Advanced Control Systems ME-524

TP2: RST Controller Design

Group L
David Wu
Sten Elling Jacobsen

June 1, 2018

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

1 Desired closed-loop poles

The order of A n_A is $length(A) - 1 = 6$, since the first element is per definition a constant. The B vector contains the delay d. n_d can be found by counting the number of leading zeros minus one, since the first coefficient of B is 0 per definition, so here $n_d = 1$. Thus, the order of B is $length(B) - d - 1 = 4$.

We then calculate the desired damping and natural frequency using

$$M_p - 1 = e^{-\frac{\zeta\pi}{1-\zeta^2}} \tag{1}$$

$$\zeta = \frac{1}{\sqrt{1 + \frac{\pi^2}{[ln(M_p-1)]^2}}} = 0.6901 \tag{2}$$

for overshoot $M_p - 1 = 0.05$.

We can then approximate the natural frequency ω_n using

$$\omega_n = \frac{2.16\zeta + 0.6}{T_r} = 4.1812 \tag{3}$$

using rise time $T_r = 0.5$.

The coefficients of the closed-loop polynomial are then given by:

$$p_1 = -2e^{-\zeta\omega_n T_s} cos(\omega_n T_s \sqrt{1-\zeta^2}) = -1.7689 \tag{4}$$

$$p_2 = e^{-2\zeta\omega_n T_s} = 0.7939 \tag{5}$$

resulting in the closed-loop polynomial

$$P(q^{-1}) = 1 - 1.7689q^{-1} + 0.7939q^{-2} \tag{6}$$

2 Pole placement

Using our poleplace function, we obtained

$$R = 0.2513 + 13.5853q^{-1} + 23.5790q^{-2} + -125.8965q^{-3}213.6546q^{-4} - 195.7242q^{-5}71.5719q^{-6} \tag{7}$$

$$S = 1.000 + 1.9660q^{-1} + 1.1756q^{-2} - 1.0556q^{-3} - 2.4399q^{-4} - 0.6461q^{-5} \tag{8}$$

Our desired poles were $0.8844 \pm 0.1079i$, and our achieved poles were $0.8844 \pm 0.1079i$, in addition to $0.0207 + 0.0000i$, $0.0159 \pm 0.0133i$, $0.0036 \pm 0.0204i$, $-0.0194 \pm 0.0071i$, $-0.0104 \pm 0.0179i$. These extra poles are likely due to numerical errors.

Since the dynamics for tracking and regulation were the same, T is given by the sum of the coefficients of R. We calculate $T = 1.0214$.

```
function [R,S]=poleplace(B,A,Hr,Hs,P)
    %Determine delay d and separate it from B
    d = 0;
    if length(B) > 1
        for i=2:length(B)
            if B(i) == 0
                d = d + 1;
```

```
            end
        end
    end
    B = B(1+d:end);

    n_a = length(A)-1;
    n_b = length(B)-1;
    n_hr = length(Hr)-1;
    n_hs = length(Hs)-1;
    n_a1 = n_a+n_hs;
    n_b1 = n_b+n_hr;
    n_s1 = n_b+n_hr+d-1;

    A1 = conv(A,Hs);
    B1 = conv(B,Hr);

    % M1, M2 are left and right halves of sylvester matrix
    M1 = tril(toeplitz([A1,zeros(1,n_b1+d-1)]));
    M1 = M1(:, 1:n_b1+d);
    M2 = tril(toeplitz([zeros(1,d+1), B1(2:end), zeros(1,n_a1-1)]));
    M2 = M2(:, 1:n_a1);

    M = [M1, M2];
    p = ([P,zeros(1, n_a1+n_b1+d-length(P))])';
    x = M\p;

    S1 = x(1:n_s1+1)';
    R1 = x(n_s1+2:end)';

    S = conv(S1,Hs);
    R = conv(R1,Hr);
end
```

3 Analysis of the closed-loop system

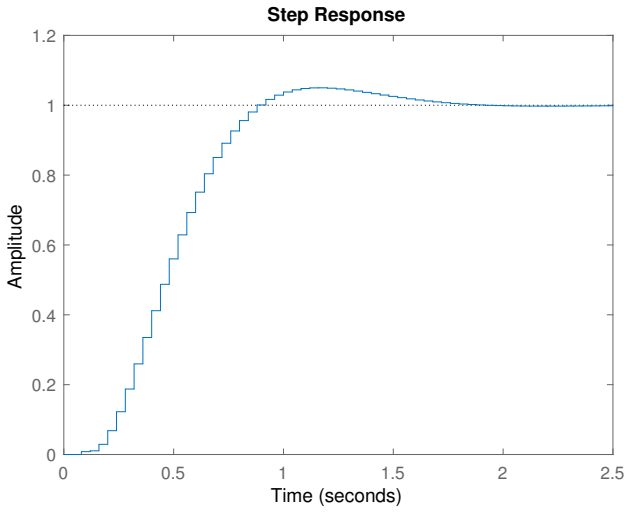

Figure 1: Tracking step response of the system with RST-controller.

Using stepinfo, we see that the rise-time is 0.52 while the overshoot is 4.99 %. It fails to achieve the desired rise-time 0.50 s, possibly because we use an approximation when calculating the natural frequency ω_n of the system. Also, our achieved poles are not exactly the same as the desired ones.

The Nyquist plot encircles -1 three times counterclockwise, which according to the Nyquist criterion indicates that the open-loop system needs 3 poles in the RHP for the closed-loop system to be stable. For our discrete system, this corresponds to poles outside the unit circle. Upon inspecting the poles, we see that there are indeed three poles outside the unit circle, located in -1.3179 and $-0.6657 \pm 1.0511j$, meaning that the closed-loop system is stable.

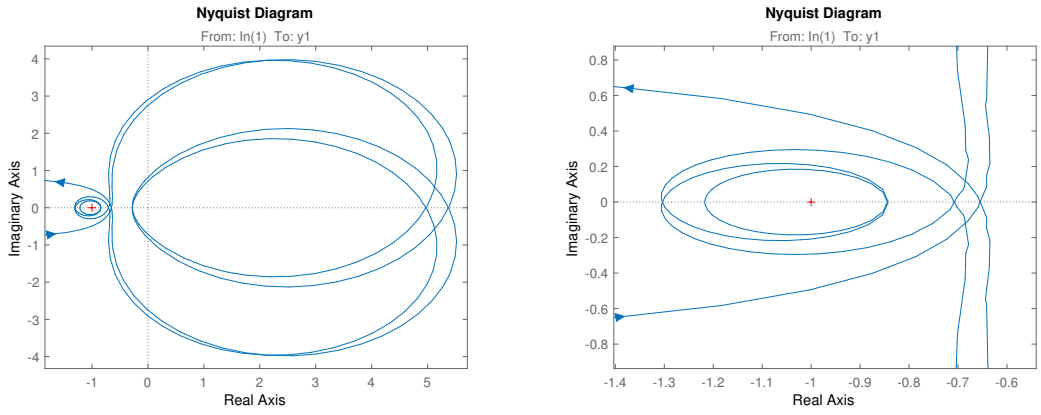

Figure 2: Left: Nyquist diagram of the open-loop system. Right: Zoomed-in version of left image.

The requirements $MM \geq 0.4$ and $||U||_\infty \leq 35$ dB are not satisfied, our system has $MM = 0.1566$ and $||U||_\infty = 74.6$ dB (see Figure 3). It should therefore not be implemented on the real system.

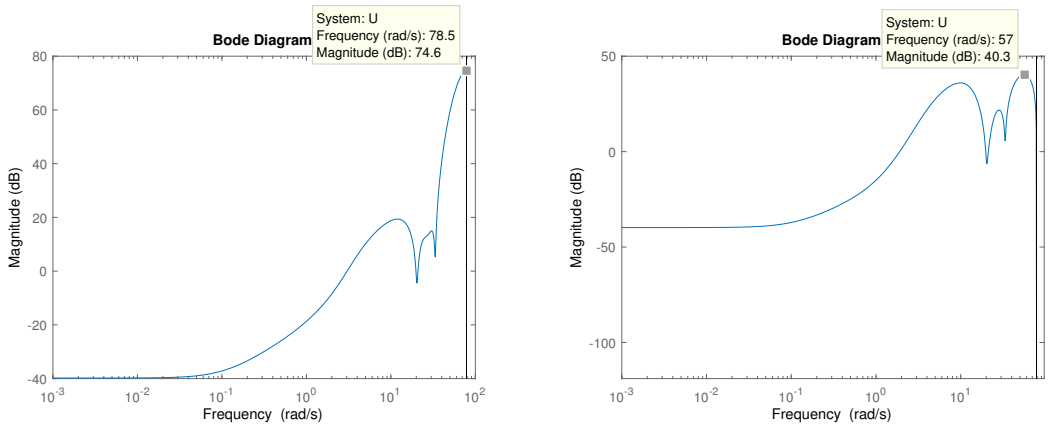

Figure 3: Magnitude plots of the input sensitivity function with auxiliary poles in zero (left) and auxiliary poles in 0.4 (right).

4 Improving the results

In general, there is always a trade-off between performance and robustness. The implementation constraints depend on the poles of the system, which depend on the specifications (overshoot and rise-time) of the system. Increasing rise-time and allowing for more overshoot (reducing performance) will make it easier to meet the implementation constraints (increase robustness).

To improve the RST-controller, we add $H_R = 1 + q^{-1}$ to $R(q^{-1})$ to dampen $||U||_\infty$. We also replace the auxiliary poles at zero with poles at α, $0.1 \leq \alpha \leq 0.5$. This tends to decrease $||U||_\infty$, and larger α seems to generally cause more attenuation. The controller was:

$$\frac{0.9426 - 22.65z^{-1} + 69.91z^{-2} - 76.57z^{-3} + 0.6977z^{-4} + 77.84z^{-5} - 71.54z^{-6} + 21.39z^{-7}}{1 - 2.034z^{-1} + 0.5062z^{-2} + 0.9872z^{-3} + 0.3475z^{-4} - 0.6138z^{-5} - 0.193z^{-6}} \tag{9}$$

Using $\alpha = 0.4$ reduces $||U||_\infty$ from 74.6 dB to 40.3 dB, which still does not meet the constraints. We try again using Q-parametrization.

For the design and implementation of the final controller, we accidentally submitted the wrong one for testing; the modulus margin on this one was only around 0.2. We will therefore present a different controller with sufficient modulus margin here, and discuss the test results separately at the end.

The final controller using Q-parametrization was found using $\alpha = 0.16$, and the corresponding $Q = -7.36 - 13.79z^{-1} - 2.646z^{-2} + 3.056z^{-3} - 2.234z^{-4} - 5.86z^{-5} - 3.121z^{-6} + 0.3228z^{-7}$. The controller is given by

$$\frac{\begin{array}{r}6.899 - 10.57z^{-1} + 0.188z^{-2} + 12.98z^{-3} - 9.088z^{-4} - 9.33z^{-5} + 3.115z^{-6} + 14.3z^{-7} - 2.824z^{-8} \\ -7.797z^{-9} - 3.549z^{-10} + 8.272z^{-11} + 1.508z^{-12} - 7.479z^{-13} + 3.841z^{-14} - 0.2825z^{-15}\end{array}}{\begin{array}{r}1 + 0.366z^{-1} - 0.8713z^{-2} - 1.093z^{-3} + 0.006521z^{-4} + 0.7574z^{-5} + 0.335z^{-6} - 0.3042z^{-} - 0.2398z^{-8} \\ + 0.1184z^{-9} + 0.1366z^{-10} - 0.07951z^{-11} - 0.1163z^{-12} - 0.01807z^{-13} + 0.00255z^{-14}\end{array}}$$

We use the model obtained through pole placement, H_r and P_f, calculated earlier, as a nominal model. We then write MATLAB functions

$$R(Q(q^{-1})) = R_0(q^{-1}) + A(q^{-1})H_R(q^{-1})H_S(q^{-1})Q(q^{-1}), \tag{10}$$

$$S(Q(q^{-1})) = S_0(q^{-1}) - q^{-d}B(q^{-1})H_R(q^{-1})H_S(q^{-1})Q(q^{-1}), \tag{11}$$

R_0 and S_0 being values from the nominal model. This allows us to shape the sensitivity functions S and U through convex optimization, without moving the closed-loop poles.

Finally, we formulate the following optimization problem:

$$\min Q \quad MM(Q) \tag{12}$$
$$s.t. \quad ||M_m S(Q)||_\infty < 1 \tag{13}$$
$$||U(Q)||_\infty < U_{max} \tag{14}$$

and solve it using MATLAB's fmincon function.

Table 1: Rise time, settling time and overshoot, sensitivity norms for high frequencies.

	Rise time [s]	Settling time [s]	Overshoot	$\|\|S\|\|_\infty$ (hi-freq)[dB]	$\|\|U\|\|_\infty$ (hi-freq) [dB]
Original	0.52	1.56	0.499	6.39	78.5
Improved	0.52	1.64	0.483	2.34	35.0

The results are summarized in Table 1. We see that the tracking step responses are fairly similar; with the first controller it is a bit faster and has a bit more overshoot. However, the control signal of the improved one is significantly better, being much smoother than the former. The norms of the output sensitivity functions translate to a modulus margin of 0.16 and 0.43 for the original and improved systems, respectively. For the input sensitivity functions, the infinity norm of the original system is 79.5 dB, which is way above the criterion of 35 dB, which the improved system satisfies.

In short, using the improved controller gives us much better robustness at the cost of having less than one-tenth of a second slower settling time.

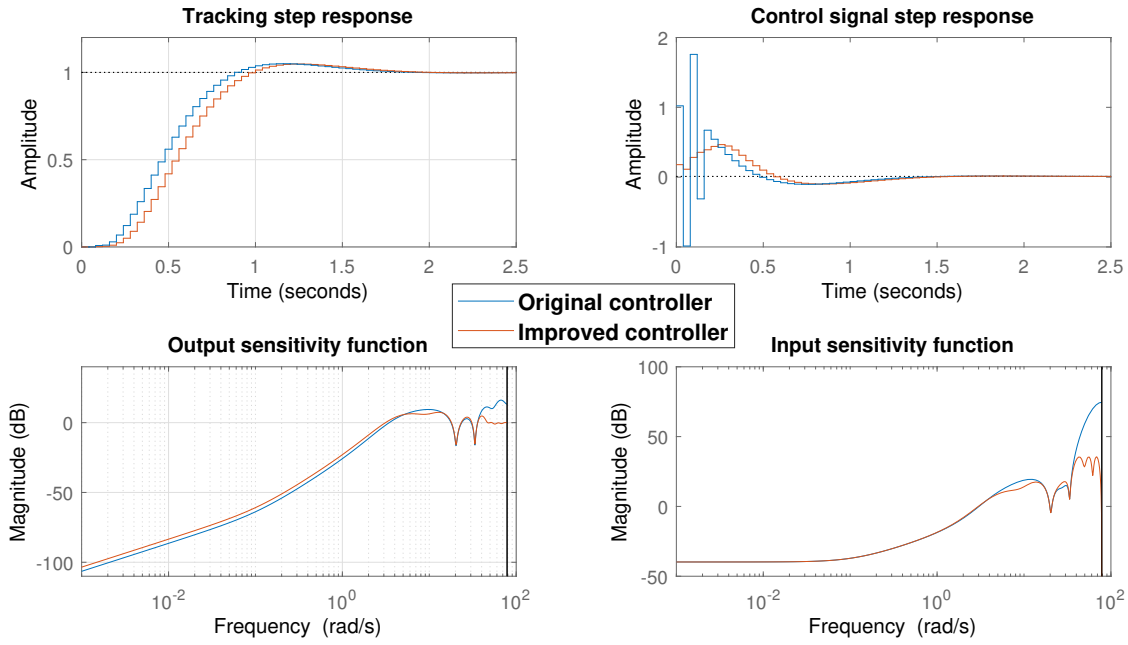

Figure 4: Comparison of performance for original and improved RST-controller.

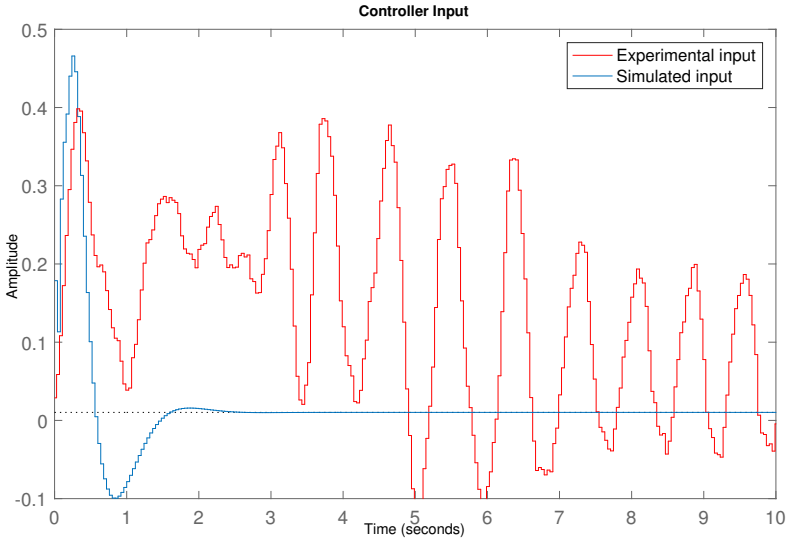

Figure 5: Experimental and simulated input of the plant.

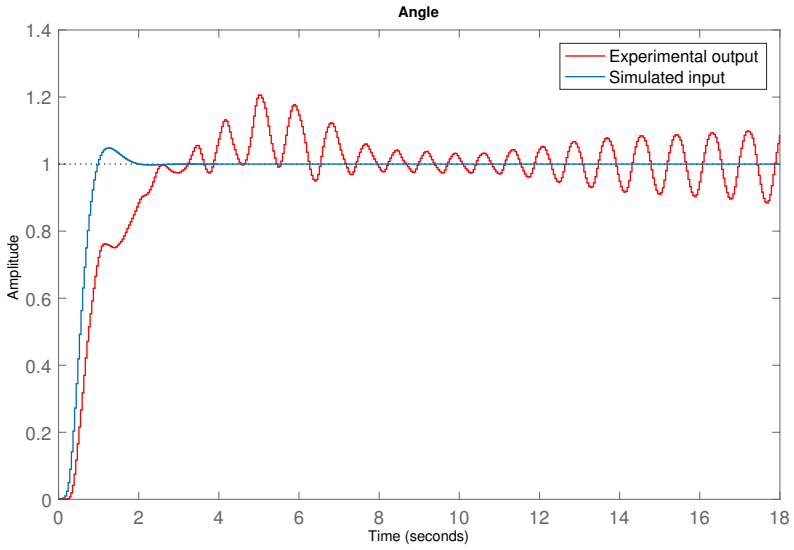

Figure 6: Experimental and simulated output of the plant.

The test results can be seen in Figures 5 and 6. Both the control signal and output oscillate a lot, and even though the output seems to center around 1, the output seems to be unstable from 11 seconds and onward. This is as mentioned caused by the too low modulus margin. Already from the beginning, the system starts diverging from the simulation. This is mainly because the static friction (and possibly higher-order dynamics) prevents it from reaching its target using the planned control signal. Had the system been more robust (≥ 0.4 modulus margin), it would likely have been able to handle the situation better.

```
%% 2.4 Q-parametrization
load('TorMod.mat');
Ts = 0.04;
A = G3.f;
B = G3.b;
p1 = -1.7689;
p2 = 0.7939;
P = [1 p1 p2];

Hs = [1 -1];
Hr = [1 1];

% Set auxiliary poles
a = 0.16;
poles_aux = [a,a,a,a,a,a,a,a,a,a];
coefs = poly(poles_aux);
P_new = conv(P, coefs);

[R, S] = poleplace(B, A, Hr, Hs, P_new) ;
P_end = conv(A,S) + conv(B,R);

B = B(2:end); %Separate delay and B
d = [0 1];

M_m = 0.4;
U_max = 56.2; %35 dB  = 56.2

% Create R and S functions
Q_length = 8;
R0 = [R, zeros(1, Q_length)];
S0 = [S, zeros(1, Q_length)];
R_new = @(Q) R0 + conv(A, conv(Hr, conv(Hs, Q)));
S_new = @(Q) S0 - conv(d, conv(B, conv(Hs, conv(Hr, Q))));

%Set inequality (c) and equality (ceq) constraints
c = @(Q) [norm(M_m*S_new(Q), Inf) - 1;
      norm(tf(conv(A, R_new(Q)), P_end, ...
              Ts, 'variable', 'z^-1'), Inf) - U_max];
ceq = @(Q) [];
Nonlincon = @(Q)deal(c(Q), ceq(Q));

% Define modulus margin and optimize
Mod_marg = @(Q) norm(tf(S_new(Q), 1 ,Ts, 'variable', 'z^-1'), Inf)^(-1);
Q_opt = fmincon(Mod_marg, zeros(1, Q_length) ,[] ,[] ,[] ,[], ...
                    [-Inf,-Inf],[Inf,Inf], Nonlincon);

% Evaluate R,S,T using optimal Q
R_final = R_new(Q_opt);
S_final = S_new(Q_opt);
T = sum(R_final);

%% Check constraints
CL = tf(conv(T,G3.b), P_end, Ts,'variable','z^-1'); %Closed loop system
```

```
54 U = tf(conv(A,R_final), P_end, Ts, 'variable', 'z^-1'); %Input sensitivity
       function
55 input = tf(conv(T,A), conv(A,S_final) + conv(conv(d, B), R_final) ,...
56             Ts,'variable','z^-1');
57
58 figure(1);
59 step(input);
60 figure(2);
61 step(CL);
62 figure(3);
63 bodemag(U);
64 MM = inv(norm(S_final,inf)) %Modulus margin
```