MF2007: Dynamic & Motion Control
Workshop 2

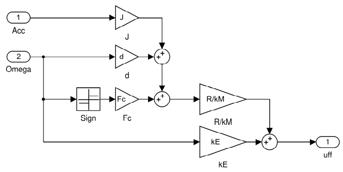
Kilian DEMEULEMEESTER, Jeremy POUECH
{kiliande,pouech}@kth.se

March 27, 2017

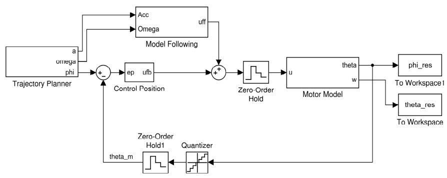
Contents

Abstract

This paper summarizes our work on servo control, code implementation on a microprocessor and analysis of the robustness to parameters uncertainties and sensor noise. The first and second parts use the result from workshop A about how to control a DC-motor in position, improving the controller by using a model following block and a trajectory planner. The last part deals with closed loop poles positionning in the case of a valved controlled hydraulic cylinder.



(a) Model following for the DC-motor



(b) Control structure for the DC-motor

Figure 1: Model following and control structure for a DC-motor

Servo Control

Model following control

Level 1

Here, we will design a model following controller by inverting the process model in the time domain.
The control structure and the model following block are depicted in Figure 1.
Since the DC-motor model we used is a second order system, the reference position must be two times differentiable.
The trajectory planner is design using the fastest

possible positionning:

$$a_{max} = \frac{\pm M_{max}}{J} \tag{1}$$

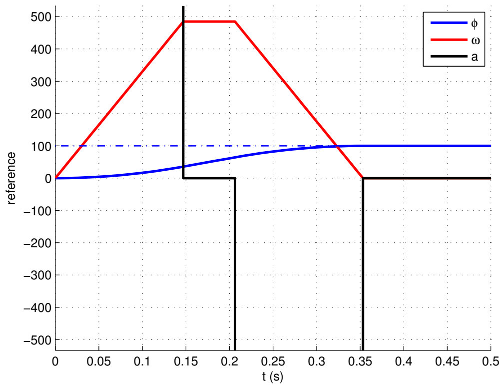
$$v_{max} = \pm \frac{U_{max} - \frac{R}{k_\varphi} F_c}{\frac{Rd}{k_\varphi} + k_\varphi} \tag{2}$$

With:
M_{max} : maximum torque of the motor
v_{max} : maximum reachable velocity of the DC-motor, computed with the DC motor model equations.
The reference signal is computed using the following equations:
Let $t_1 = \frac{v_{max}}{a_{max}}, t_2 = \frac{Rs}{v_{max}}$ and $t'_1 = \sqrt{\frac{Rs}{a_{max}}}$

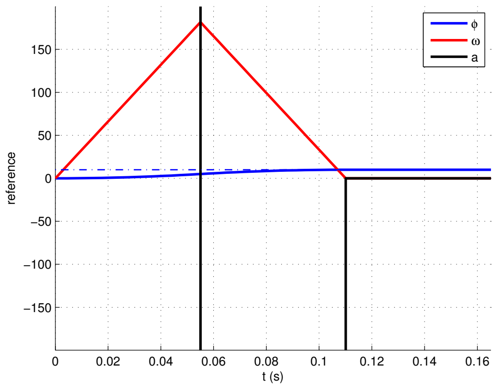
$$a(t) = \begin{cases} a_{max} & , \quad t < t_1 \\ 0 & , \quad t_1 < t < t_2 \\ -a_{max} & , \quad t_2 < t < t_1 + t_2 \\ 0 & , \quad t > t_1 + t_2 \end{cases} \text{, if } t_1 < t_2 \tag{3}$$

$$a(t) = \begin{cases} a_{max} & , \quad t < t'_1 \\ -a_{max} & , \quad t'_1 < t < 2t'_1 \\ 0 & , \quad t > 2t'_1 \end{cases} \text{, if } t_1 > t_2 \tag{4}$$

The reference signal is then created as depicted in Figure 2.
Once the reference signal is designed, we simulate the system (simulation and real-case). Figure 3 shows the results.
The Servo control model leads to an excellent control law of the motor for the two set points. Since the trajectory computed by the trajectory planner is completly reachable by the real-motor (no saturation), the planned trajectory and the ral one are almost completly merged.
Remark: In order to be sure of our control design, we reduced the value of a_{max} and v_{max} to 75% of the value computed theoretically. Indeed, using the maximum value of those values drive the motor to a dangerous area (too close to saturation).

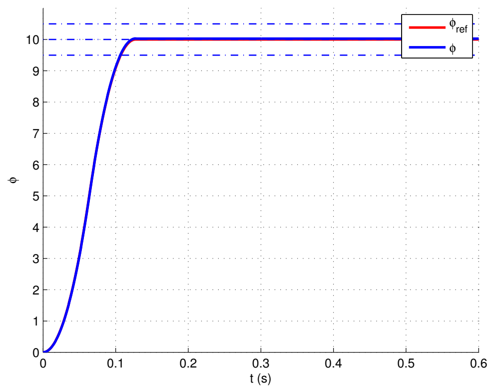


(a) Reference signal for $Rs = 100$ [rad]

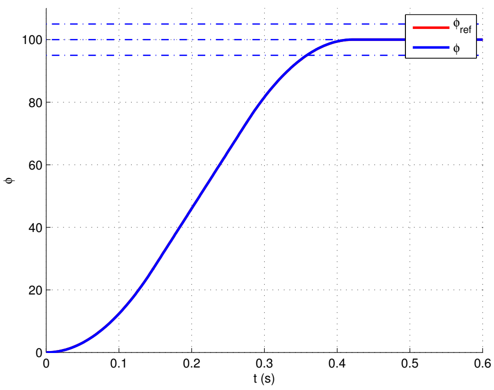


(b) Reference signal for $Rs = 10$ [rad]

Figure 2: Reference signal



(a) $Rs = 10$[rad]



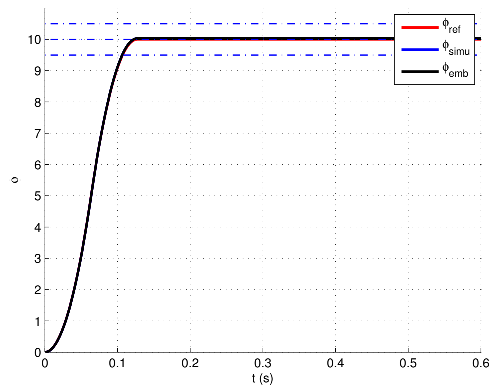
(b) $Rs = 100$[rad]

Figure 3: Simulated step responses with the position controller

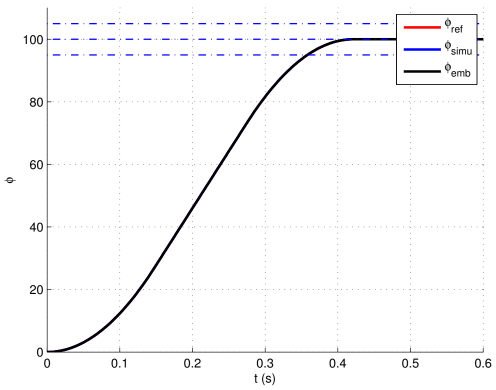
Writing Controller Code

Level 1

The embedded Matlab implementation leads to the exact same result than the implementation using ordinary Simulink blocks. Indeed, since we use a discretized controller, transposing the simulink code to the embedded Matlab implementation produces a code with the same behavior as the one produced by Simulink.
Figure 4 shows the superposition of the embedded implementation and the one using Simulink.
See Appendix A for the Matlab code.



(a) $Rs = 10$[rad]

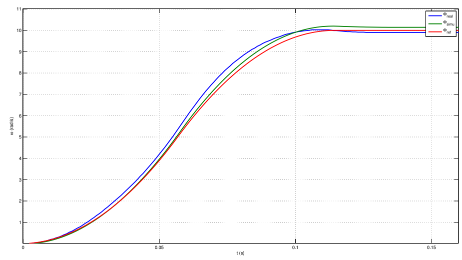


(b) $Rs = 100$[rad]

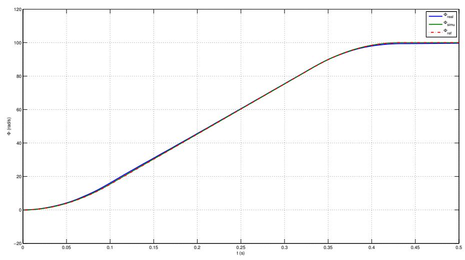
Figure 4: Result with a embedded controller

Level 2

Once we had the model following controller with both Simulink and embedded Matlab implementmation, we ran it on the real motor. Figure 5 shows the two step responses, using the two differents methods. As previously, the two curves are almost alentirely merged.
Moreover, since our controller is designed in such a way that the motor is accelerating at $0.75a_{max}$ and moving at most at a velocity of $0.75v_{max}$, the trajectory planner provide safe trajectories. The step response are therefore almost the same as the one simulated.



(a) $R_s = 10$[rad]



(b) $R_s = 100$[rad]

Figure 5: Simulation and real results
(blue) – Real step response
(green) – Simulated step response
(red) – Reference

Robustness to Parameter Uncertainty and Sensor Noise

This part aims to control a valve for a hydraulic cylinder. Figure 6 shows the system.

Figure 6: Valve controlled hydraulic cylinder

Level 2

We want to design a velocity controller for the valve controlled hydraulic cylinder – using a continuous controller.
The system will be simulate using the following reference:

- $r(t) = 0.5$ m/s
- Zero external force

Real model

The system is described by the following equations:

$$
\begin{aligned}
m\dot{v} &= p_1 A_1 - p_2 A_2 - dv - f_e \\
Q_{1v} &= R_v \sqrt{p_s - p_1} x_v \\
Q_{2v} &= R_v \sqrt{p_2 - p_r} x_v \\
Q_1 &= Q_{1v} - Q_c \\
Q_2 &= -Q_{2v} + Q_c \\
Q_c &= Av \\
C_f \dot{p_1} &= Q_1 \\
C_f \dot{p_2} &= Q_2
\end{aligned}
$$

With:
p_i: internal pression in area i
m: mass of piston
A_i: effective piston area
f_e: external force
d: friction coefficient
Q_c: volume flow due to piston velocity
Q_{iv}: flow from/out area i
C_f: fluid capacitance
p_s, p_t: supplied pression, tank pression
R_v: flow constant

Linear model

We linearize the model around an operating point:

$$
\begin{aligned}
x_v &= x_{vQ} + \Delta x_v \\
p_1 &= p_{1Q} + \Delta p_1 \\
p_2 &= p_{2Q} + \Delta p_2
\end{aligned}
$$

Let

$$x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = \begin{bmatrix} \Delta v & \Delta p_1 & \Delta p_2 \end{bmatrix}^T$$

and

$$u = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^T = \begin{bmatrix} \Delta x_v & f_e \end{bmatrix}^T$$

Then:

$$
\begin{aligned}
\boldsymbol{A} &= \left(\frac{\partial f_i}{\partial x_j}\right)_{i,j} \\
\boldsymbol{B} &= \left(\frac{\partial f_i}{\partial u_j}\right)_{i,j}
\end{aligned}
$$

Thus:

$$
\begin{aligned}
\dot{x} &= \boldsymbol{A}x + \boldsymbol{B}u \\
y &= \boldsymbol{C}x + \boldsymbol{D}u
\end{aligned}
$$

With:

$$
\boldsymbol{A} = \begin{pmatrix} -\frac{d}{m} & \frac{A}{m} & -\frac{A}{m} \\ -\frac{A}{C_f} & -\frac{1}{C_f}\frac{R_v x_{vQ}}{2\sqrt{p_s - p_{1Q}}} & 0 \\ \frac{A}{C_f} & 0 & \frac{1}{C_f}\frac{-R_v x_{vQ}}{2\sqrt{p_{2Q} - p_t}} \end{pmatrix}
$$

$$
\boldsymbol{B} = \begin{pmatrix} 0 & \frac{1}{m} \\ \frac{R_v\sqrt{p_s - p_{1Q}}}{C_f} & 0 \\ \frac{-R_v\sqrt{p_s - p_{1Q}}}{C_f} & 0 \end{pmatrix}
$$

$$
\boldsymbol{C} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}
$$

$$
\boldsymbol{D} = 0
$$

Control design

The close-loop transfert function is equal to:

$$H(s) = \frac{B(s)}{A(s)} = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} \quad (5)$$

The block diagram with the controller is depicted on Figure 7.
The control law is given by:

$$u(s) = \frac{T(s)}{R(s)}r - \frac{S(s)}{R(s)}y$$

The closed loop response is:

$$y = \frac{B}{AR+BS}r + \frac{AR}{AR+BS}v - \frac{BS}{AR+BS}n$$

Pole placement gives:

$$AR + BS = A_m A_0 \tag{6}$$

We start with:

$$\begin{aligned} A_m(s) &= s^2 + 2\xi_m\omega_m s + \omega_m^2 \\ A_0(s) &= s + \omega'_m \end{aligned}$$

Therefore since:

$$\begin{aligned} A(s) &= s^2 + 2\xi_0\omega_0 s + \omega_0^2 \\ B(s) &= K_0 \end{aligned}$$

Equation (6) leads to:

$$\begin{aligned} R(s) &= s + r_c \\ S(s) &= K_C(s + s_C) \end{aligned}$$

We then select T such as $T = A_0 t_0$.
Then:

$$\begin{cases} r_c &= w'_m + 2(\omega_m\xi_m - \omega_0\xi_0) \\ K_C &= \frac{1}{K_0}\left(\omega_m^2 + 2\omega_m'^2\xi_m\omega_m - 2r_c\xi_0\omega_0 - \omega_0^2\right) \\ s_C &= \frac{1}{K_0K_C}(\omega'_m\omega_m - r_c\omega_0^2) \\ t_0 &= \frac{\omega_m^2}{K_0} \end{cases}$$

Using the method described in the subject, we obtain the following value (satisfying step response) for ω_m and ω'_m, leading to the following A_m and A_0:

$$\begin{aligned} A_m &= s^2 + 3600s + 4e6 \\ A_0 &= s + 2e3 \end{aligned}$$

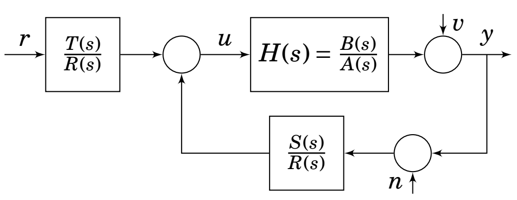
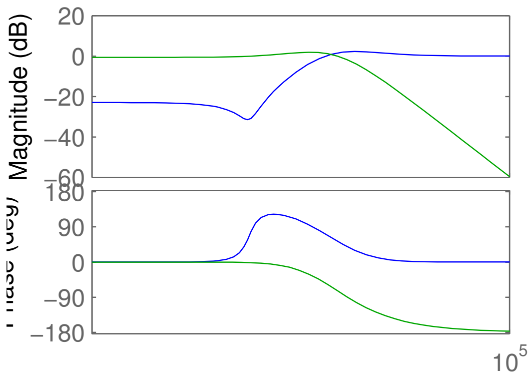


Figure 7: Block diagram of the system with the controller

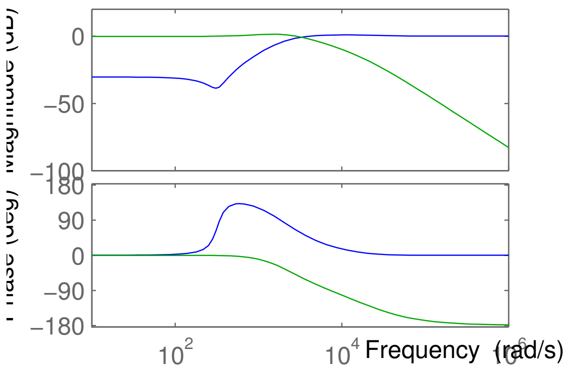
The sensitivity function and the complementary sensitivity function are defined by:

$$\begin{aligned} S_e(s) &= \frac{A(s)R(s)}{A_m(s)A_0(s)} \\ T_e(s) &= \frac{B(s)S(s)}{A_m(s)A_0(s)} \end{aligned}$$

Figure 8 shows the sensitivity function and the complementary sensitivity function bode diagrams.



(a) $A_0 = s + 2e3$



(b) $A_0 = s + 2e4$

Figure 8: Bode diagram of the Sensitivity and complementary sensitivity functions
(green) – Sensitivity function
(blue) – Complementary sensitivity function

Figure 9 shows the step response of the system without any perturbation.
The step responses shows that:

- Linearized model and real one are close to each other (the linearization was legitimate a posteriori).

- The system react as a second order one (as defined is the design).
- The step responses present no overshoot and are fast.
- The non-linearized is almost as fast as the linearized one.

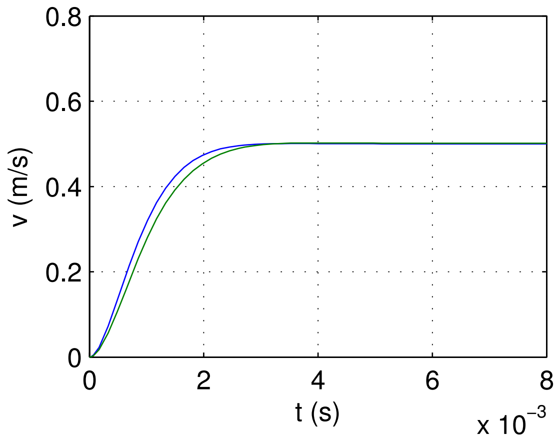
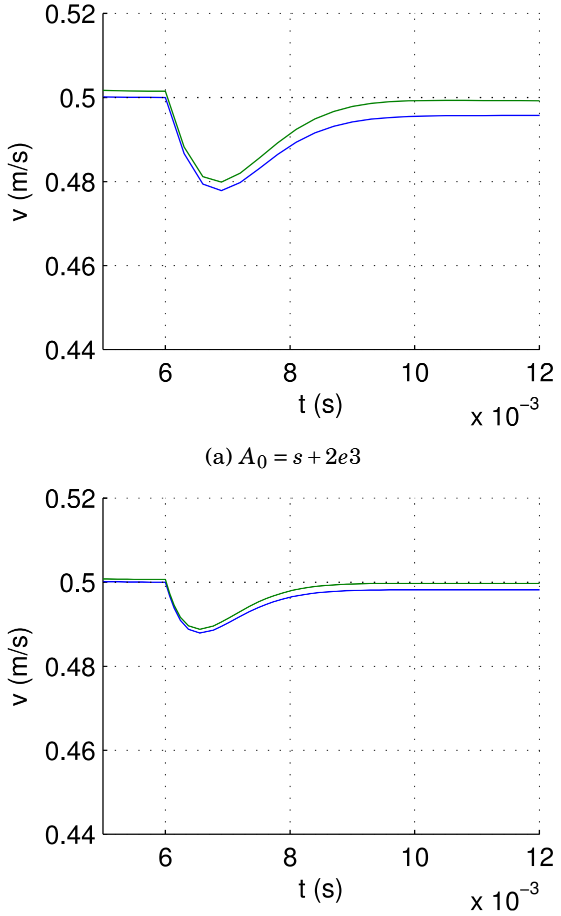


Figure 9: Step response of the sytem without any perturbation
(green) – real system
(blue) – linearized system

Step responses with perturbations

Keeping the same $A_m = s^2 + 3600s + 4e6$, figure 10 shows the step responses using a perturbation $f_e = 5000$N.



(a) $A_0 = s + 2e3$

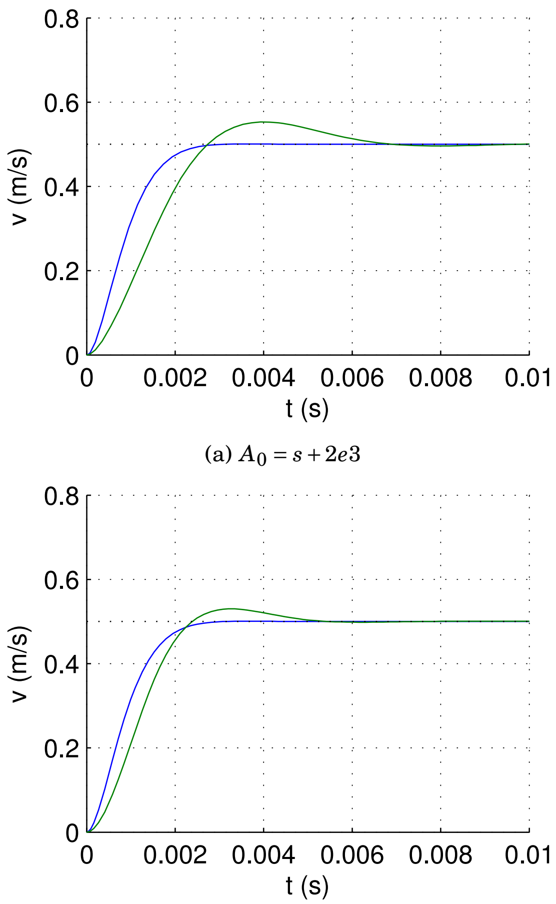
(b) $A_0 = s + 2e4$

Figure 10: Step responses with a perturbation $f_e = 5000$N
(green) – real system
(blue) – linearized system

In the first case ($A_0 = s + 2e3$), the steady state error is bigger with the perturbation. This can be explain by the fact that, looking at the bode diagram, the gain is not equal to $-\infty$ with a perturbation. But the bigger ω'_m is, the smaller the gain is. This is why looking at the second step response, we have a smaller steady state error.
Moreover, the system is balancing the perturbation faster with a bigger ω'_m .

Mass variation

We change the mass from 100kg to 200kg. Figure 11 shows the step responses using the two same A_0 as in the previous subsubsection.



(a) $A_0 = s + 2e3$

(b) $A_0 = s + 2e4$

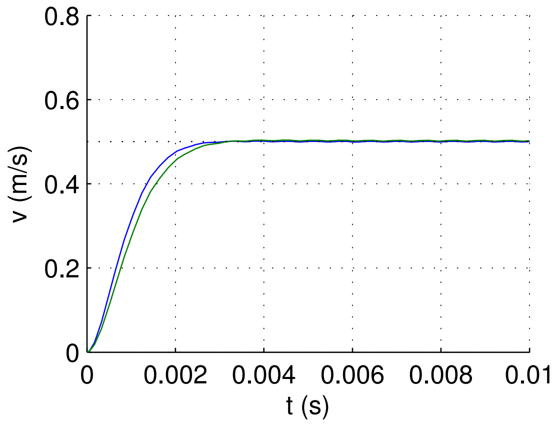
Figure 11: Step responses with a 200kg mass
(green) – real system
(blue) – linearized system

Changing the mass, the linearized system response do not change.
However, the quality of the real system response is damage by this change. Increasing the value of ω'_m leads to a better step response.
This misestimation of the parameter m can be seens as the addition of noise in the model. As

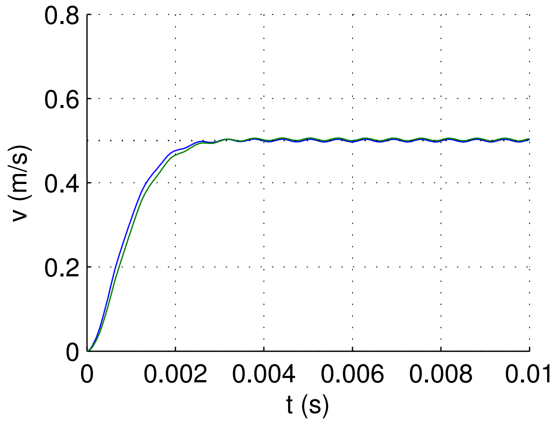
previously, the impact of this noise is reduced by a raise of ω'_m.

Noise effect

In order to analyse the impact of the noise in our system, we added some in the feedback loop. Figure 12 shows the step responses with two different A_0.



(a) $A_0 = s + 2e3$



(b) $A_0 = s + 2e4$

Figure 12: Step responses with noise
(green) – real system
(blue) – linearized system

The system response is worse in the second case ($\omega'_m = 2e4$). Indeed, the bode diagram shows that the cutoff frequency in this case is bigger than in the other case. The bigger ω'_m, the more the noise impacts the system response.

A Writing Controller Code

```
function [ac,vc,rc,u] = fcn(Rs,t)
%#codegen
%% Variable declarations

Ts = 1e-3;
amax = 3000;
vmax = 300;
d = 5.8e-6;
J = 1/2200000;
Fc = 1e-4;
R = 24;
kM = 8.64e-3;
kE = 30/pi*0.905e-3;

%% Trajectory computation
t1 = vmax/amax;
t2 = Rs/vmax;
t1p = sqrt(Rs/amax);

if t1 > t2
    t1 = t1p;
    t2 = t1p;
end

t3 = t1+t2;

ac = 0;
vc = 0;
rc = 0;
if t < t1
    ac = amax;
    vc = amax*t;
    rc = amax*t^2/2;
end
if t >= t1 && t < t2
    ac = 0;
    vc = amax*t1;
    rc = amax*t1^2/2 + amax*t1*(t-t1);
end
if t >= t2 && t < t3
    ac = -amax;
    vc = amax*t1 - amax*(t-t2);
    rc = amax*t1^2/2 + amax*t1*(t2-t1) +amax*t1*(t-t2) - amax*(t-t2)^2/2;
end
if t >= t3
    ac = 0;
    vc = 0;
```

```
    rc = Rs;
end

%% Feedforward loop
u = R*(J*ac + d*vc + sign(vc) * Fc)/kM + kE * vc;
```