Exercise 1

The equation representing “time-constant dynamics” is:

T\*dS/dt + S = K\*(C + L)

To calculate parameters by using least square, we can transform it into:

dS/dt = [C; 1; -S] [K/T; L\*K/T; 1/T]’

dS/dt is the observed variable, [C; 1; -S] is the regression vector, [K/T; L\*K/T; 1/T]’ is the unknown parameter vector.

In Matlab, I calculated dS/dt using Euler forward method:

%ds\_dt is the known value

ds\_dt = zeros(L-1,1);

%Calculate ds\_dt using Euler forward

for i=1:L-1

ds\_dt(i) = (S(i+1)-S(i))/dt;

end

Then defined the regression vector using:

%Define the Regression matrix

Regression=[C(1:L-1) ones(L-1,1) -S(1:L-1)];

%Calculate the unknown values

b = ((Regression'\*Regression)\Regression')\*ds\_dt;

The calculated values are:

K = 0.1772

L = -0.0625

T = 0.3035

Exercise 2

We have the following equations:

Tout = Tenv + Theat

thetat \* d(Theat)/dt = - Theat + Kh \* u(t-thetad)

Because Theat = Tout - Tenv

So we can calculate d(Theat)/dt using the known value, Tout

d(Theat)/dt = ((Theat)i+1 - (Theat)i)/dt = ((Tout - Tenv)i+1 - (Tout - Tenv)i)/dt = ((Tout)i+1 - (Tout)i)/dt

Please note Tenv is a constant:

dTHeat\_dt = zeros(L-1,1);

%Calculate dTHeat\_dt using Euler forward

for i=1:L-1

dTHeat\_dt(i) = (Tout(i+1)-Tout(i))/dt;

end

The objective function is implemented as:

%Objective function:

u\_delay = zeros(L-1,1);

u\_delay(thera\_d:end) = u(1:L-thera\_d);%implement time delay

Tout\_est = Tenv + (Kh\*u\_delay - thera\_t\*dTHeat\_dt);%calculate Tout

f = sqrt(mean((Tout(1:L-1) - Tout\_est).^2));%calculate root measn square error

To implement the grid search, I used four ‘for loops’ for thera\_d, Tenv, thera\_t and Kh respectively.

for k\_theta\_d=1:length(theta\_d\_array)

theta\_d=round(theta\_d\_array(k\_theta\_d));%use round to ensure the time delay is an integer

for k\_Tenv=1:length(Tenv\_array)

Tenv=Tenv\_array(k\_Tenv);

for k\_theta\_t=1:length(theta\_t\_array)

theta\_t=theta\_t\_array(k\_theta\_t);

for k\_Kh=1:length(Kh\_array)

Kh=Kh\_array(k\_Kh);

Exercise 3

The objective function is implemented in a function Matlab file:

function f=fun\_Ex5\_objective(x,u,Tout,L,dTHeat\_dt)

thera\_d=round(x(1));

Tenv=x(2);

thera\_t=x(3);

Kh=x(4);

u\_delay = zeros(L-1,1);

u\_delay(thera\_d:end) = u(1:L-thera\_d);

Tout\_est = Tenv + (Kh\*u\_delay - thera\_t\*dTHeat\_dt);

f = sqrt(mean((Tout(1:L-1) - Tout\_est).^2));

end

Exercise 4

In both grid search and fmincon method, I used same ranges for the four parameters:

thera\_d\_min=1;thera\_d\_max=10;

Tenv\_min=15;Tenv\_max=25;

thera\_t\_min=15;thera\_t\_max=25;

Kh\_min=1;Kh\_max=10;

The grid search gives the following optimal solution:

f\_min = 8.4132

thera\_d\_opt = 10

Tenv\_opt = 25

thera\_t\_opt = 15

Kh\_opt = 2

The fmincon gives the following optimal solution:

f\_min = 8.4344

theta\_d\_opt = 4.9165

Tenv\_opt = 24.8736

thera\_t\_opt = 15.0000

Kh\_opt = 1.8767

Both methods present similar results, but fmincon is much faster than grid search.

Exercise 5

I first load the data into Matlab:

%Loads data from file into workspace.

load airheater\_logfile.txt;

Ts=0.1; %Sampling interval

L=length(airheater\_logfile);%length of logfile

F=0.5;%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);

Then estimated paramters using subspace method:

model\_est=n4sid(iddata(y\_estim,u\_estim,Ts),modelorder,'InputDelay',3);