Notice: You can see all scripts and pictures from my GitHub: https://github.com/pppoe80/lego_work/tree/firstlesson

K and Tm for each voltage

k_from_w Tm_from_w k_from_th Tm_from_th
2.04287927923512,0.0871907717612732,1.82327727100957,0.000364984652944915
1.99302954145547,0.0761024442329195,1.80773175577971,0.000488631768910535
1.96145002591095,0.0769624193291128,1.79128840588144,1.94087511044914e-05
1.84609345370023,0.0715928208093208,1.69322274585986,0.000543955424862383
1 ,1 ,1 ,0.5
1.82156239698612,0.0742567648248263,1.67448012503667,0.000969562997862289
1.93474847501476,0.078915612660436,1.75787575688357,7.02606619202911e-06
1.96811268704505,0.0772081265907237,1.78197594454092,6.51168318644445e-05

MATLAB Code

```
global k_from_w Tm_from_w;
global k_from_th Tm_from_th;
for n = 1:8
    data = readmatrix("data"+n+".txt");
    U = data(1,1);
    t = data(:,2);
    w = data(:,3);
    therta = data(:,4);

    w = w*pi/180;
    therta = therta*pi/180;

    w_function = @(P,t) P(1)*U-P(1)*U*exp(-t/P(2));

    P_guess = [1,1];
    P_approx = lsqcurvefit(w_function,P_guess,t,w);
    k = P_approx(1);
    Tm = P_approx(2);

    k_from_w(n) = k;
    Tm_from_w(n) = Tm;

    figure('name','real_w and approx_w');
    plot(t,w,'Marker','+');
    xlabel('time,sec');
    ylabel('speed,rad/sec');
```

```
    grid on;
    hold on;
    grid minor;

    w_approx = k*U-k*U*exp(-t/Tm);
    plot(t,w_approx,'r');
    print('real_w and approx_w'+string(n),'-djpeg');

    simOut = sim("call_model.slx");
    figure('name','sim_from_w');
    hold on;
    grid on;
    grid minor;
    xlabel('time,sec');
    ylabel('angle or speed,red is speed');
    plot(simOut.yout{1}.Values);
    plot(simOut.yout{2}.Values,'r');
    print('sim_w_from_w and sim_th_from_w'+string(n),'-djpeg');

    therta_function = @(P,t) P(1)*U*t+P(1)*P(2)*U*exp(-t/P(2))-
P(1)*P(2)*U;;

    P_guess = [1,0.5];
    P_approx = lsqcurvefit(therta_function,P_guess,t,therta);
    k = P_approx(1);
    Tm = P_approx(2);

    k_from_th(n) = k;
    Tm_from_th(n) = Tm;

    figure('name','real_th and approx_th');
    plot(t,therta,'Marker','+');
    xlabel('time,sec');
    ylabel('angle,rad');
    grid on;
    hold on;
    grid minor;
    therta_approx = k*U*t+k*Tm*U*exp(-t/Tm)-k*U*Tm;
    plot(t,therta_approx,'r');
    print('real_th and approx_th'+string(n),'-djpeg');

    open_system("call_model.slx");
    load_system("call_model.slx");
```

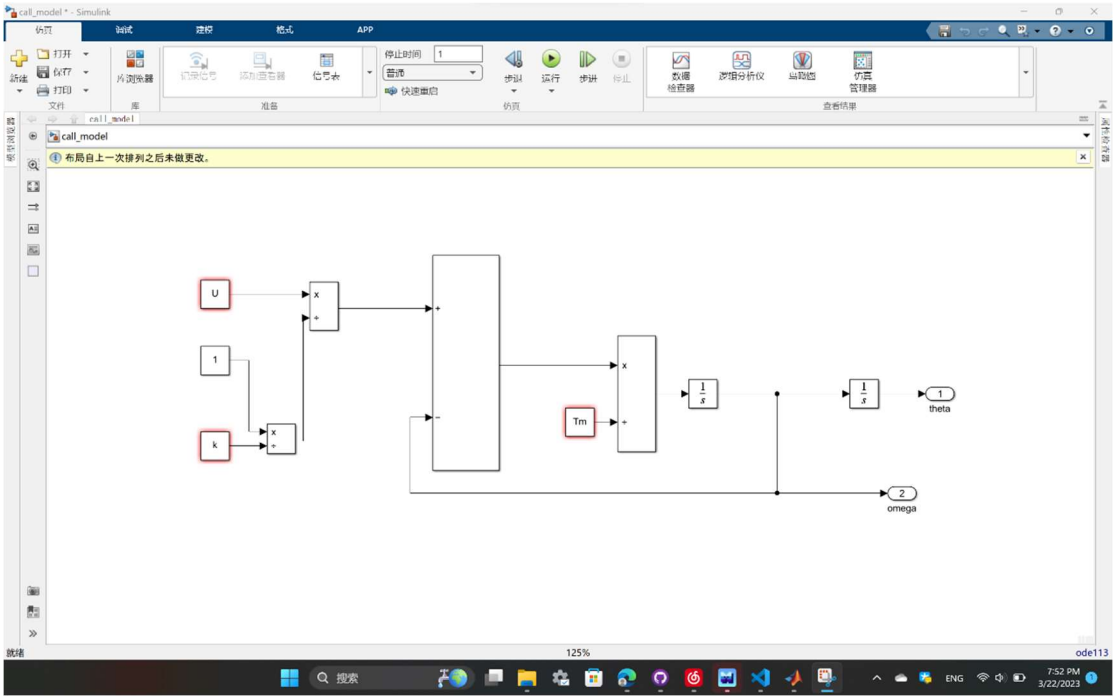
```
    simOut = sim("call_model.slx");
    figure('name','sim_from_th');
    hold on;
    grid on;
    grid minor;
    xlabel('time,sec');
    ylabel('angle or speed,red is speed');
    plot(simOut.yout{1}.Values);
    plot(simOut.yout{2}.Values,'r');
    print('sim_w_from_th and sim_th_from_th'+string(n),'-djpeg');


    close all;
end

writelines("k_from_w    Tm_from_w   k_from_th   Tm_from_th  ","k_Tm_Data
_out.txt");
writematrix([k_from_w.',Tm_from_w.',k_from_th.',Tm_from_th.'],'k_Tm_Dat
a_out',WriteMode='append');
```

Python code

```
#!/usr/bin/env python3
from ev3dev.ev3 import *
from ev3dev2.power import PowerSupply
from math import *
import time

motor = LargeMotor('outA')

max_voltage = round(PowerSupply().measured_volts,2)

motor_input =100    # Min = -100, Max = 100
total_time = 1      # After this many seconds, the motor will stop

for i in range(8):

    data = open('data' + str(i+1)+ '.txt','w')
    motor.run_direct(duty_cycle_sp = 0)
    time.sleep(1)
```

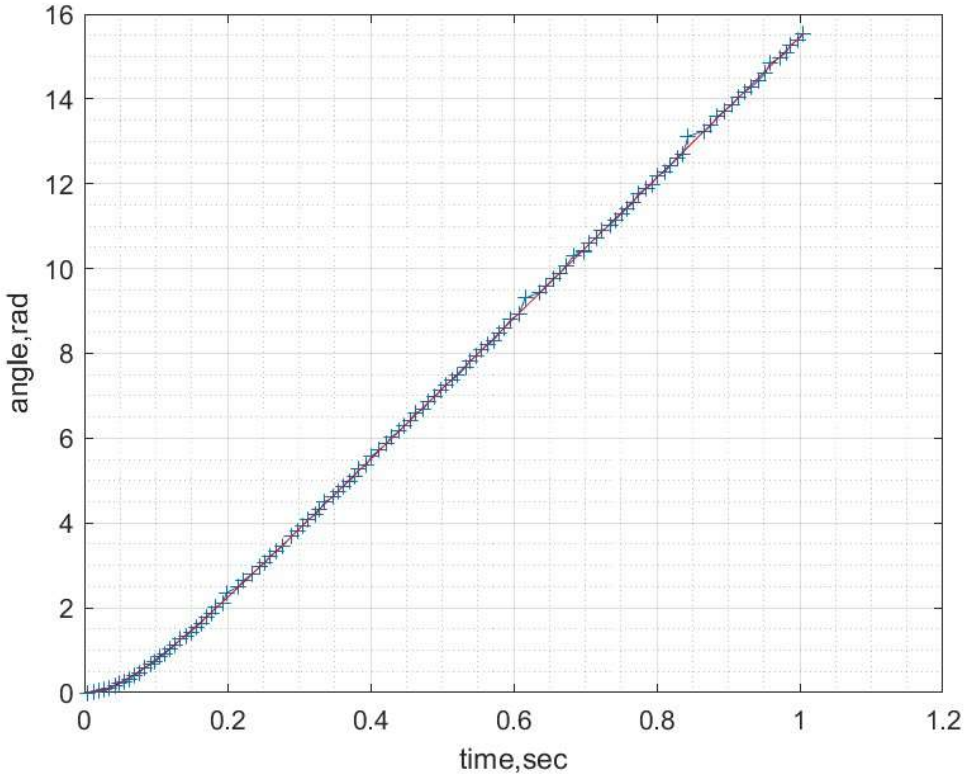
```
    timestart = time.time()
    motor.position = 0
    while True:

        # Calculating time up to 3 decimal places


        timenow = round(time.time()-timestart, 3)

        # Applying voltage
        motor.run_direct(duty_cycle_sp = motor_input-i*25)

        # Writing the following data: applied voltage (volts), time
passed (seconds), motor speed (degrees per second)

        data.write(str(max_voltage*(motor_input-i*25)/100)+'
'+'{0:.3f}'.format(timenow)+' '+str(motor.speed)+'
'+str(motor.position)+'\n')

        if timenow > total_time:
            motor.run_direct(duty_cycle_sp = 0)
            break
```

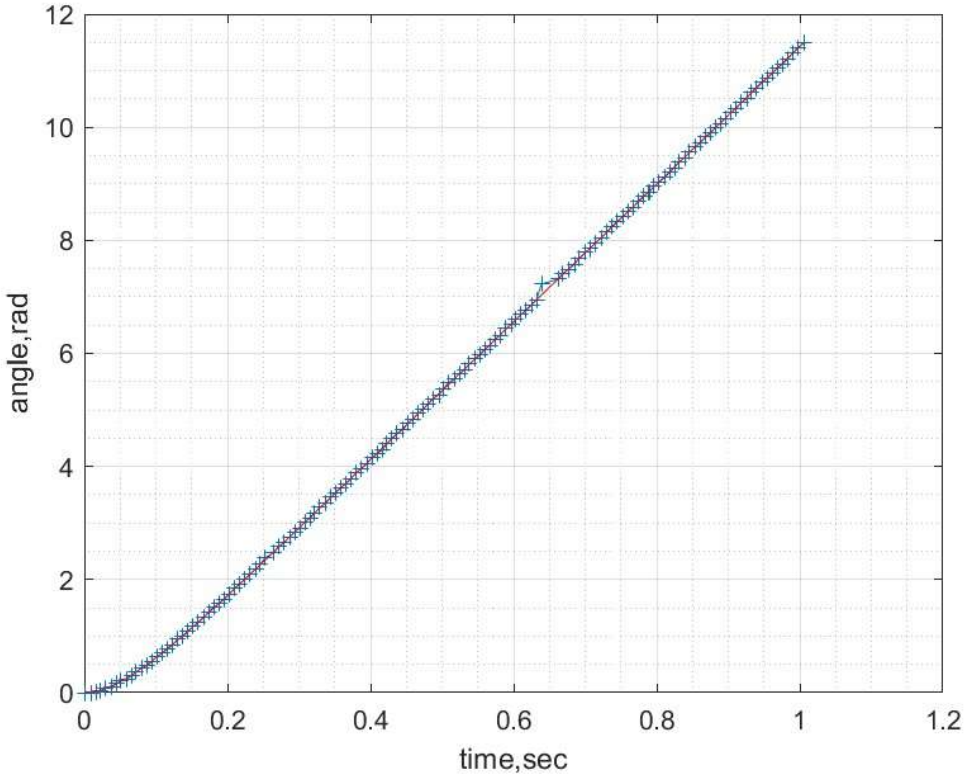
Model

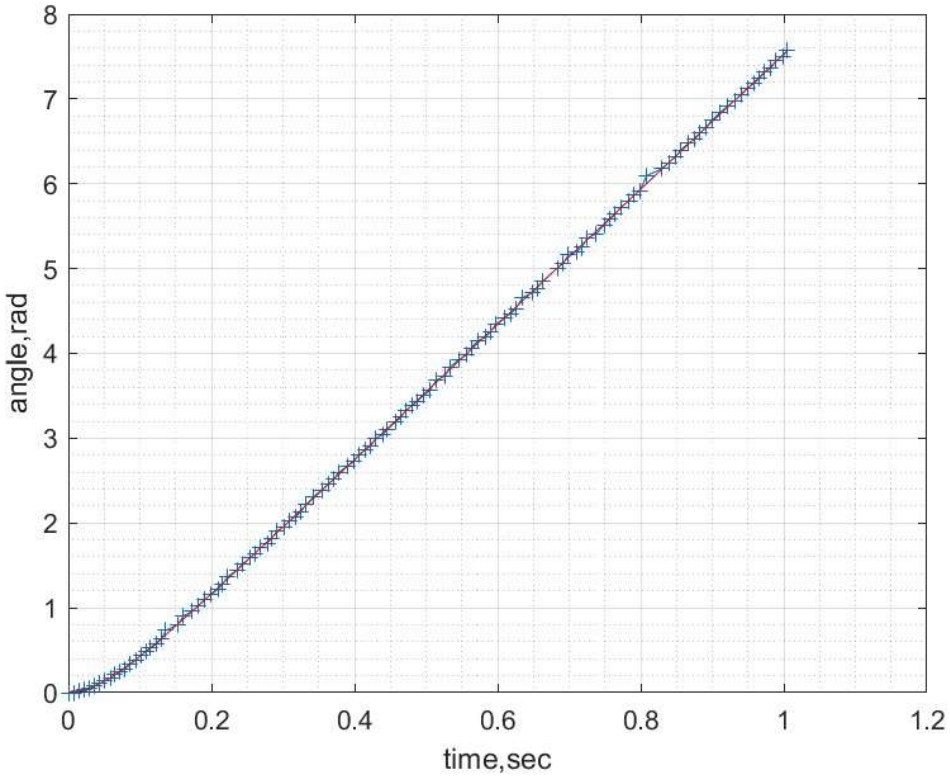


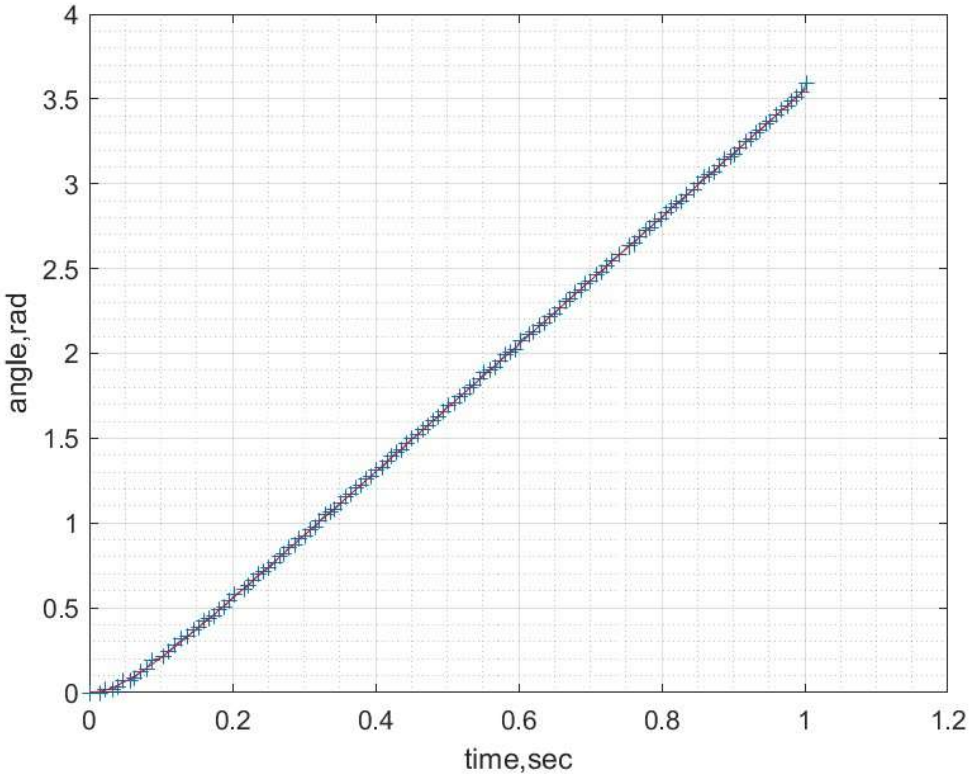
Plots in order of

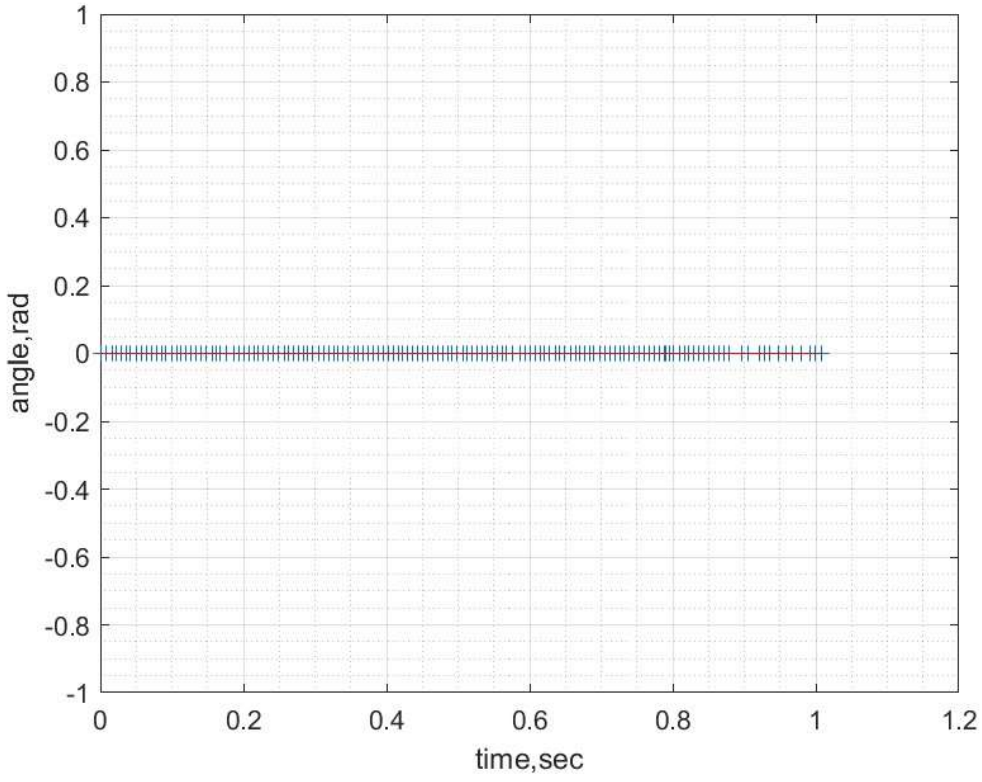
(r&a_w_th, r&a_w_w, sim_w&th_th, sim_w&th_w)

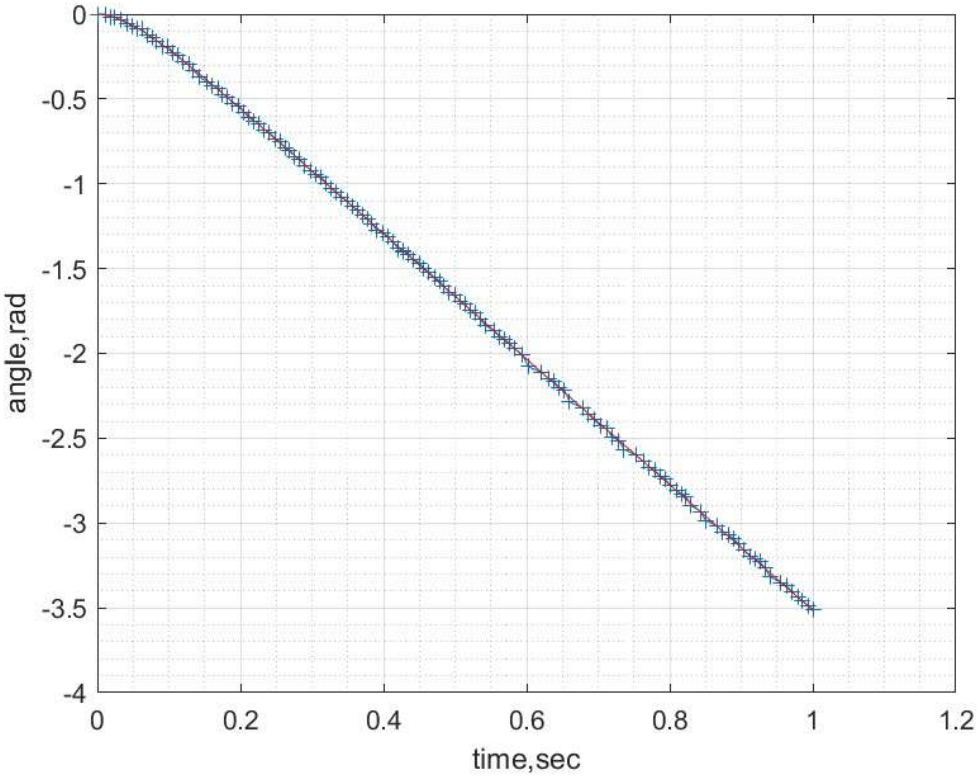


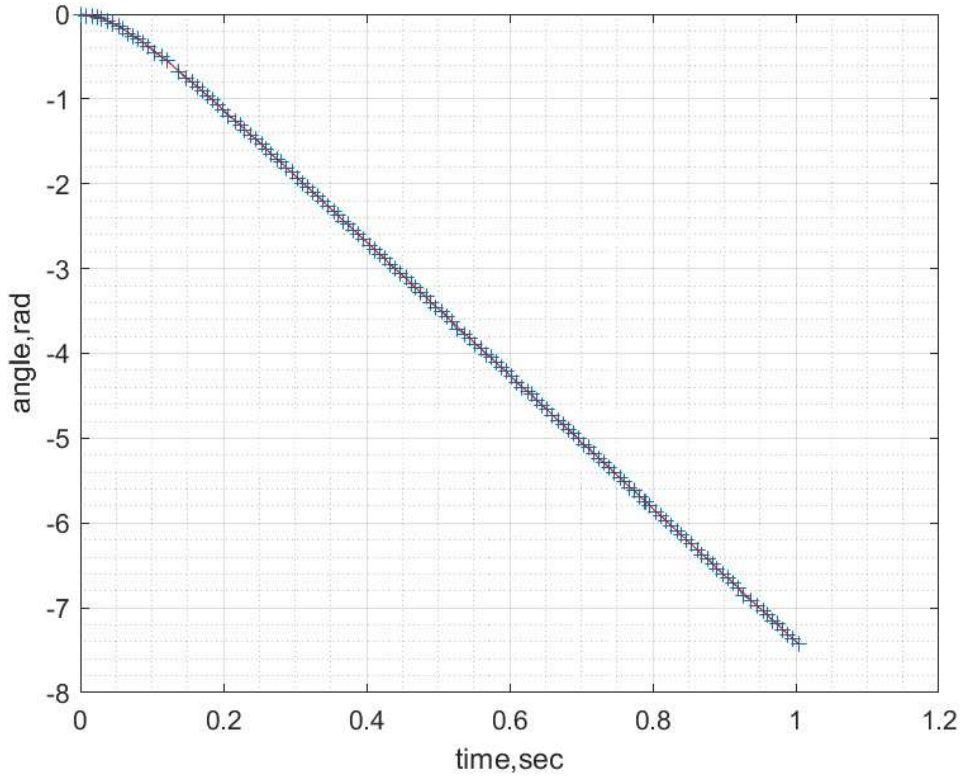

12
10
8
6
4
2
0
angle,rad
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

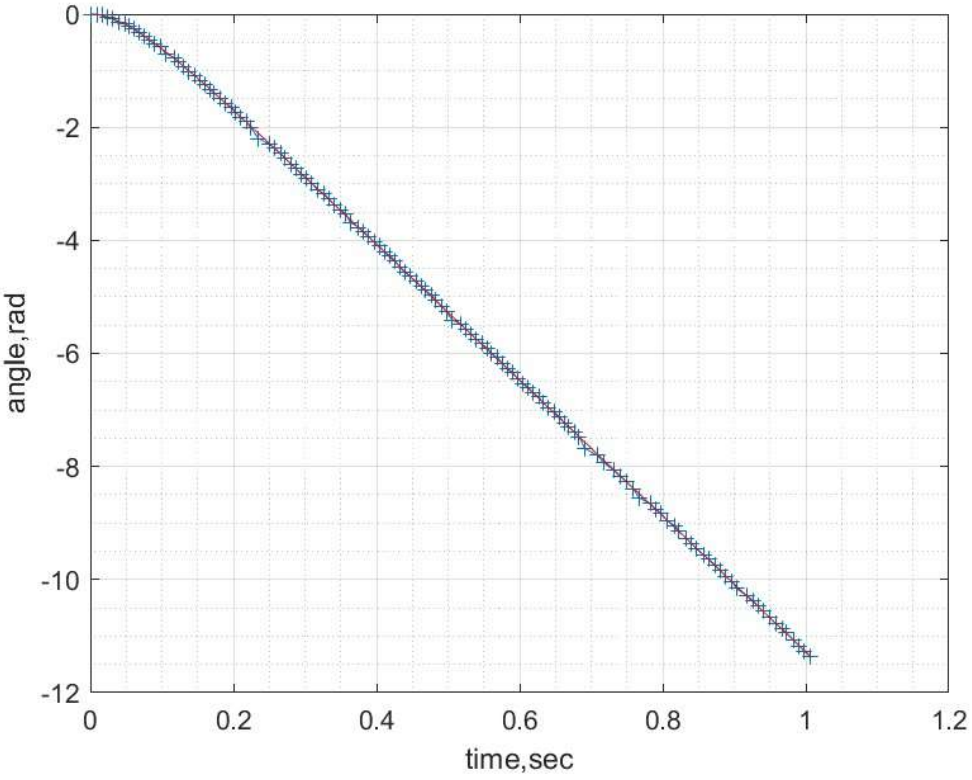

8
7
6
5
4
3
2
1
0
angle,rad
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

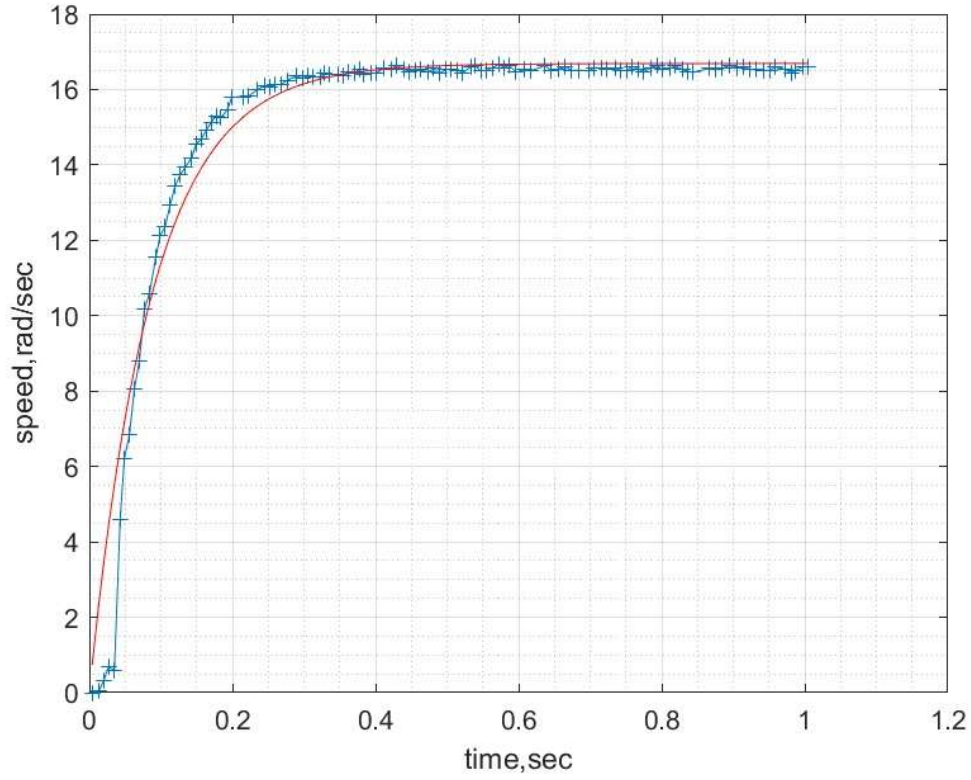

4
3.5
3
2.5
2
1.5
1
0.5
0
angle,rad
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

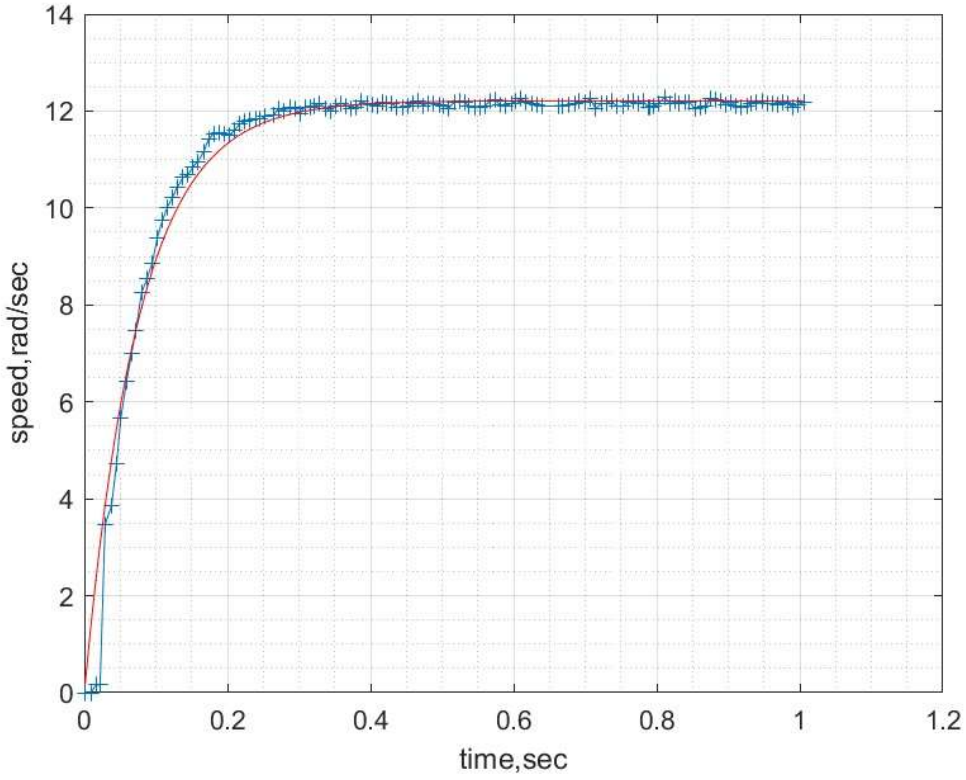

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
angle,rad
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

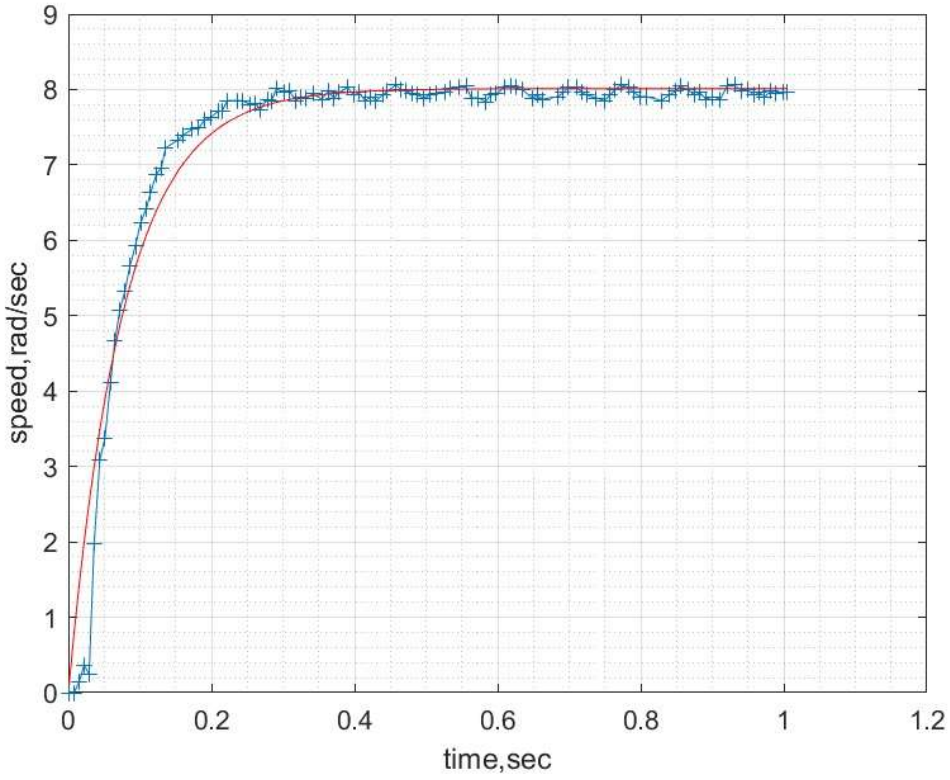

0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
angle,rad
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

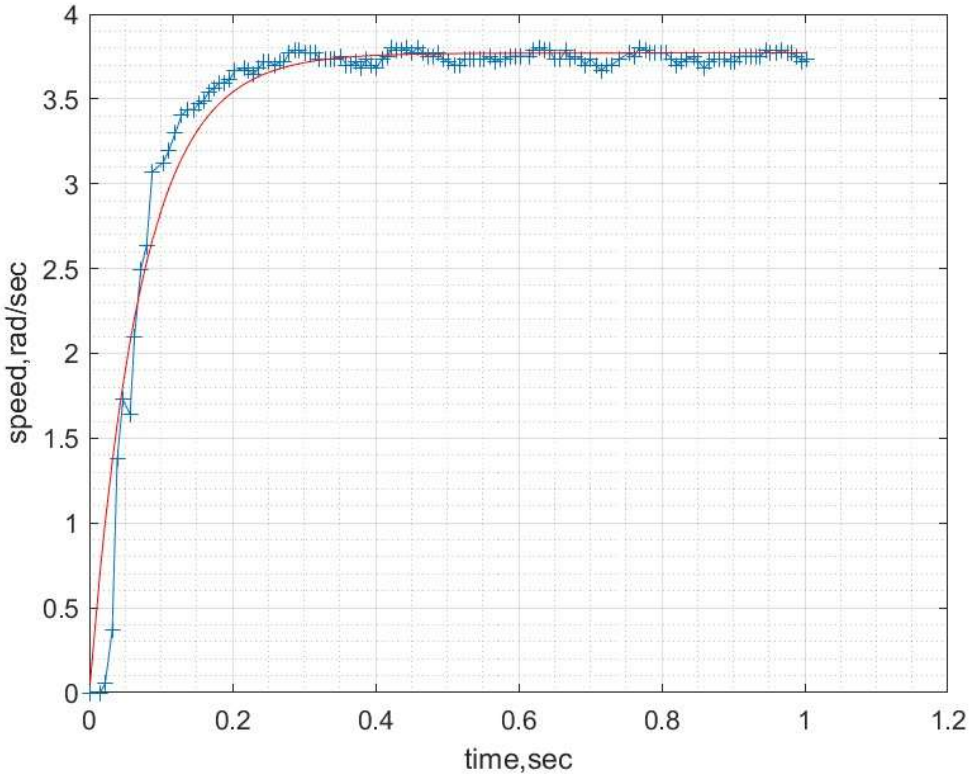

0
-1
-2
-3
-4
-5
-6
-7
-8
angle,rad
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

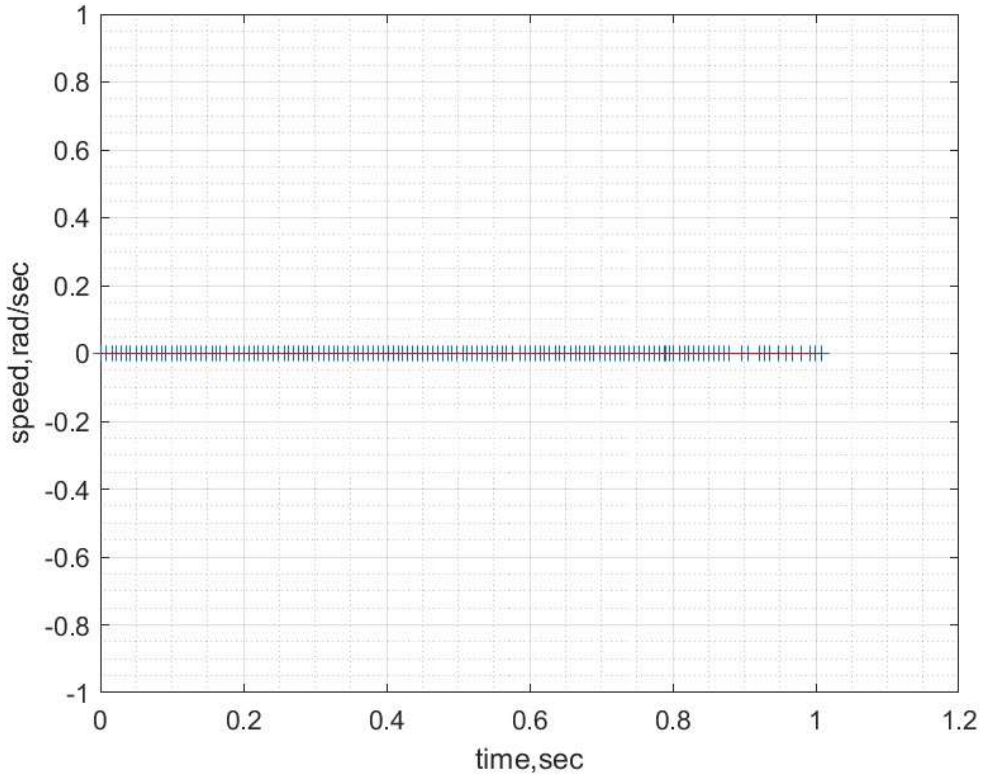

0
-2
-4
-6
-8
-10
-12
angle,rad
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

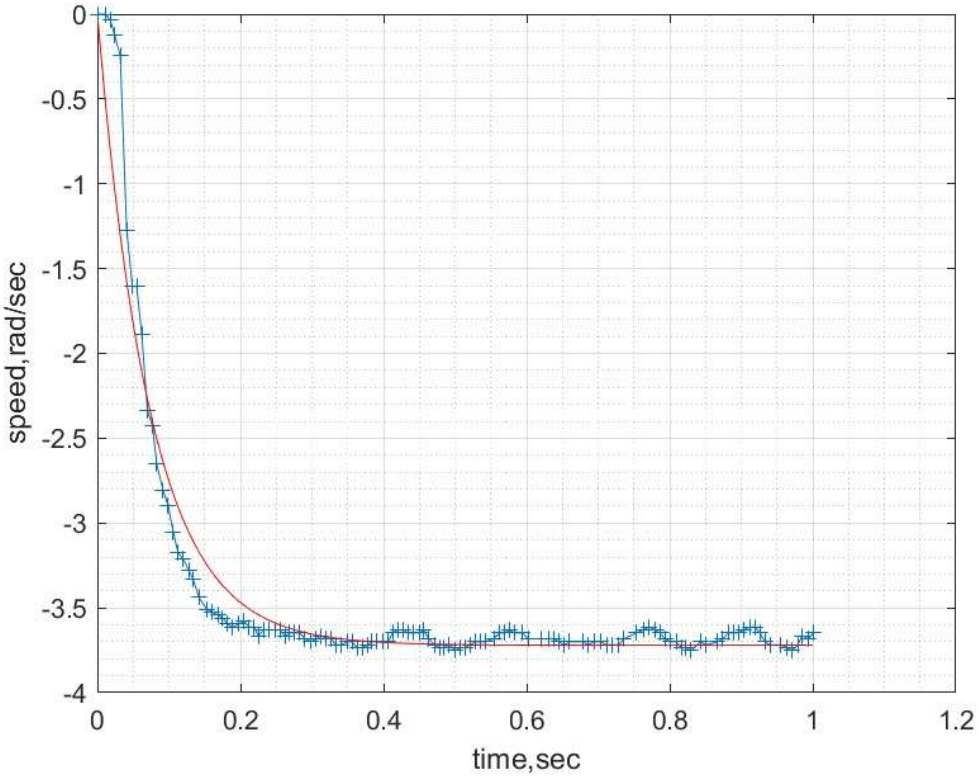

18
16
14
12
10
8
6
4
2
0
speed,rad/sec
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

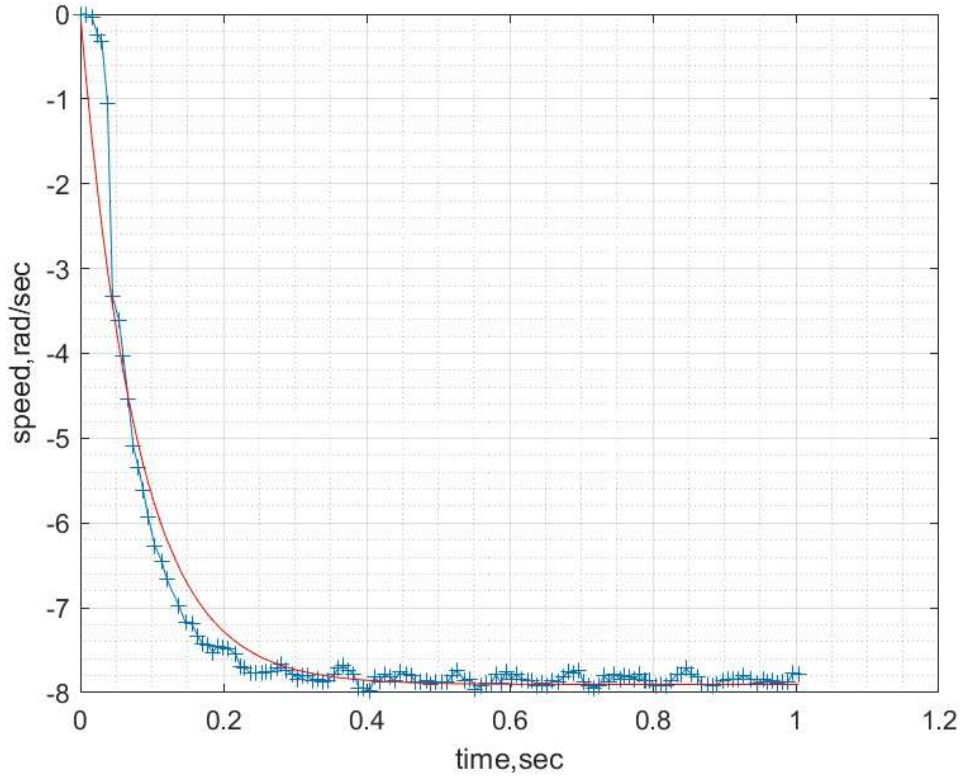

14
12
10
8
6
4
2
0
speed,rad/sec
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

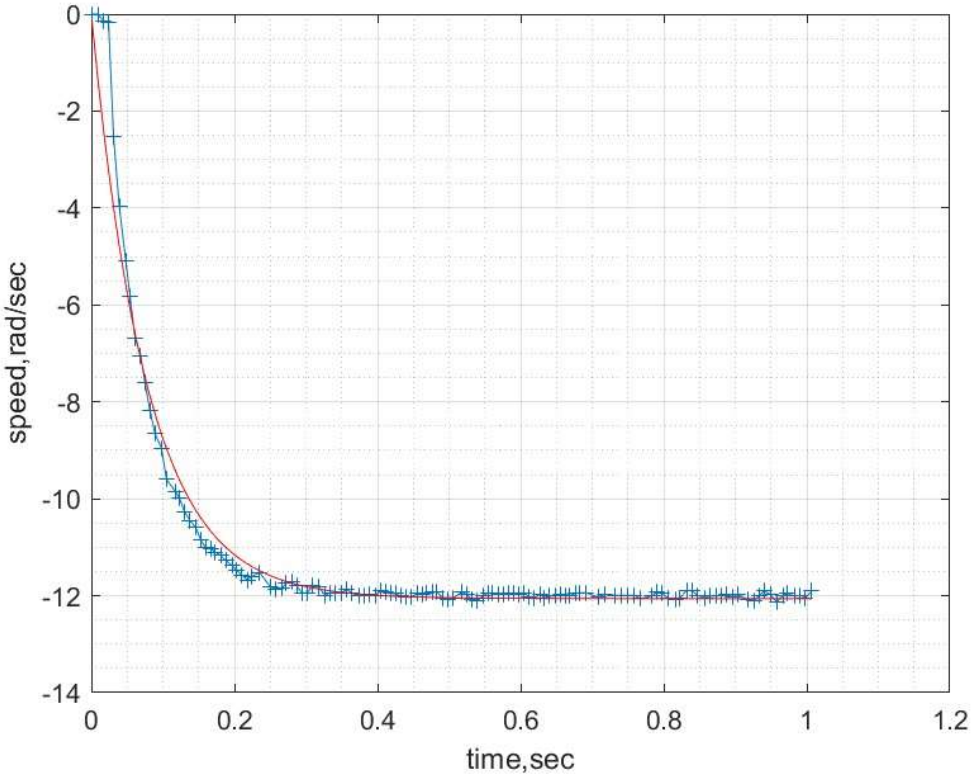

9
8
7
6
5
4
3
2
1
0
speed,rad/sec
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

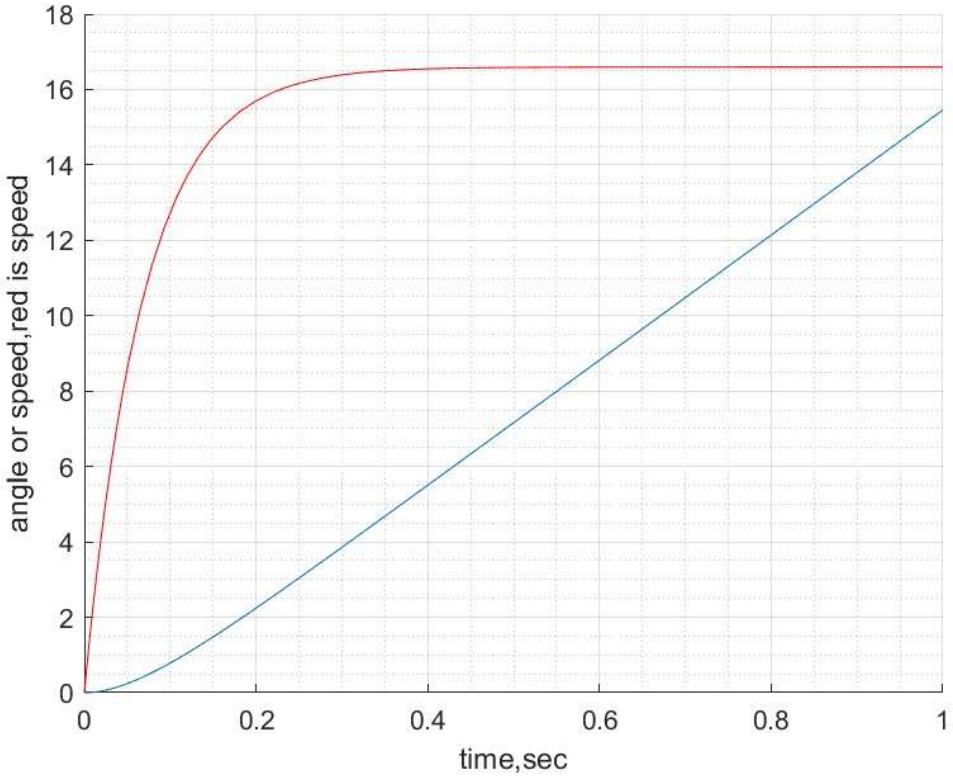

4
3.5
3
2.5
2
1.5
1
0.5
0
speed,rad/sec
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

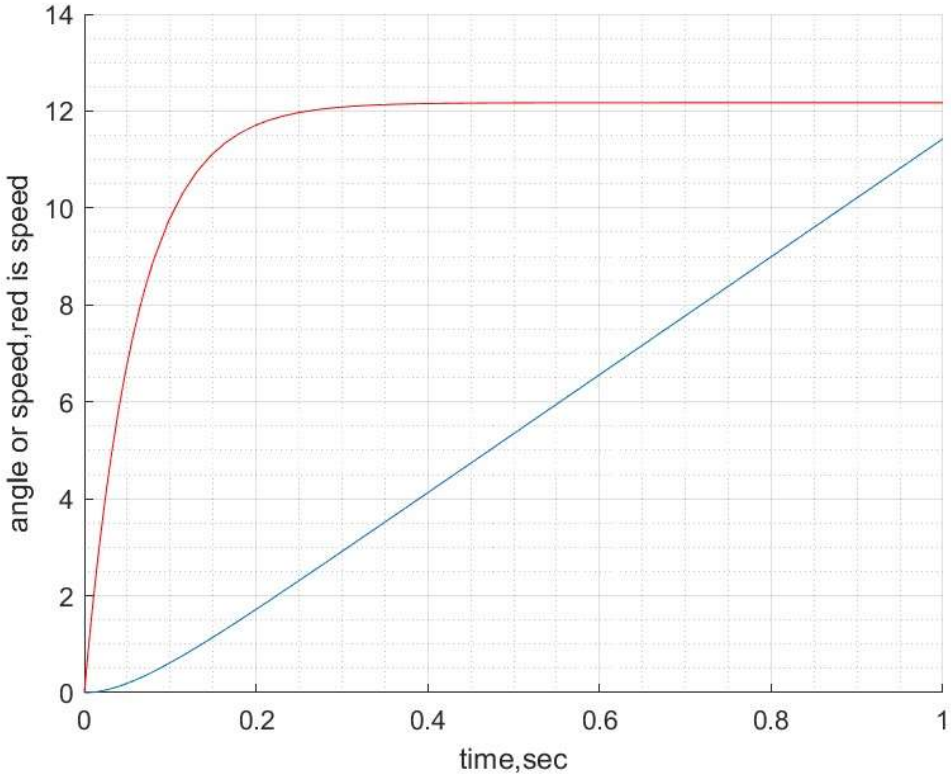

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
speed,rad/sec
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

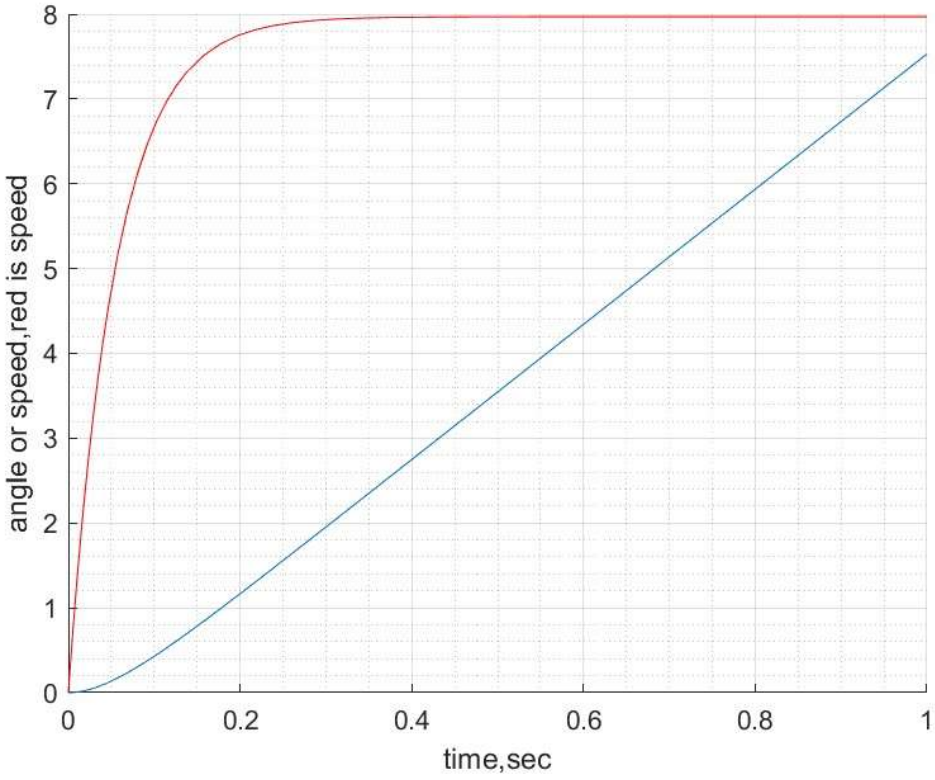

0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
speed,rad/sec
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

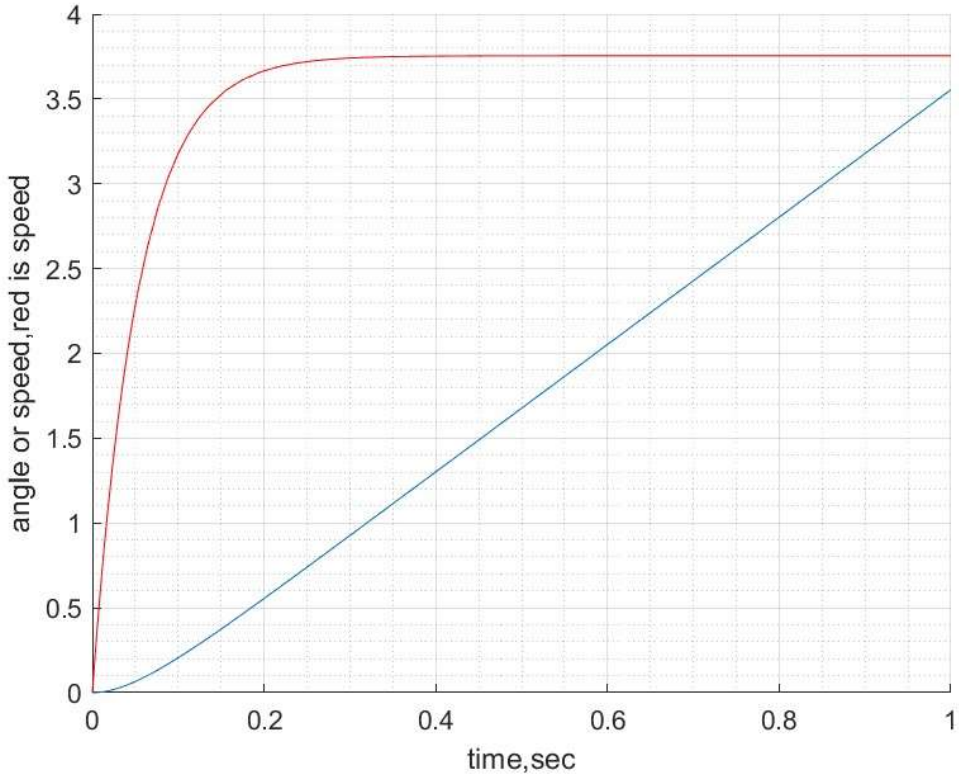

0
-1
-2
-3
-4
-5
-6
-7
-8
speed,rad/sec
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

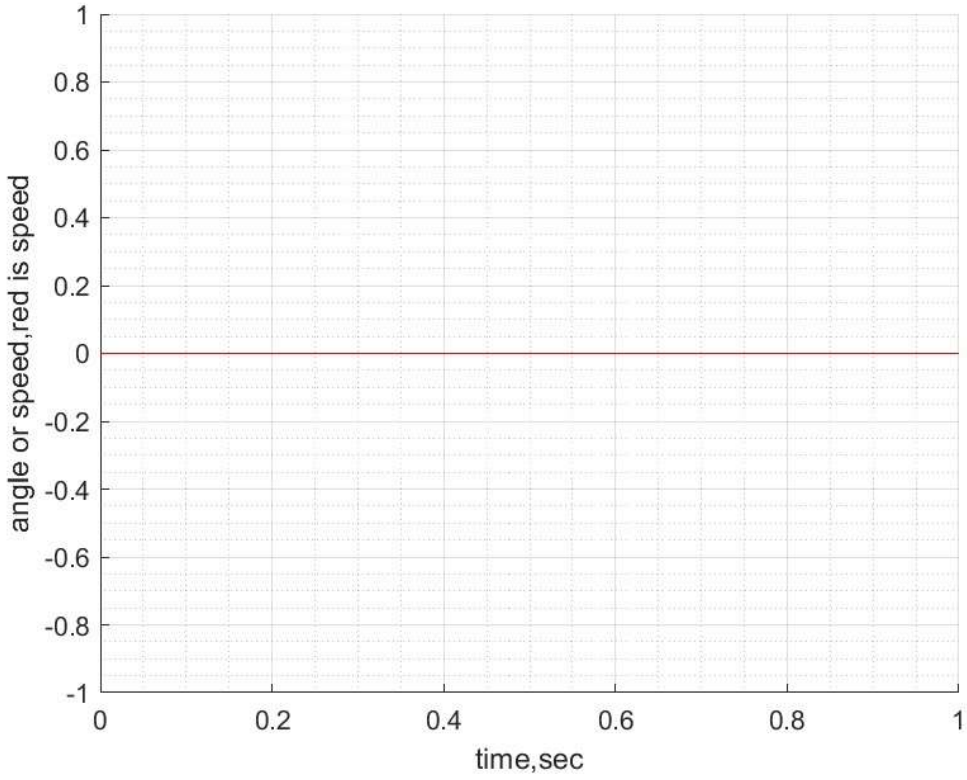

0
-2
-4
-6
-8
-10
-12
-14
speed,rad/sec
0
0.2
0.4
0.6
0.8
1
1.2
time,sec

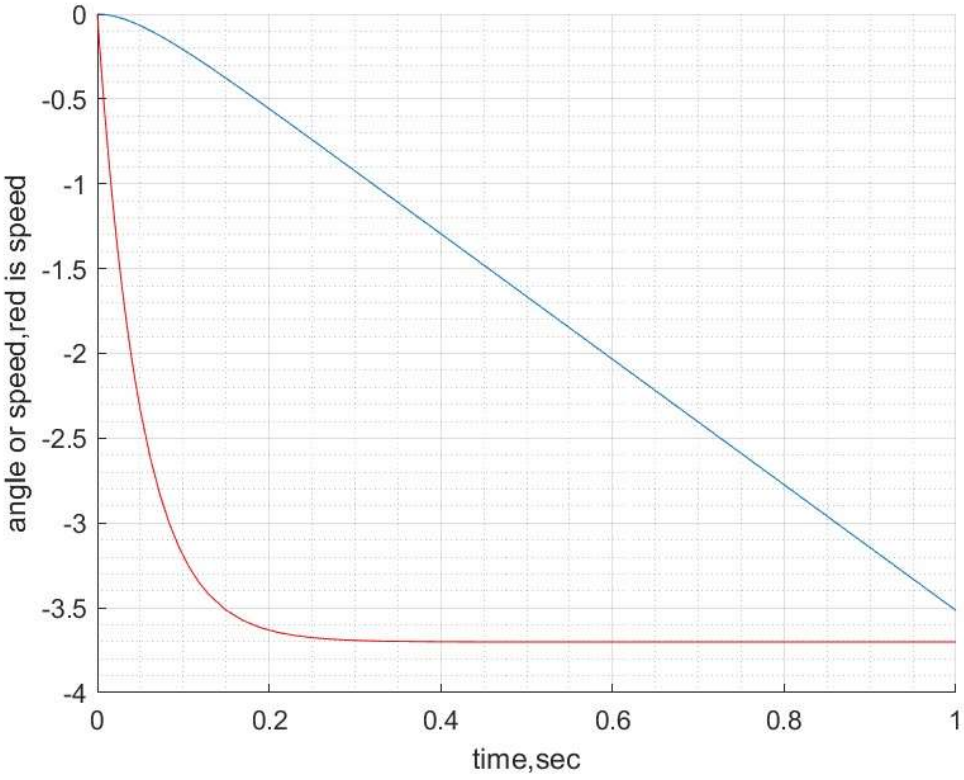

18
16
14
12
10
8
6
4
2
0
angle or speed,red is speed
0
0.2
0.4
0.6
0.8
1
time,sec

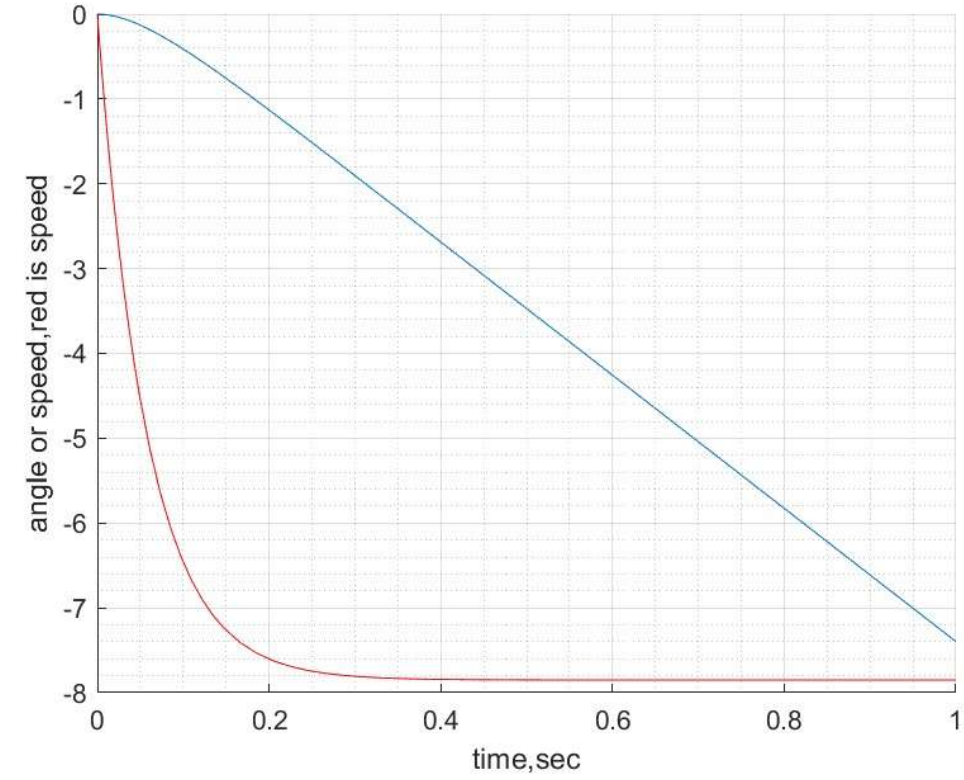

14
12
10
8
6
4
2
0
0
0.2
0.4
0.6
0.8
1
angle or speed,red is speed
time,sec

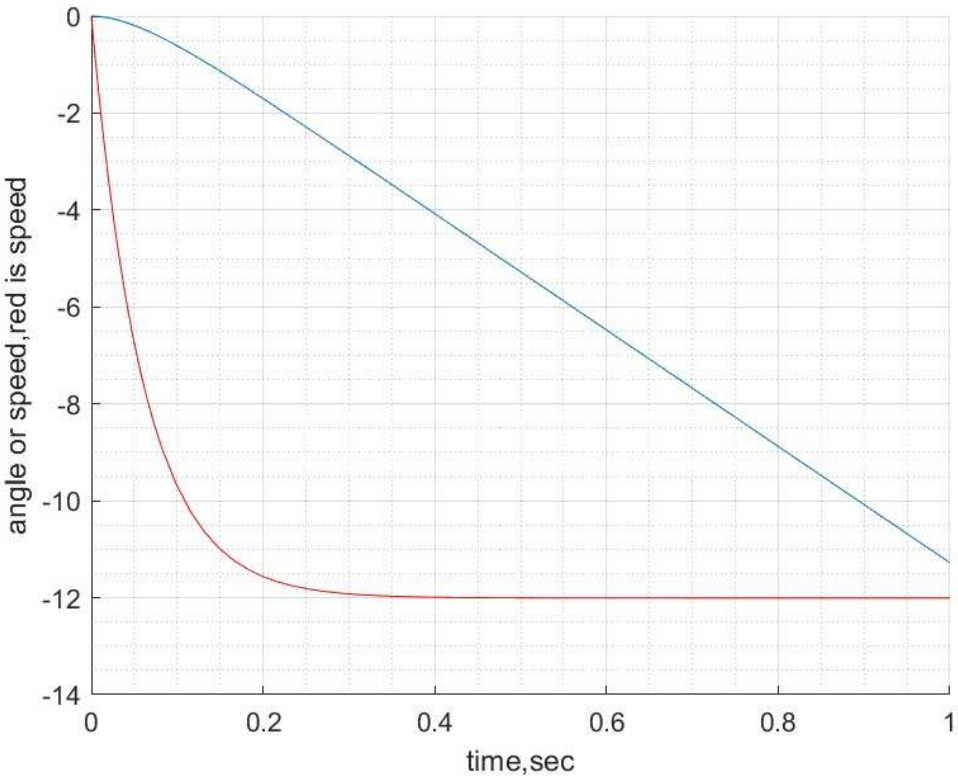

8
7
6
5
4
3
2
1
0
0
0.2
0.4
0.6
0.8
1
angle or speed,red is speed
time,sec

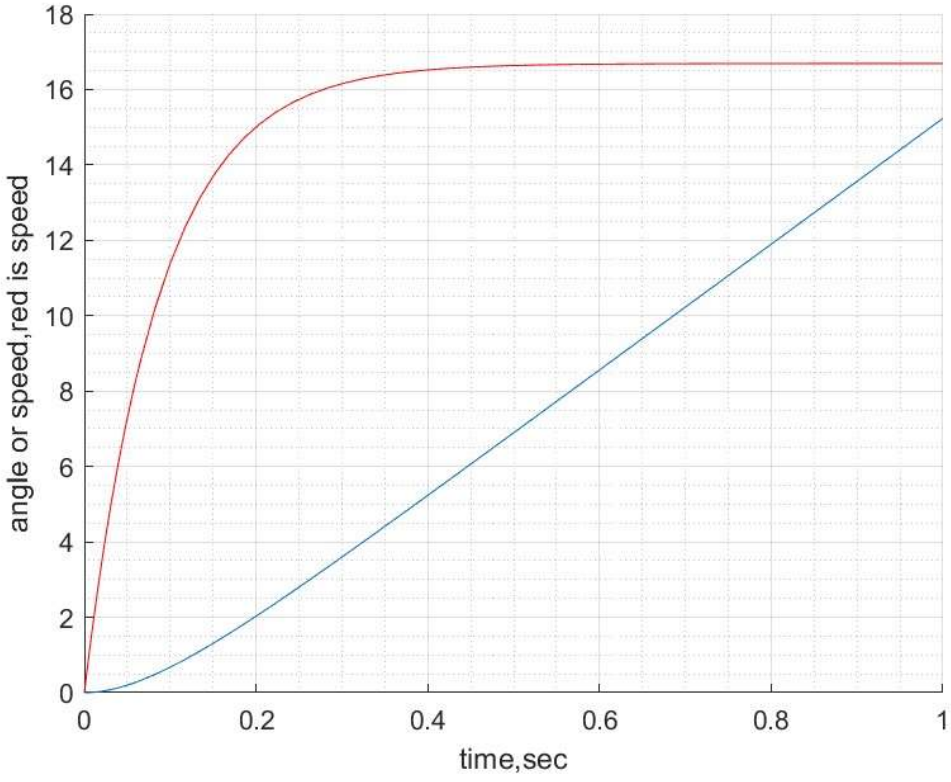

4
3.5
3
2.5
2
1.5
1
0.5
0
angle or speed,red is speed
0
0.2
0.4
0.6
0.8
1
time,sec

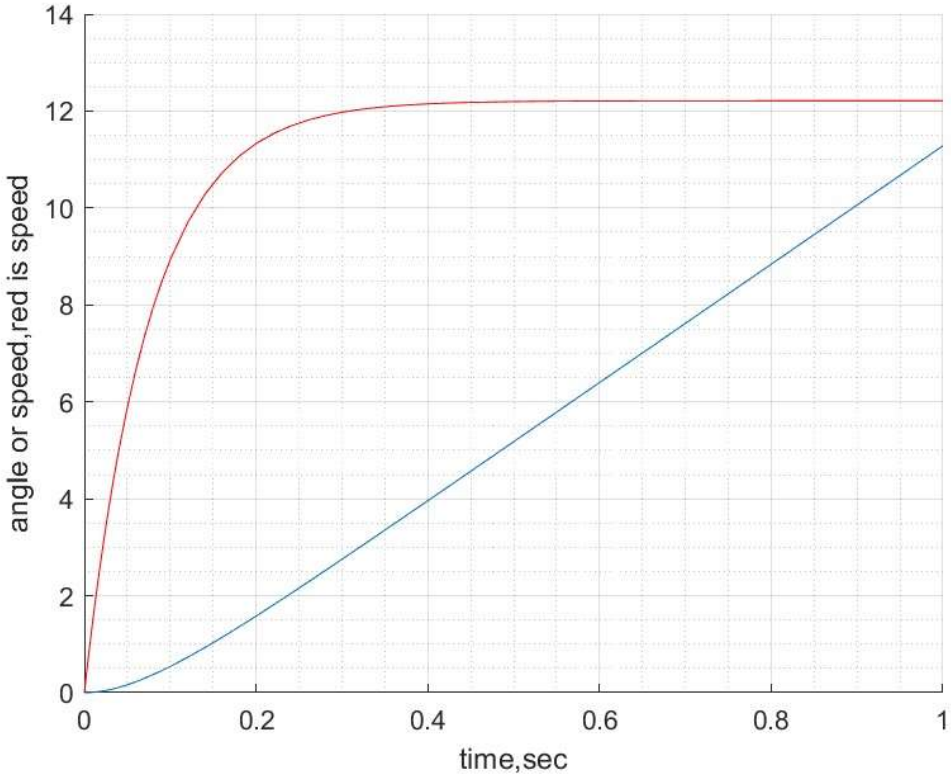

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
angle or speed,red is speed
0
0.2
0.4
0.6
0.8
1
time,sec

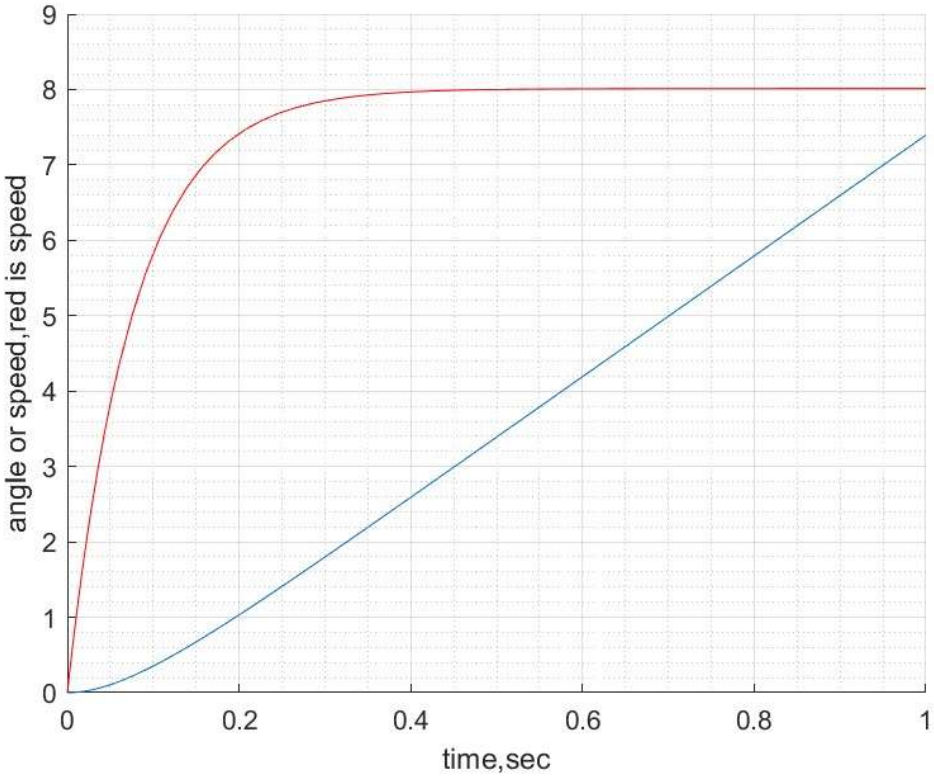

0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
angle or speed,red is speed
0
0.2
0.4
0.6
0.8
1
time,sec

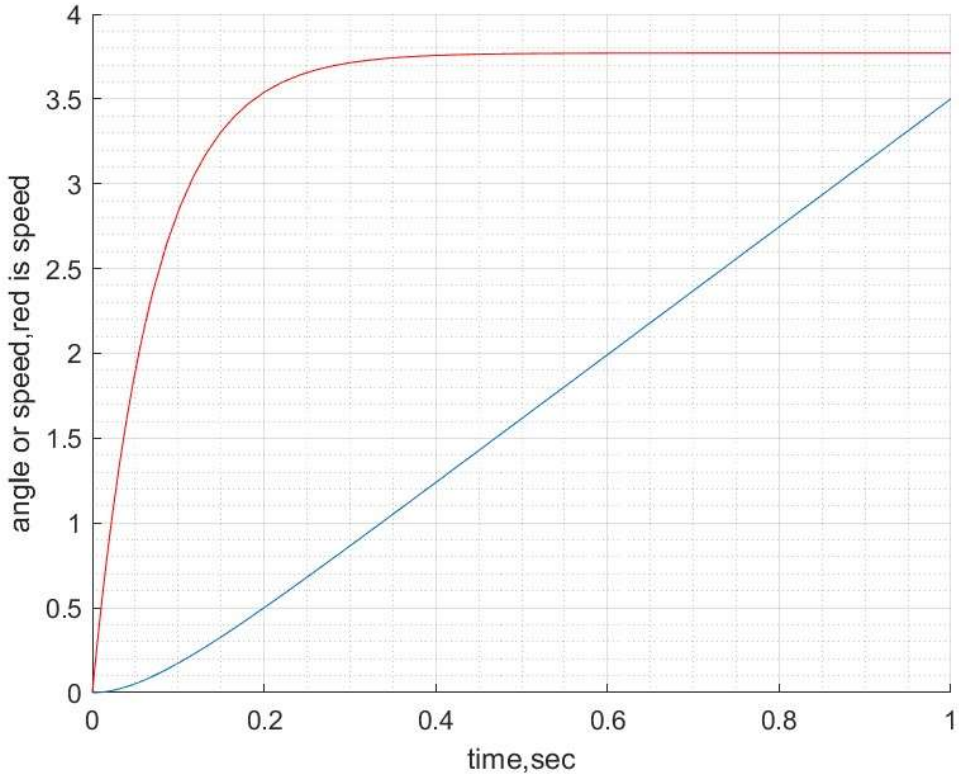

0
-1
-2
-3
-4
-5
-6
-7
-8
angle or speed,red is speed
0
0.2
0.4
0.6
0.8
1
time,sec

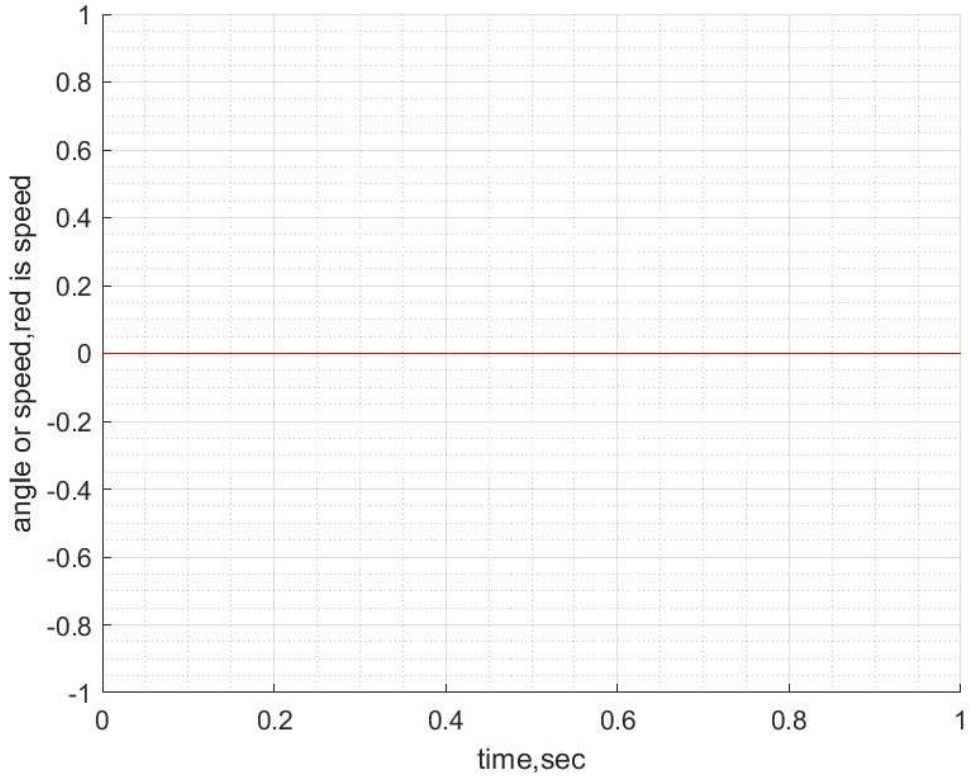

0
-2
-4
-6
-8
-10
-12
-14
angle or speed,red is speed
0
0.2
0.4
0.6
0.8
1
time,sec

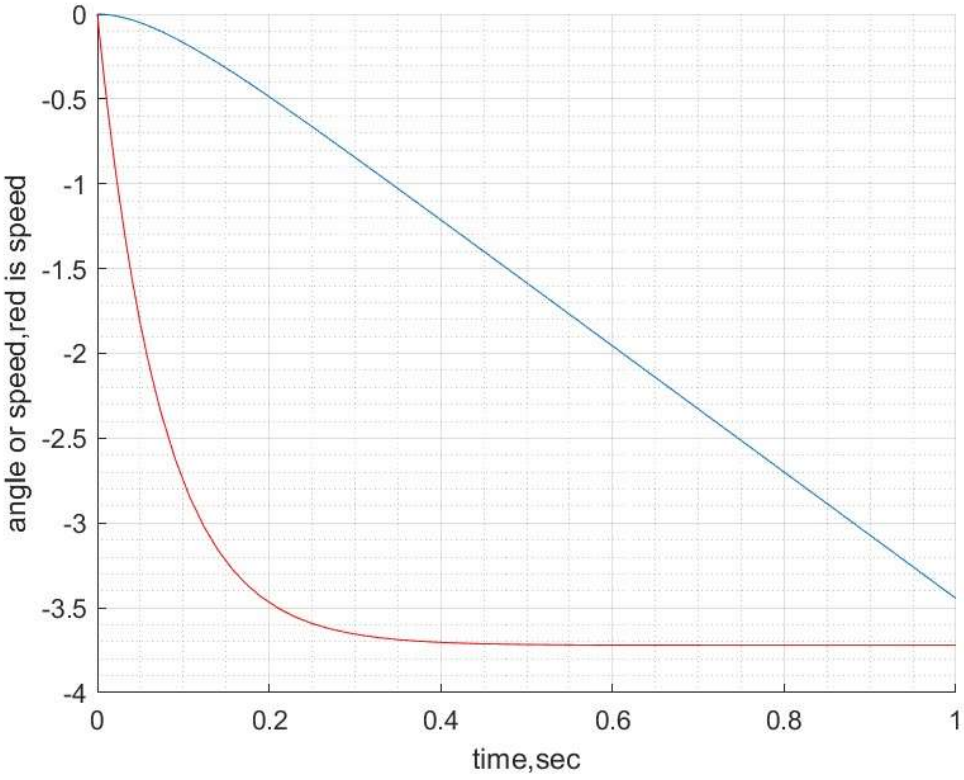

18
16
14
12
10
8
6
4
2
0
angle or speed,red is speed
0
0.2
0.4
0.6
0.8
1
time,sec

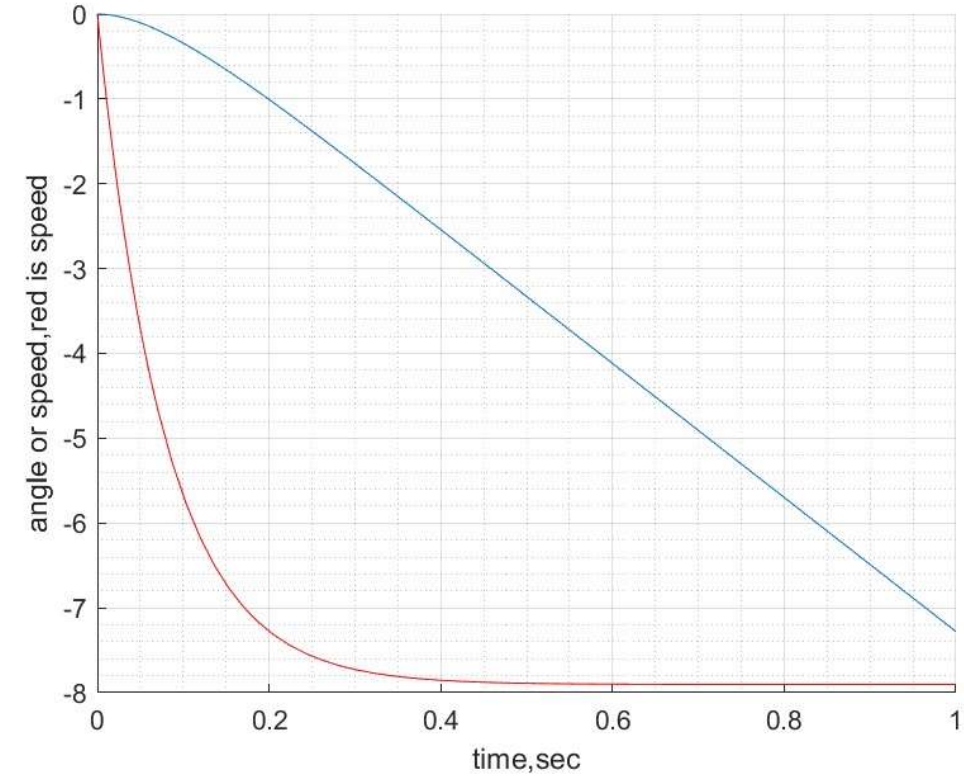

14
12
10
8
6
4
2
0
0
0.2
0.4
0.6
0.8
1
angle or speed,red is speed
time,sec

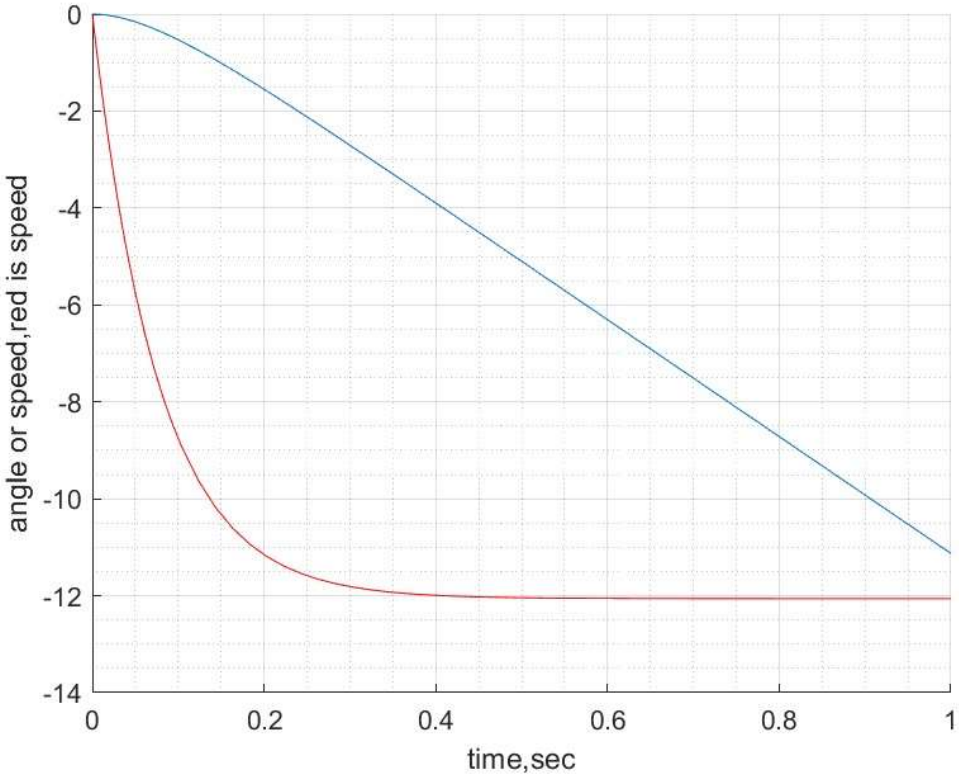

9
8
7
6
5
4
3
2
1
0
0
0.2
0.4
0.6
0.8
1
angle or speed,red is speed
time,sec


4
3.5
3
2.5
2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1
angle or speed,red is speed
time,sec


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.2
0.4
0.6
0.8
1
angle or speed,red is speed
time,sec


0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
angle or speed,red is speed
0
0.2
0.4
0.6
0.8
1
time,sec


0
-1
-2
-3
-4
-5
-6
-7
-8
angle or speed,red is speed
0
0.2
0.4
0.6
0.8
1
time,sec



My answers for question:

Why graphics look different: The influence from time on angle have a delay.
Why graphics look the same: The influence from time on omega almost have no delay.
Why from step 7: We want to fastest way to control the motor to right angle, so we should focus on angle rather speed.