output of the system is not exactly a linear function, but its overall behavior can be approximated by such.

A MATLAB code

TransferScript1624.m

```
%% Initialize workspace
clear; format short e;
%% Load motor constants variables
MotorConstants1624

s = tf('s');

Gpos = (Kt)/(s* ((Jm*s + Dm) * (La*s + Ra) + Kt * Kb ))

figure(1); clf;
tModel = linspace (0 , 0.220 , 1000);
eModel = (6).*( tModel >=0);
OmegaModel = lsim (s * Gpos , eModel , tModel );
plot ( tModel , OmegaModel , 'k - ');
ylabel (' Angular Velocity ( rad /s) ');
title (' 6 Volt Step Response of Velocity for Faulhaber 1624 ');

w_max = max(OmegaModel); %% 183.28 (At 1V)
0.632 * w_max %% 115.83 (At 1V)

%% Pulse Train Response
figure (2); clf
NP = 100000; AMP = 6; T = 0.04; DC = 0.75;
tModel = linspace (0 , 1.2 , NP );
eModel = AMP *( mod ( tModel , T) <( DC *T ));
OmegaModel = lsim (s * Gpos , eModel , tModel );
subplot (2 ,1 ,1)
plot ( tModel , OmegaModel , 'k - ')
axis ([0.0 1.2 0.0 1.1* max(OmegaModel)])
xlabel (' Time (s ) '); ylabel (' Angular Velocity ( rad /s) ')
title (' Pulse Train Response of Velocity for Faulhaber 1624 ')
subplot (2 ,1 ,2)
plot ( tModel , eModel , 'k - ')
axis ([0.0 1.2 -1.0 7.0])
xlabel (' Time (s ) ');
ylabel (' Input Voltage (V) ');
```

TransferScript1841.m

```
%% Initialize workspace
clear; format short e;
%% Load motor constants variables
MotorConstants1841
```

```
s = tf('s');

Gpos = (Kt)/(s* ((Jm*s + Dm) * (La*s + Ra) + Kt * Kb ))

figure(1); clf;
tModel = linspace (0 , 0.270 , 1000);
eModel = (6).*( tModel >=0);
OmegaModel = lsim (s * Gpos , eModel , tModel );
plot ( tModel , OmegaModel , 'k - ');
ylabel (' Angular Velocity ( rad /s) ');
title (' 6 Volt Step Response of Velocity for Faulhaber 1841 ');

w_max = max(OmegaModel); %% 183.28 (At 1V)
0.632 * w_max %% 115.83 (At 1V)

%% Pulse Train Response
figure (2); clf
NP = 1000; AMP = 6; T = 0.4; DC = 0.75;
tModel = linspace (0 , 1.2 , NP );
eModel = AMP *( mod ( tModel , T) <( DC *T ));
OmegaModel = lsim (s * Gpos , eModel , tModel );
subplot (2 ,1 ,1)
plot ( tModel , OmegaModel , 'k - ')
axis ([0.0 1.2 0.0 1.1* max(OmegaModel)])
xlabel (' Time (s ) '); ylabel (' Angular Velocity ( rad /s) ')
title (' Pulse Train Response of Velocity for Faulhaber 1841 ')
subplot (2 ,1 ,2)
plot ( tModel , eModel , 'k - ')
axis ([0.0 1.2 -1.0 7.0])
xlabel (' Time (s ) ');
ylabel (' Input Voltage (V) ');
```

RunOpen1642.m

```
%% Initialize workspace
clear ; format short e
%% Load global variables
MotorConstants1624
%% Open the Simulink model
ModelName = 'OpenLoopMotor';
open_system(ModelName);

%% Change values and save
set_param ([ ModelName '/DesSpeedFcn'] , 'Expression', '100')
save_system ( ModelName )

%% Run simulation
sim(ModelName)
```

```
%% Plot
figure(1); clf
plot(t, Omega, 'k--', t, OmegaDes, 'r');
title('Shaft Speed vs. Time for 1642 (mrg)');
legend('Motor speed', 'Desired Speed');
xlabel('Time (s)');
ylabel('Angular Speed (rad/s)');
```

RunClosed1841.m

```
%% Initialize workspace
clear ; format short e
%% Load global variables
MotorConstants1841
%% Open the Simulink model
ModelName = 'ClosedLoopMotor';
open_system(ModelName);

%% Change values and save
set_param ([ ModelName '/DesSpeedFcn'] , 'Expression', '100*u')
set_param ([ ModelName '/KP'] , 'Gain', '10')
save_system ( ModelName )

%% Run simulation
sim(ModelName)


%% Plot
figure(1); clf;
plot(t, Omega, 'k--', t, OmegaDes, 'r');
title('Shaft Speed vs. Time for 1841 (K=10) - Ramp Input');
legend('Motor speed', 'Desired Speed');
xlabel('Time (s)');
ylabel('Angular Speed (rad/s)');
ylim([0, 110])

ess = ( OmegaDes ( end )- Omega ( end )) / OmegaDes ( end )
```

MotorConstants1624.m

```
global Jm Dm Kt Kb La Ra Kn;

Jm = 0.68 * (1/1000) * (1/100)^2;
% g cm ^2 * kg / g *  (m / cm )^2 -> kg m ^2

Dm = 90.9e-9; %% kg * m^2/s

Kt = 5.30 * (1/1000);
```

```
% mN m/A * (N/mN) -> N m/A

Kb = 0.555 * (1/1000) * (60/1) * (1/(2*pi));
% mV/rmp -> V*s/rad

La = 200e-6;

Ra = 9.1;

Kn = 1800 * (1/60) * 2*pi;
```

MotorConstants1841.m

```
global Jm Dm Kt Kb La Ra Kn Kemf Ktach;

Jm = 0.85 * (1/1000) * (1/100)^2;
% g cm ^2 * kg / g *  (m / cm )^2 -> kg m ^2

Dm = 90.9e-9; %% kg * m^2/s

Kt = 5.30 * (1/1000);
% mN m/A * (N/mN) -> N m/A

Kb = 0.555 * (1/1000) * (60/1) * (1/(2*pi));
% mV/rmp -> V*s/rad

La = 200e-6;

Ra = 9.1;

Kn = 1800 * (1/60) * 2*pi;

Kemf = 9.55 * (1/1000);
% mV /( rad /s) * (V/ mV ) -> V /( rad /s)

Ktach = 1 / ( Kemf * Kn );
% 1 / (( V /( rad /s )) * (( rad /s )/ V )) -> unity
```