TTK4215 - Assignment 7

Sigurd Totland — MTTK

October 16, 2019

Problem 1 I&S 4.13 d, e

d)

d.1) Adaptive law derivation

We write the system with the common bilinear parametrization

$$\begin{aligned} y &= \frac{1}{k}(u - ms^2y - \beta sy) \\ &= \rho^* \left(- \begin{bmatrix} \beta & m \end{bmatrix} \begin{bmatrix} sy \\ s^2y \end{bmatrix} + u \right). \end{aligned} \tag{1}$$

Because we have derivatives, We need to filter this with at least a second-order filter to make it realizable. We choose $W(s)L(s) = \frac{1}{\Lambda(s)}$ as our filter, where $\Lambda(s)$ is a second order Hurwitz polynomial in s. With that, we can write

$$\begin{aligned} z = \frac{y}{\Lambda} &= W(s)L(s)\rho^*(\theta^{*\top}\phi + z_1) \\ &= \frac{1}{\Lambda(s)}\rho^* \left(\begin{bmatrix} \beta & m \end{bmatrix} \begin{bmatrix} s \\ s^2 \end{bmatrix} (-y) + u \right), \end{aligned} \tag{2}$$

where $\theta^{*\top} = \begin{bmatrix} \beta & m \end{bmatrix}$, $\rho^* = \frac{1}{k}$, $\phi = \begin{bmatrix} s & s^2 \end{bmatrix}^\top (-y)$ and $z_1 = u$. This let's us use an adaptive law from table 4.4 in I&S, namely

$$\dot{\theta} = \Gamma\epsilon\phi \mathrm{sgn}(\rho^*) = \Gamma\epsilon\phi \tag{3}$$

since we know a priori that $\mathrm{sgn}(\rho^*) = 1$, as well as

$$\dot{\rho} = \gamma\epsilon\xi. \tag{4}$$

We here define the estimation error to be

$$\epsilon = z - \hat{z}. \tag{5}$$

Note that we have let the normalization term $n_s^2 = 0$. Furthermore, we have

$$\xi = \theta^\top\phi + z_1. \tag{6}$$

Finally, $\hat{z}$ is defined as

$$\hat{z} = W(s)L(s)\rho(\theta^\top\phi + z_1) = \frac{1}{\Lambda}\rho(\theta^\top\phi + u) \tag{7}$$

d.2) Simulation with constant mass

When simulating this, we filter ϕ and ρ individually since

$$\hat{z} = \frac{1}{\Lambda}\rho(\theta^\top\phi + u) = \rho(\frac{1}{\Lambda}\theta^\top\phi + \frac{1}{\Lambda}u). \tag{8}$$

From this, we also note that

$$\hat{z} = \rho\frac{1}{\Lambda}\xi, \tag{9}$$

i.e. ξ filtered with our stable filter and scaled by ρ. This let's us calculate the filtered ξ and then use it for both (4) and (9). The main simulation loop that achieves this is shown in listing 1.

Listing 1: Main simulation loop

```
% Main simulation loop
for n = 1:N-1
    % Generate input
    t(n+1) = n*h;
    u = 10*sin(2*t(n)) + 10*cos(3.6*t(n)) + 20*cos(1*t(n));
    U(n) = u;

    % Changing mass after t = 20s
    if(varying_mass && t(n) > 20)
        m = 20*(2-exp(-0.01*(t(n) - 20)));
        A = [0, 1; -k/m, -beta/m];
        B = [0; 1/m];
    end

    % Save actual model parameters at timestep for comparison later
    theta_star(:, n+1) = [beta, m];
    rho_star(:, n+1) = k;

    % Simulate actual system
    x_dot = A*x(:, n) + B*U(n);
    x(:, n+1) = x(:, n) + h*x_dot;
    y = x(1, n);

    % Calculate z
    x_z_n = x_z + (A_f*x_z + B_f*y)*h; % Euler integration
    z = C_f_1*x_z; % Fetch output

    % Calculate phi
    x_phi_n = x_phi + (A_f*x_phi + B_f*(-y))*h; % Euler integration
    phi_filt = [C_f_2*x_phi + D_f_2*(-y); C_f_3*x_phi + D_f_3*(-y);]; % Fetch output

    % Calculate z_1 = u_filt
    x_u_filt_n = x_u_filt + (A_f*x_u_filt + B_f*u)*h; % Euler integration
    u_filt = C_f_1*x_u_filt; % Fetch output


    % Calculate z_hat
    xi_filt = theta(:, n)'*phi_filt + u_filt;
    z_hat = rho(:, n)*xi_filt;

    % Estimation error
    epsilon = z - z_hat;

    % Propagate parameter estimates using Euler intagration
    theta_dot = Gamma*epsilon*phi_filt;
    theta(:, n+1) = theta(:, n) + theta_dot*h;

    rho_dot = gamma_k * epsilon * xi_filt;
    rho(:, n+1) = rho(:, n) + rho_dot * h;

    % Set variables for next iteration
    x_phi = x_phi_n;
    x_z = x_z_n;
    x_u_filt = x_u_filt_n;
end
```

The results are shown in figure 1.

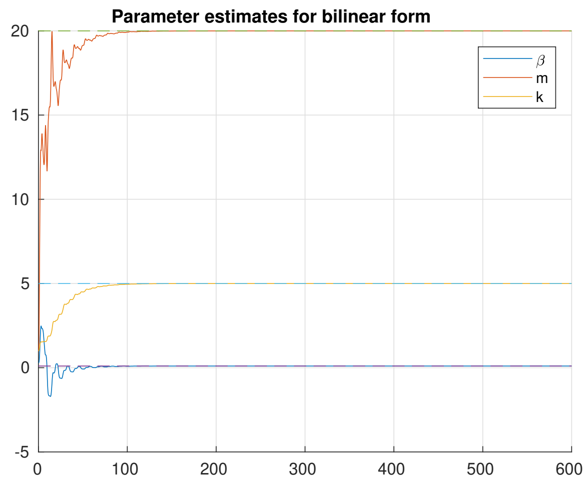

Figure 1: Parameter estimation using bilinear form

e) Varying mass

Letting the mass equal

$$m = 20(2 - e^{-0.01(t-20)}) \tag{10}$$

for $t \geq 20$, we get the results shown in figure 2.

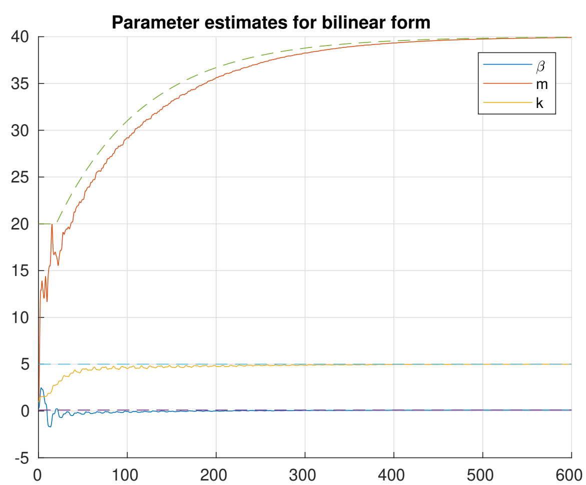

Figure 2: Bilinear form parameter estimation with varying mass