MF2007 - Workshop B

Adam Lang
861110-3956

Gabriel Andersson Santiago
910706-4538

Andreas Fröderberg
880730-7577

April 7, 2016

1 Servo Control

Level 1

The part for following the model was derived by inverting each block of the DC motor model step by step. This was then verified by using outputs from the DC motor model as input to the inversed model. This should result in that the input to the DC motor model is the same as the output for the inverse model. Since this is the case, this part can be seen as verified.

Values for a_{max} and v_{max} needed to be obtained in order to design the trajectory planner. v_{max} was read from the velocity plot when the motor model was fed with 24 V. a_{max} was calculated to a value around 650 $\mathrm{m\,s^{-2}}$ but since this value saturates the voltage from the model follower, therefore was a_{max} tweaked into a value which never saturated the voltage. These derived values can be seen in Table 1.

Table 1: Max values for acceleration and velocity

a_{max}	255
v_{max}	270

The signal from the trajectory planner with Rs = 10 and Rs=100 can be seen in Figure 3.

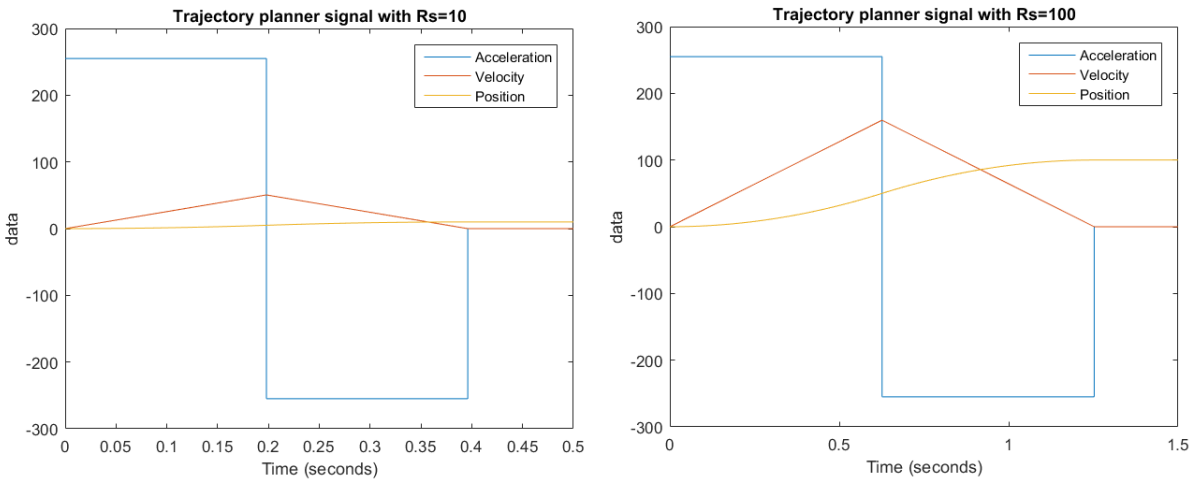

Figure 1: The trajectory planner with different Rs values

2 Writing Controller Code

Level 1

When comparing the Simulink blocks to the code equivalent it could be seen that this gave similar results, this can also be seen in Figure 2 and Figure 3. A sampling time of 10 ms was used for the code, and when experimenting with different sampling times it was noticed that a higher sampling time gave an increase on the trajectory planner. When a higher sampling time was used, was the change of acceleration missed by a few milliseconds and is therefore impacting the velocity and position. This gives the difference in the position of the motor which is seen in Figure 3.

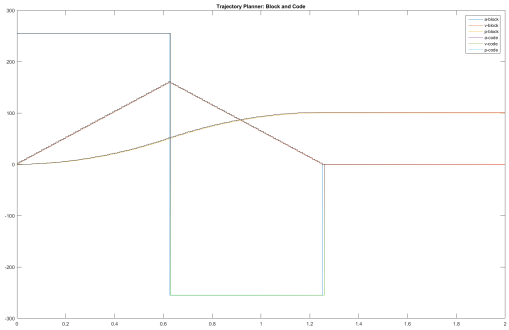

Figure 2: Trajectory planner signal comparison

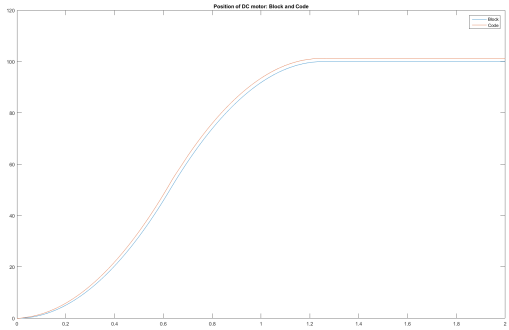

Figure 3: Motor position comparison

A close-up of the sampling time problem is shown in Figure 4. When the velocity increases it the code signal follows perfectly but when there is a change in acceleration the velocity code signal gets "out-of-sync".

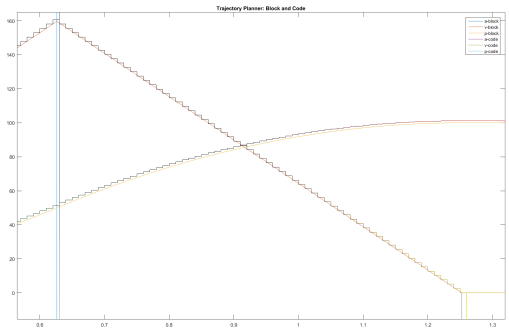

Figure 4: Close-up of trajectory planner signal

Level 2

Some modifications to the code had to be made to make it work in dSPACE but the same code is used for the plot in Figure 5. The performance of the trajectory planner in dSPACE worked excellent. It behaved almost exactly like in Simulink on the DC-motor model.

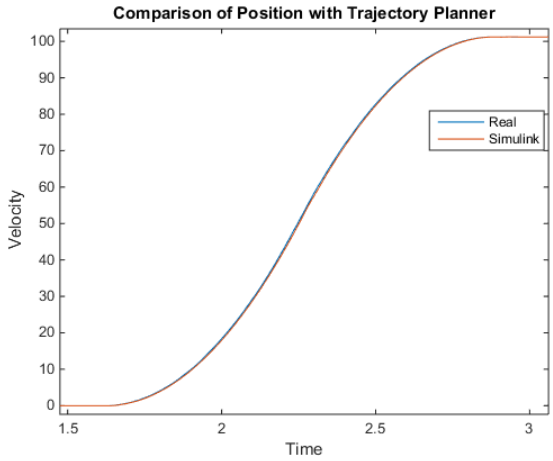

Figure 5: Comparison on the performance of the trajectory planner

The code for the embedded controller can be found in Listing 1.

Listing 1: Embedded Matlab Code

```
function [acceleration,velocity,position,voltFollowing,volt,err]  = Controller(
    feedback)
%#codegen
% PERSISTENT VARIABLES
persistent time;
if isempty(time)
    time = 0;
end

persistent oldAcc;
if isempty(oldAcc)
    oldAcc = 0;
end

persistent oldVel;
if isempty(oldVel)
    oldVel = 0;
end
```

```
persistent oldPos;
if isempty(oldPos)
    oldPos = 0;
end
%Update "time"


% Trajectory Planner Variables
Ts = 0.01;
vmax = 270;% 252;
amax = 255; %369;
amin = -amax;
rs = 10;
t1 = sqrt(rs/amax);
t2 = t1;
t3 = t1+t2;
voltmax = 24;

%Initial Value block
out1 = amax;
%%SIGNAL GENERATOR
%Reset to 0 at t1 block
out2 = 0;
if(time >= t1)
    out2 = amin;
end

%retardation at t2 block
out3 = 0;
if(time >= t2)
    out3 = amin;
end

% Reset at t3 block
out4 = 0;
if(time >= t3)
    out4 = amax;
end

acceleration = out1+out2+out3+out4;
oldAcc = acceleration;
%% INTEGRATOR PARTS
%For velocity
velocity = oldVel + Ts*(acceleration+oldAcc)/2;
oldVel = velocity;

%For position
position = oldPos + Ts*(velocity+oldVel)/2;

oldPos = position;

%% MODEL FOLLOWING PART

% Model Following variables
J1 = 6e-4;
motor_j = 7.46e-7;
n = 5;
motor_k = 0.0697;
motor_R = 112;
motor_d = 1e-5;
%% Input Acceleration Part

accPart = acceleration*(J1/(n^2)+motor_j);
```

```
%% Input Velocity Part

velPartK = velocity*motor_k;
velPartD = velocity*motor_d;

%%Output

voltFollowing = (accPart+velPartD)*motor_R/motor_k + velPartK;


%% PID CONTROLLER

% PERSISTENT VARIABLES
persistent oldErr
if isempty(oldErr)
    oldErr = 0;
end
persistent olderErr
if isempty(olderErr)
    olderErr=0;
end
persistent oldPIDVolt
if isempty(oldPIDVolt)
    oldPIDVolt = 0;
end
persistent olderPIDVolt
if isempty(olderPIDVolt)
    olderPIDVolt = 0;
end
% PID parameters
s2 = 1348;
s1 = -1899;
s0 = 692;
r0 = 0.89;

% Error
err = position - feedback;

%%PID
PIDVolt = s2*err+s1*oldErr+s0*olderErr+r0*olderPIDVolt-(r0-1)*oldPIDVolt;
olderPIDVolt = oldPIDVolt;
oldPIDVolt = PIDVolt;
olderErr = oldErr;
oldErr = err;
%% OUTPUT

volt = voltFollowing+PIDVolt;

if(volt >= voltmax)
    volt = voltmax;
elseif(volt <= -voltmax)
    volt = -voltmax;
end


time = time +Ts;

end
```

3 Robustness to parameter uncertainty and sensor noise

Level 1

Since the plant have a transfer function as,

$$G_o(s) = \frac{25.15}{s + 2.157} \tag{1}$$

and that an $A_m(s)$ of same order as the plant should be chosen, the following functions was used,

$$A_m(s) = s + \omega_1$$
$$A_o(s) = s + \omega_2$$

with

$$\omega_1 = 10$$
$$\omega_2 = 5.$$

These values gives the plot seen in figure 6.

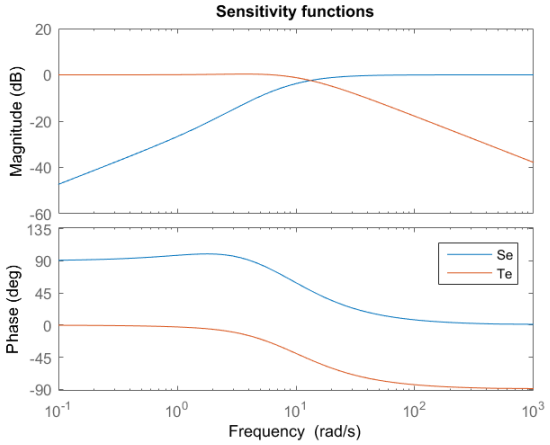

Figure 6: The sensitivity and conmplementary sensitivity function.

When the poles for $A_0(s)$ is increased to five times $A_m(s)$ the results in figure 7 is obtained.

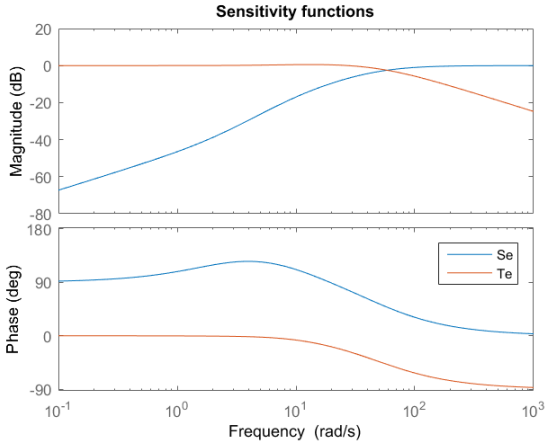

Figure 7: S_e and T_e with five times faster $A_o(s)$ poles.

From the plots we can see that there is a possibility to adjust the system to be less sensitive to noise and disturbances from either the sensor or the plant. This means that if there is a good plant model, there is less sensitivity to sensor noise and vice versa.

Level 2

The velocity controller from Exercise 3 was used as controller. By following the first part of the task was the values in Table 2 chosen.

Table 2: Values for the velocity controller

Am	500
Ao_1	1000
Ao_2	2000

1

Figure 8 shows the step responses for two different values of A_o. A force of 5000 N is applied at the time 0.03 seconds.

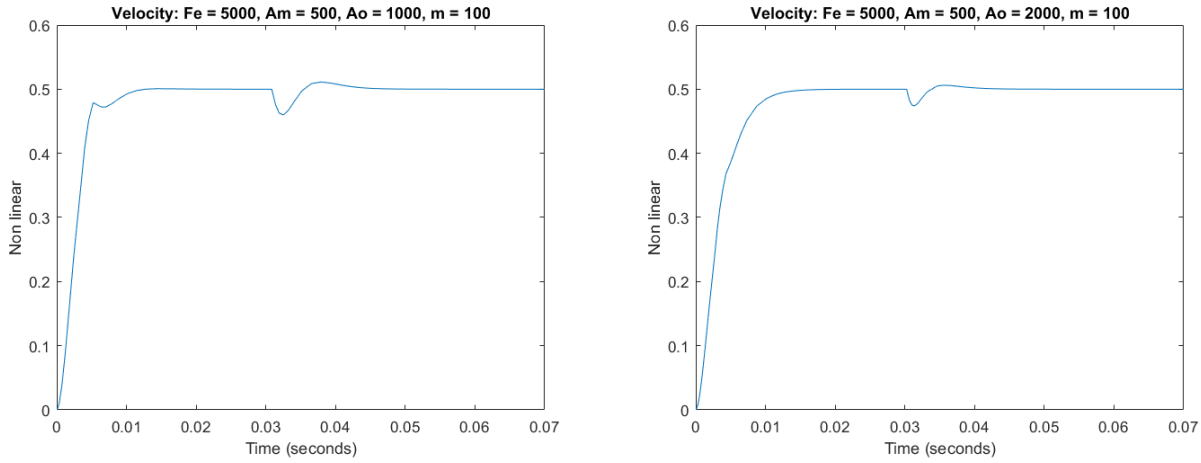

Figure 8: Comparison between the two different A_o Values

2

Figure 9 shows the step responses for two different values of A_o and now with a mass of 200 kg instead of 100 kg. The system behaves more or less the same as with a mass of 100 kg.

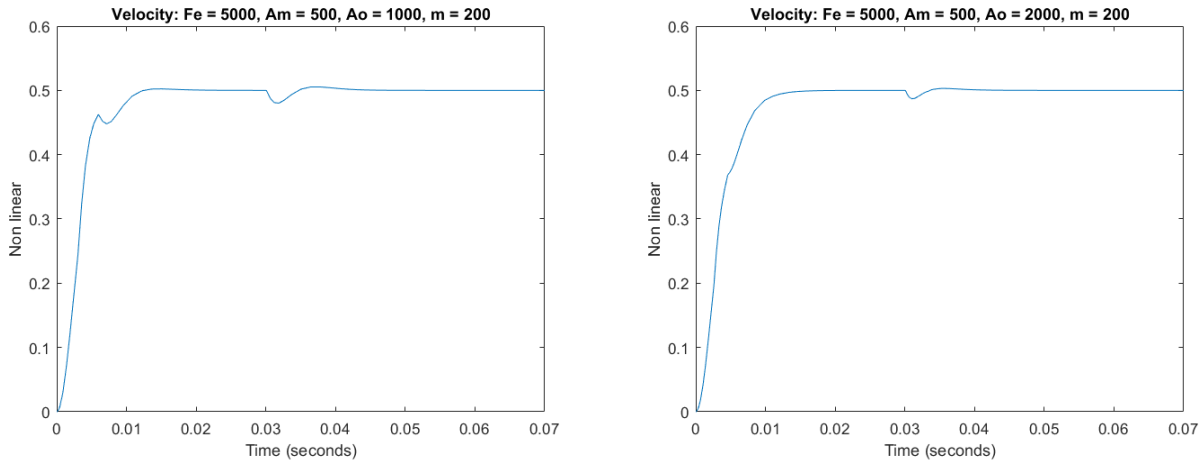

Figure 9: Comparison between the two different A_o values

3

Figure 10 shows the step responses for two different values of A_o with a sine wave as noise. The sine wave has a frequency of 1700 rad/s.

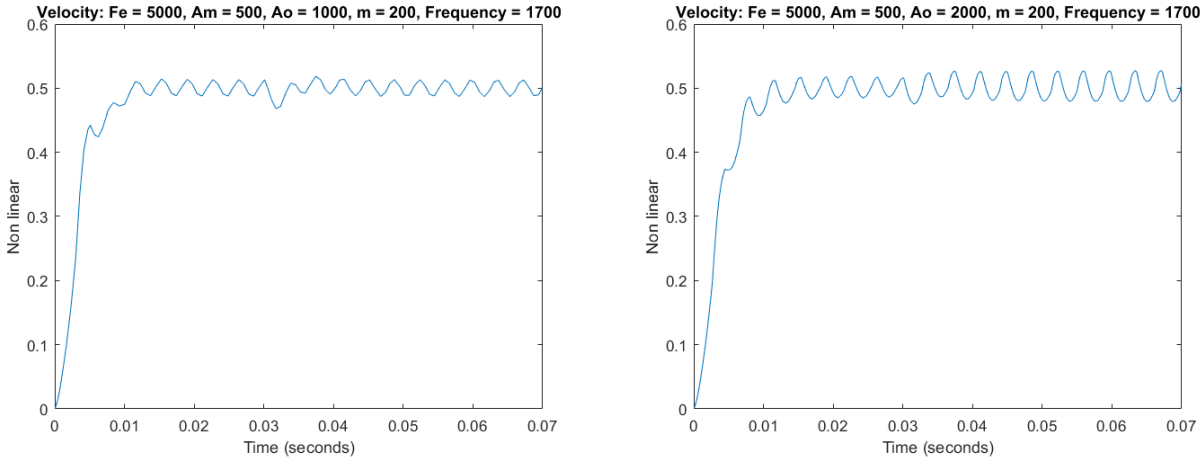

Figure 10: Comparison between the two different A_o values

Figure 11 shows the complementary sensitivity function for the two different A_o values. The magnitude when the frequency is 1700 rad/s is shown which shows that a lower A_o value dampens the noise more. It was hard to find a frequency were the noise dampening were very noticeable but the dampening can be seen in Figure 10 were the noise affects the system less when A_o=1000 compared to A_o=2000.

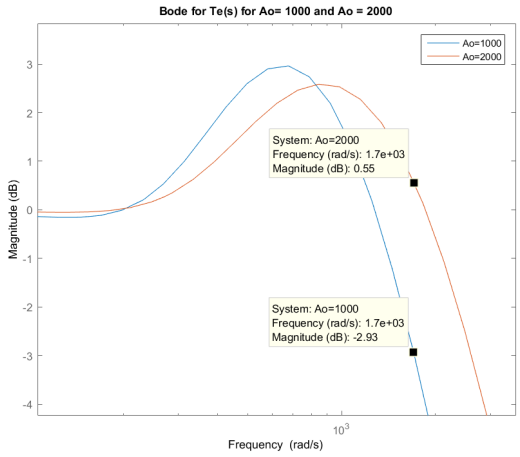

Figure 11: Comparison between the complementary sensitivity functions