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\_Data\_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)*

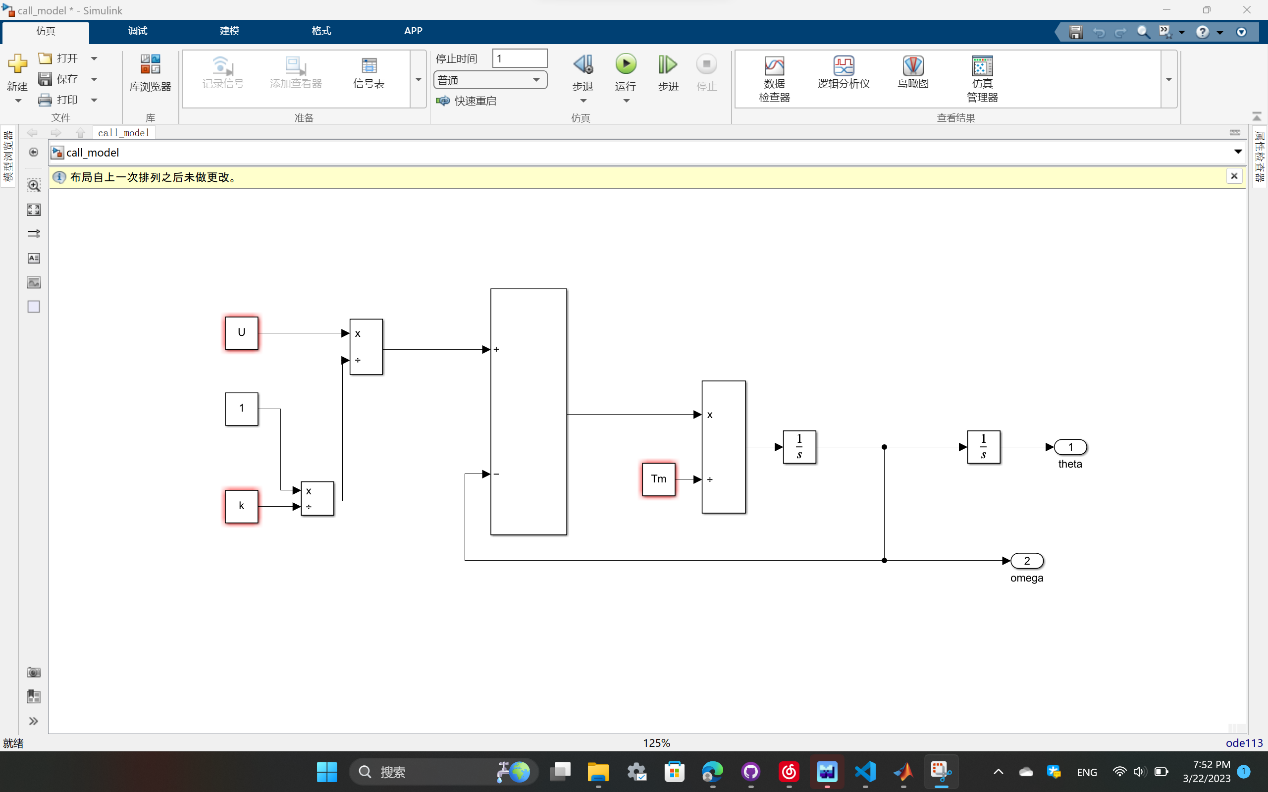
        data.write(str(max\_voltage\*(motor\_input-i\*25)/100)+' '+'{0:.3f}'.format(timenow)+' '+str(motor.speed)+' '+str(motor.position)+'\n')

        if timenow > total\_time:

            motor.run\_direct(duty\_cycle\_sp = 0)

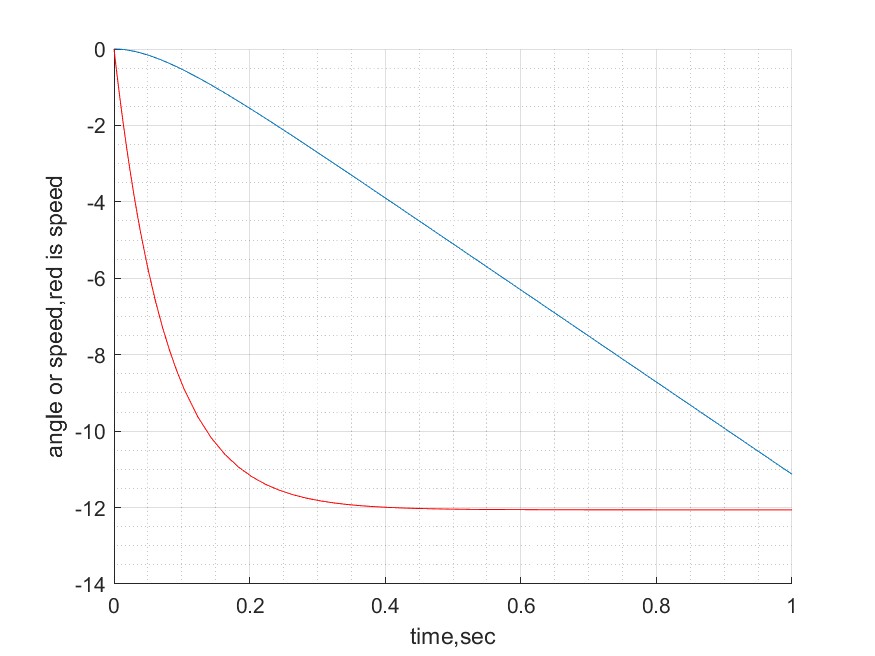
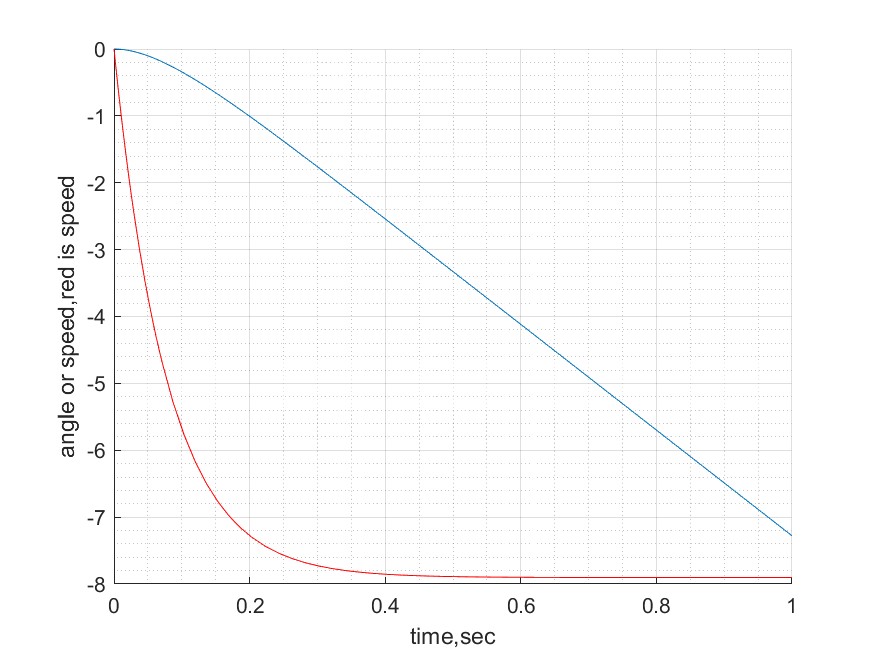
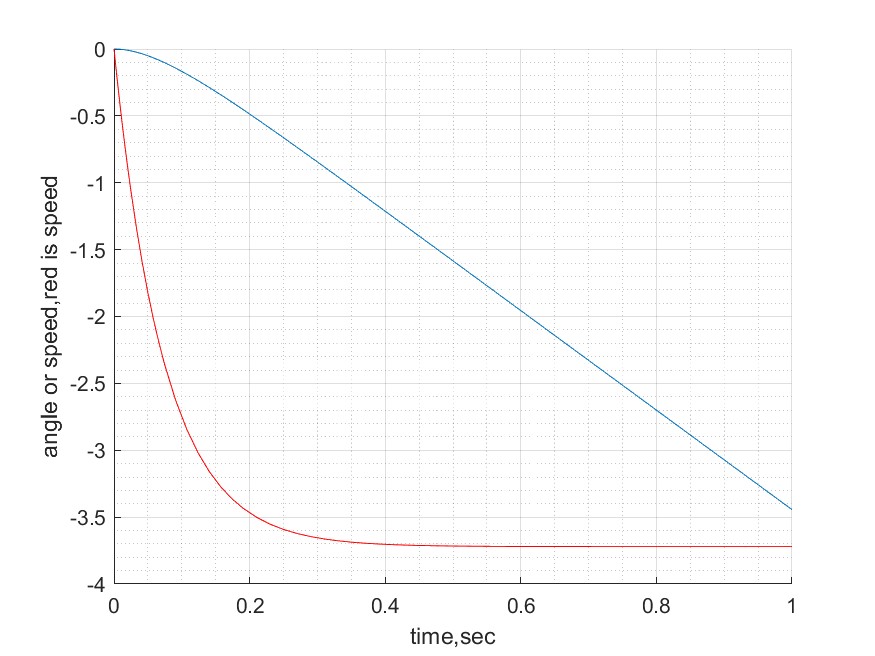
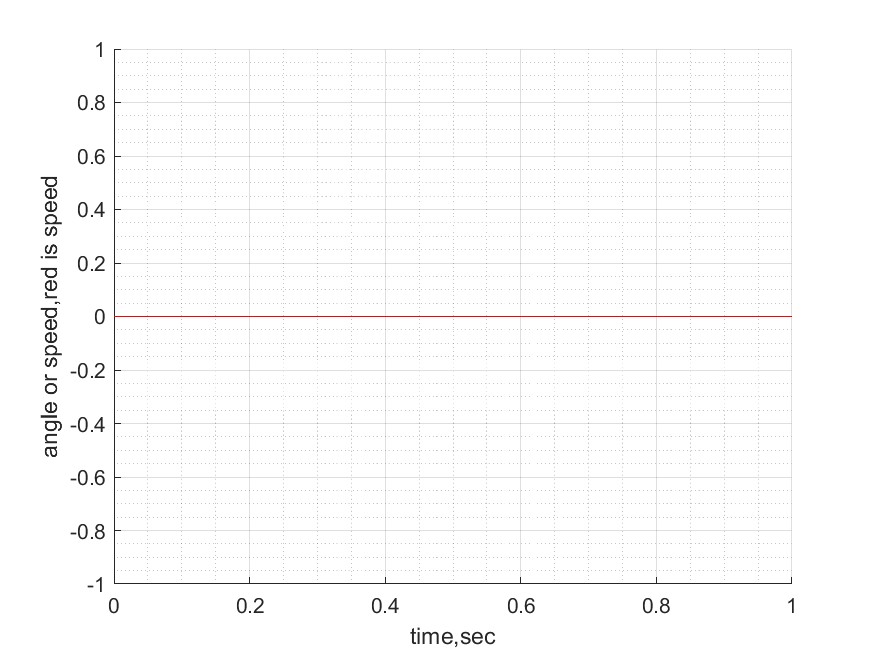
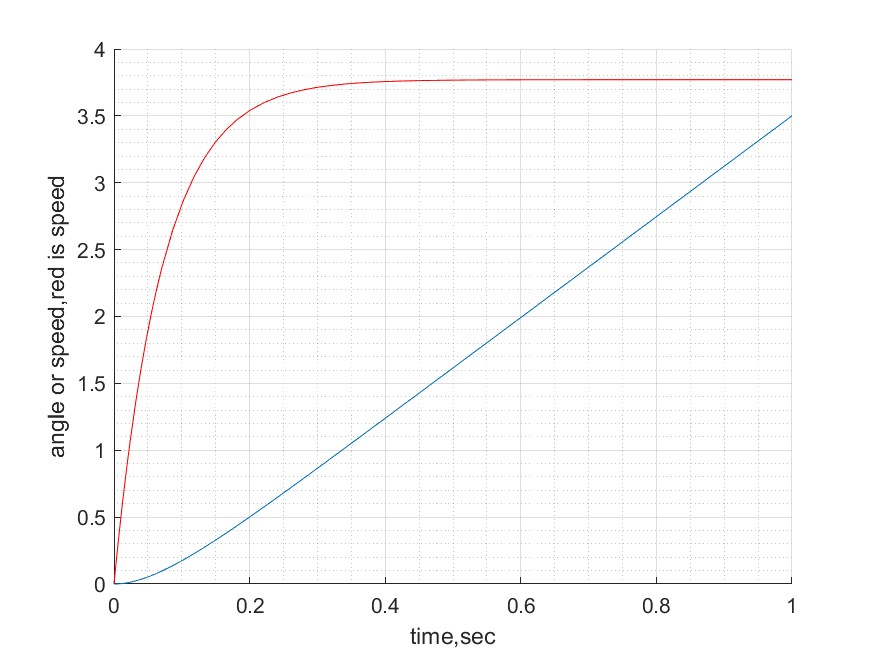
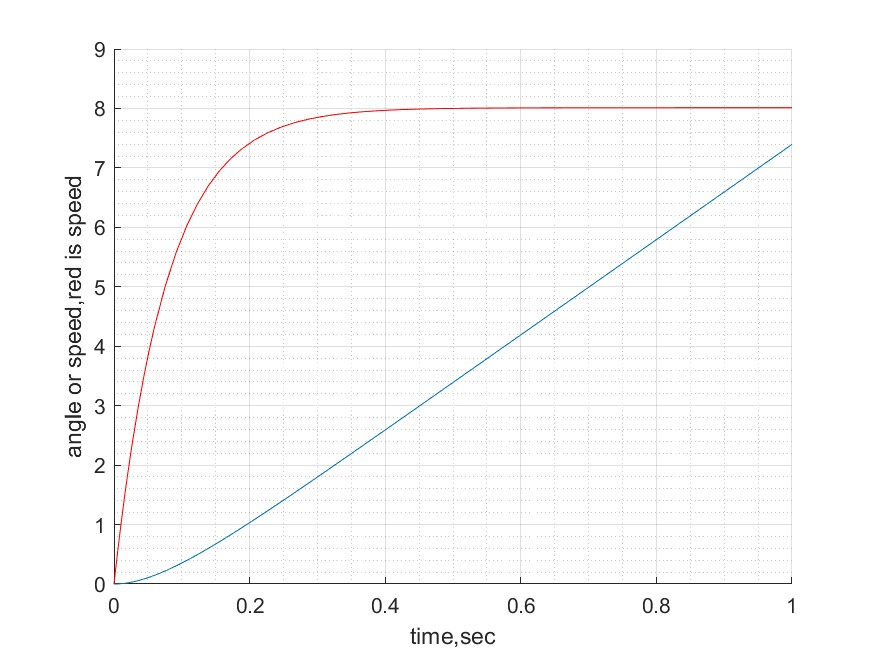
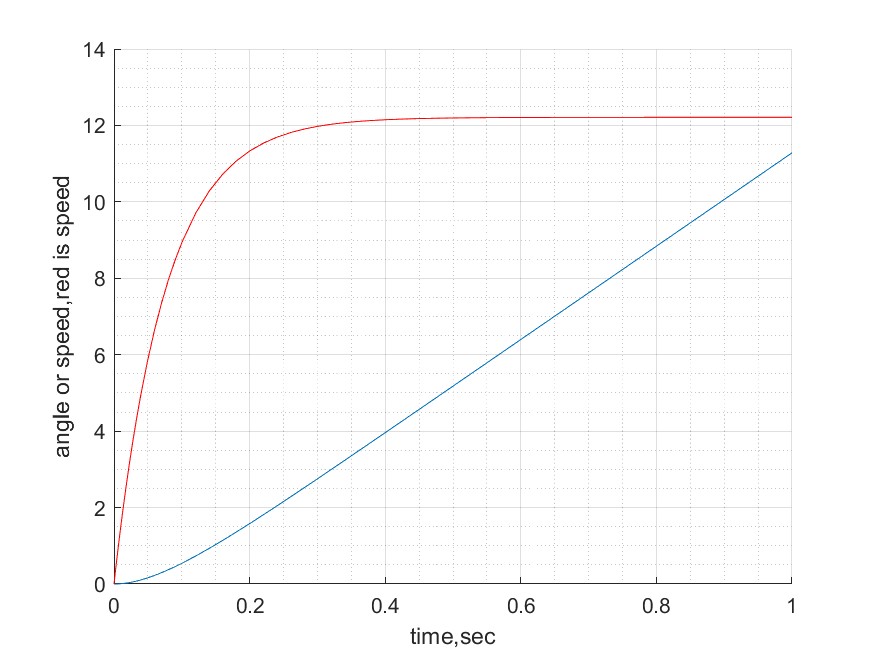
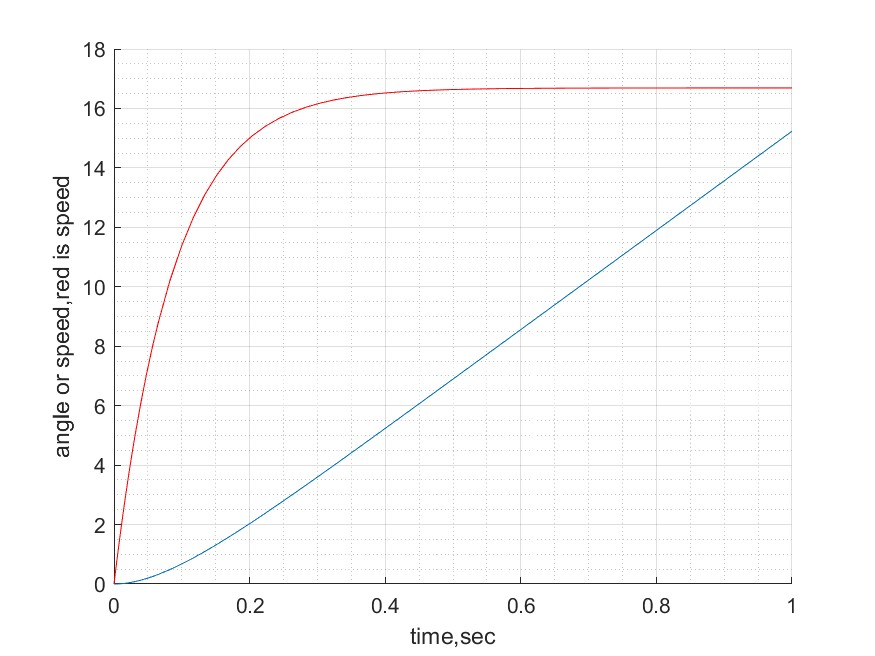
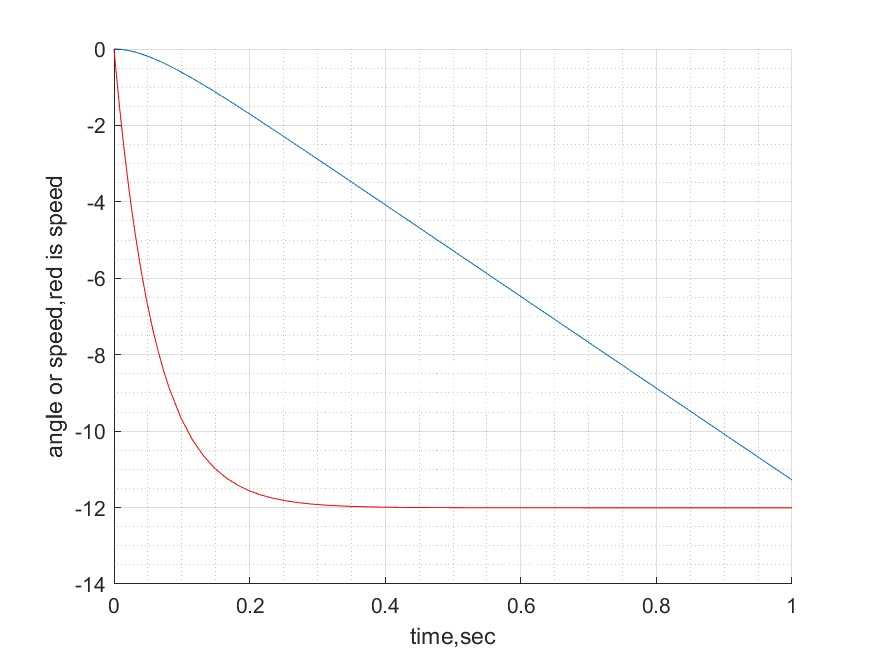
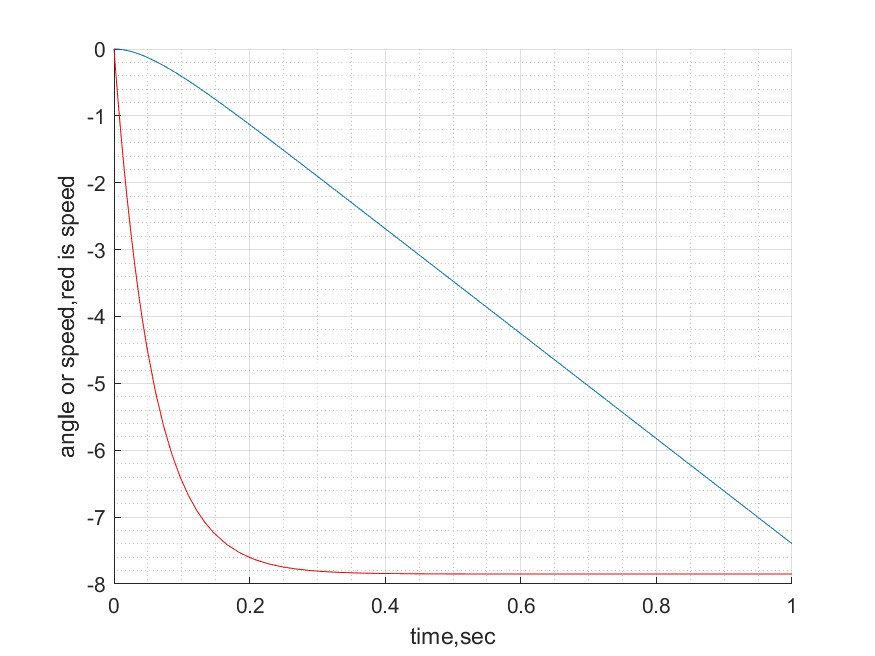
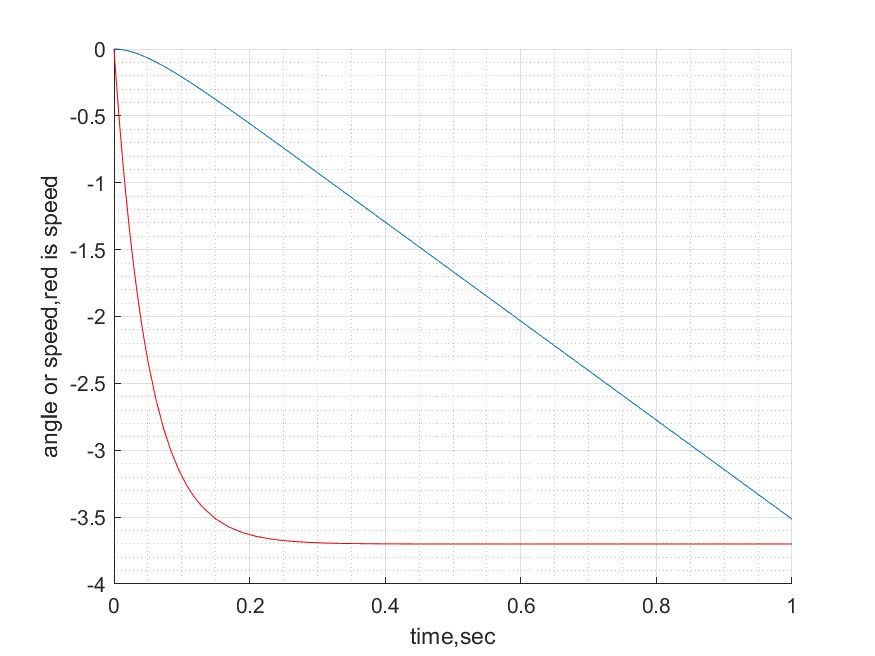
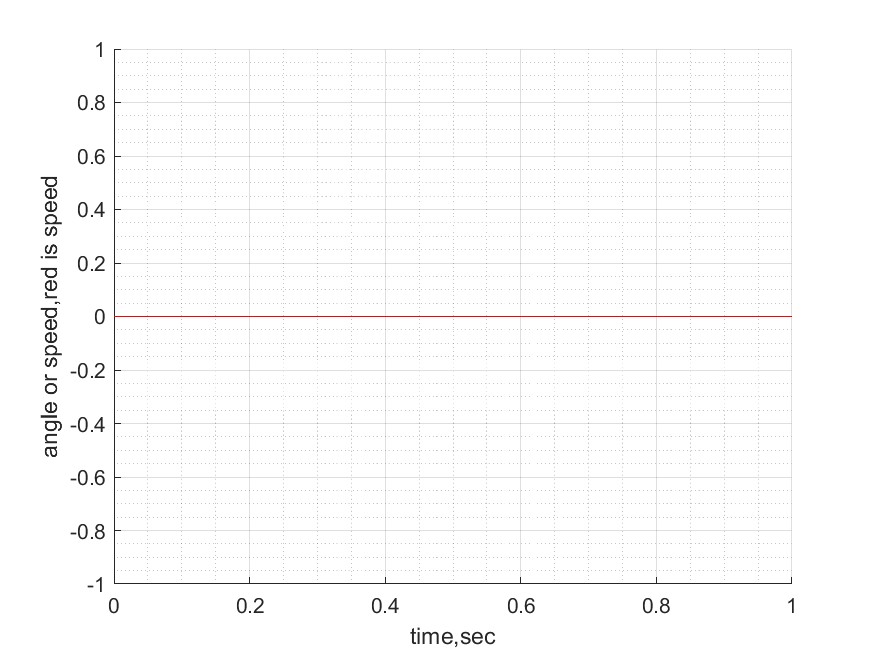
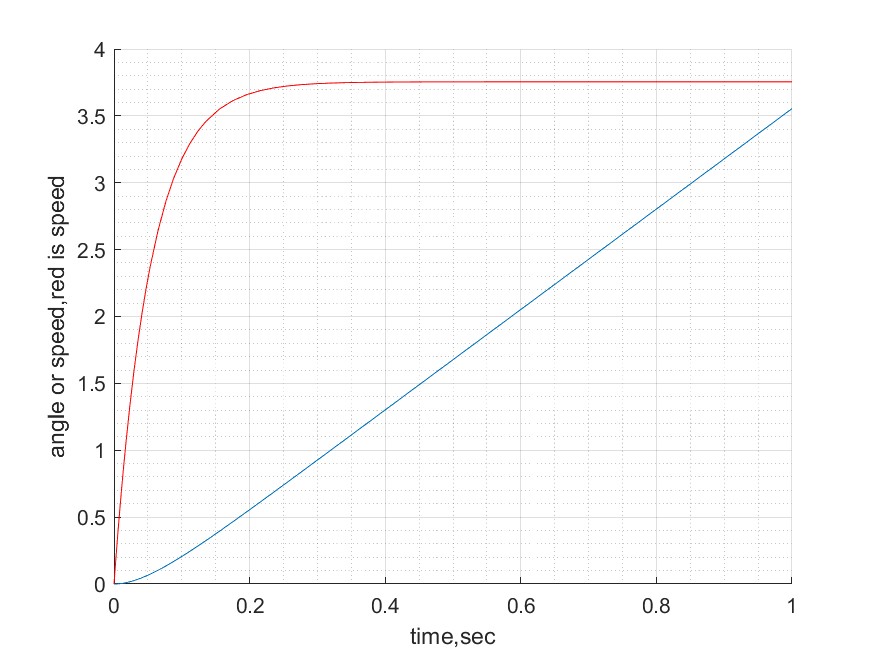
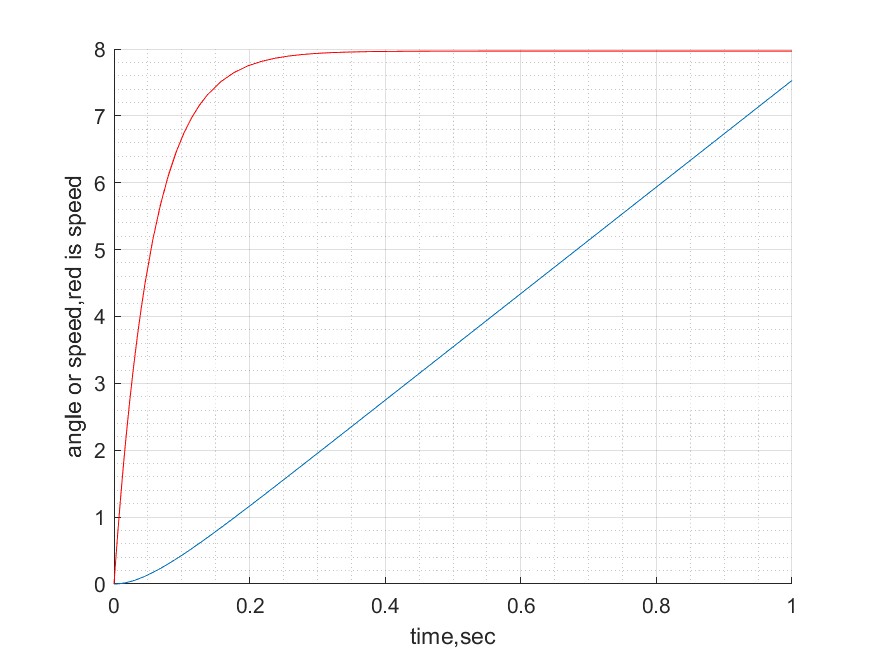
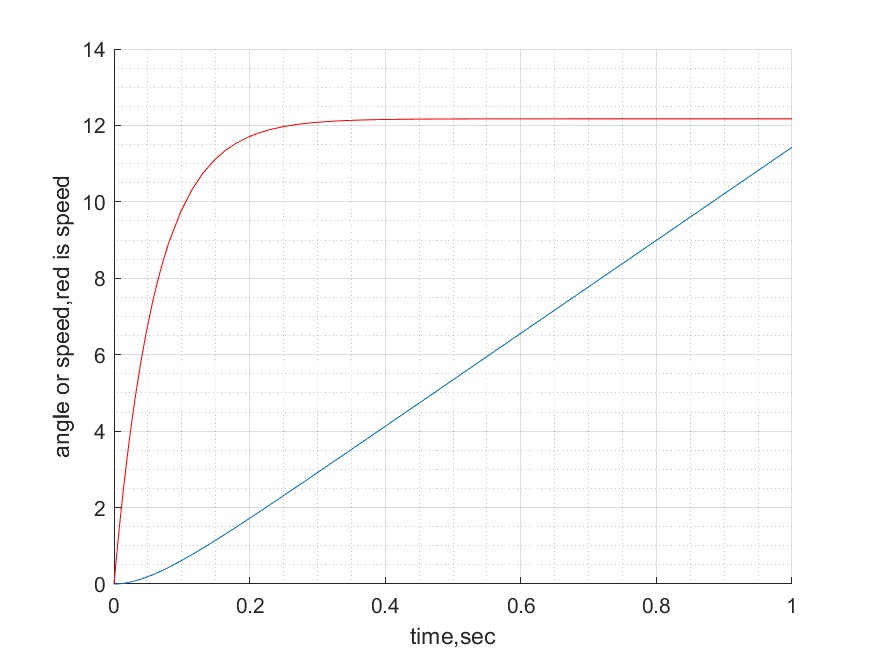
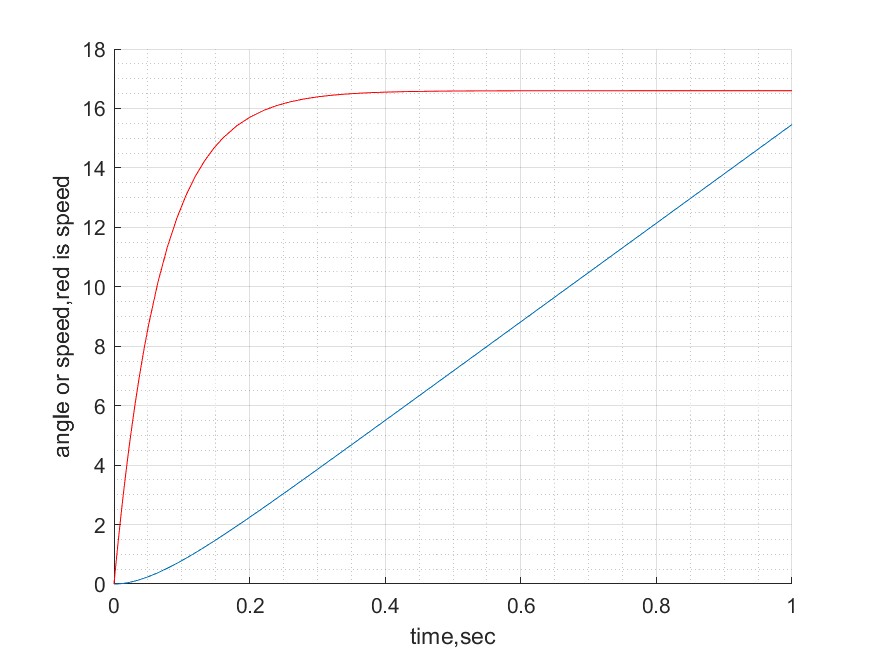
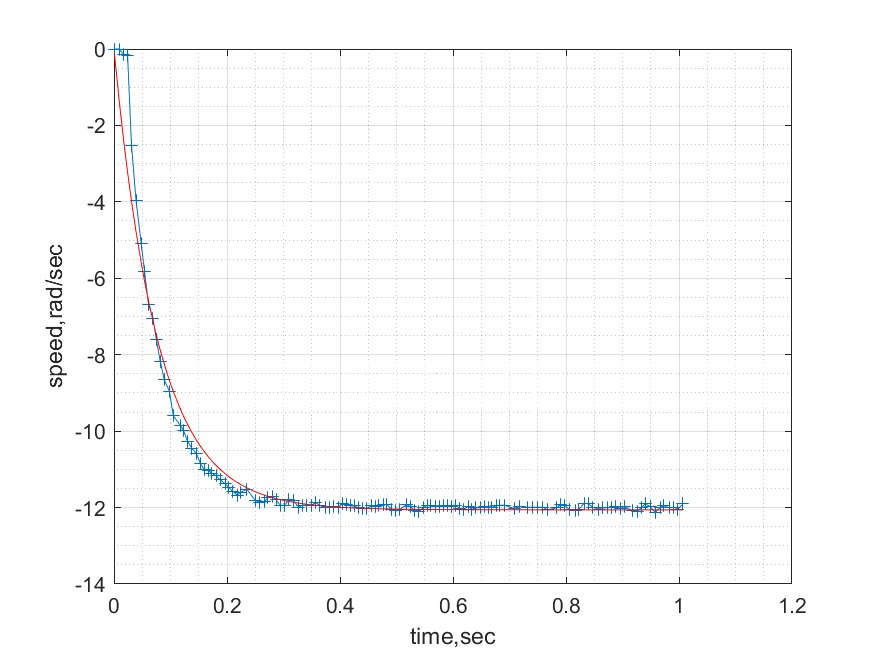
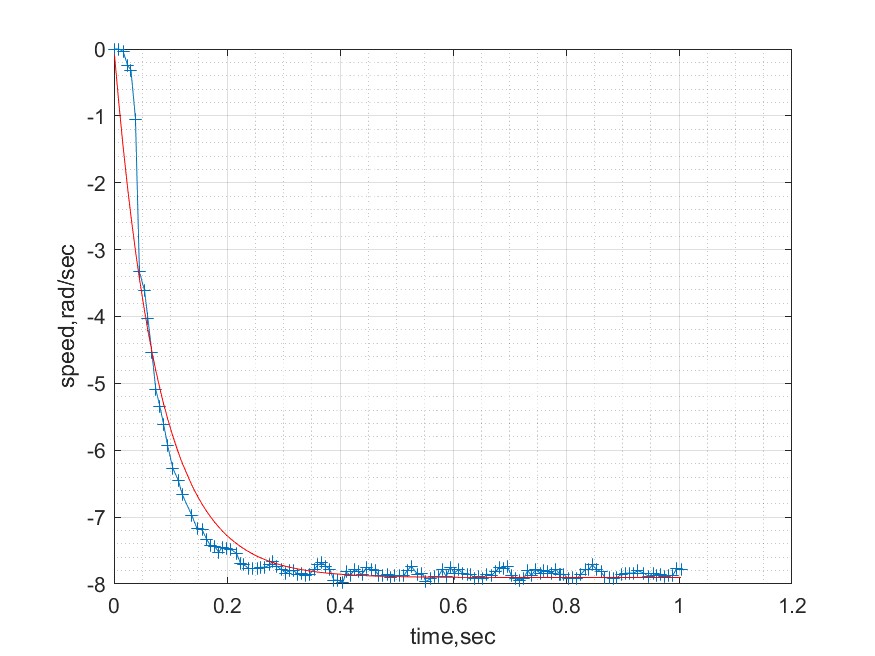
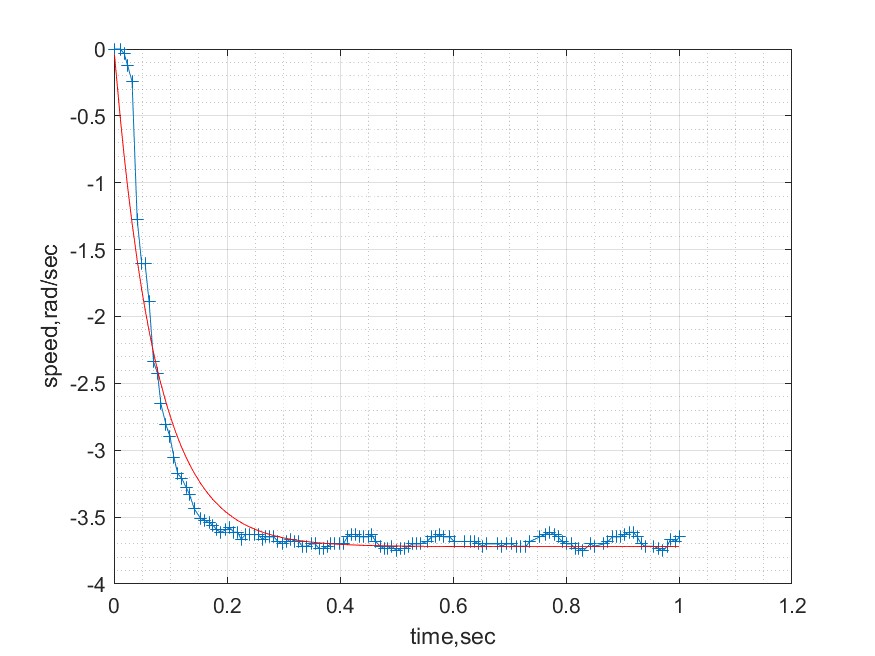
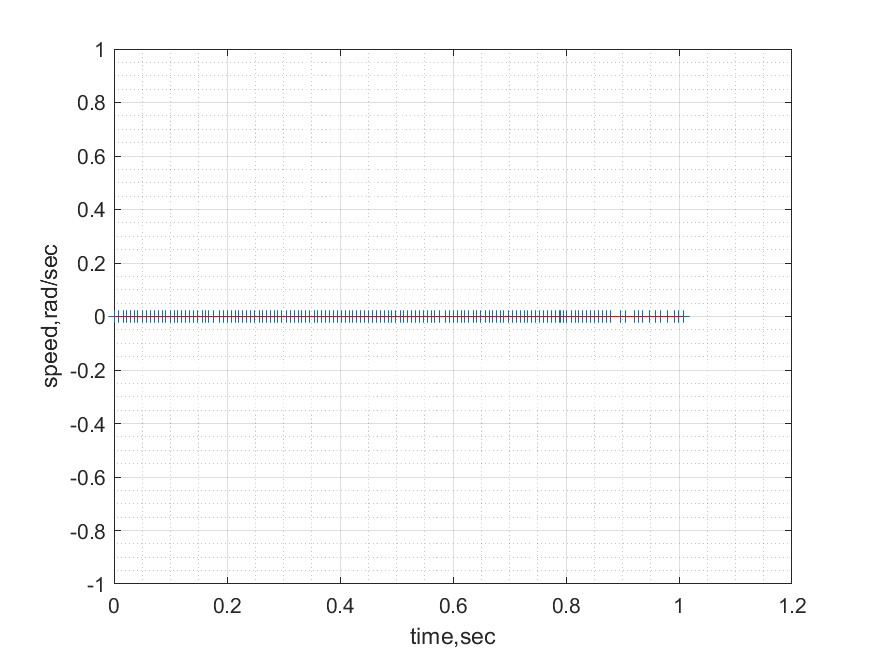
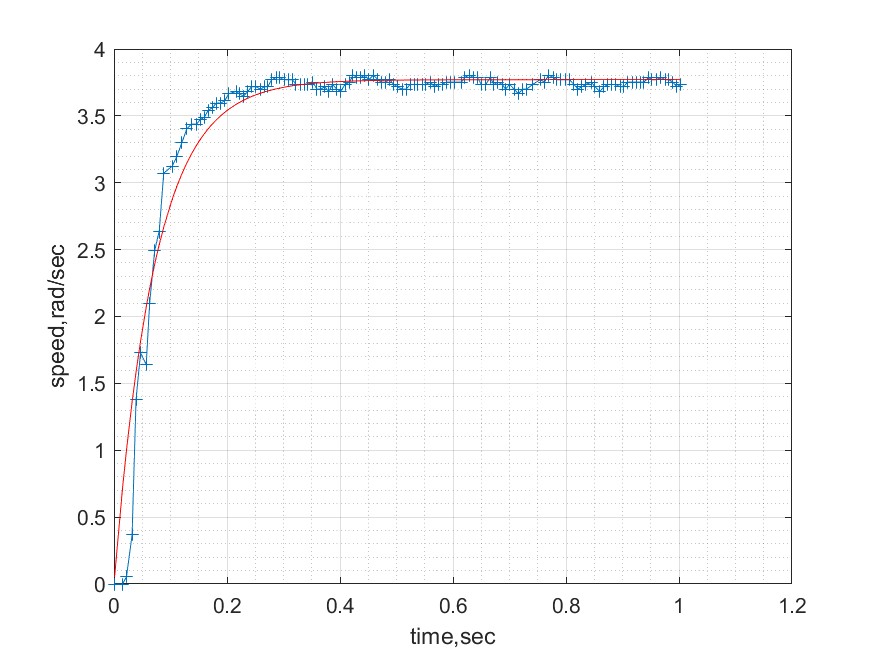
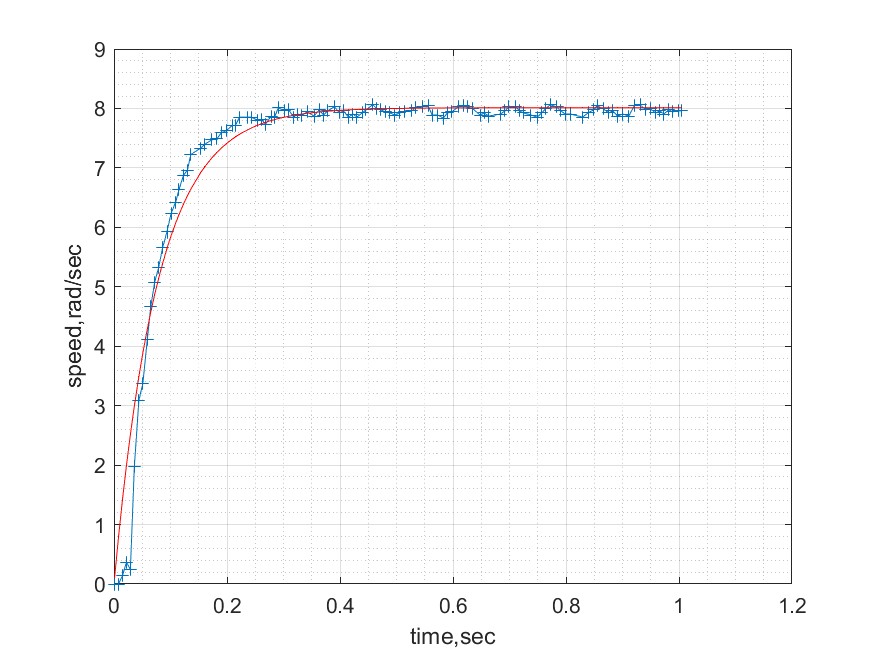
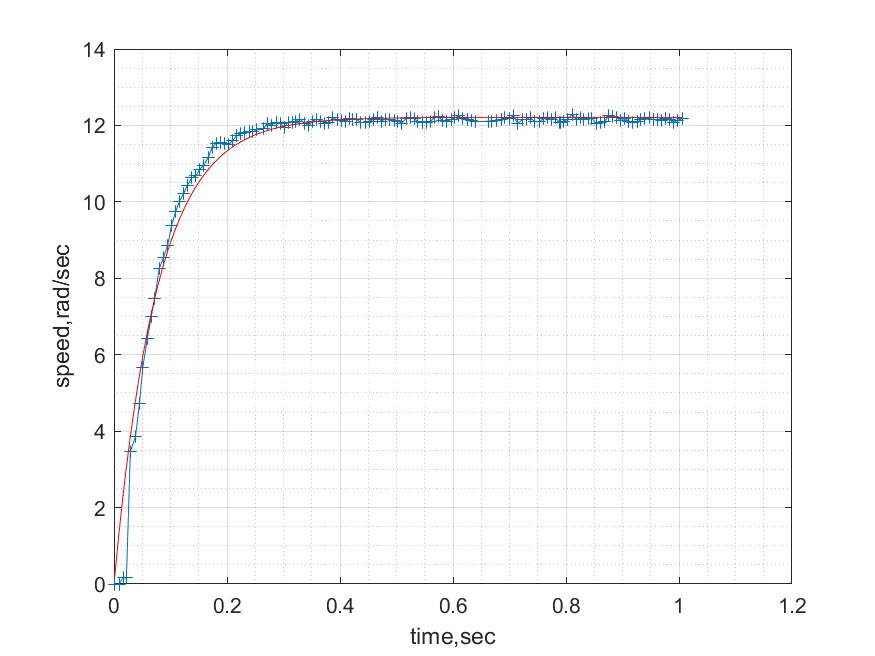
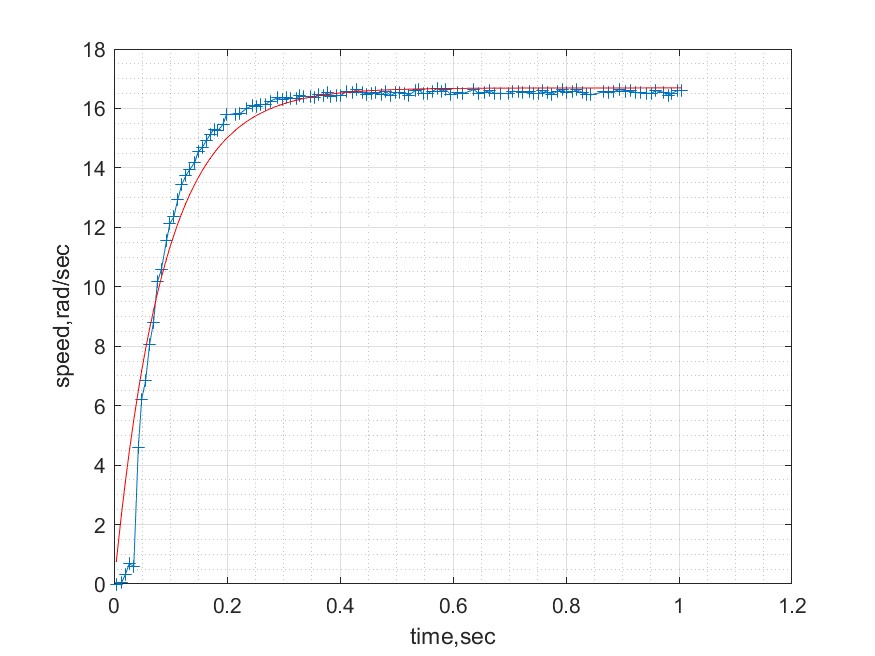
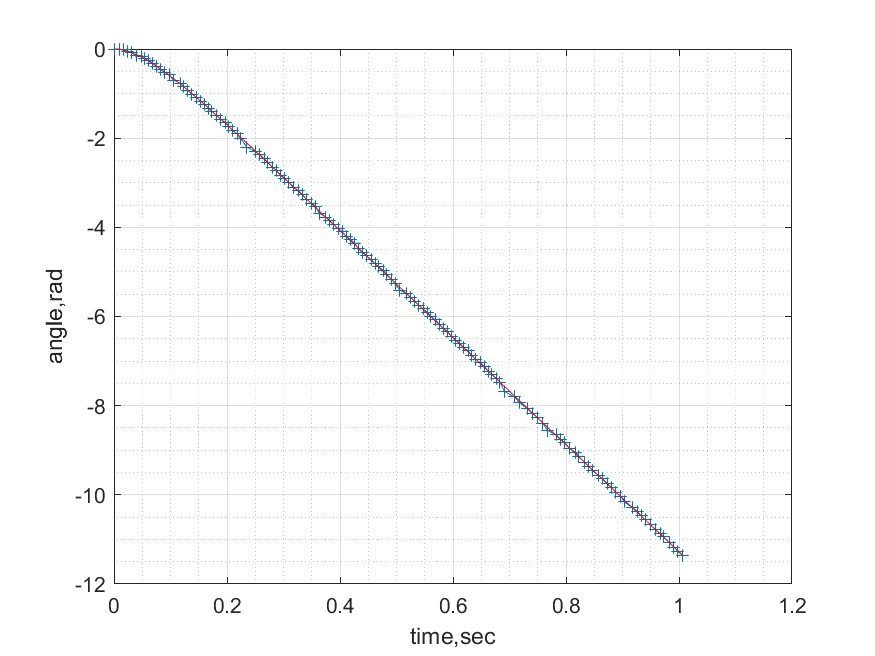
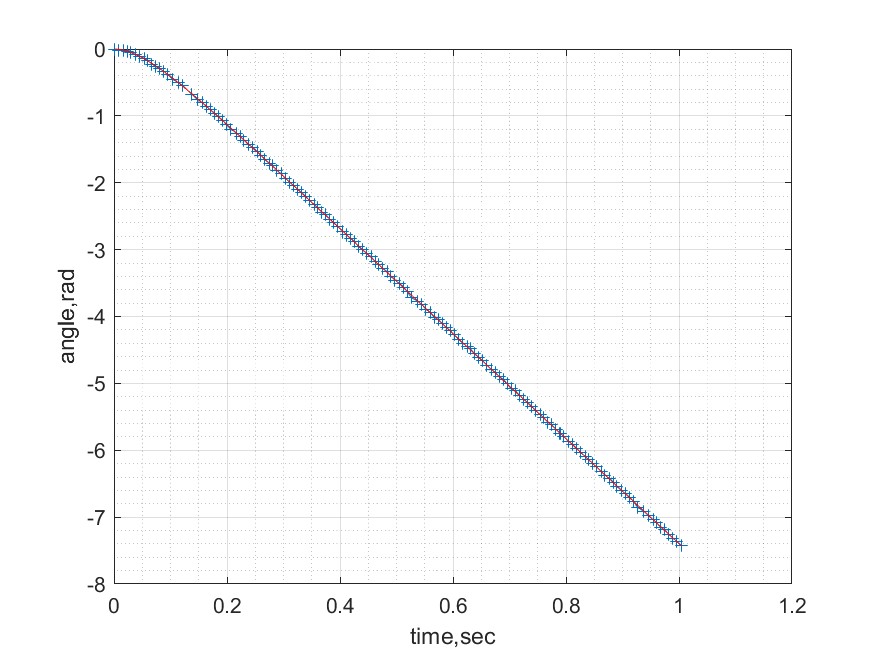
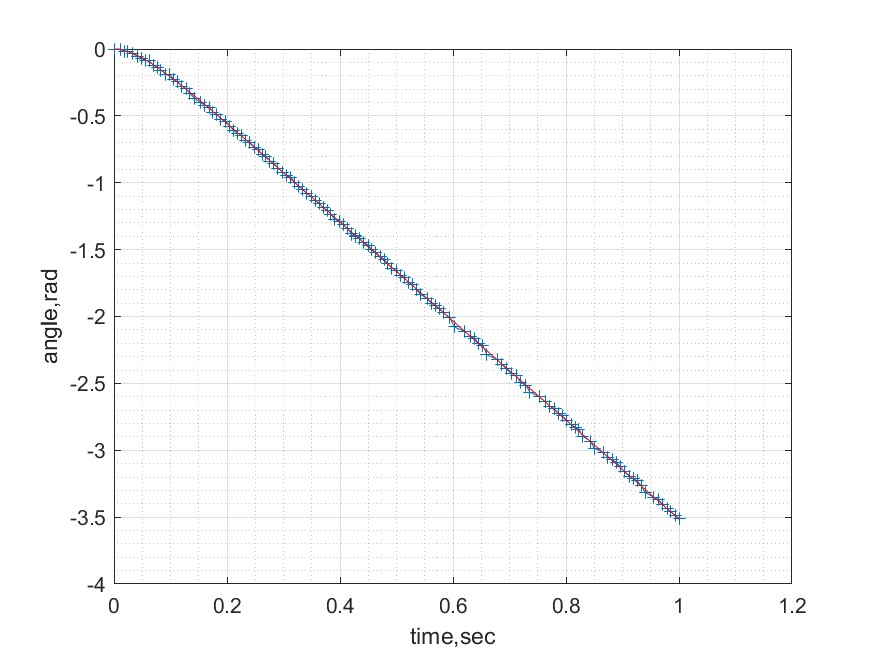
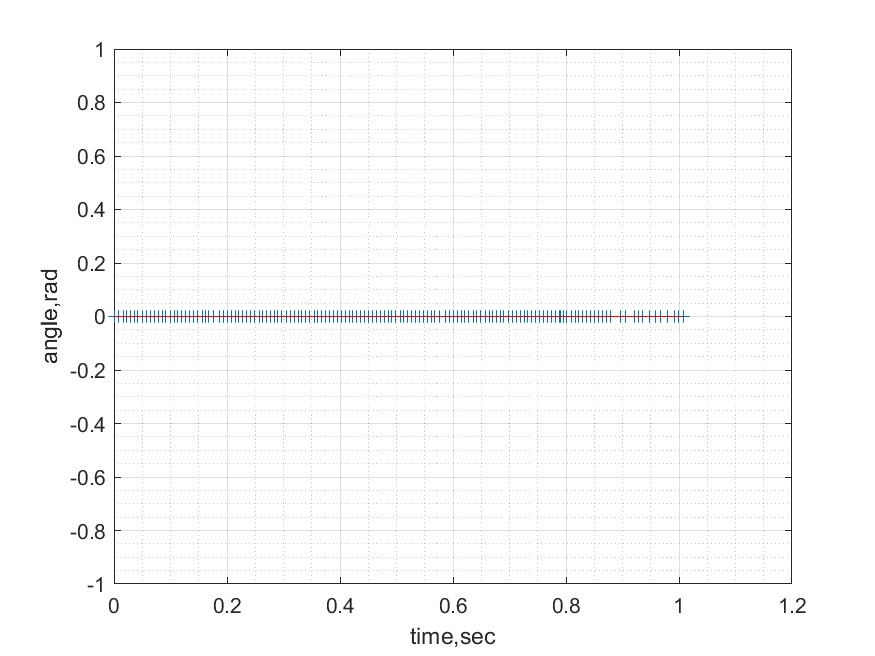
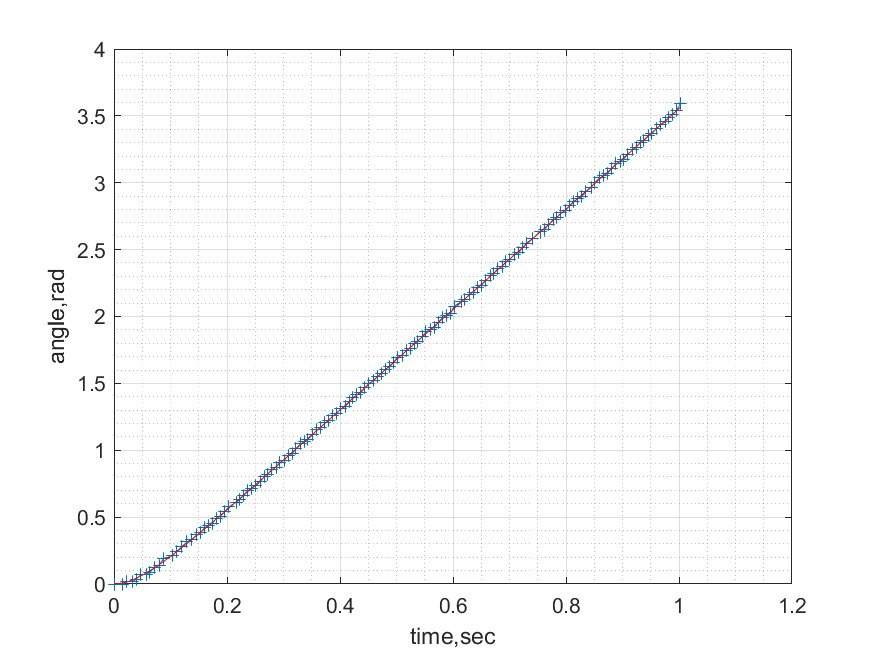
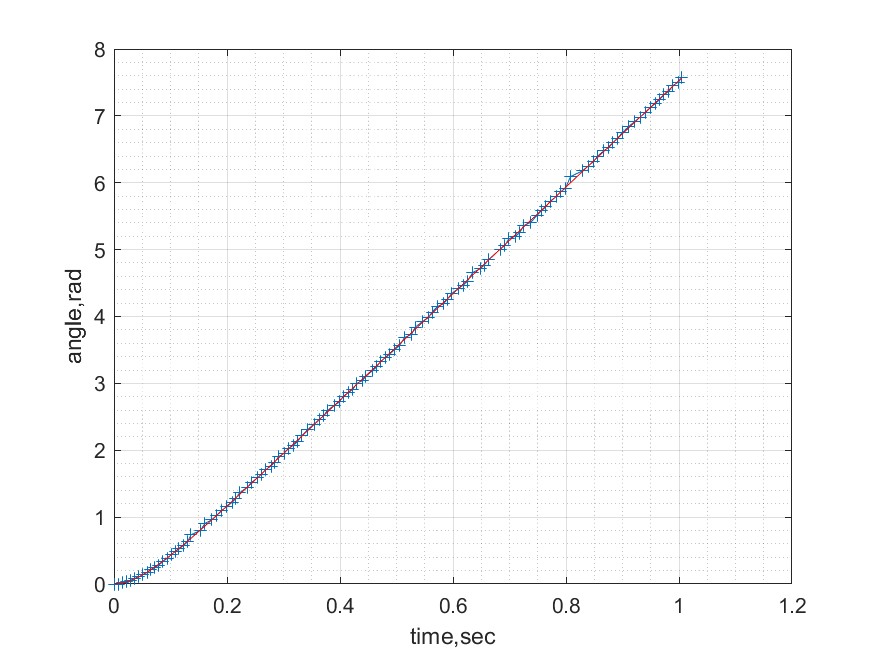
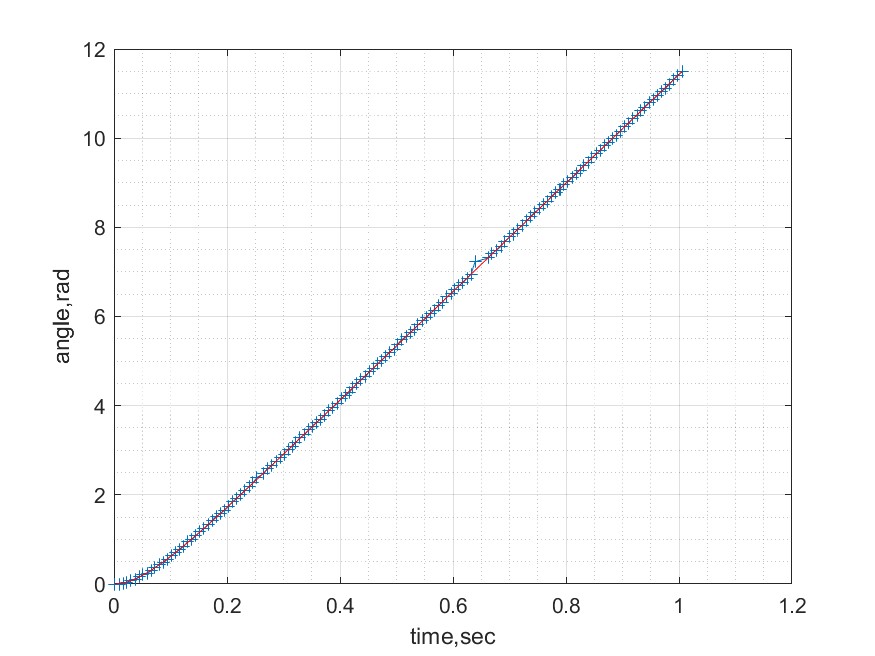
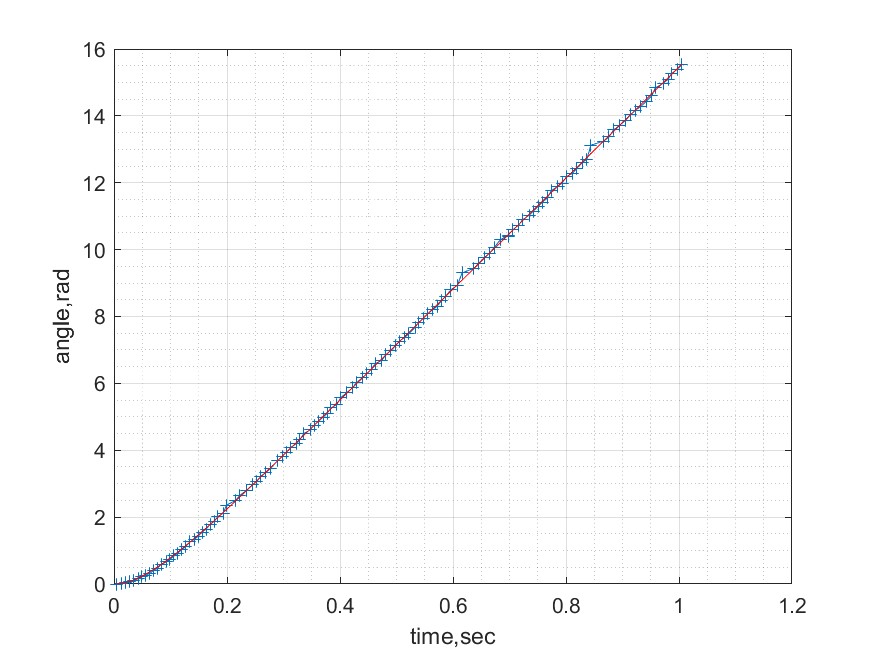
            break

Model



Plots in order of

(r&a\_w\_th, r&a\_w\_w, sim\_w&th\_th, sim\_w&th\_w)



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.