Otto-von-Guericke-University Magdeburg
Faculty of Electrical Engineering and Information Technology
Chair for Automation/Modelling

Project Report

State Estimation

Group 2

Temperature Control Lab

Submitted: October 15, 2021

by: Arman Ahmed Khan (230601)
Shailesh Kumar (229916)

Contents

1 Introduction **2**

1.1 Project Task I 2
1.2 Model equations 2
1.3 Matlab Energy Balance Equation Code 3
1.4 Matlab S-Function code 4
1.5 Temperature Control Lab Model 7

2 Project tasks - II **10**

2.1 Linear case - Open-Loop 10
2.2 Linear case - Closed-Loop 19
2.3 Nonlinear case 26
2.4 Simultaneous state and parameter estimation 31

Bibliography **38**

List of Figures

1.1 Simulink Model of TC Lab 6

2.1 Energy balance Model Step response 10
2.2 Heater waveform as step input 11
2.3 TC Lab Step response 12
2.4 Comparison of Linearized Simulink output vs real process model output vs KF estimate 13
2.5 Comparison of Linearized Simulink output vs real process model output vs KF estimate 15
2.6 Comparison of Linearized Simulink output vs real process model output vs KF estimate with 30 percent increment in model parameter 16
2.7 Comparison of Linearized Simulink output vs real process model output vs KF estimate with 30 percent decrement in model parameter 16
2.8 Process model and measurement from sensors in Simulink 18
2.9 Linear Closed loop Simulink Model with Kalman filter 20
2.10 Integral state feedback controller Simulink Model 21
2.11 Integral state feedback controller block diagram 21
2.12 Comparison of Estimated state vs Measured state vs Prior State using Linear Kalman Filter in Closed loop linear model 22
2.13 Comparison of Estimated state vs Measured state vs Prior State using Linear Kalman Filter in Closed loop linear model when As is increased 30 percent 23
2.14 Comparison of Estimated state vs Measured state vs Prior State using Linear Kalman Filter in Closed loop linear model when As is decreased 30 percent 23
2.15 Linear Kalman Filter with Integral State feedback controller with Q=.3 & R=1 and N=0 24
2.16 Linear Kalman Filter with Integral State feedback controller with Q=.1 & R=1 and N=0 (30% decrement) 24
2.17 linear Kalman Filter with Integral State feedback controller with Q=.7 & R=1 and N=0 (30% increment) 25
2.18 Non-Linear Extended Kalman Filter Simulink model 28
2.19 Non-Linear Extended Kalman Filter state estimation vs noisy measurement 28
2.20 Non-Linear Extended Kalman Filter Model parameter and noise increased 29

2.21 Non-Linear Extended Kalman Filter Model parameter and noise decreased 30
2.22 RGA analysis with Model linearization Step plot 34
2.23 Simulink model of Augmented system with measurement and process noise 35
2.24 Simulink model of Augmented system with measurement noise 35
2.25 Augmented system with measurement noise vs Extended Kalman filter state estimation 36
2.26 Augmented system with measurement and process noise vs Extended Kalman filter state estimation 37

List of Tables

1.1 Table 1: Parameters for nonlinear model 3

1 Introduction

1.1 Project Task I

The Temperature Control Lab is an application of feedback control with an Arduino, a Light Emitting diode, two heaters, and two temperature sensors. The heater power output is adjusted to maintain a desired temperature set point. Thermal energy from the heater is transferred by conduction, convection, and radiation to the temperature sensor. Heat is also transferred away from the device to the surroundings [1].

1.2 Model equations

In the following, the nonlinear equations for dynamics of the Temperature Control Lab (TCL) elements are given. The energy balance for the heating elements is given as:

$$mc_p[\frac{dT_{H1}}{dt}] = kA(T_\infty - T_{H1}) + \epsilon\sigma A(T_\infty^4 - T_{H1}^4) + Q_{con,1,2} + Q_{rad,1,2} + \alpha, Q_1, \tag{1.1}$$

$$mc_p[\frac{dT_{H2}}{dt}] = kA(T_\infty - T_{H1}) + \epsilon\sigma A(T_\infty^4 - T_{H1}^4) + Q_{con,1,2} + Q_{rad,1,2} + \alpha, Q_2, \tag{1.2}$$

with heat exchange between two heaters by convection and radiation is defined as

$$Q_{rad,1,2} = k_s A_s(T_{H2} - T_{H1}) \tag{1.3}$$

$$Q_{rad,1,2} = \epsilon\sigma A_s(T_{H2}^4 - T_{H1}^4) \tag{1.4}$$

Heat transfer between heater and sensor is restricted to convection where the heat loss of the heater is negligible. Thus, dynamics of the sensor temperatures read as

$$\frac{dT_{s1}}{dt} = \frac{1}{\tau}(T_{H1} - T_{s1}) \tag{1.5}$$

$$\frac{dT_{s2}}{dt} = \frac{1}{\tau}(T_{H2} - T_{s1}) \tag{1.6}$$

and the lag constant is given by

$$\tau = \frac{m_s c_p, s}{kA_s} \tag{1.7}$$

Table 1.1: Table 1: Parameters for nonlinear model

S/N	Parameter	Value	Unit
1	α_1	0.01	$W(\%)^{-}1$
2	α_2	0.0075	$W(\%)^{-}1$
3	A	10	cm^2
4	A_s	2	cm^2
5	k	10	$\frac{W}{m^2K}$
6	k_s	20	$\frac{W}{m^2K}$
7	Q_1	[0,100]	W
8	Q_2	[0,100]	W
9	τ	20	s

We started by creating an energy balance equation file in Matlab, and this is the following code.

1.3 Matlab Energy Balance Equation Code

```
% define energy balance model
function dTdt = energy_bal(t,x,u)

    global Ta k m Cp A As alpha1 alpha2 eps sigma tao ks

    % Parameters
    Ta = 23 ;            %  C
    k = 10.0;            % W/m^2- C
    m = 4.0/1000.0;      % kg
    Cp = 0.5 * 1000.0;   % J/kg- C
    A = 10.0 / 100.0^2; % Area in m^2
    As = 2.0 / 100.0^2; % Area in m^2
    alpha1 = 0.0100;     % W / % heater 1
    alpha2 = 0.0075;     % W / % heater 2
    eps = 0.9;           % Emissivity
    sigma = 5.67e-8;     % Stefan-Boltzman
    tao = 20;            % second
    ks = 20;             % W/m^2- C

    % Temperature States
    Th1 = x(1);
    Th2 = x(2);
    Ts1 = x(3);
    Ts2 = x(4);
    Q1 = u(1);
    Q2 = u(2);

```

```
28
29      % Heat Transfer Exchange Between 1 and 2
30      conv12 = ks*As*(Th2-Th1);
31      rad12  = eps*sigma*As * (Th2^4 - Th1^4);
32
33      % Nonlinear Energy Balances
34      dTh1dt = (1.0/(m*Cp))*(k*A*(Ta-Th1) + eps * sigma * A * (Ta^4 - ...
            Th1^4) + conv12 + rad12 + alpha1*Q1);
35      dTh2dt = (1.0/(m*Cp))*(k*A*(Ta-Th2) + eps * sigma * A * (Ta^4 - ...
            Th2^4) - conv12 - rad12 + alpha2*Q2);
36      dTs1dt = (1.0/tao)*(Th1-Ts1);
37      dTs2dt = (1.0/tao)*(Th2-Ts2);
38
39      % Output
40      dTdt = [dTh1dt,dTh2dt,dTs1dt,dTs2dt]';
41  end
```

The energy balance equation model is the mathematical and non-linear model of TC Lab, which is based on Conduction and radiation of heat. It takes two input that are the heater 1 signal and heater 2 signal. Our mathematical model has 4 states which defined with 4 differential equation and the solution of these equations is calculated by @ode45 MATLAB function. After that, we made an S-Function to connect the energy balance equation with the Simulink model, and this is the following code.

1.4 Matlab S-Function code

```
1  function [sys,x0] = energy_bals(t,x,u,flag)
2  %
3  %  Simulink interface to TC Lab
4  %
5  % Inputs:    t    - time in [seconds].
6  %            x    - 4 states, Temperature of Heater 1.
7  %                   Temperature of Heater 2.
8  %                   Temperature of Sensor 1.
9  %                   Temperature of Sensor 2.
10 %           u(1) = Q1   - Heater 1 level (0-100%)
11 %           u(2) = Q2   - Heater 2 level (0-100%)
12 %
13 % Outputs:   sys and x0 as described in the SIMULINK manual.
14 %            when flag is 0 sys contains sizes and x0 contains
15 %            initial condition.
16 %            when flag is 1, sys contains the derivatives,
17 %            and when flag is 4 sys contains outputs;
18 %            y(1)    - Temperature of Heater 1.
```

```
19 %            y(2)    - Temperature of Heater 2.
20 %            y(3)    - Temperature of Sensor 1.
21 %            y(4)    - Temperature of Sensor 2.
22 % Initialize :  define initial conditions,  X0
23 %               define system in terms of number of states, inputs etc.
24 %             e.g. sys = [2, 0, 2, 1, 0, 0];
25 %            1st array :  number of continuous states
26 %            2nd array :  number of discrete states
27 %            3rd array :  number of outputs
28 %            4th array :  number of inputs
29 %            5th array :  flag for direct feedthrough
30 %            6th array :  the number of sample times
31
32
33 if flag == 0
34   % Initialize the system
35   x0 = [ 23 23 23 23 ]
36   sys = [4, 0, 4, 2, 0, 0];
37 elseif abs(flag) == 3
38   % Return system outputs.
39   sys(1,1,1) = x(1);       %  Th1
40   sys(2,1,1) = x(2);       %  Th2
41   sys(3,1,1) = x(3);       %  Ts1
42   sys(4,1,1) = x(4);       %  Ts2
43 elseif  abs(flag) == 1
44   % Return state derivatives.
45   sys = energy_bal(t,x,u);
46 else
47   sys = [];
48 end
```

The Simulink model of energy balance equation is attached below. Here we can see the Step response waveform applied to heater 1 & 2.

Simulink Model

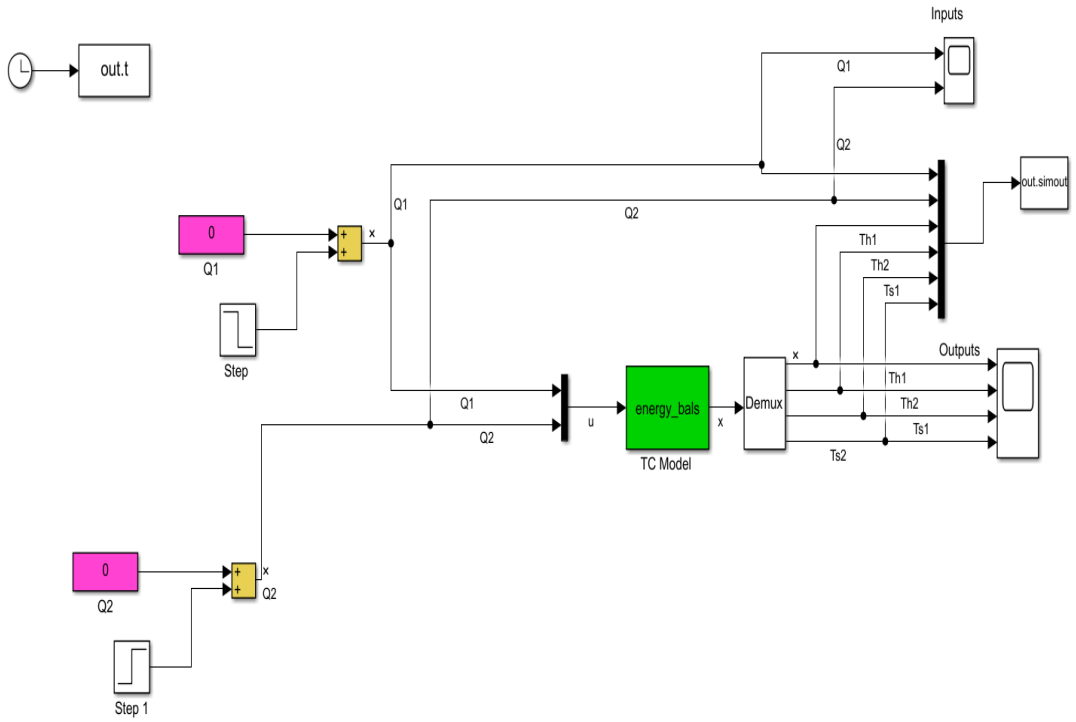

Figure 1.1: Simulink Model of TC Lab

After the completion of Simulink model, we created a MATLAB function which uses a predefined function to read and write data from and to Arduino. The Arduino send data regarding the voltage provided to different heater, and we calculate the temperature accordingly. The Matlab code for reading data from TC Lab is given below [1].

```
% connect to Arduino
try
    a = arduino;
    disp(a)
catch
    warning('Unable to connect, user input required')
    disp('For Windows:')
    disp('  Open device manager, select "Ports (COM & LPT)"')
    disp('  Look for COM port of Arduino such as COM4')
    disp('For MacOS:')
    disp('  Open terminal and type: ls /dev/*.')
    disp('  Search for /dev/tty.usbmodem* or /dev/tty.usbserial*. The ...
        port number is *.')
```

```
    disp('For Linux')
    disp('  Open terminal and type: ls /dev/tty*')
    disp('  Search for /dev/ttyUSB* or /dev/ttyACM*. The port number ...
        is *.')
    disp('')
    com_port = input('Specify port (e.g. COM4 for Windows or ...
        /dev/ttyUSB0 for Linux): ','s');
    a = arduino(com_port,'Uno');
    disp(a)
end

% voltage read functions
v1 = @() readVoltage(a, 'A0');
v2 = @() readVoltage(a, 'A2');

% temperature calculations as a function of voltage for TMP36
TC = @(V) (V - 0.5)*100.0;          % Celsius
TK = @(V) TC(V) + 273.15;           % Kelvin
TF = @(V) TK(V) * 9.0/5.0 - 459.67; % Fahrenhiet

% temperature read functions
T1C = @() TC(v1());
T2C = @() TC(v2());

% LED function (0 ≤ level ≤ 1)
led = @(level) writePWMDutyCycle(a,'D9',max(0,min(1,level)));  % ON

% heater output (0 ≤ heater ≤ 100)
% limit to 0-0.9 (0-100%)
h1 = @(level) writePWMDutyCycle(a,'D3',max(0,min(100,level))*0.9/100);
% limit to 0-0.5 (0-100%)
h2 = @(level) writePWMDutyCycle(a,'D5',max(0,min(100,level))*0.5/100);
```

1.5 Temperature Control Lab Model

To obtain the data from TC lab, we give it a step response and plotted the temperature 1 and temperature 2 result on a graph. The Matlab code for step response of TC Lab is given below [1].

```
close all; clear all; clc
% include tclab.m for initialization
tclab;

disp('Test Heater 1')
```

```
6  disp('LED Indicates Temperature')
7
8  figure(1)
9  t1s = [];
10 t2s = [];
11 h1s = [];
12 h2s = [];
13 % initial heater values
14 ht1 = 0;
15 ht2 = 0;
16 h1(ht1);
17 h2(ht2);
18
19 for i = 1:600
20     tic;
21     if i==1
22         disp('Turn on heater 1 & Turn off heater 2')
23         ht1 = 80;
24         h1(ht1);
25         ht2 = 0;
26         h2(ht2);
27     end
28     if i==300
29         disp('Turn off heater 1 & Turn on heater 2')
30         ht1 = 0;
31         h1(ht1);
32         ht2 = 80;
33         h2(ht2);
34     end
35     % read temperatures
36     t1 = T1C();
37     t2 = T2C();
38
39     % LED brightness
40     brightness1 = (t1 - 30)/50.0;  % <30degC off, >100degC full ...
           brightness
41     brightness2 = (t2 - 30)/50.0;  % <30degC off, >100degC full ...
           brightness
42     brightness = max(brightness1,brightness2);
43     brightness = max(0,min(1,brightness)); % limit 0-1
44     led(brightness);
45
46     % plot heater and temperature data
47     h1s = [h1s,ht1];
48     h2s = [h2s,ht2];
49     t1s = [t1s,t1];
50     t2s = [t2s,t2];
```

```
51      n = length(t1s);
52      time = linspace(0,n+1,n);
53      clf
54      subplot(2,1,1)
55      plot(time,t1s,'r.','MarkerSize',10);
56      hold on
57      plot(time,t2s,'b.','MarkerSize',10);
58      ylabel('Temperature (degC)')
59      legend('Temperature 1','Temperature 2','Location','NorthWest')
60      subplot(2,1,2)
61      plot(time,h1s,'r-','LineWidth',2);
62      hold on
63      plot(time,h2s,'b--','LineWidth',2);
64      ylabel('Heater (0-100%)')
65      xlabel('Time (sec)')
66      legend('Heater 1','Heater 2','Location','NorthWest')
67      drawnow;
68      t = toc;
69      pause(max(0.01,1.0-t))
70  end
71
72  disp('Turn off heaters')
73  h1(0);
74  h2(0);
75
76  disp('Heater Test Complete')
```

2 Project tasks - II

2.1 Linear case - Open-Loop

a) Linearize the model equations around an operation point, e.g. the room temperature,to derive the linearized process model.
Linearized System of Elergy balance equation at 27 degree celsius
A = [-0.007 0.002 0 0; 0.002 -0.007 0 0; 0.050 0 -0.050 0; 0 0.050 0 -0.050];
B = [0.005 0; 0 0.0038; 0 0; 0 0];
C = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
D = zeros(4,2);

b) Use MATLAB, Simulink or Python for numerical solution of the process model!

Energy Balance Model Step Response

These are the result obtained from the Energy balance model step response. Here, we give 80 % to Heater 2 for first 300 second, and then we give 80 % to Heater 1 for rest 300 seconds.

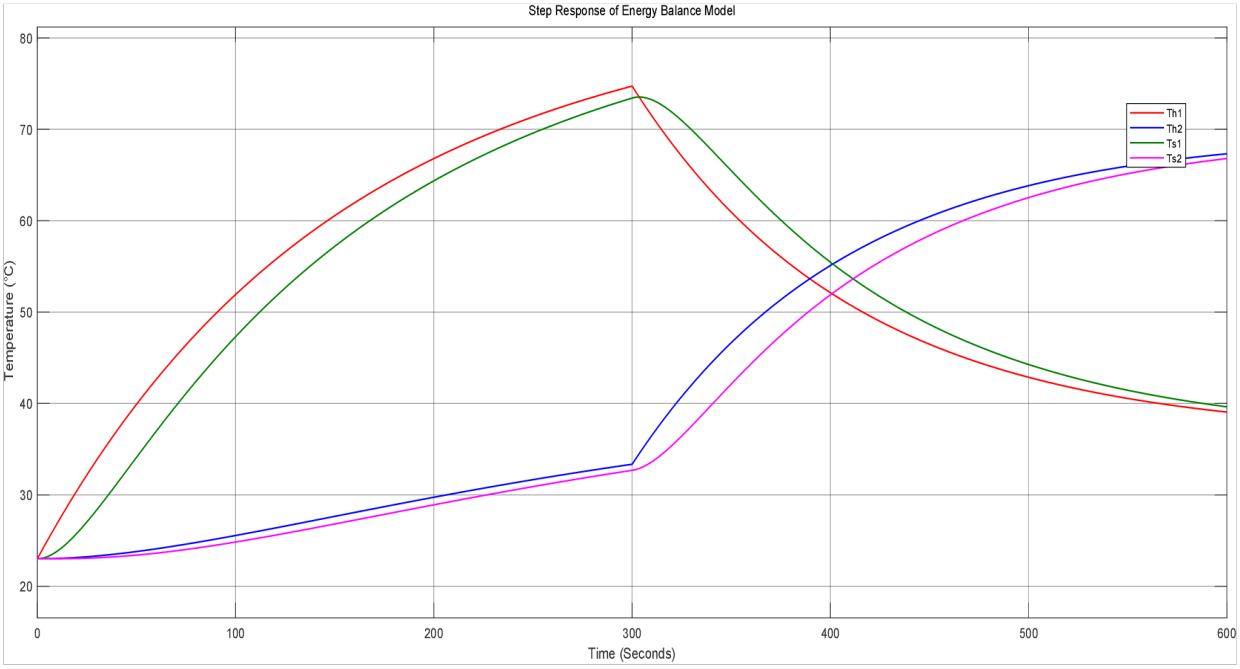

Figure 2.1: Energy balance Model Step response

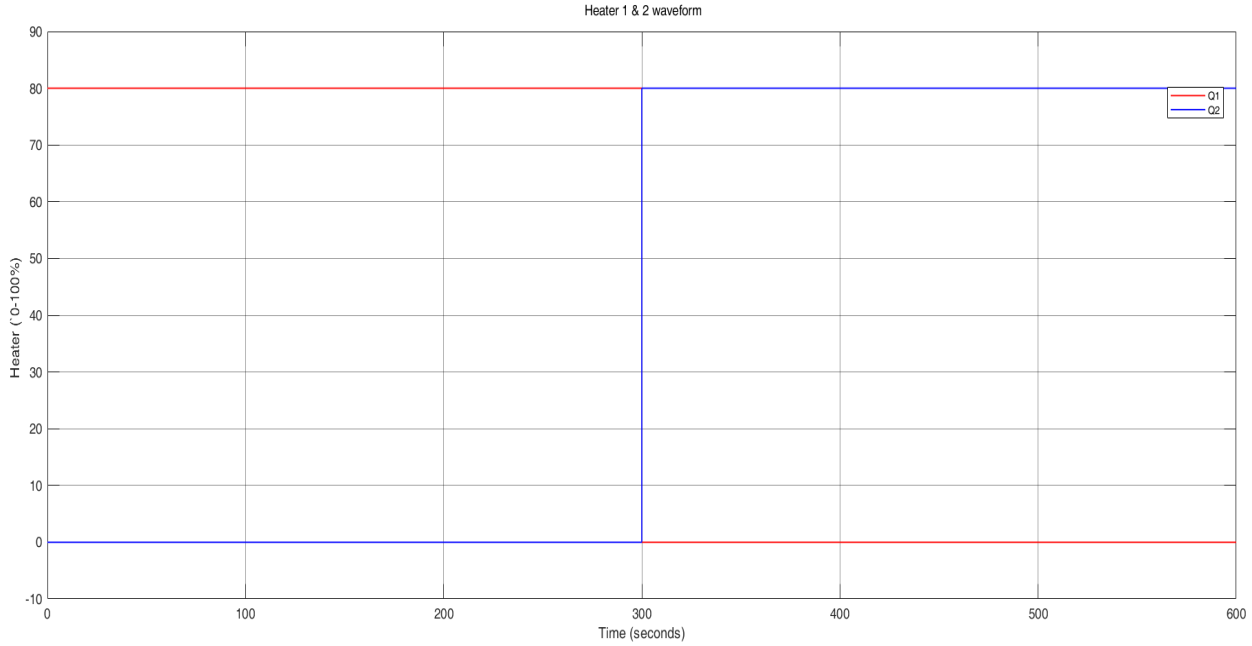

Figure 2.2: Heater waveform as step input

TC Lab step response

These are the result obtained from the TC Lab step response.

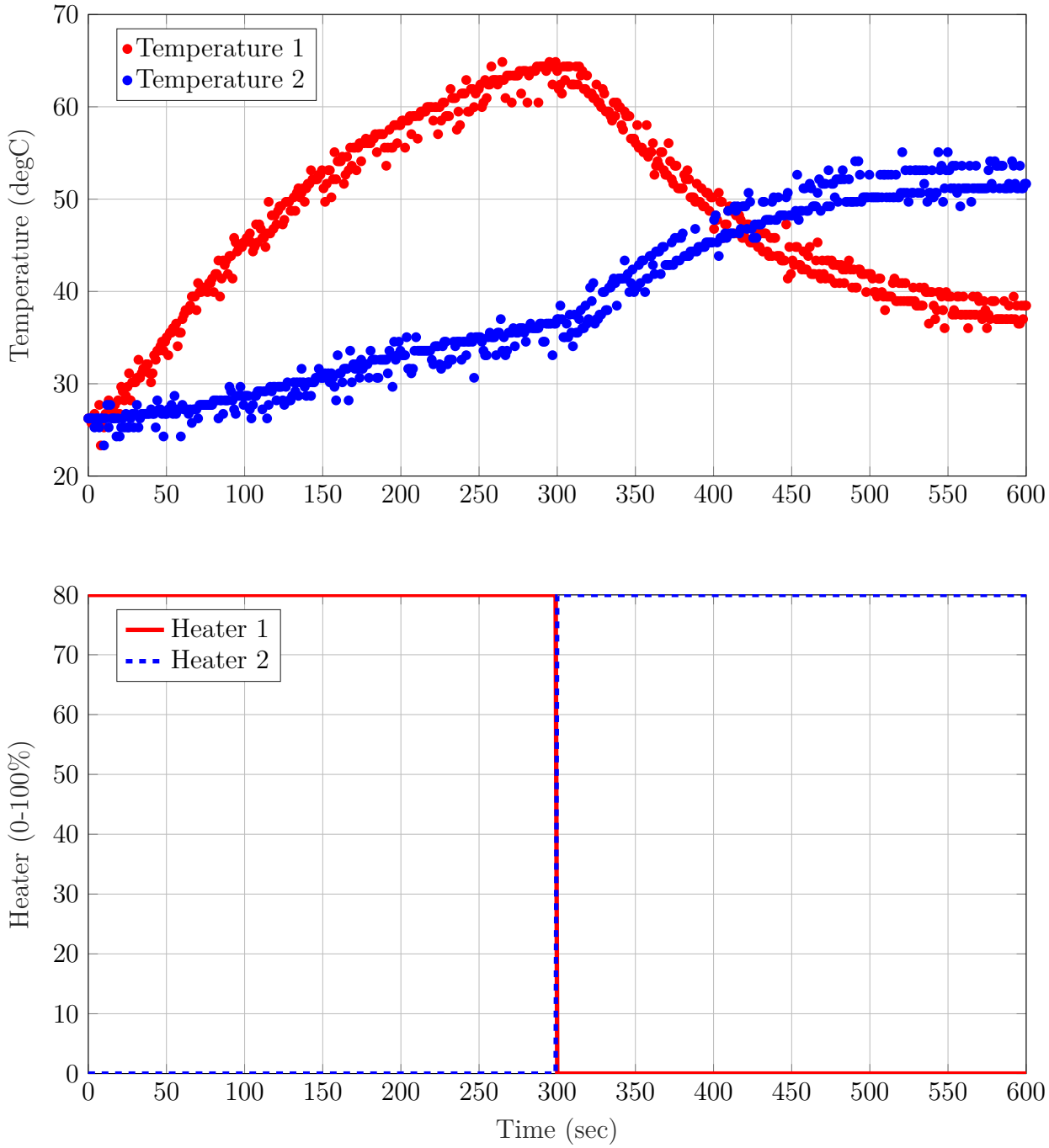

Figure 2.3: TC Lab Step response

c) Compare the simulations to the real process output for different input signals (e.g. step response and more complex series of steps)!

Here we applied a two pulses for Q1 and Q2 with pulse width 100 and 50 seconds. The duty cycle is 50% for both pulses. In the graph we can see the change in temperature

measured from sensor 1 & 2. we can also see the KF estimate for both sensor temperature and the wave form from the nonlinear energy balance model given the same pulsating input.

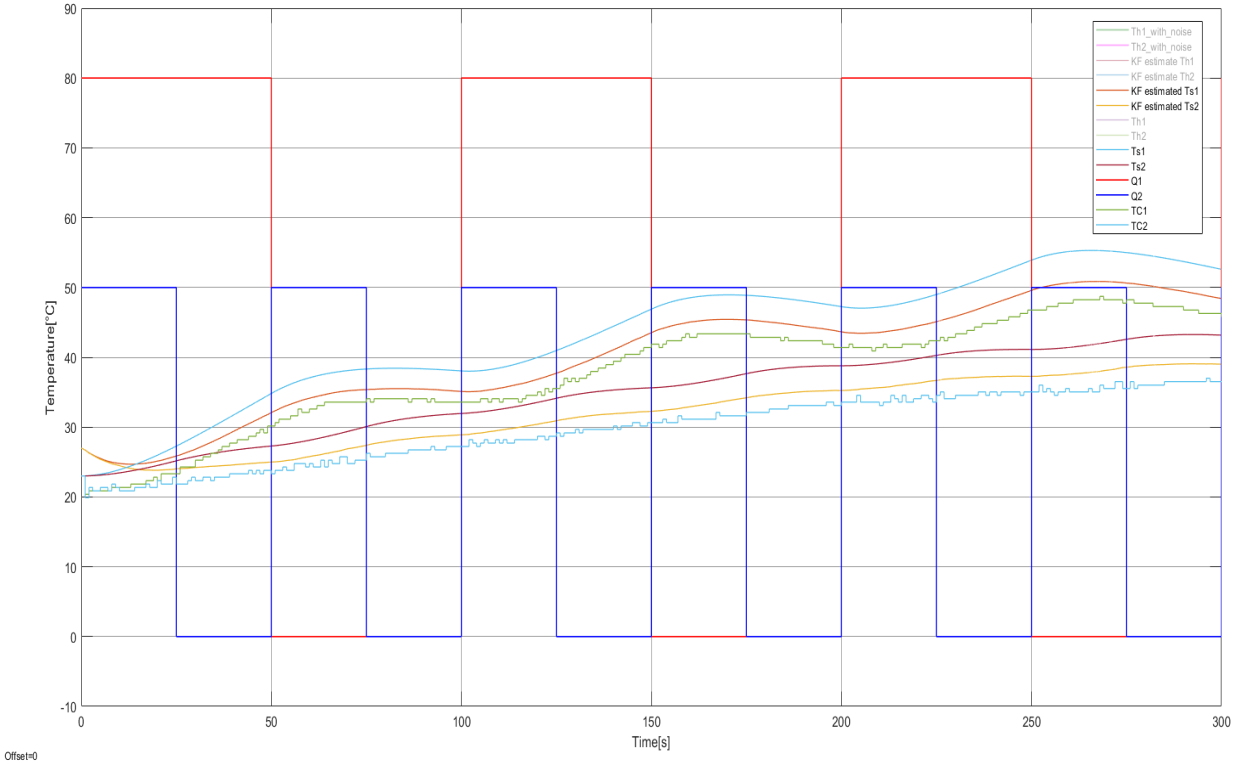

Figure 2.4: Comparison of Linearized Simulink output vs real process model output vs KF estimate

d) Compare simulations and process for changes in the model parameters (+- 30 Percent)

We changed As from $2/100^2$ Area in m^2 to $2.66/100^2$ that is 30 % increase and to $1.33/100^2$ that is 30% decrease. The graphs are given below.

e) Use the linearization based Kalman-Filter provided in the exercises and repeat the previous two tasks

We implemented a Linear Kalman Filter with Process noise (Q) = 10 and Measurement noise (R) = 10/50. We took the initial states to be 27 degree Celsius each. Here we only have two state measurement which are Ts1 (Temperature sensor 1) and Ts2 (Temperature sensor 2). So in order to have the four states, we gave a constant value of 0 to two other states [2].

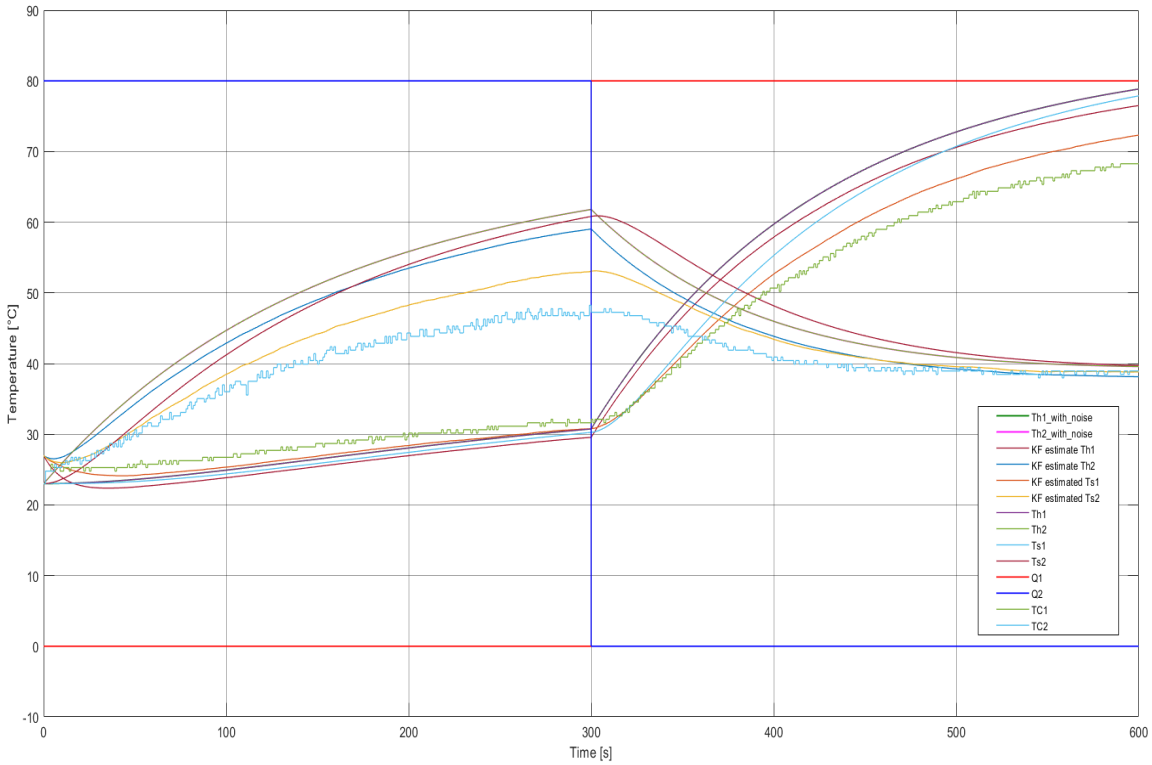

Figure 2.5: Comparison of Linearized Simulink output vs real process model output vs KF estimate

We can see that the result from the energy balance model has slightly higher value when compared to the actual values from the TC lab. This can be due to the inappropriate parameters used in the energy balance model.

Now we will change the model parameters by 30 %. In the graphs below, first we incremented the process model parameter by 30 % and then in the next graph we decreased it by 30 %.

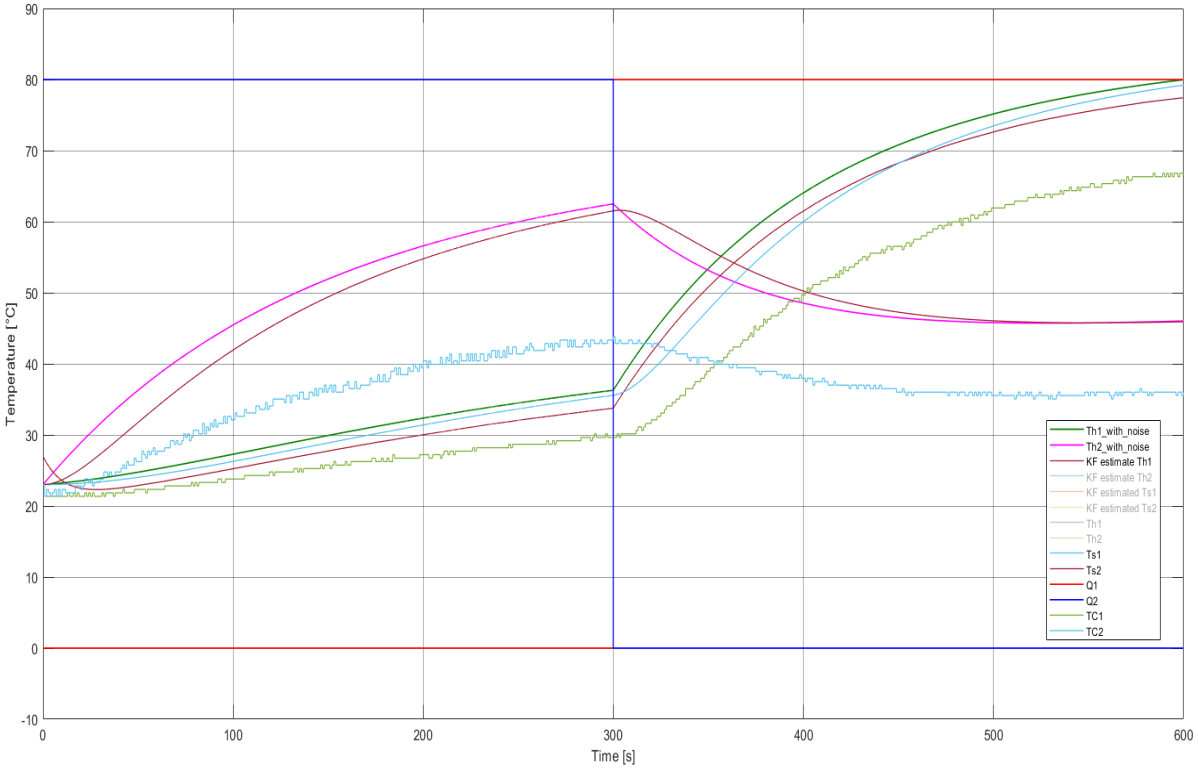

Figure 2.6: Comparison of Linearized Simulink output vs real process model output vs KF estimate with 30 percent increment in model parameter

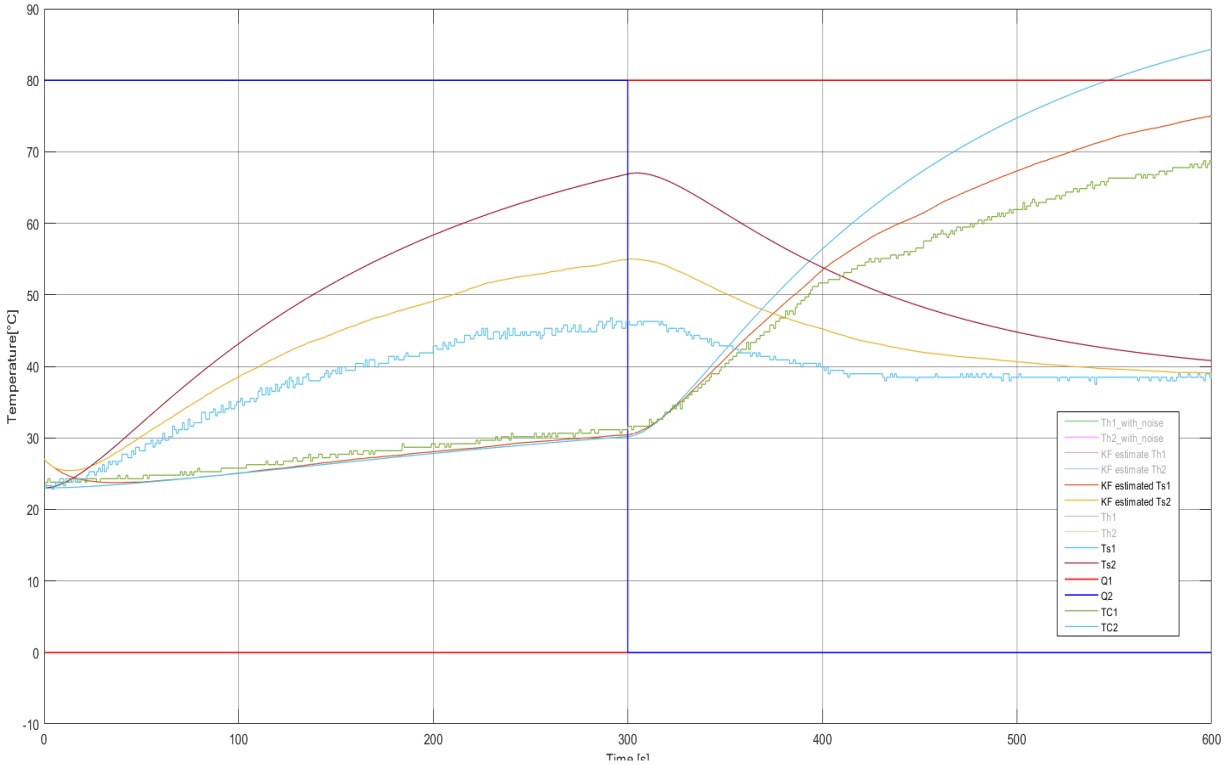

Figure 2.7: Comparison of Linearized Simulink output vs real process model output vs KF estimate with 30 percent decrement in model parameter

As we can see from the above graph that the linearized process model has slightly different values from the real process model because of the different initial conditions and also because of process noise and measurement noise in the real process.

Matlab's code for the TC Control Lab to read temperature sensor data [1]

```
function TC = arduino_tclab(heater)

persistent icount a

if (isempty(icount))
    % include tclab.m for initialization
    tclab;
    icount = 0;
else
    % voltage read functions
    v1 = @() readVoltage(a, 'A0');
    v2 = @() readVoltage(a, 'A2');

    % temperature calculations as a function of voltage for TMP36
    TC = @(V) (V - 0.5)*100.0;          % Celsius
    TK = @(V) TC(V) + 273.15;           % Kelvin
    TF = @(V) TK(V) * 9.0/5.0 - 459.67; % Fahrenhiet

    % temperature read functions
    T1C = @() TC(v1());
    T2C = @() TC(v2());

    % LED function (0 ≤ level ≤ 1)
    led = @(level) writePWMDutyCycle(a,'D9',max(0,min(1,level)));  % ON

    % heater output (0 ≤ heater ≤ 100)
    % limit to 0-0.9 (0-100%)
    h1 = @(level) ...
        writePWMDutyCycle(a,'D3',max(0,min(100,level))*0.9/100);
    % limit to 0-0.5 (0-100%)
    h2 = @(level) ...
        writePWMDutyCycle(a,'D5',max(0,min(100,level))*0.5/100);
end

% increment counter
icount = icount + 1;

% read temperature
```

```
37 TC1 = T1C();
38 TC2 = T2C();
39 TC = [TC1,TC2];
40
41 % write heater levels
42 h1(heater(1));
43 h2(heater(2));
44
45 % indicate high temperature with LED
46 if (TC1 || TC2) > 40
47     led(1)
48 else
49     led(0)
50 end
```

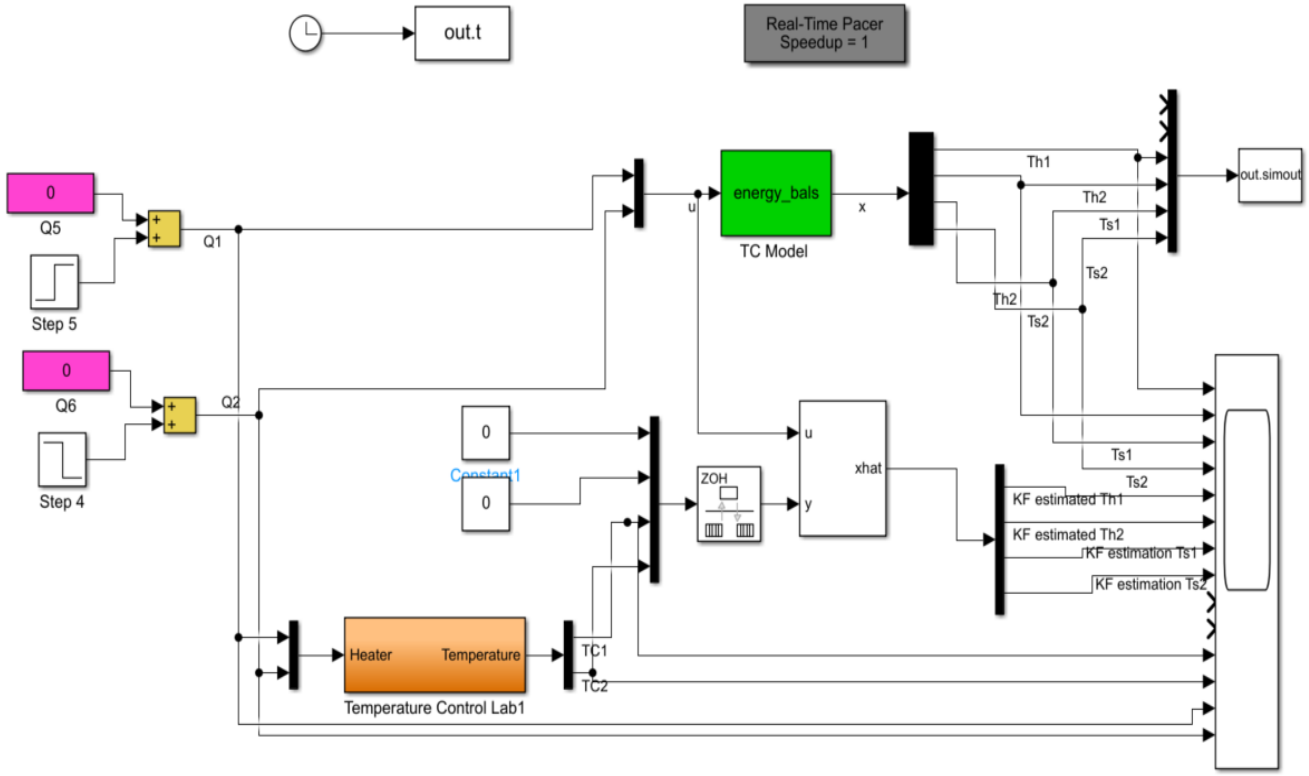

Figure 2.8: Process model and measurement from sensors in Simulink

2.2 Linear case - Closed-Loop

a) Design and implement an integral state-feedback controller using the KF-reconstructions!

We implemented a PI controller with P= 10 I = 10/50, D = 0. we also applied Kalman filter with initial condition to be [27 27 27 27], and linearized model of energy balance model with C = diag([0 0 1 1]) as we only got two temperature values from real process measurement. We initiated process noise (Q) = 10 and Measurement noise (R) = 10/50 in KF block in Simulink. We gave the reference value for temperature sensor 1 (Ts1) equals to 40 and Ts2 = 50. We tuned the PID controller such that we can decrease the steady state error to the minimum, and we can reach the state in optimal time. Hence, we used PI controller. We also tuned the Kalman filter such that we can remove the measurement and process noise error as much as possible without sacrificing the steady state error. We are able to use PID controller because the output and the state were the same in our case and the model was linearized also.

In the figure (2.10) we can see that we have implemented Integral state feedback controller with KF reconstruction. The control action upper and lower limits are set to be 80°C and 30°C as the room temperature is 25°C and we didn't want to damage the sensors by going above 80. Here we use LQR command to calculate Controller gain matrix which is then multiplied with estimated state from Kalman filter. The main idea in LQR (Linear Quadratic Regulator) control design is to minimize the quadratic cost function of $int(x^TQx + u^TRu)dt$. It turns out that regardless of the values of Q and R, the cost function has a unique minimum that can be obtained by solving the Algebraic Riccati Equation. The parameters Q and R can be used as design parameters to penalize the state variables and the control signals. The larger these values are, the more we penalize these signals. Basically, choosing a large value for R means we try to stabilize the system with less (weighted) energy. This is usually called expensive control strategy. On the other hand, choosing a small value for R means we don't want to penalize the control signal (cheap control strategy). Similarly, if we choose a large value for Q means we try to stabilize the system with the least possible changes in the states and large Q implies less concern about the changes in the states. Since there is a trade-off between the two, we may want to keep Q as I (identity matrix) and only alter R. One can choose a large R, if there is a limit on the control output signal (for instance, if large control signals introduce sensor noise or cause actuator's saturation) , and choose a small R if having a large control signal is not a problem for the system.

```
% LQR command to calculate Controller gain matrix
>> K = lqr([-0.007 0.002 0 0; 0.002 -0.007 0 0; 0.050 0 -0.050 0; 0 ...
    0.050 0 -0.050],[0.005 0; 0 0.0038; 0 0; 0 0],.07,1,0)
K =
    0.0501    0.0147    0.0031    0.0001
    0.0112    0.0384    0.0001    0.0023
```

We also took two reference for Temperature sensor 1 (Ts1) and Temperature sensor 2 (Ts2) which are 60°C and 50°C respectively. These references are then added with negative feedback of output from sensors. The resultant is then integrated and added with estimated state from Kalman filter. This final result then matrix multiplicated with the 'K' i.e controller gain matrix to yield the final input result to be fed to sensors hence achieving reference tracking.

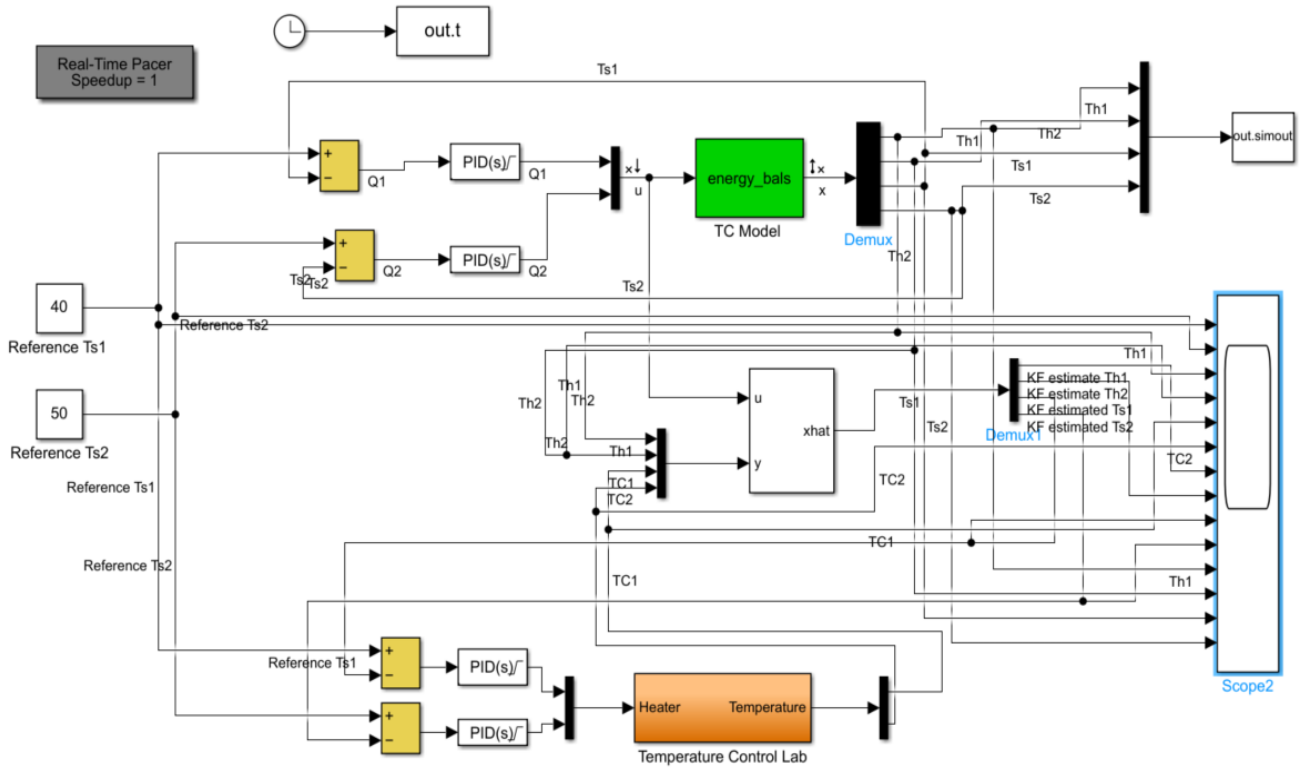

Figure 2.9: Linear Closed loop Simulink Model with Kalman filter

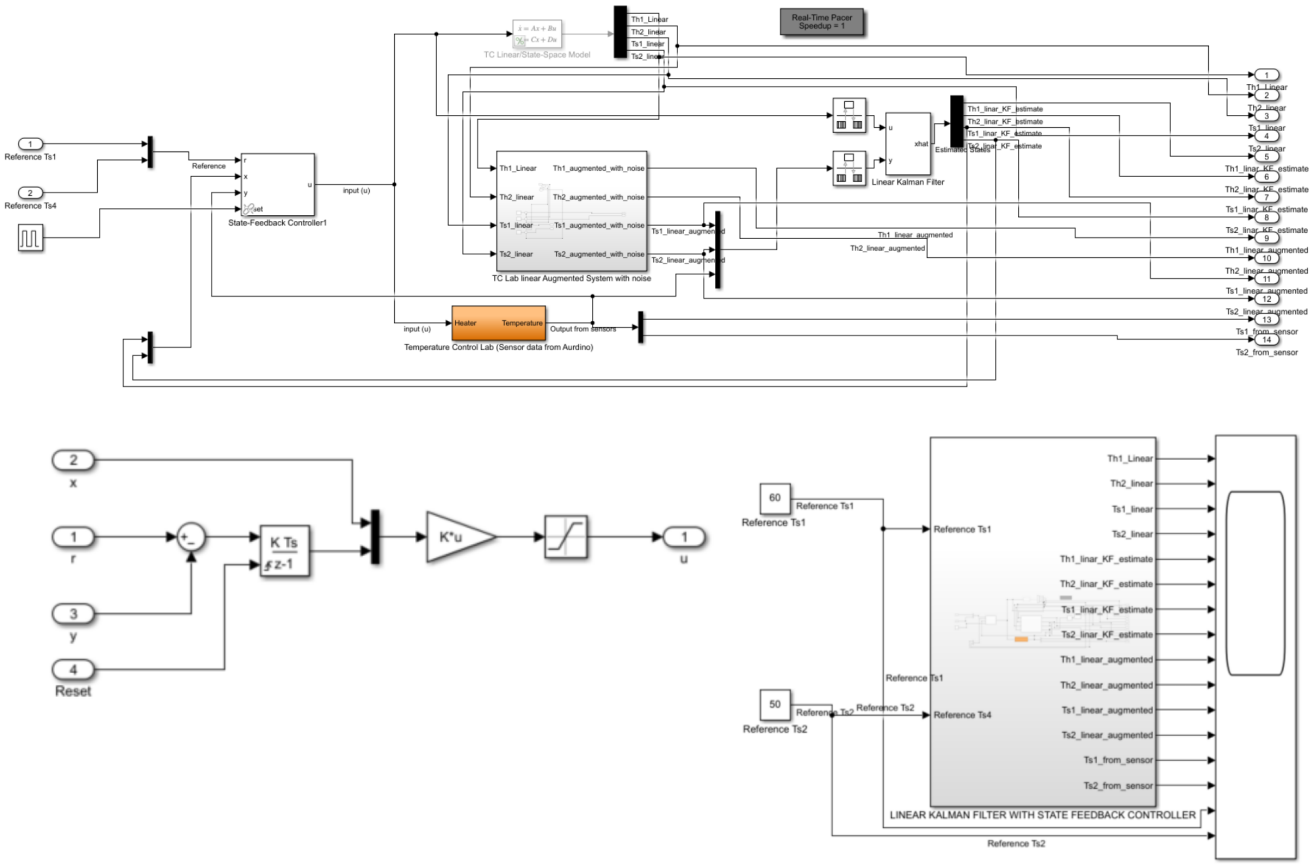

Figure 2.10: Integral state feedback controller Simulink Model

Integral action for set-point tracking

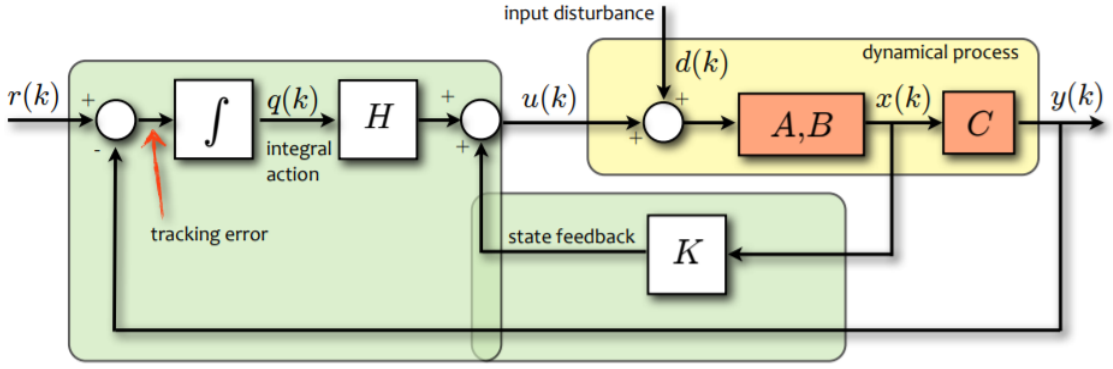

Figure 2.11: Integral state feedback controller block diagram

b) Evaluate the overall closed-loop performance for changes in the model parameters (+-30 %)!

As we can see below in the graphs, when we increased and decreased the model parameters (As) by 30 %, it shifted linearized model states towards and away from the measured states respectively. Finally, we came to the conclusion that "As" equal to $2/100^2 m^2$ yields the best result.

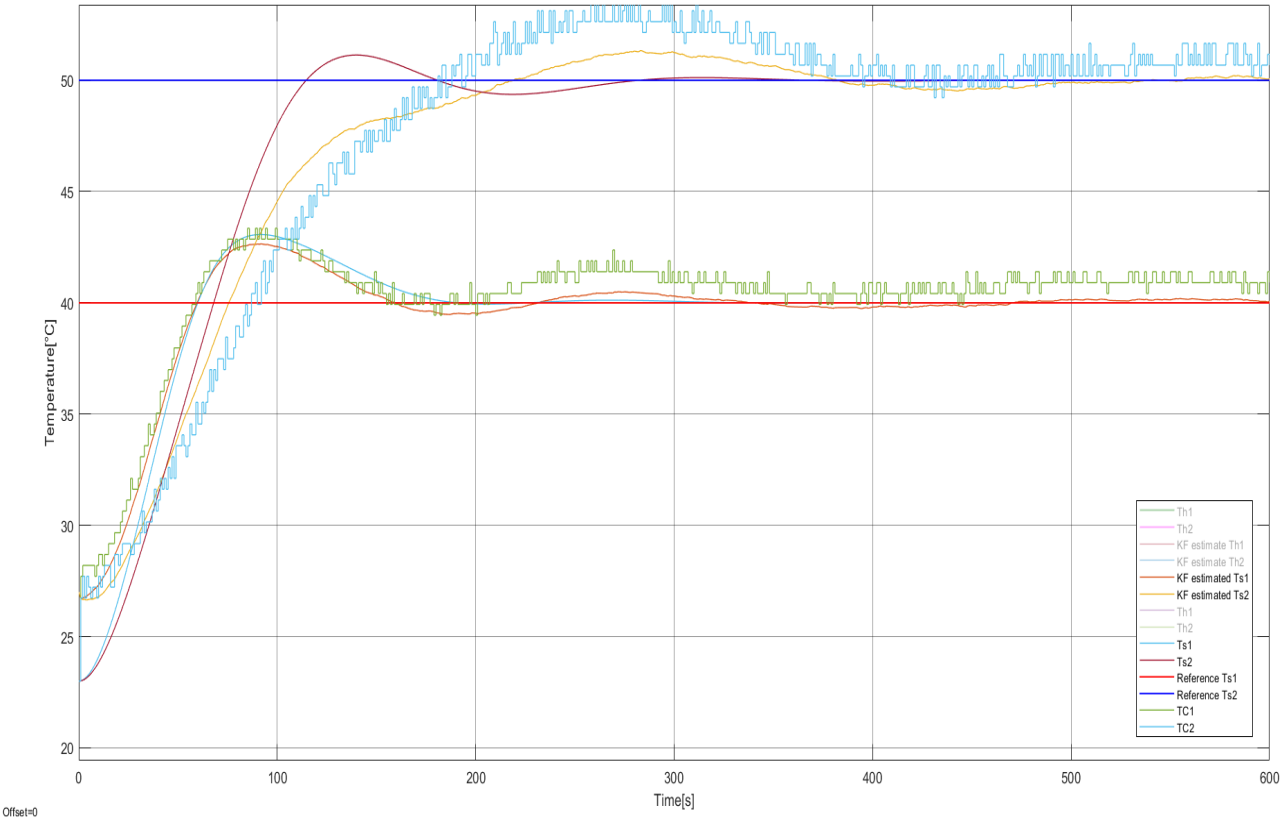

Figure 2.12: Comparison of Estimated state vs Measured state vs Prior State using Linear Kalman Filter in Closed loop linear model

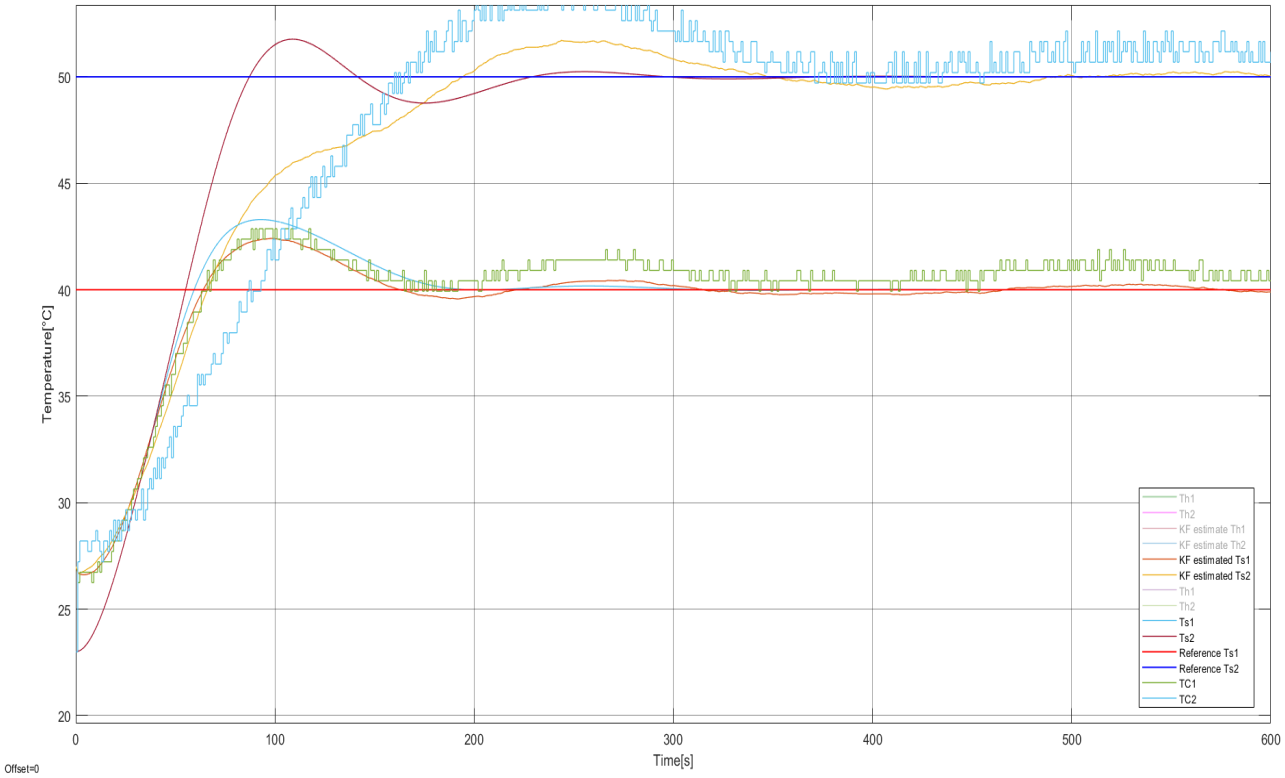

Figure 2.13: Comparison of Estimated state vs Measured state vs Prior State using Linear Kalman Filter in Closed loop linear model when As is increased 30 percent

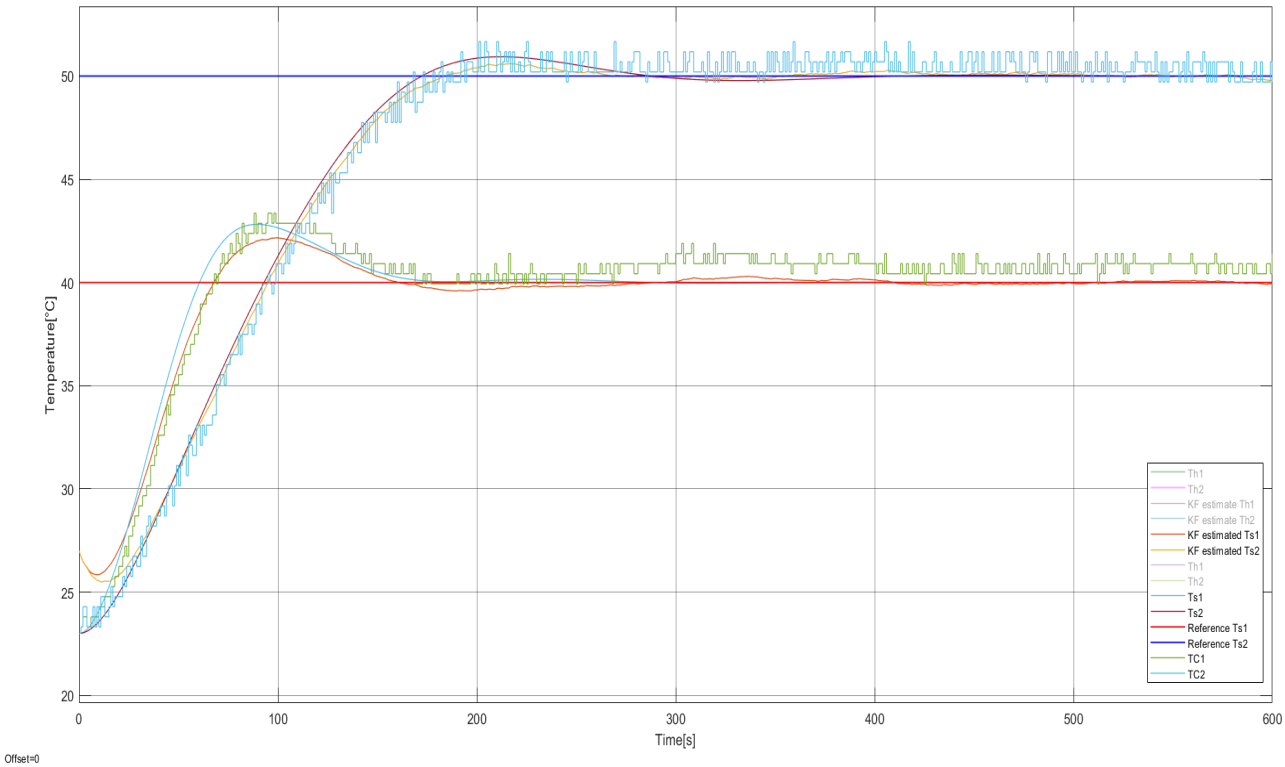

Figure 2.14: Comparison of Estimated state vs Measured state vs Prior State using Linear Kalman Filter in Closed loop linear model when As is decreased 30 percent

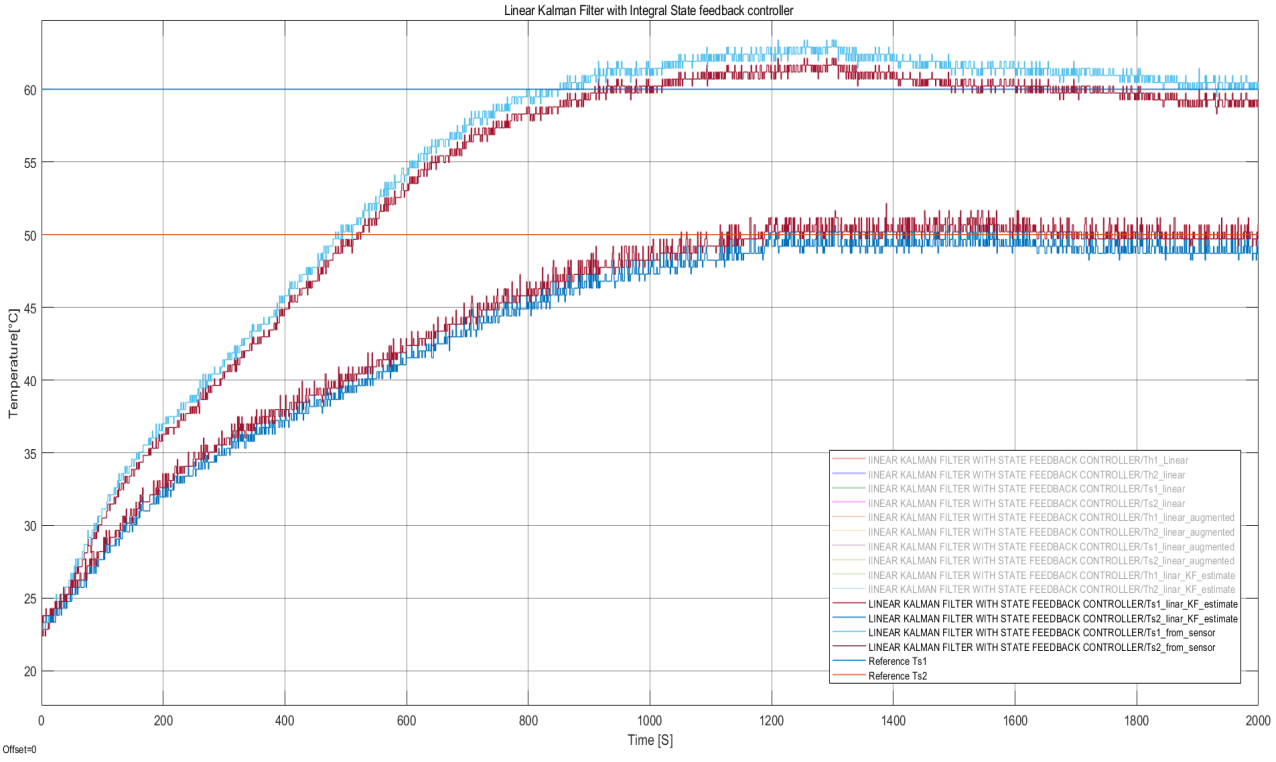

Figure 2.15: Linear Kalman Filter with Integral State feedback controller with Q=.3 & R=1 and N=0

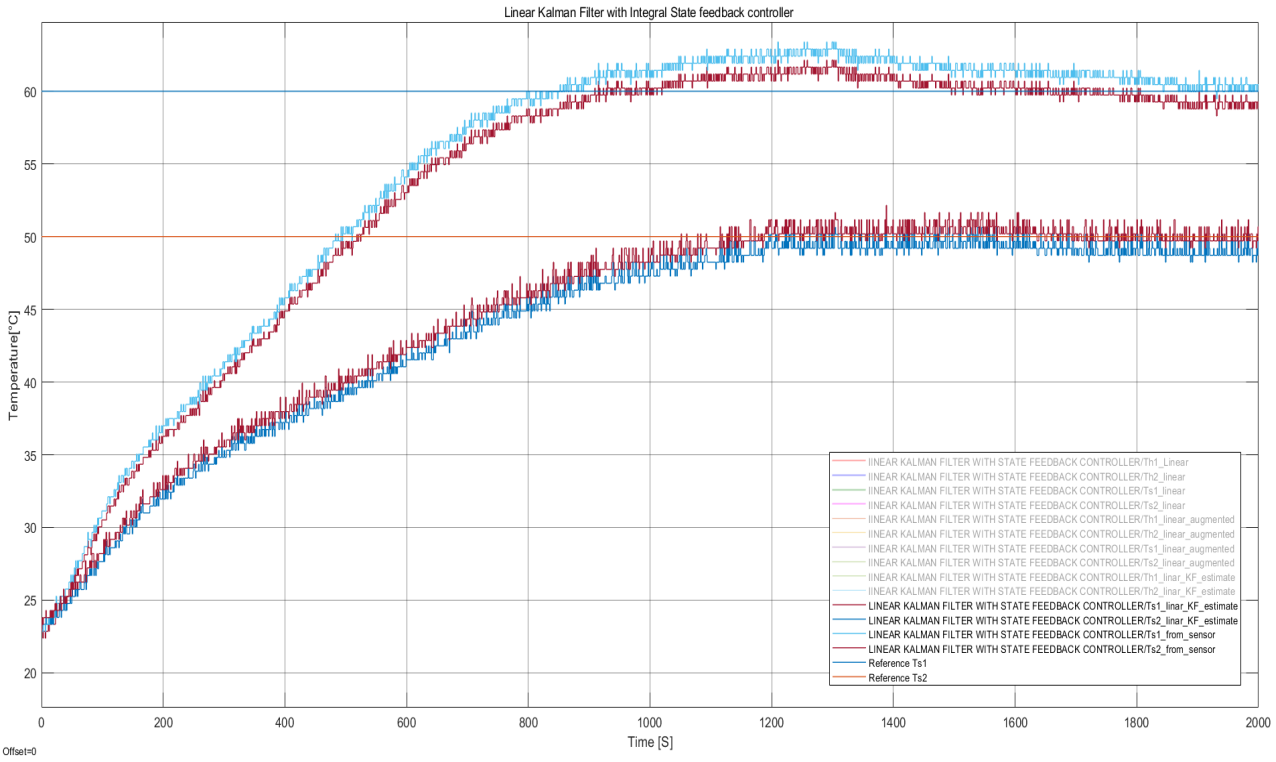

Figure 2.16: Linear Kalman Filter with Integral State feedback controller with Q=.1 & R=1 and N=0 (30% decrement)

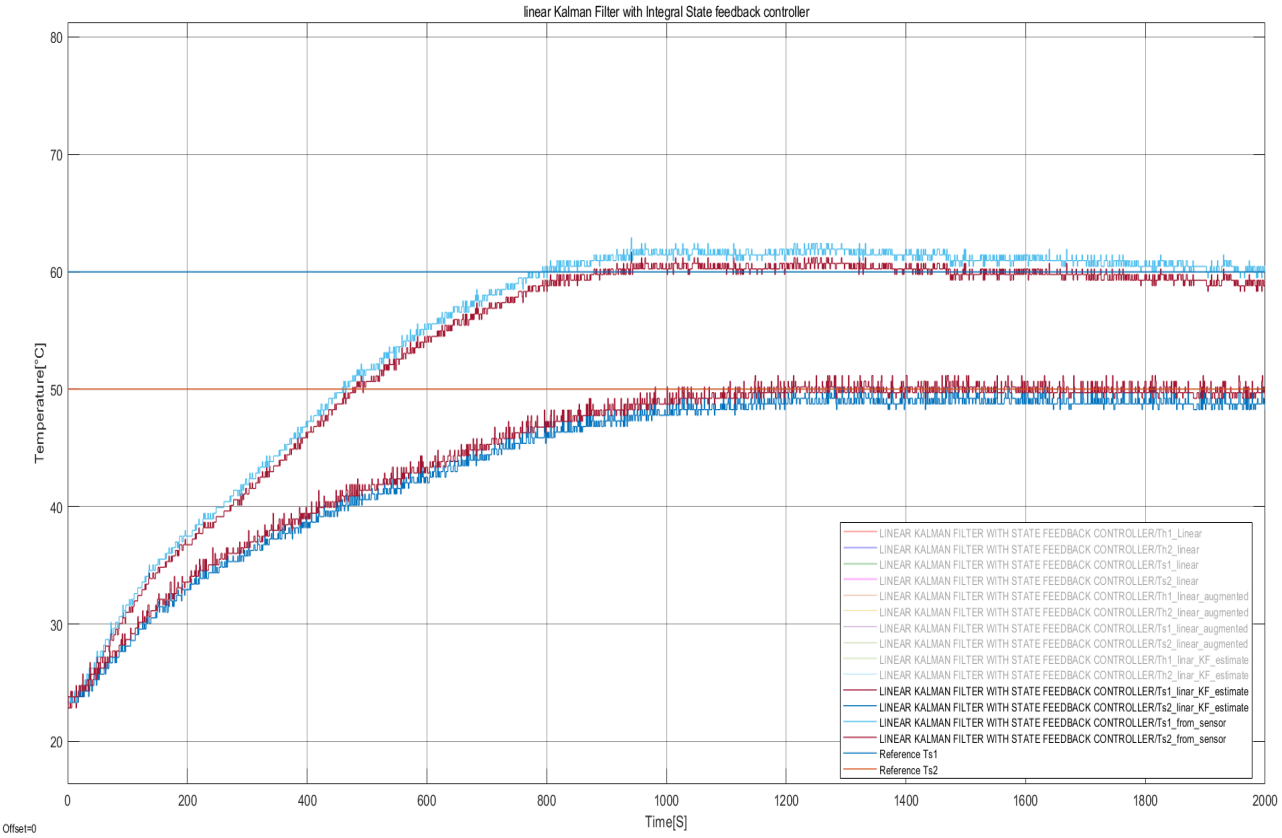

Figure 2.17: linear Kalman Filter with Integral State feedback controller with Q=.7 & R=1 and N=0 (30% increment)

2.3 Nonlinear case

a) Under the assumption, that Heater 2 can be neglected, show that the nonlinear model for Heater 1 is locally weak observable!

As soon as we neglect Heater 2 which can be achieved by putting an insulation sheet between the two heater-sensor pair, we stop the flow of heat from conduction and radiation from one heater to the other which in turn don't allow Heater 1 to go to all the states (especially the higher temperature states) as to reach higher temperature states the Heater 1 relies on it's 100 % with the conduction and radiated heat from heater 2. Hence, this causes the heater 1 to reach only a certain amount of temperature and hence it is locally weak observable.

b) Design, implement and evaluate an Extended Kalman Filter for the system with two heaters!

We made the Extended Kalman filter by using Process noise mean equal to 0 and covariance matrix = diag([1e-1 1e-1 1e-2 1e-2]) while the process noise mean equal to 0 and covariance matrix diag([1e-4 1e-4 5e-5 5e-5]). we took the noise as additive noise. we added measurement noise to output of heater temperature 1 and 2 and gave the signal along with noise to extended Kalman filter alongside the two measurement of sensor 1 & 2. We also made two MATLAB function. One is state transition function, which contained energy balance model equations and differential equations of state and model parameters. We also made a measurement function y=x as our outputs are also the states.

Matlab Code for State transition function for EKF

```
1  unction dTdt = myStateTransitionFcn(x,u)
2
3  %     global Ta k m Cp A As alpha1 alpha2 eps sigma tao ks
4
5      % Parameters
6      Ta = 27 ;            %  C
7      k = 10.0;            % W/m^2- C
8      m = 4.0/1000.0;      % kg
9      Cp = 0.5 * 1000.0;   % J/kg- C
10     A = 10.0 / 100.0^2; % Area in m^2
11     As = 2.0 / 100.0^2; % Area in m^2
12     alpha1 = 0.0100;     % W / % heater 1
13     alpha2 = 0.0050;     % W / % heater 2
14     eps = 0.9;           % Emissivity
```

```
15      sigma = 5.67e-8;     % Stefan-Boltzman
16      tao = 20;            % second
17      ks = 20;             % W/m^2- C
18
19      % Nonlinear Energy Balances
20      dTh1dt = (1.0/(m*Cp))*(k*A*(Ta-x(1)) + eps * sigma * A * (Ta^4 - ...
            x(1)^4) + (ks*As*(x(2)-x(1))) + (eps*sigma*As * (x(2)^4 - ...
            x(1)^4) + alpha1*u(1)));
21      dTh2dt = (1.0/(m*Cp))*(k*A*(Ta-x(2)) + eps * sigma * A * (Ta^4 - ...
            x(1)^4) - (ks*As*(x(2)-x(1))) - (eps*sigma*As * (x(2)^4 - ...
            x(1)^4) + alpha2*u(2)));
22      dTs1dt = (1.0/tao)*(x(1)-x(3));
23      dTs2dt = (1.0/tao)*(x(1)-x(4));
24
25      % Output
26      dTdt = [dTh1dt,dTh2dt,dTs1dt,dTs2dt]';
27  end
```

Matlab Code for Measurement function for EKF

```
1  % Measurement function for EKF
2  function y = myMeasurementFcn(dTdt)
3  y = dTdt;
4  end
```

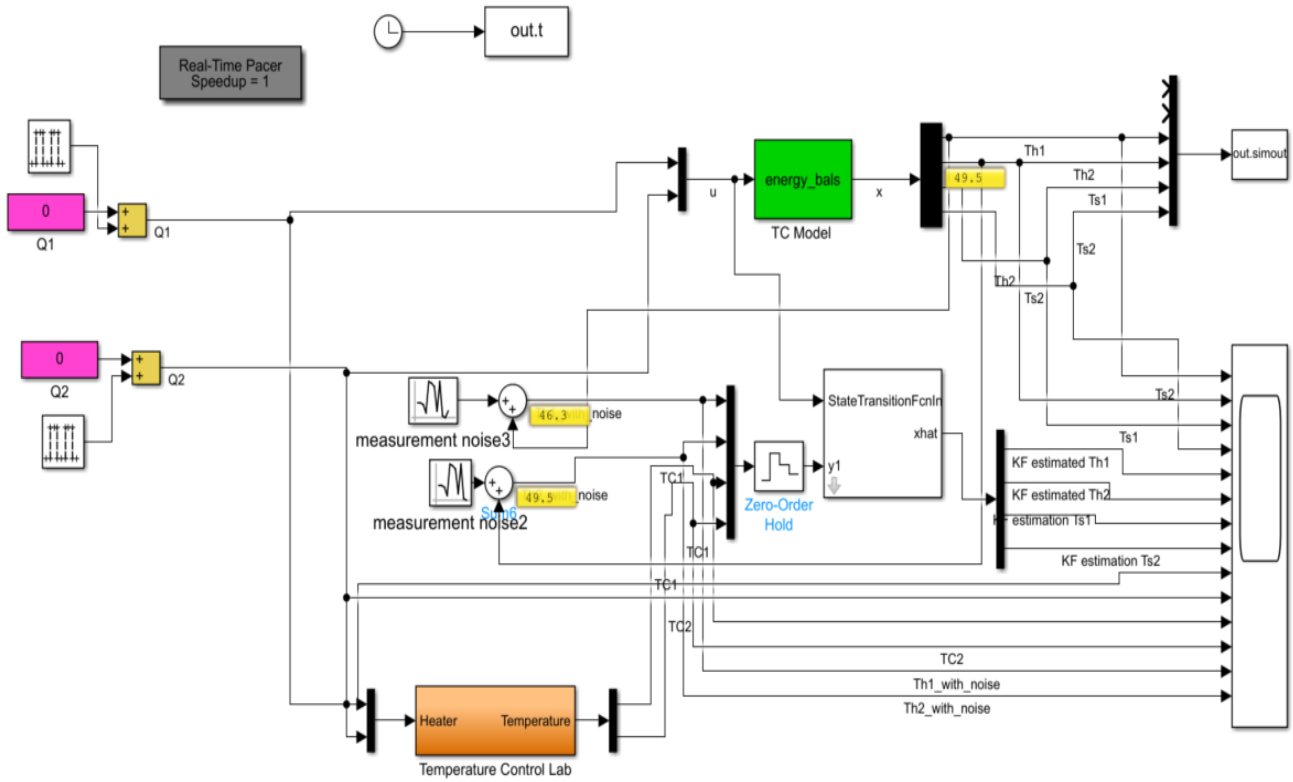

Figure 2.18: Non-Linear Extended Kalman Filter Simulink model

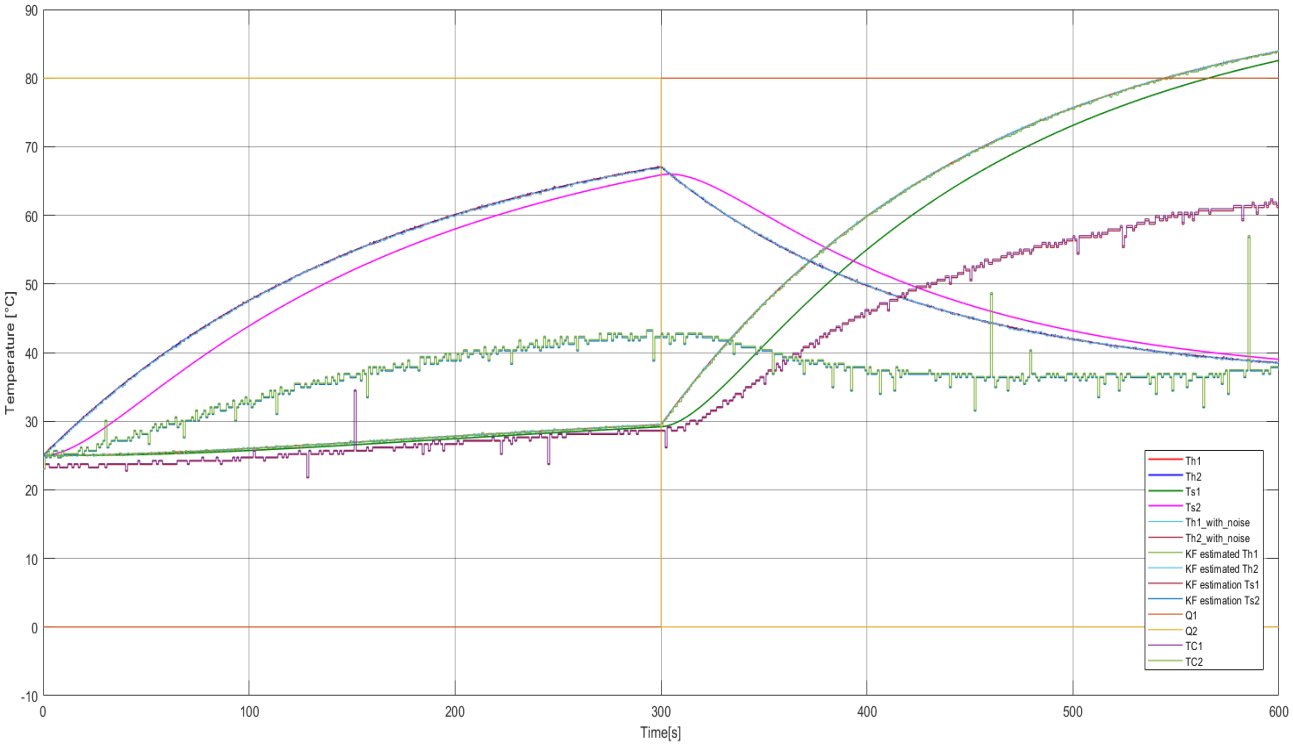

Figure 2.19: Non-Linear Extended Kalman Filter state estimation vs noisy measurement

c) Evaluate the performance of the EKF for changes in the noise matrices and the model parameters!.

In the graph below, we changed the model parameter "As" from $2/100^2 m^2$ to $1/100^2 m^2$ and the measurement noise matrix to diag([1e-4 1e-4 5e-6 5e-6]).

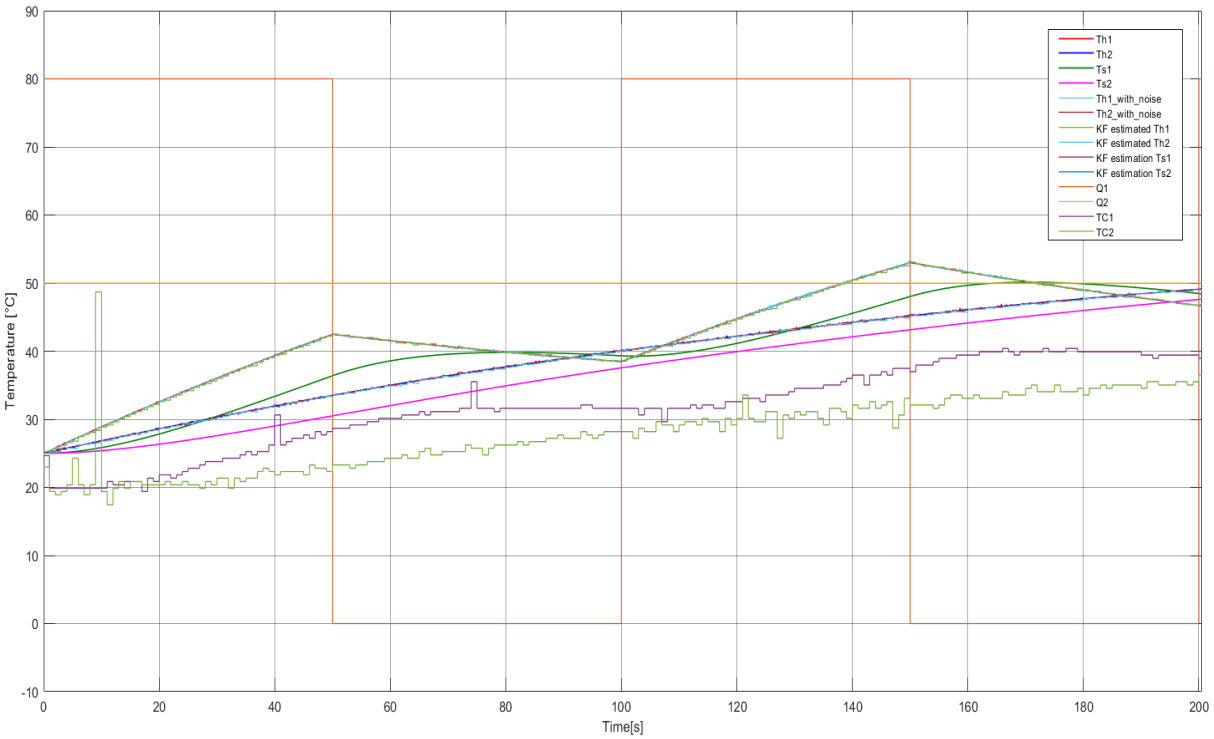

Figure 2.20: Non-Linear Extended Kalman Filter Model parameter and noise increased

In the next graph, we changed the model parameter "As" from $2/100^2 m^2$ to $2.5/100^2 m^2$ and the measurement noise matrix to diag([1e-4 1e-4 1e-5 1e-5]).

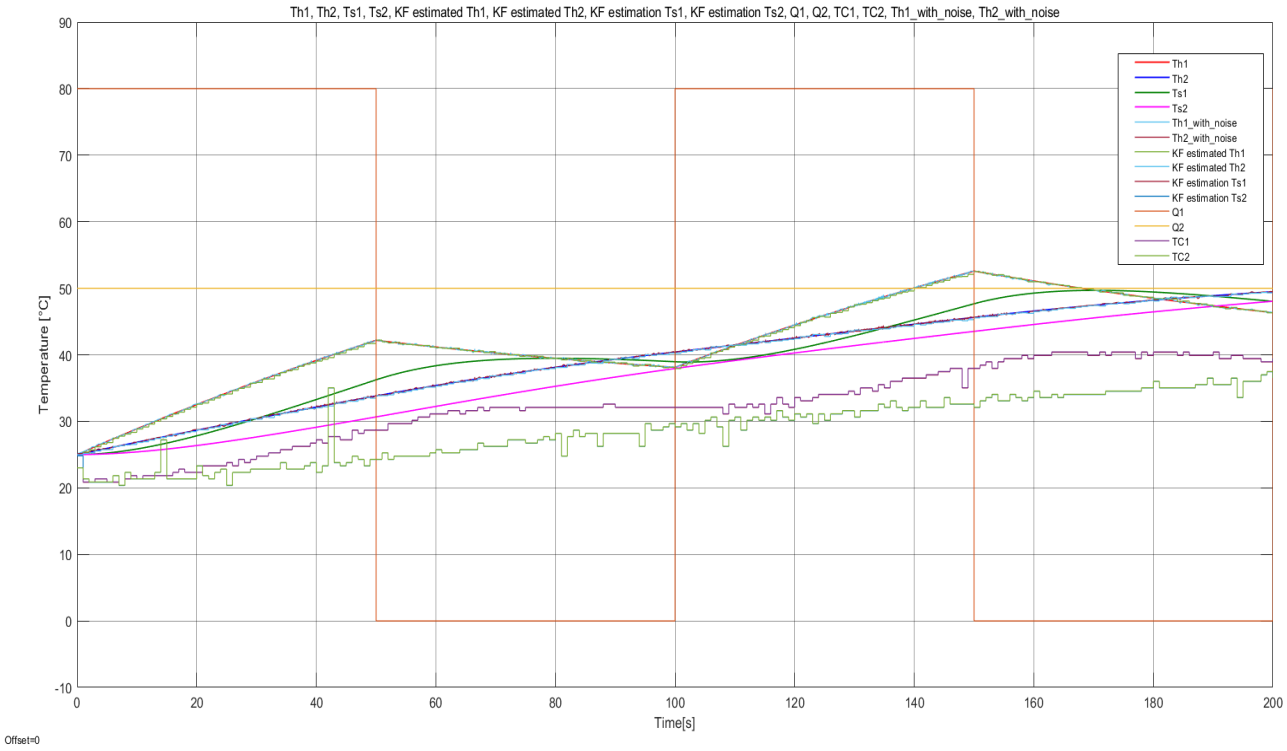

Figure 2.21: Non-Linear Extended Kalman Filter Model parameter and noise decreased

From the above graphs, we can see that the best result is when we keep the "As" small and also decreased the measurement noise.

2.4 Simultaneous state and parameter estimation

a) Augment the process model by the dynamics of the two unknown parameters K and τ .

We augmented the process model with measurement noise equals to $1e^{-2}$ for each state and process noise equal to $1e^{-2}$ also. We also augmented the process model with only measurement noise too. Below are the two Simulink model respectively, can be seen in figure (2.23) & (2.24).

For calculating K and τ we did the RGA analysis first and found out that Ts1 and Th1 are heavily impacted (around 80%) by Q1 and Ts2 and Th2 are heavily impacted (around 80%) by Q2. This can be seen in the figure (2.22) below. After that we did the following computation.

Obtain the process output step response curve in the case of a unit step change in the input :

$$>> G = tf(linsys1) \tag{2.1}$$

G =.....Transfer function matrix
From input "Reference(1)" to output...

$$x(1) = \frac{0.005s + 3.501e^{-05}}{s^2 + 0.014s + 4.503e^{-05}} \tag{2.2}$$

$$x(2) = \frac{1e^{-05}}{s^2 + 0.014s + 4.503e^{-05}} \tag{2.3}$$

$$x(3) = \frac{0.00025s + 1.75e^{-06}}{s^3 + 0.064s^2 + 0.0007452s + 2.251e^{-06}} \tag{2.4}$$

$$x(4) = \frac{5.001e^{-07}}{s^3 + 0.064s^2 + 0.0007452s + 2.251e^{-06}} \tag{2.5}$$

From input "Reference(2)" to output...

$$x(1) = \frac{7.501e^{-06}}{s^2 + 0.014s + 4.503^{e-05}} \tag{2.6}$$

$$x(2) = \frac{0.00375s + 2.626e^{-05}}{s^2 + 0.014s + 4.503e^{-05}} \tag{2.7}$$

$$x(3) = \frac{3.751e^{-07}}{s^3 + 0.064s^2 + 0.0007452s + 2.251e^{-06}} \tag{2.8}$$

$$x(4) = \frac{0.0001875s + 1.313e^{-06}}{s^3 + 0.064s^2 + 0.0007452s + 2.251e^{-06}} \tag{2.9}$$

Linearization at model initial condition Continuous-time transfer function.

```
%%Store the output response data under variable , y and time vector,t .

>> [y,t] = step(G)

%%Slope at each point :

>> slope1 = gradient(y(:,3,1),t)
slope2 = gradient(y(:,4,2),t)

%%Coordinates of the point of inflection :

[tslope1,idx1] = max(slope1)

tIP1 = t(idx1)

yIP1 = y(idx1,3,1)

[tslope2,idx2] = max(slope2)

tIP2 = t(idx2)

yIP2 = y(idx2,4,2)

tslope1 = 0.0036

idx1 = 26

tIP1 = 46.0517

yIP1 = 0.1243

tslope2 = 0.0027

idx2 = 26

tIP2 = 46.0517

```

```
yIP2 = 0.0932

%%The tangent line of the step response curve at the inflection point :
>> yTangentLine1 = tslope1*(t-tIP1) + yIP1
yTangentLine2 = tslope2*(t-tIP2) + yIP2

%%Estimate parameters of the transfer function ( FOPTD) model :

%%Time delay (Dead time) , Td :

>> Td1 = tIP1-(yIP1/tslope1)
Td2 = tIP2-(yIP2/tslope2)

Td1 = 11.8713

Td2 = 11.8713

%%Time constant , Tau :

>> Tau1 = y(end,3,1)/tslope1
Tau2 = y(end,4,2)/tslope2

Tau1 = 213.6024

Tau2 = 213.6024

%%Process gain, K :

>> K1 = y(end,3,1)/1
K2 = y(end,4,2)/1

K1 = 0.7769

K2 = 0.5827

%%Approximate transfer function as first order system:

>> G12 = tf(K1,[Tau1],'InputDelay',Td1)

G12 = exp(-11.9*s) * (0.003637)

Continuous-time transfer function.

>> G21 = tf(K2,[Tau2],'InputDelay',Td2)

G21 = exp(-11.9*s) * (0.002728)
Continuous-time transfer function.
```

From the above computation we tried to approximate the Transfer function between Ts1 with Q1 and Ts2 with Q2 to a first order with time delay transfer function. And hence we got the gain(K) and time delay (τ) for both transfer function.

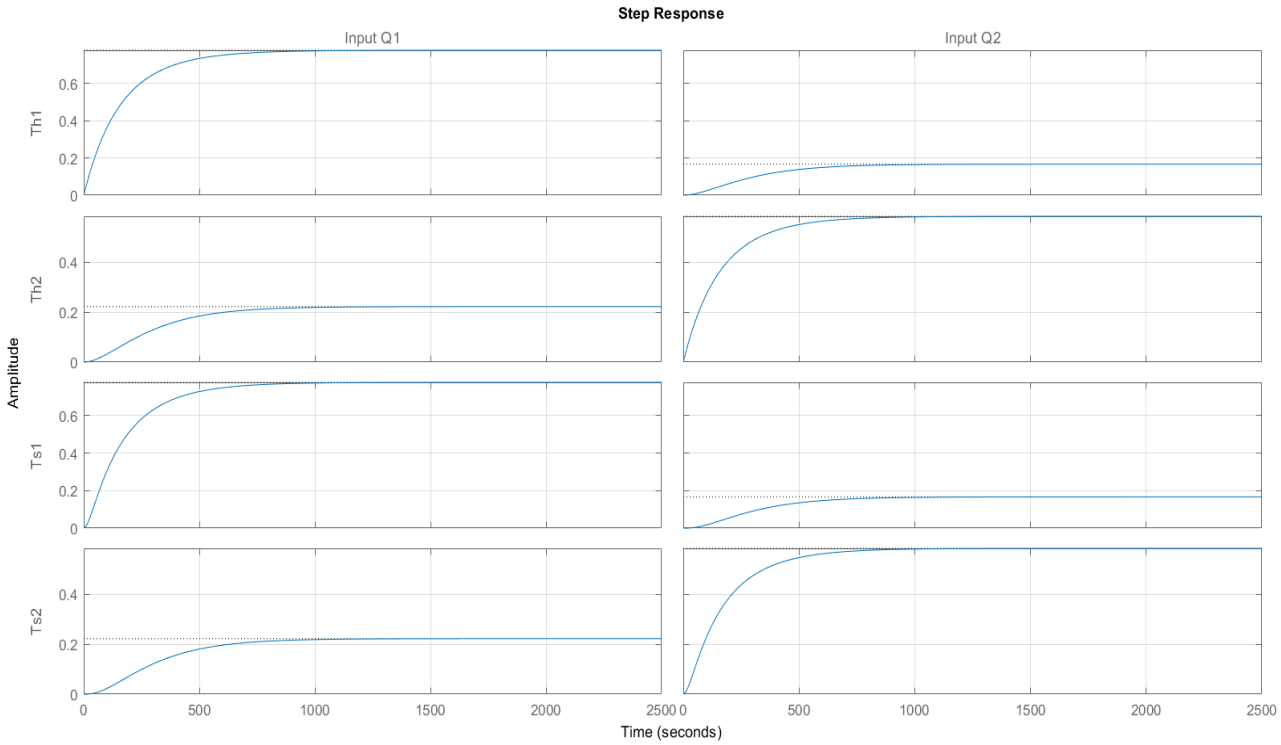

Figure 2.22: RGA analysis with Model linearization Step plot

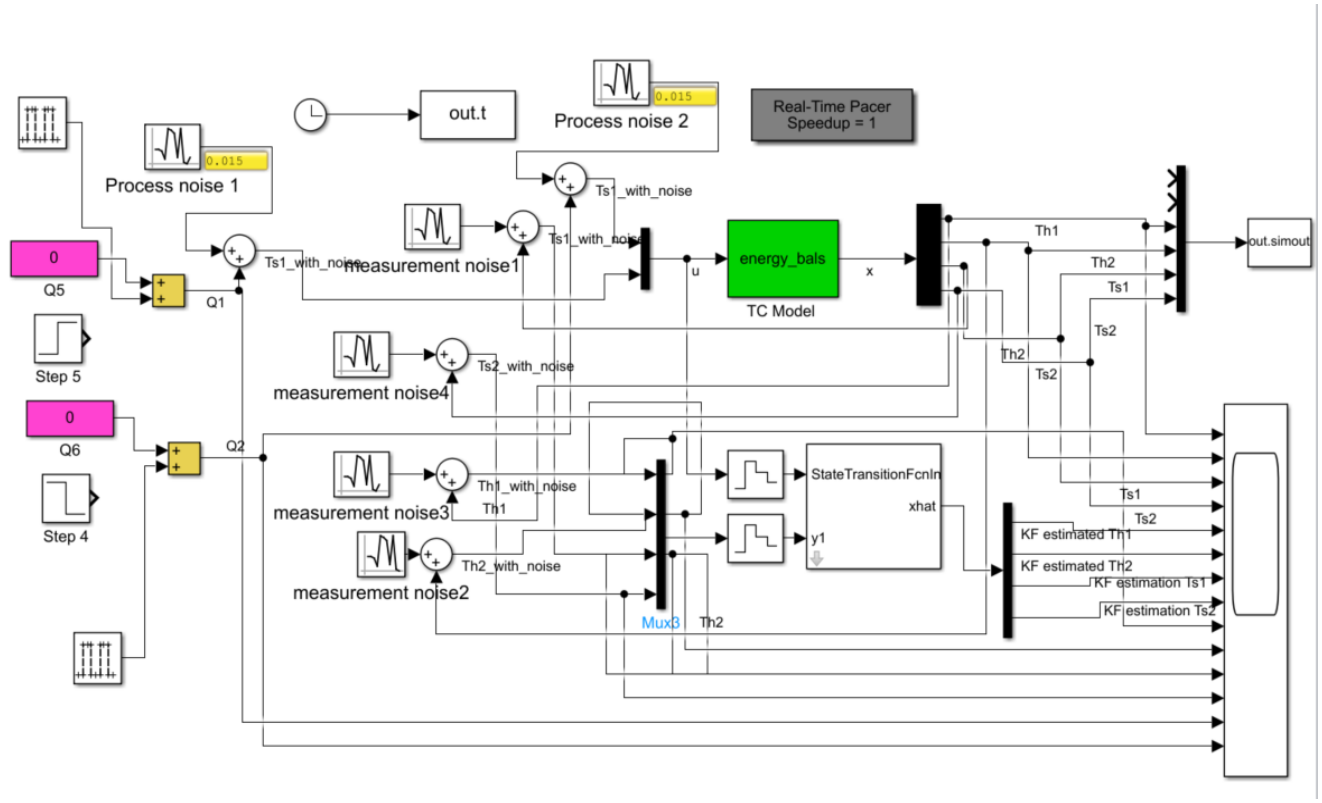

Figure 2.23: Simulink model of Augmented system with measurement and process noise

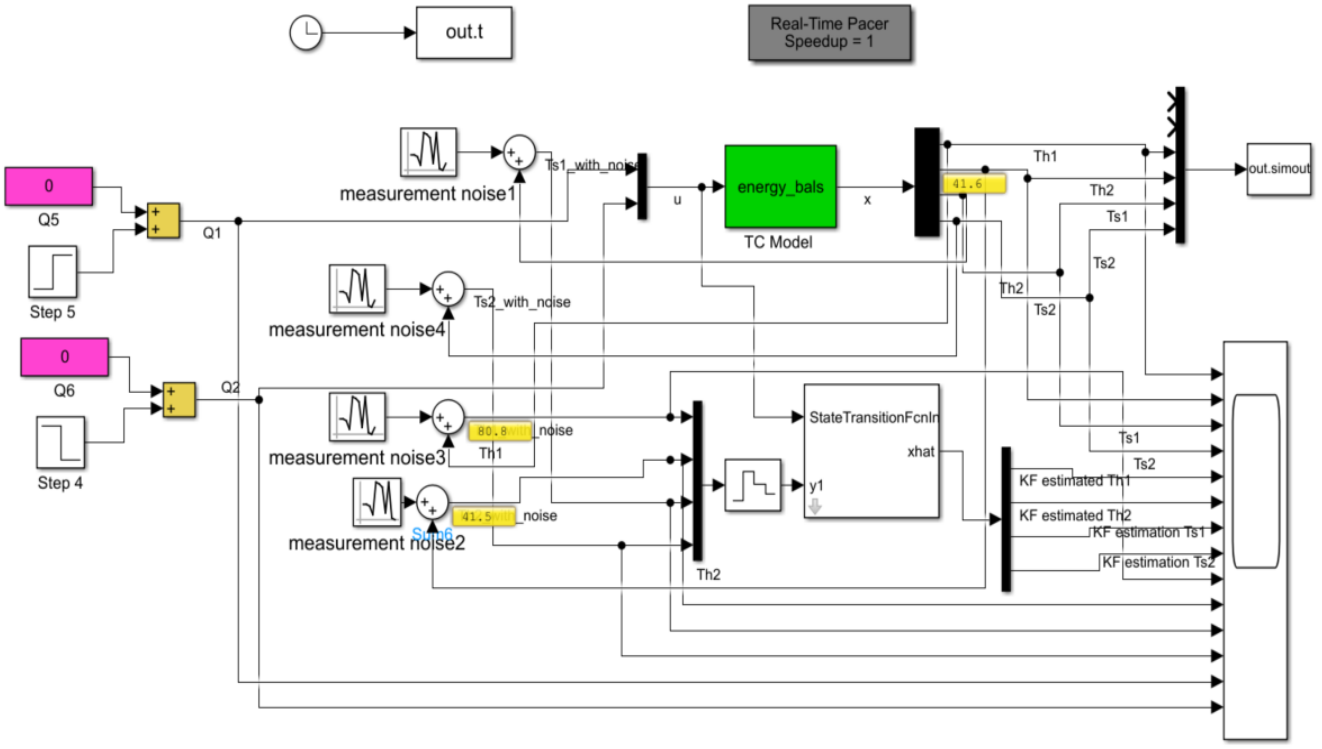

Figure 2.24: Simulink model of Augmented system with measurement noise

b) Analyze the observability of the linearized system!

```
% Checking Observability
obsv(A,C);
rank(obsv(A,C))

ans =

4
```

as we can see it is a full rank hence globally observable.

c) Design, implement and validate an EKF for the augmented system.

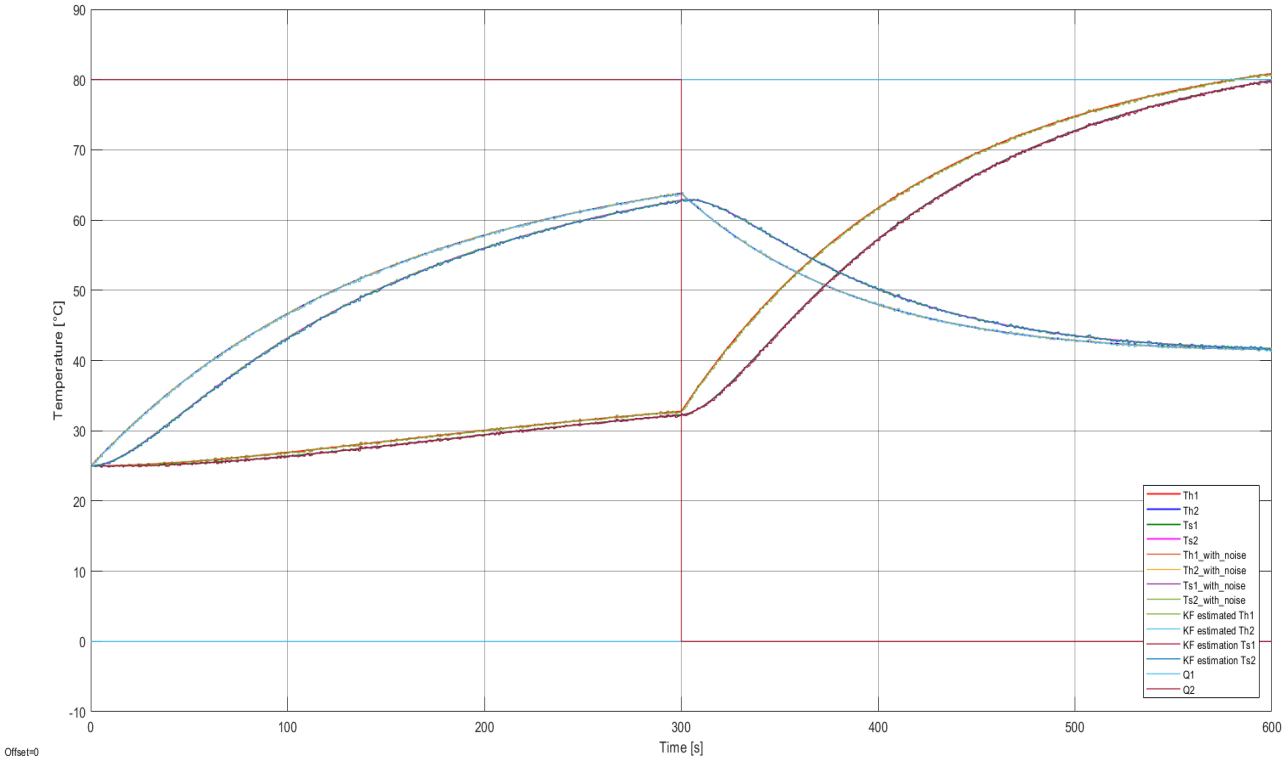

Figure 2.25: Augmented system with measurement noise vs Extended Kalman filter state estimation

From the above graphs, we can see that the extended Kalman filter is able to estimate the augmented system with and without process noise and measurement noise with high accuracy.

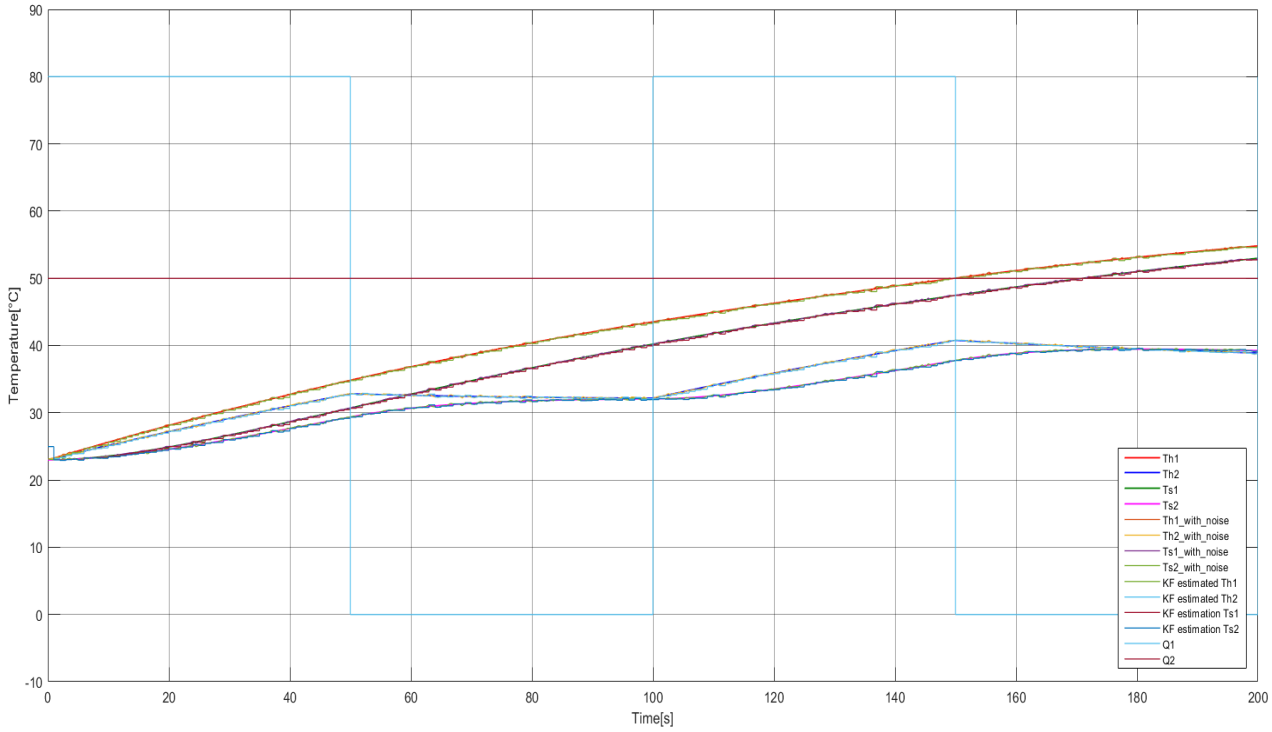

Figure 2.26: Augmented system with measurement and process noise vs Extended Kalman filter state estimation

Bibliography

[1] Dynamics and Control https://apmonitor.com/pdc/index.php/Main/ArduinoTemperatureControl. Temperature control lab. In *apmonitor*, 2016.

[2] MATLAB. *kalman filter version 7.10.0 (R2010a)*. The MathWorks Inc., https://de.mathworks.com/matlabcentral/fileexchange/28197-learning-kalman-filter-implementation-in-simulink-r, 2010.