Compulsory assignment 5

Aleksander Hykkerud

Parameter estimation of a DC motor with least squares (LS) method

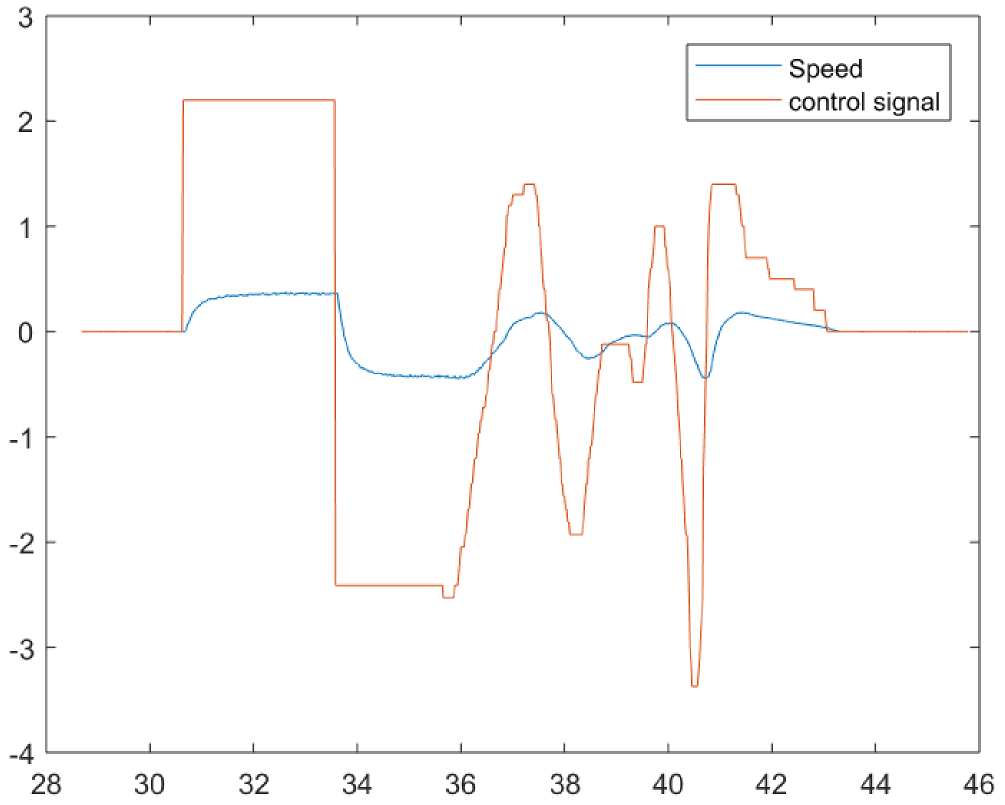

Figure 1: Actual response to manual manipulation

The system is represented by the differential equation

$$T * \dot{S} + S = K * (C + L)$$

Rearranging the equation to get the Response $\dot{S}$ on one side

$$\dot{S} = K * \frac{(C + L)}{T} - \frac{S}{T} = K \cdot \frac{C}{T} + K \cdot \frac{L}{T} - \frac{S}{T}$$

This is represented by the vector form

$$\vec{Y} = \Phi\theta$$

giving

$$\vec{S} = [C, 1, -\dot{S}] \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Where

$$a = K$$

$$b = K \cdot \frac{L}{T}$$

$$c = \frac{1}{T}$$

This can be solved by the least square method

$$\vec{\theta} = \Phi^t \Phi \; \Phi^t Y$$

The solution was implemented in matlab by importing the experimental data and converting the different inputs (time, control signal and speed) into column vectors.

Script

```
formatspec = '%f %f %f';
sizeA = [3 inf];
infile = fopen('logfile1.txt','r');

% time (t), control signal (C), speed (S)
A = fscanf(infile,formatspec,sizeA)';

tstep = A(2)-A(1); % assuming same timestep for all steps
da = (A(2:end,3)-A(1:end-1,3))/tstep;
S = A(1:end-1,3);
C = A(1:end-1,2);
t = A(1:end-1,1);
% C, 1, -S
phi = [C,ones(length(A(1:end-1,1)),1),-S];

% delta = [a,b,c] : a = K/T, b=K*L/T, c = 1/T
delta = inv(phi'*phi)*phi'*da;

T = 1/delta(3);
K = delta(1)*T;
L = delta(2)*(T/K);

plot(t,S,t,C)
legend('Speed','control signal')



%numsteps = int16(3/tstep);
numsteps = length(t);
tstart = 1;
tsim = t;
usim = C;

s_sims = zeros(numsteps,1);
s_sim = 0;
ds_sim = 0;
for step = 2:numsteps
    s_sims(step)=K*(usim(step)+L)-T*ds_sim;
    ds_sim = s_sims(step)-s_sims(step-1);
end

% plot the results
disp(['K=',num2str(K),'  L=',num2str(L),'  T=',num2str(T)])
plot(tsim,s_sims,tsim,usim,t,S)
legend('s simulated','u','Measured S')
```

Results

The values estimated for K, L and T are as follows

K=0.88586 L=-0.06248 T=0.30349

The simulated vs the real response show below reveals that the model responds poorly to binary changes and responds faster than the true reaction.

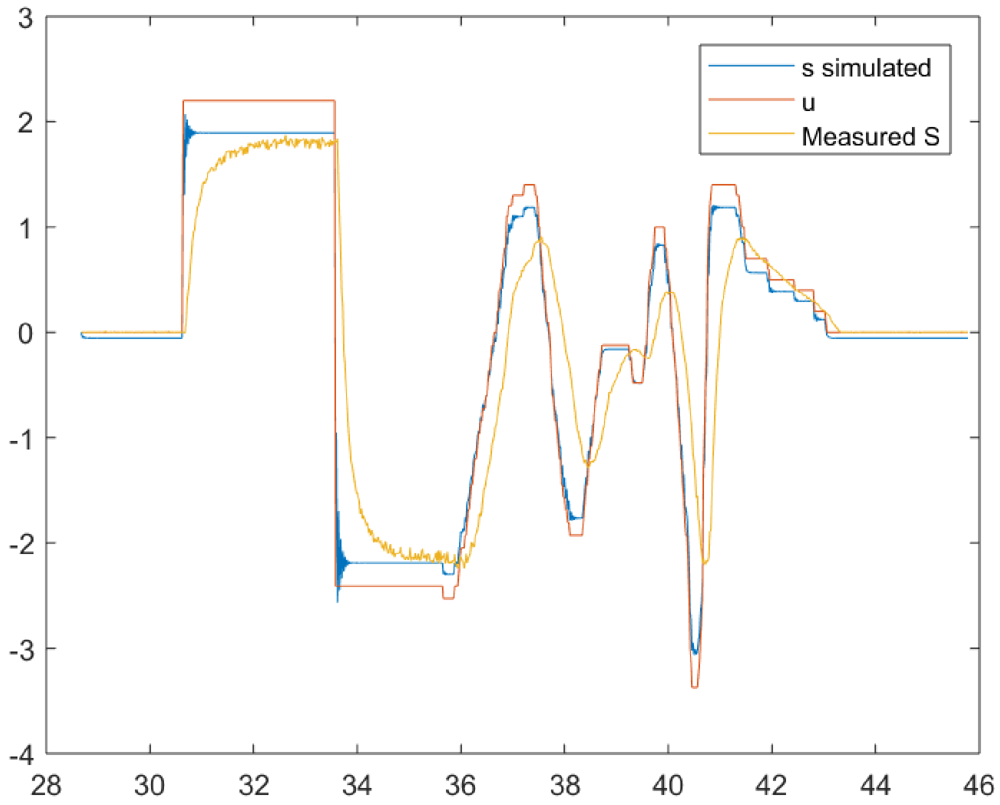

Figure 2: Real vs simulated response

Parameter estimation of an air heater using the grid optimization method

The log file was imported and plotted to get a better view of the response and the system. The goal is to find T_{env} K_h, θ_t, θ_r of the system given by

$$\theta_t * \frac{d(T_{heat})}{dt} = -T_{heat} + K_h * u(t - \theta_d)$$

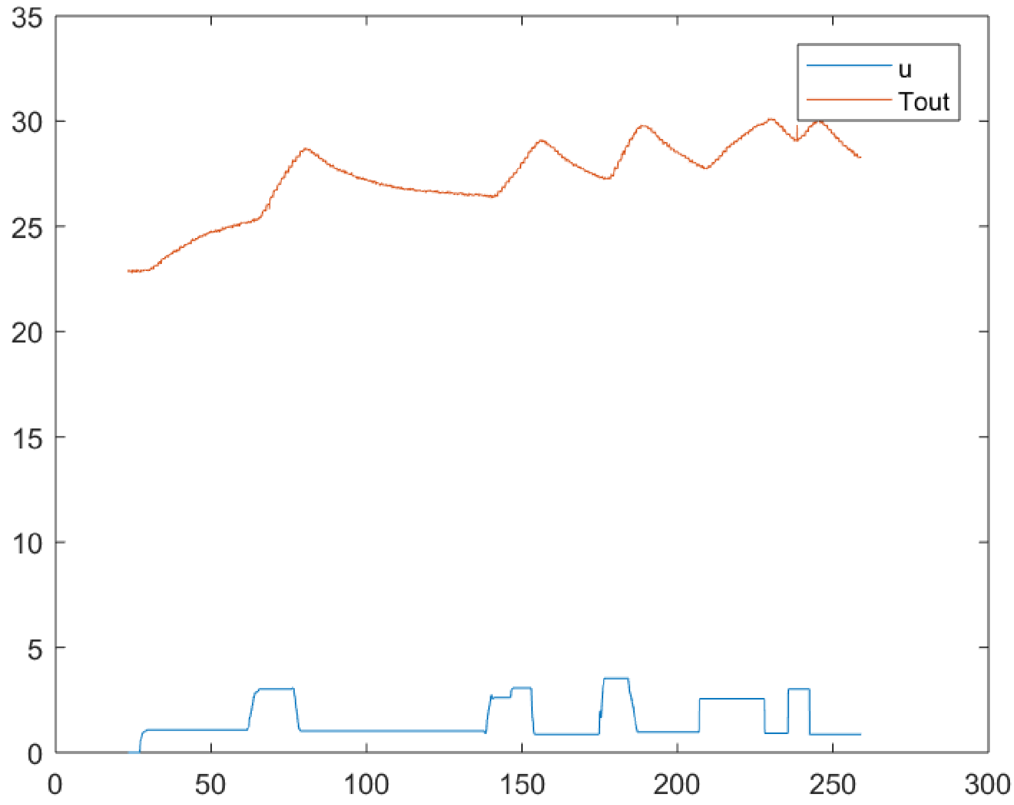

Figure 3: Heater singnal response

The grid search parameters were set up with constraints of their values.

$$T_{env} \rightarrow [15,30]$$

$$K_h \rightarrow [0,8]$$

$$\theta_t \rightarrow [1,250]$$

$$\theta_r \rightarrow [1,8]$$

For each grid, a simulation was run and the square error sum was calculated and used as a metric for the goodness of fit. The grid search yielded the following parameters as the best fit in the search area

$$T_{env} = 22.5$$

$$K_h = 3.5$$

$$\theta_t = 232$$

$$\theta_r = 2.8$$

The grid search was performed with the script below. When the search was done, a simulation with the optimal parameter values were done to get a plot.

```
% thetat * d(Theat)/dt = - Theat + Kh * u(t-thetad)
load('airheater_logfile.txt')
```

```
t_arr = airheater_logfile(:,1); % time
u_arr = airheater_logfile(:,2); % control signal u
T_out_arr= airheater_logfile(:,3); % Out temperature T_out

plot(t_arr,u_arr,t_arr,T_out_arr)
legend('u','Tout')

% ranges of the parameters
T_env_r = [15,30,0.5]; % T_env
Kh_r = [0,8,0.5]; % K_h
etha_t_r = [1,250,2]; % time constant
etha_d_r = [1,8,0.3]; % time delay

T_env_min = inf;
kh_min = inf;
etha_t_min = inf;
etha_d_min = inf;


timestep = t_arr(5)-t_arr(4);

numsteps = length(t_arr)-int16((etha_d_r(2)-etha_d_r(1))/timestep) % how
many simulationsteps to run
% Must subtract the time delay so there won't be index out of range
issues

min_error = inf; % the square error will be stored here
simcounter = 0; % simple counter for the number of simulations done
for Tenv = T_env_r(1):T_env_r(3):T_env_r(2)
    for Kh = Kh_r(1):Kh_r(3):Kh_r(2)
        for etha_t = etha_t_r(1):etha_t_r(3):etha_t_r(2)
            for etha_d = etha_d_r(1):etha_d_r(3):etha_d_r(2)
                Theat = 0; % initial heating set to zero for each
                Tout = Tenv; % initial Tout equal to the environment
temperature
                errorsum = 0;
                simcounter = simcounter +1;
                for count = 1:numsteps
                    if count<=int16(etha_d/timestep)
                        dTheat = -Theat/etha_t+Kh*0/etha_t;
                        Theat = Theat + dTheat;
                        Tout = Tenv + Theat;
                    else
                        dTheat = -Theat/etha_t+Kh*u_arr(count-
int16(etha_d/timestep))/etha_t;
                        Theat = Theat + dTheat;
                        Tout = Tenv + Theat;

                    end
                    errorsum = errorsum+(Tout-T_out_arr(count))^2; % sum
the errors

                end
                % check if the new error is less then the old one, and
save
                % the parameters
                 if errorsum < min_error
                    min_error = errorsum;
                    disp(['found a new min ',num2str(errorsum)])
                    T_env_min = Tenv;
                    kh_min = Kh;
```

```
                    etha_t_min = etha_t;
                    etha_d_min = etha_d;
                end
            end
        end
    end
end


% Single simulation to display the results
Tenv = T_env_min;
Tout = T_out_arr(1);
Theat = Tout-Tenv; % initial Theat
Kh = kh_min;
theta_t = etha_t_min;
theta_d = etha_d_min;
testarr = zeros(numsteps,1);
testarr2 = zeros(numsteps,1);
testarr3 = zeros(numsteps,1);
testarr4 = zeros(numsteps,1);
for count = 1:numsteps
    if count<=int16(theta_d/timestep)
        dTheat = -Theat/theta_t+Kh*0/theta_t;
    else
        dTheat = -Theat/theta_t+Kh*u_arr(count-
int16(theta_d/timestep))/theta_t;
    end
    Theat = Theat + dTheat;
    Tout = Tenv + Theat;
    testarr2(count)=Tout; % final temperature
    testarr(count)=Theat; % heating value
    testarr3(count)=(T_out_arr(count)-Tout)^2; % errors
    testarr4(count)=T_out_arr(count); % the actual temperature
end
testerror = sum(testarr3);
pt = t_arr(1:numsteps);
plot(pt,testarr,pt,u_arr(1:numsteps),pt,testarr2,pt,testarr3,pt,testarr4)
legend('Theat','u','Tout simulated','error','Tout
real','Location','NorthEastOutside')
T_env_min
kh_min
etha_t_min
etha_d_min
```

In figure 4 below, the simulation with the real control variables from the experiment is shown. The result shows good agreement with the real and the simulated response.

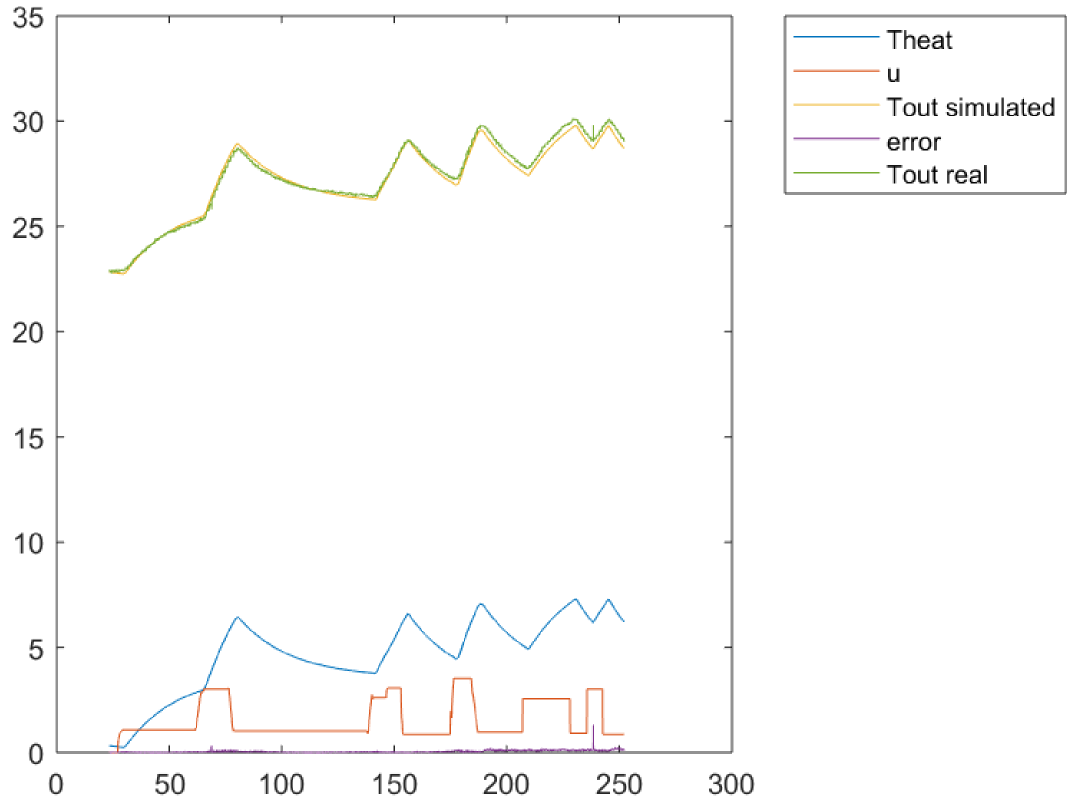

Figure 4: simulated vs real response of a real input signal

Parameter estimation of the air heater using the nonlinear least squares (NLS) method

For the life of me I was unable to implement the fmincon solution with time delay

Comparison of estimation results

For the life of me I was unable to implement the fmincon solution with time delay

Subspace identification of the air heater

For the life of me I was unable to implement the subspace solution with time delay