**Problem 1**

Model

Standard form of the system

Discretize the system

From Equation 1 and 2

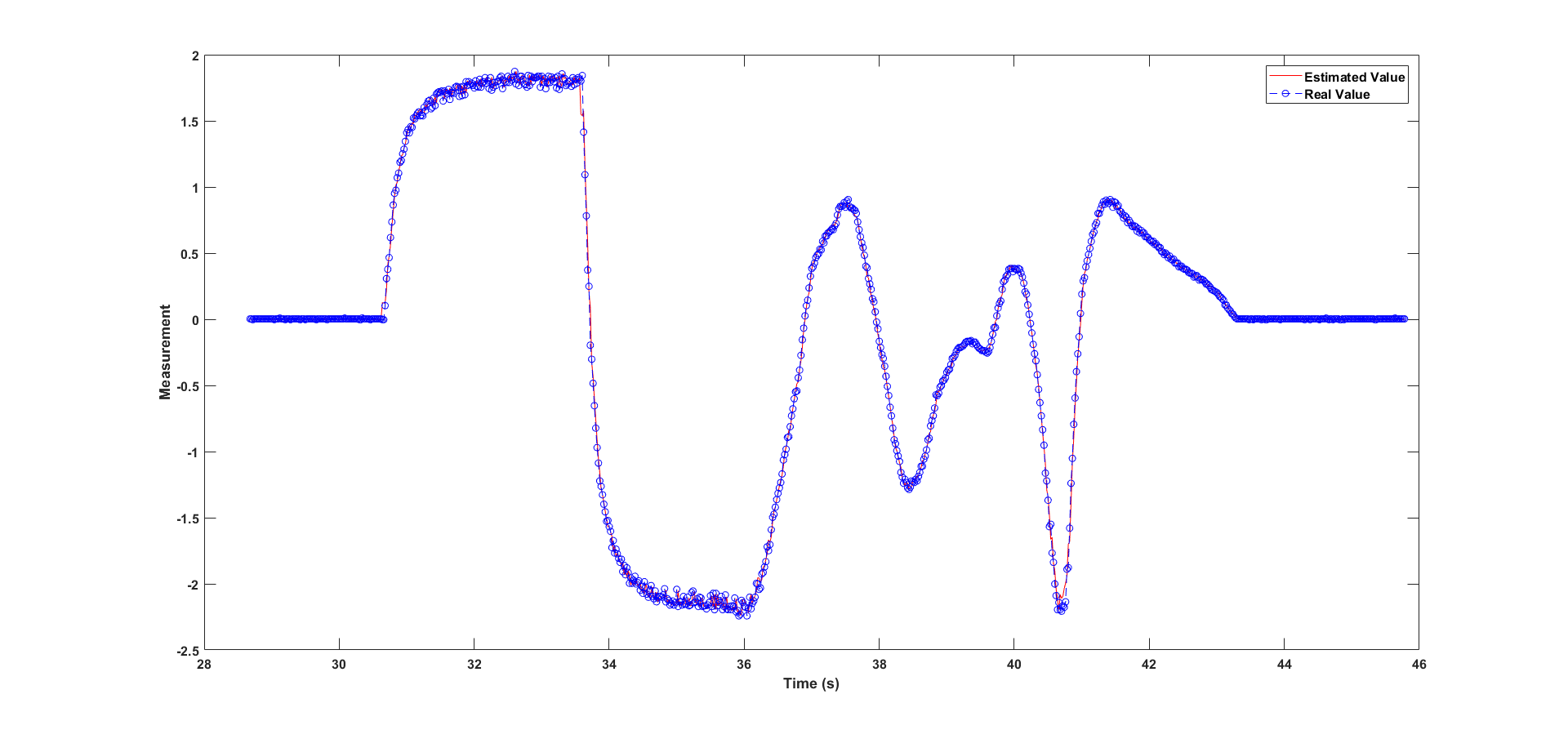
There are 3 parameters and the system is linear in its parameters

According to the formulae provided in the book for LS parameter estimation

Implementing the code in MATLAB with the file name Problem\_1.m. The code can also be found in the appendix.

The result from the matlab code provides the following values of the parameters

The model is simulated with the parameters obtained from the LS method. Figure 1. Shows the comparison between the model-generated values and the real values.



**Figure 1.** Comparison between model and real values.

From the plot presented in Figure 1. it can be observed that the values predicted by the model has a clear match with the data from the tachometer. Thus validating the parameter estimation.

**Problem 2**

The model

Substituting the value of Tk in the state equation

In the optimization problem statement, the parameters are the decision variables and the objective function is the error between the value of predicted by the model and the value obtained from the experiment. The grid search method is used to find the point which gives the minimum value of the least square error.

**Solution from Grid Search method**

The model is simulated with the parameters estimated using grid search method. The plot presented below shows a comparison between the estimated values of Y and the measured values.



**Figure 2**. Comparison between the model and the real value (Grid-search method)

The plot shows a good match between the values predicted by the model and the real measurement obtained from the temperature sensor.

**Problem 3**

The system described in the previous problem is solved using non-linear least square method.

The objective function is the least square error between the measured variable and the model predicted value. The decision variables are the unknown parameters of the model. The interior point algorithm is implemented using the ‘*fmincon’* function in matlab. The code for the solver is presented in Appendix.

The following values are obtained by solving the optimization problem.

Figure. 3 shows the comparison between the model predicted value and the real value of temperature sensor. The plot displays that the parameters estimated by the optimization solver is fairly accurate.



**Figure 3**. Comparison between the model and the real value (fmincon)

The plot presented in Figure 3 shows that there is a close match between the values from the model and the real values, which infers that the parameters estimated are correct.

**Problem 4**

**Table 1.** Comparison between Grid search and fmincon

|  |  |  |
| --- | --- | --- |
|  | **Grid Search (N=304)** | **fmincon** |
| Simulation Time | 17.8713 | 0.601137 |
| Objective function | 4.2919 | 5.3843 |
| Initial Guess | Not Required | Required |
| Minimum | Global Minimum | Local Minimum |

**Simulation time:**

The solver using fmincon is faster compared to the grid search method. The accuracy of solution and simulation time for the grid search method is highly depended on the number of grid points chosen. The figure below shows a comparison between the number of discrete points and the simulation time. It can be observed that increasing the number of grid points can exponentially increase the simulation time. The code for the sensitivity analysis is included in the appendix as well is attached as a file ‘sensitivity.m’.



**Figure 4.** Sensitivity analysis of objective function and simulation time with no. of discrete points.

While looking at the simulation time as a criteria, the grid-search is computationally intensive compared to the fmincon.

**Objective function**

The objective function obtained by the grid search method is lower, which indicates that the solver has identified a global minimum. In case of fmincon, the method is dependent on the initial guess, since there is a high possibility that the solver could be stuck in a local minimum.

From the comparison made above, it is observed that the nonlinear LS method using fmincon function in matlab is faster since the simulation time is about 0.6911 s compared to the grid search method which has a simulation time of 17 seconds (for N = 304).

**Initial guess**

Another problem with fmincon is the dependency to the initial condition. For the objective function described here, there are multiple local minimum, hence there is a possibility that the interior point optimization solver could be stuck in a local minimum. Here is a few example where changing the initial guess could give different values of objective function.

|  |  |  |
| --- | --- | --- |
| **Initial Guess** | **Solution** | **Objective function** |
| [5,60,10,28] | [1.0; 50.8; 10.02; 27.2] | 5.6611 |
| [2,20,6,21] | [2.30; 20.65; 6.05; 24.36] | 5.5309 |
| [2,20,5,28] | [2.4; 19.8; 5.3; 24.06] | 5.4869 |
| [3,23,3,21] | [3.14; 21.76; 3.25; 23.11] | 5.3860 |

Therefore, the closer the initial guess is to the real solution, the higher are the chance of obtaining a solution, which satisfies global minimum. This problem does not exist while using the grid-search method since the solver searches over the entire solution space.

**Problem 5:**

A linear state-space model is built using the data for Air Heater. The subspace identification toolbox of Matlab has been used for the identification. The number of states has been fixed at 1. The code can be found in Appendix.

The delay can be added to the model by using the term ('InputDelay',delay)in the n4sid command.



Assuming that we have no knowledge about the time delay for the system. The grid search method is implemented to identify the value of time delay, which has the minimum least square error.

From the Matlab code the optimal value of time delay in the control function is obtained as 2.3 s.

The values of A, B, C and D are obtained from the Matlab code ‘Problem\_5.m’ (also presented in Appendix) as follows.

Input delays (sampling periods): 2.3 s

**Appendix**

1. **Problem 1**

clear all; clc;

%% Extract data from the txt file into matlab workspace and define known parameters

data\_sys=load('logfile1.txt');

K\_m=5;

del\_t=0.02;

%% Selecting the estimation and validation dataset

L=length(data\_sys);

F=0.7;%First fraction (portion) of logfile to be used for model

N=floor(L\*F); %N is number of samples used for estimation.

%Rest used for visual validation.

t\_estim=del\_t\*[1:N]';

u\_estim=data\_sys(1:N,2);

y\_estim=data\_sys(1:N,3);

t\_valid=del\_t\*[N+1:L]';

u\_valid=data\_sys(N+1:L,2);

y\_valid=data\_sys(N+1:L,3);

t\_total=del\_t\*[1:L]';

u\_total=data\_sys(1:L,2);

y\_total=data\_sys(1:L,3);

%% Creating the system

C = u\_estim(1:end-1,:); %control signal

S = y\_estim(1:end-1,:)./K\_m; %speed in the previous time step

Y = y\_estim(2:end,:)./K\_m; %speed

phi=[S C ones(length(C),1)]; %Ø – phi

theta = inv(phi'\*phi)\*phi'\*Y;%Parameter estimation using Least square.

%% Calculating the parameters T, K and L

T=del\_t/(1-theta(1))

K=theta(2)\*T/(del\_t)

L=theta(3)\*T/(K\*del\_t)

%% Validating the model

C = u\_total(1:end-1,:); %control signal

S = y\_total(1:end-1,:)./K\_m; %speed in the previous time step

Y = y\_total(2:end,:)./K\_m; %speed in the current time step

y\_est=[S C ones(length(C),1)]\*theta;

%% Plot and data Visualisation

plot(t\_total(2:end),y\_est,'-r',t\_total(2:end),Y,'b--o');

xlabel('Time (s)');

ylabel('Measurement');

legend('Estimated Value','Real Value');

1. **Problem 2**

%Finn Haugen (finn@techteach.no)

%----------------------------------------------------------

disp('-------------------')

disp('Grid search method to solve an')

disp('optimization problem:')

disp('min\_x f(x)')

disp('s.t.')

disp('model: f= function which calculates the error "AC\_heater\_error";')

disp('constraints:')

disp('2 <= x1 <= 5')%Kh

disp('15 <= x2 <= 25')%theta t

disp('0 <= x3 <= 4')%theta d

disp('15 <= x4 <= 30')%T\_env

%----------------------------------------------------------

clear all

close all

format compact

data\_sys = load('airheater\_logfile.txt');

%% Selecting the estimation and validation dataset

L=length(data\_sys);

F=0.7;%First fraction (portion) of logfile to be used for model adaption:

N=floor(L\*F); %N is number of samples used for estimation.

del\_t=0.1;

%Rest used for visual validation.

t\_estim=del\_t\*[1:N]';

u\_estim=data\_sys(1:N,2);

y\_estim=data\_sys(1:N,3);

t\_valid=del\_t\*[N+1:L]';

u\_valid=data\_sys(N+1:L,2);

y\_valid=data\_sys(N+1:L,3);

t\_total=del\_t\*[1:L]';

u\_total=data\_sys(1:L,2);

y\_total=data\_sys(1:L,3);

%% Grid Search Method

%Initialization:

x1\_min=2;x1\_max=5;N\_x1=50;

x1\_array=linspace(x1\_min,x1\_max,N\_x1);

x2\_min=15;x2\_max=25;N\_x2=50;

x2\_array=linspace(x2\_min,x2\_max,N\_x2);

x3\_min=0;x3\_max=4;N\_x3=101;

x3\_array=linspace(x3\_min,x3\_max,N\_x3);

x4\_min=15;x4\_max=30;N\_x4=50;

x4\_array=linspace(x4\_min,x4\_max,N\_x4);

f\_min=inf;

x1\_opt=-inf;

x2\_opt=-inf;

x3\_opt=-inf;

x4\_opt=-inf;

tic;

for k\_x1=1:length(x1\_array)

x1=x1\_array(k\_x1);

for k\_x2=1:length(x2\_array)

x2=x2\_array(k\_x2);

for k\_x3=1:length(x3\_array)

x3=x3\_array(k\_x3);

for k\_x4=1:length(x4\_array)

x4=x4\_array(k\_x4);

%Objective function:

f=AC\_heater\_error(x1,x2,x3,x4,u\_estim, y\_estim);

%Improving the previous solution:

if f <= f\_min,

f\_min=f;

x1\_opt=x1;

x2\_opt=x2;

x3\_opt=x3;

x4\_opt=x4;

end

end

end

end

end

%--------------------------------------------------

%Displaying the optimal solution:

disp('-------------------')

disp('Optimal solution:')

f\_min

K\_opt=x1\_opt

thetaT\_opt=x2\_opt

thetaD\_opt=x3\_opt

T\_env=x4\_opt

%--------------------------------------------------

%Calculation of execution times:

disp('-------------------')

disp('Execution times of running the nested for-loops:')

dt\_elapsed\_if=toc

dt\_elapsed\_per\_cycle\_if=toc/(N\_x1\*N\_x2\*N\_x3\*N\_x4)

N\_tot=N\_x1\*N\_x2\*N\_x3\*N\_x4;

%% Model Validation

A=(1-del\_t/thetaT\_opt);

B=[del\_t\*K\_opt/thetaT\_opt del\_t/thetaT\_opt] ;

C=1;

D=[0 0];

model\_est=ss(A,B,C,D,del\_t,'InputDelay',round(thetaD\_opt/del\_t));

u\_total=[u\_total T\_env.\*ones(length(u\_total),1)];

x0=y\_total(1);

[y\_sim,t,x] = lsim(model\_est,u\_total,[],x0);

%% Plots

h=figure; %Getting figure handle

fig\_posleft=8;fig\_posbottom=1.5;fig\_width=24;fig\_height=20;

fig\_pos\_size\_1=[fig\_posleft,fig\_posbottom,fig\_width,fig\_height];

set(gcf,'Units','centimeters','Position',fig\_pos\_size\_1);

figtext='Estimation of model params of Air-Heater';

set(gcf,'Name',figtext,'NumberTitle','on')

subplot(2,1,1)

figure(1)

plot(t\_total,y\_sim,'r',t\_total,y\_total,'b');

title('Real y (blue). Simulated y with adapted model (red). Green: Interval for adaption.')

grid minor

ylim([20,35]);

ylabel('[V]');xlabel('t [s]')

subplot(2,1,2)

plot(t\_total,u\_total,'b');

title('Applied control sigal, u, to both real and simulated process. Green: Interval for adaptation.')

grid minor

ylim([0,4]);

ylabel('[V]');xlabel('t [s]')

%% Function for calculating the error

function f = AC\_heater\_error(x1,x2,x3,x4,u,y)

Kh=x1; % Parameter Kh

theta\_t=x2; % Parameter theta\_t – θt

dt=0.1;

theta\_d=max(0,round(x3/dt)-1); % Parameter theta\_d – θd

Tprev\_meas=y(1:end-1);

Tout\_meas=y(2:end);

u = [u(1)\*ones(theta\_d,1); u(1:end-theta\_d-1)]; % include delay in the control signal

T\_env=x4.\*ones(length(u),1); % Parameter environment temperature – Tenv

T\_est=((1-dt/theta\_t).\*Tprev\_meas)+((Kh\*dt/theta\_t).\*u)+(dt/theta\_t.\*T\_env); % State Eq

error=Tout\_meas-T\_est;

f=error'\*error; % least square error

end

1. **Problem 3**

%----------------------------------------------------------

disp('-------------------')

disp('Using fmincon() in Matlab solve an optimization')

disp('(Nonlinear Programming, NLP) problem:')

disp('min\_x f(x)')

disp('s.t.')

disp('model: f= AC\_heater\_error(x1,x2,x3,x4,data)')

disp('constraints:')

disp('1 <= x1 <= 10')

disp('10 <= x2 <= 100')

disp('1 <= x3 <= 20')

disp('15 <= x4 <= 30')

%----------------------------------------------------------

format compact

clear all

close all

%--------------------------------------------------

% Load the data

global u\_estim y\_estim

data\_sys = load('airheater\_logfile.txt');

%% Selecting the estimation and validation dataset

L=length(data\_sys);

F=0.7;%First fraction (portion) of logfile to be used for model adaption:

N=floor(L\*F); %N is number of samples used for estimation.

del\_t=0.1;

%Rest used for visual validation.

t\_estim=del\_t\*[1:N]';

u\_estim=data\_sys(1:N,2);

y\_estim=data\_sys(1:N,3);

t\_valid=del\_t\*[N+1:L]';

u\_valid=data\_sys(N+1:L,2);

y\_valid=data\_sys(N+1:L,3);

t\_total=del\_t\*[1:L]';

u\_total=data\_sys(1:L,2);

y\_total=data\_sys(1:L,3);

%% Parameter estimation using Nonlinear least squares (fmincon)

% Contraints:

x\_lb=[1,10,1,15]';

x\_ub=[10,100,20,30]';

%--------------------------------------------------

%fmincon:

x\_guess=[7,2,4,20]';

Aineq=[]; Bineq=[]; Aeq=[]; Beq=[];

tic

fun\_objective\_handle = @(x)fun\_objective(x);%Local function

fun\_constraints\_handle = @(x)fun\_constraints(x);%Local function

optim\_options=optimset('Display','on');

%

[x\_opt,fval,exitflag,output,lambda,grad,hessian] =...

fmincon(fun\_objective\_handle,x\_guess,Aineq,Bineq,Aeq,Beq,x\_lb,x\_ub,...

fun\_constraints\_handle,optim\_options);

toc

%--------------------------------------------------

%Displaying the optimal solution:

disp('-------------------')

disp('Optimal solution:')

fval

x\_opt

K\_opt=x\_opt(1)

thetaT\_opt=x\_opt(2)

thetaD\_opt=x\_opt(3)

T\_env=x\_opt(4)

%% Model Validation

del\_t=0.1; % Time step

A=(1-del\_t/thetaT\_opt);

B=[del\_t\*K\_opt/thetaT\_opt del\_t/thetaT\_opt] ;

C=1;

D=[0 0];

model\_est=ss(A,B,C,D,del\_t,'InputDelay',round(thetaD\_opt/del\_t)); % Create a state-space model

u\_total=[u\_total T\_env.\*ones(length(u\_total),1)];

x0=y\_total(1); % Define the initial value of the simulation

[y\_sim,t,x] = lsim(model\_est,u\_total,[],x0);

%% Plots

h=figure; %Getting figure handle

fig\_posleft=8;fig\_posbottom=1.5;fig\_width=24;fig\_height=20;

fig\_pos\_size\_1=[fig\_posleft,fig\_posbottom,fig\_width,fig\_height];

set(gcf,'Units','centimeters','Position',fig\_pos\_size\_1);

figtext='Estimation of model params of Air-Heater';

set(gcf,'Name',figtext,'NumberTitle','on')

subplot(2,1,1)

figure(1)

plot(t\_total,y\_sim,'r',t\_total,y\_total,'b');

title('Real y (blue). Simulated y with adapted model (red). Green: Interval for adaption.')

grid minor

ylim([15,35]);

ylabel('[V]');xlabel('t [s]')

subplot(2,1,2)

plot(t\_total,u\_total,'b');

title('Applied control sigal, u, to both real and simulated process. Green: Interval for adaptation.')

grid minor

ylim([0,4]);

ylabel('[V]');xlabel('t [s]')

%% Function for calculating the error

function f = fun\_objective(x)

global u\_estim y\_estim

Kh=x(1);

theta\_t=x(2);

dt=0.1;

theta\_d=round(x(3)/dt)+1;

Tprev\_meas=y\_estim(1:end-1);

Tout\_meas=y\_estim(2:end);

u=u\_estim(1:end);

u = [u(1)\*ones(theta\_d,1); u(1:end-theta\_d-1)];

T\_env=x(4).\*ones(length(u),1);

T\_est=((1-dt/theta\_t).\*Tprev\_meas)+((Kh\*dt/theta\_t).\*u)+(dt/theta\_t.\*T\_env);

error=Tout\_meas-T\_est;

f=error'\*error;

end

% Defining constraints

function [cineq,ceq]=fun\_constraints(x)

cineq = []; % Compute nonlinear inequalities.

% cineq = -x(2)+x(1)+2; % Compute nonlinear inequalities.

%Inequality constraint: x2 >= x1+2

ceq = []; % Compute nonlinear equalities.

end

1. **Problem 5**

%----------------------------------------------------------

%Parameter estimation of a AirHeater with n4sid subspace identification.

%Finn Haugen (finn.haugen@usn.no)

%----------------------------------------------------------

clear all

close all

%Loads data from file into workspace.

load airheater\_logfile.txt;

Ts=0.1; %Sampling interval

%% Selecting the estimation and validation dataset

L=length(airheater\_logfile);%(Matrix name becomes same as logfile name.)

F=0.8;%First fraction (portion) of logfile to be used for model adaption:

N=floor(L\*F); %N is number of samples used for estimation.

%Rest used for visual validation.

t\_estim=Ts\*[1:N]';

u\_estim=airheater\_logfile(1:N,2);

y\_estim=airheater\_logfile(1:N,3);

t\_valid=Ts\*[N+1:L]';

u\_valid=airheater\_logfile(N+1:L,2);

y\_valid=airheater\_logfile(N+1:L,3);

t\_total=Ts\*[1:L]';

u\_total=airheater\_logfile(1:L,2);

y\_total=airheater\_logfile(1:L,3);

%% Grid Search method to identify the optimal value of control signal delay

x1\_min=1;x1\_max=40;N\_x1=40;

x1\_array=linspace(x1\_min,x1\_max,N\_x1);

f\_min=inf;

x1\_opt=-inf;

for k\_x1=1:length(x1\_array)

x1=x1\_array(k\_x1);

% Subspace model identification

modelorder=1;%Defines order of estimated model.

% Estimation of model. Model is on internal theta-format:

data\_estim=iddata(y\_estim,u\_estim,Ts); %data used for estimation

[model\_est]=n4sid(data\_estim,modelorder,'InputDelay',x1);

f=model\_est.Report.Fit.MSE; %least square error

if f <= f\_min,

f\_min=f;

x1\_opt=x1;

end

end

data\_total=iddata(y\_total,u\_total,Ts); % the complete dataset

[model\_est,x0]=n4sid(data\_total,modelorder,'InputDelay',x1\_opt); % Generating subspace model

[y\_sim,t,x] = lsim(model\_est,data\_total.InputData,[],x0); %simulating the

LSE=model\_est.Report.Fit.MSE %least square error

%% Plots

h=figure; %Getting figure handle

fig\_posleft=8;fig\_posbottom=1.5;fig\_width=24;fig\_height=20;

fig\_pos\_size\_1=[fig\_posleft,fig\_posbottom,fig\_width,fig\_height];

set(gcf,'Units','centimeters','Position',fig\_pos\_size\_1);

figtext='Estimation of model params of DC motor';

set(gcf,'Name',figtext,'NumberTitle','on')

subplot(2,1,1)

figure(1)

plot(t\_total,y\_sim,'r',t\_total,y\_total,'b');

title('Real y (blue). Simulated y with adapted model (red). Green: Interval for adaption.')

grid minor

ylim([15,45]);

ylabel('[Temperature]');xlabel('t [s]')

subplot(2,1,2)

plot(t\_total,u\_total,'b');

title('Applied control signal, u, to both real and simulated process. Green: Interval for adaptation.')

grid minor

ylim([0,4]);

ylabel('[V]');xlabel('t [s]')

1. **Sensitivity- GridSearch**

%% sensitivity analysis

format compact

data\_sys = load('airheater\_logfile.txt');

%% Selecting the estimation and validation dataset

L=length(data\_sys);

F=0.7;%First fraction (portion) of logfile to be used for model adaption:

N=floor(L\*F); %N is number of samples used for estimation.

del\_t=0.1;

%Rest used for visual validation.

t\_estim=del\_t\*[1:N]';

u\_estim=data\_sys(1:N,2);

y\_estim=data\_sys(1:N,3);

t\_valid=del\_t\*[N+1:L]';

u\_valid=data\_sys(N+1:L,2);

y\_valid=data\_sys(N+1:L,3);

t\_total=del\_t\*[1:L]';

u\_total=data\_sys(1:L,2);

y\_total=data\_sys(1:L,3);

%% Grid Search Method

for i=1:10

%Initialization:

x1\_min=2;x1\_max=10;N\_x1=10+(i-1)\*5;

x1\_array=linspace(x1\_min,x1\_max,N\_x1);

x2\_min=10;x2\_max=30;N\_x2=10+(i-1)\*5;

x2\_array=linspace(x2\_min,x2\_max,N\_x2);

x3\_min=0;x3\_max=5;N\_x3=10+(i-1)\*5;

x3\_array=linspace(x3\_min,x3\_max,N\_x3);

x4\_min=15;x4\_max=35;N\_x4=10+(i-1)\*5;

x4\_array=linspace(x4\_min,x4\_max,N\_x4);

f\_min=inf;

x1\_opt=-inf;

x2\_opt=-inf;

x3\_opt=-inf;

x4\_opt=-inf;

tic;

for k\_x1=1:length(x1\_array)

x1=x1\_array(k\_x1);

for k\_x2=1:length(x2\_array)

x2=x2\_array(k\_x2);

for k\_x3=1:length(x3\_array)

x3=x3\_array(k\_x3);

for k\_x4=1:length(x4\_array)

x4=x4\_array(k\_x4);

%Objective function:

f=AC\_heater\_error(x1,x2,x3,x4,u\_estim, y\_estim);

%Improving the previous solution:

if f <= f\_min,

f\_min=f;

x1\_opt=x1;

x2\_opt=x2;

x3\_opt=x3;

x4\_opt=x4;

end

end

end

end

end

disp('-------------------')

disp('Optimal solution:')

f\_min

dt\_elapsed\_if=toc

obj(i)=f\_min;

sim\_time(i)=dt\_elapsed\_if;

X(i)=N\_x1;

dt\_elapsed\_per\_cycle\_if=toc/(N\_x1\*N\_x2\*N\_x3\*N\_x4)

N\_tot=N\_x1\*N\_x2\*N\_x3\*N\_x4;

end

%% Plot the results

yyaxis left

plot(X,obj);

yyaxis right

plot(X,sim\_time);

yyaxis left

title('Grid-search with different resolution')

xlabel('Number of discrete points')

ylabel('objective function')

yyaxis right

ylabel('Simulation time')

%% Function for calculating the error

function f = AC\_heater\_error(x1,x2,x3,x4,u,y)

Kh=x1;

theta\_t=x2;

dt=0.1;

theta\_d=max(0,round(x3/dt)-1);

Tprev\_meas=y(1:end-1);

Tout\_meas=y(2:end);

u = [u(1)\*ones(theta\_d,1); u(1:end-theta\_d-1)];

T\_env=x4.\*ones(length(u),1);

T\_est=((1-dt/theta\_t).\*Tprev\_meas)+((Kh\*dt/theta\_t).\*u)+(dt/theta\_t.\*T\_env);

error=Tout\_meas-T\_est;

f=error'\*error;

end