```
clc;
clear;
close all;

BoilingWater2Air = readtable("DataQuench_BoilingWater2Air_2023.xlsx");
BoilingWater2IceWater =
 readtable("DataQuench_BoilingWater2IceWater_2023.xlsx");
IceWater2BoilingWater =
 readtable("DataQuench_IceWater2BoilingWater_2023.xlsx");

experimentNames = ["$ Quench: \ Boiling \ Water \ To \ Air $","$ Quench: \
 Boiling \ Water \ To \ Ice \ Water $" "$ Quench: \ Ice \ Water \ To \ Boiling
 \ Water $"];
experimentLinearPartsCount = [1,1,2];
%the linear parts before the exponentials will be removed. How many
%linear parts needs to be seen by the user, and inputted

%looks like the 101 columns are the actual data
data = {BoilingWater2Air,BoilingWater2IceWater,IceWater2BoilingWater};

for i = 1:length(data)
    t = data{i}{:,2}; %using the time step of the experiment
    T_t = data{i}{:,3}; %temperature at each time

    [T_0,T_final,t,T_t] =
 trueStartGraph(t,T_t,data{i},experimentLinearPartsCount(i)); %Resets the
 graph so that it starts when the quench does
    t;

    figure; %plotting the logarithmic line
    [timeConst,t,T_t] =
 timeConstantSolver(T_t,T_final,T_0,t,experimentNames(i)); %solves the time
 constant, but loses some increments of t and T_t
    hold off;

    figure;

 experimentalAndTheoreticalPlotter(T_final,T_0,t,timeConst,T_t,experimentNames(i))
    hold off;
end

function [initTemp,finalTemp,newTime,newTemps] =
 trueStartGraph(time,Temp,dataTable,linearPartCount)
    %Seems like it wasn't quenched initially at the first time entry
    %Will be detecting when the temp suddenly changes to exponential decay,
    %the linear parts before the exponentials will be removed. How many
    %linear parts needs to be seen by the user, and inputted
    [TF,S1,S2] = ischange(Temp,'linear');
    breakIndecies = find(TF==1);
    startingInd = breakIndecies(linearPartCount) - 1; %starting temp is right
 before the break, since it cuts it off when it stops being linear
    startingTime = time(startingInd)
```

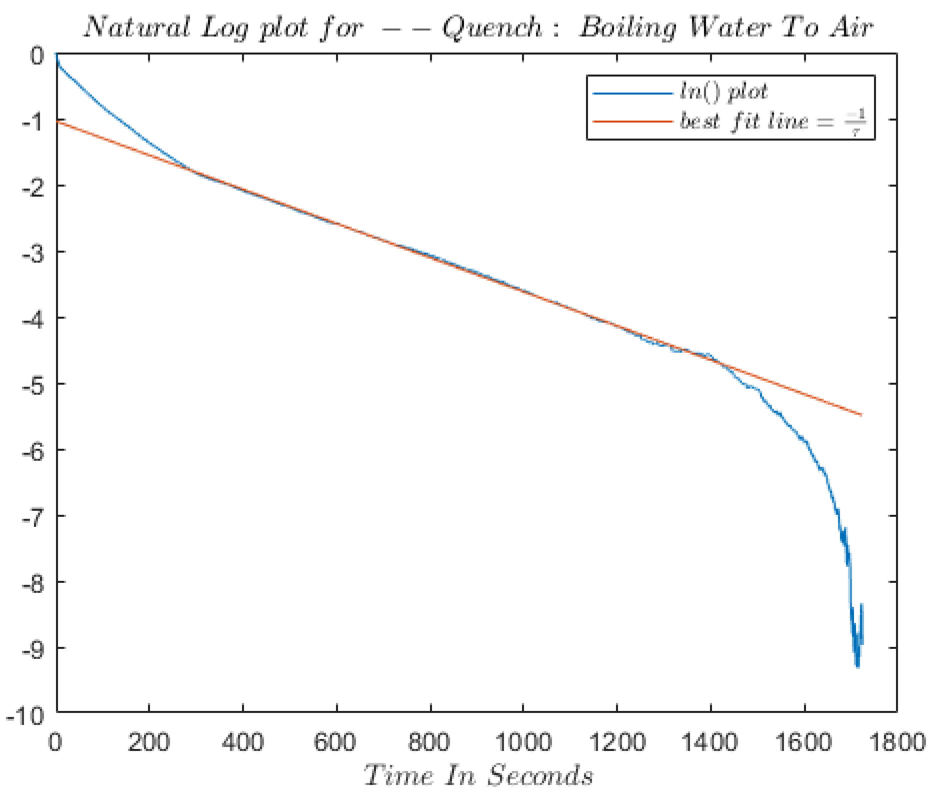
```
    initTemp = dataTable{startingInd,3};
    finalTemp = dataTable{end,3};

    time(1:startingInd-1) = []; %removing all indecies before the quench even
 starts, but not the starting time itself
    Temp(1:startingInd-1) = [];
    newTemps = Temp;
    newTime = time - startingTime; %setting the starting index to t = 0
end


function [timeConstant,filtered_t,filtered_T_t] =
 timeConstantSolver(Temps,finalTemp,initTemp,time,graphTitle)
    logarithmicPlot = log(abs((Temps - finalTemp)./(initTemp - finalTemp)));
    logarithmicPlot = smooth(logarithmicPlot); %smoothing out the plot with a
 moving average
    badIndecies = find(abs(logarithmicPlot) == Inf);%removes temp indecies
 before is exponential
    %must use abs, since some temps before the final in steady state dip below
    %the final temp, and it causes a negative inside the ln, which isn't
 possible
    %around the steady state point, can ignore values of inf. Is a
    %good approximation because is around steady state
    usefulIndecies = setdiff(1:height(Temps) , badIndecies); %set diff returns
 all
    %elements of input 1 that aren't in input 2
    time = time(usefulIndecies); % how to use only the good indecies
    filtered_t = time;
    Temps = Temps(usefulIndecies);
    filtered_T_t = Temps;
    logarithmicPlot = logarithmicPlot(usefulIndecies);

    %%solving fot the time constant
    %the test data's ln plot isn't very linear at the ends or beginning, so I
    %will have to filter out the section that is most linear
    [TF,S1,S2] = ischange(logarithmicPlot,'linear');
    % this will find the "abrupt" changes in the graph from curved to linear
    brkpt=time(TF==1); % gives the values of time where the abrupt changes
 happen
    %in my case, it seems to happen between the first and second abrupt
 changes
    breakIndecies = [find(time == brkpt(1)), find(time == brkpt(2)) ];

    bestFitLine =
 polyfit(time(breakIndecies(1):breakIndecies(2)),logarithmicPlot(breakIndecies(1):breakInd
 is in bestFitLine(1)
    %bestFitLine = polyfit(t,logarithmicPlot,1); %slope is in bestFitLine(1)
    timeConstant = -1/bestFitLine(1);

    %%dealing with the plot
    plot(time,logarithmicPlot)
    hold on;
    plot(time,bestFitLine(1)*time+bestFitLine(2))
```

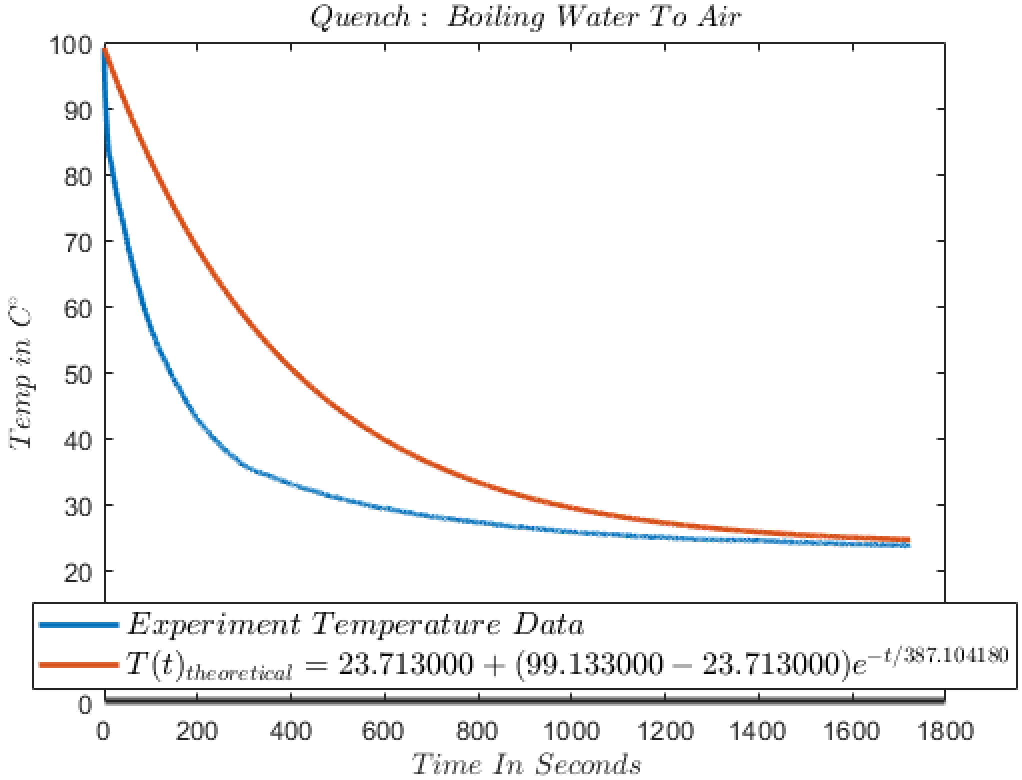
```
    lgnd= legend({"$ ln() \ plot $"," $ best \ fit \ line  =  \frac{-1}
{\tau}$"});
    set(lgnd, 'Interpreter','latex')
    graphTitle = erase(graphTitle,"$"); %removes both the $ so I can add the
 strings together
    title("$ Natural \ Log \ plot \ for \ -- \" +graphTitle
 +"$",'Interpreter','latex')
    xlabel("$ Time \ In \ Seconds $",'Interpreter','latex')
end

function [graph] =
 experimentalAndTheoreticalPlotter(finalTemp,initTemp,time,timeConstant,Temps,graphTitle)
    modelT_t = finalTemp + (initTemp - finalTemp)*exp(1).^(-time/
timeConstant);
    y_plots = [Temps , modelT_t];
    [rows,cols]=size(y_plots);

    for i = 1:cols
        plot(time,y_plots(:,i),LineWidth=2)
        hold on;
    end

    graphTitles = [graphTitle,"$ Time \ In \ Seconds $","$ Temp \ in \
 C^{\circ} $"];
    legendStuff = {'$ Experiment \ Temperature \ Data
 $',sprintf('$ T(t)_{theoretical} = %f + (%f - %f)e^{-t/%f}
 $',finalTemp,initTemp,finalTemp,timeConstant)};
    lgnd= legend(legendStuff);
    set(lgnd, 'Interpreter','latex')
    lgnd.FontSize = 12;
    lgnd.Location = 'best';
    title(graphTitles(1),'Interpreter','latex')
    xlabel(graphTitles(2),'Interpreter','latex')
    ylabel(graphTitles(3),'Interpreter','latex')
    yline(0,'LineWidth',3,'HandleVisibility','off')
end
```

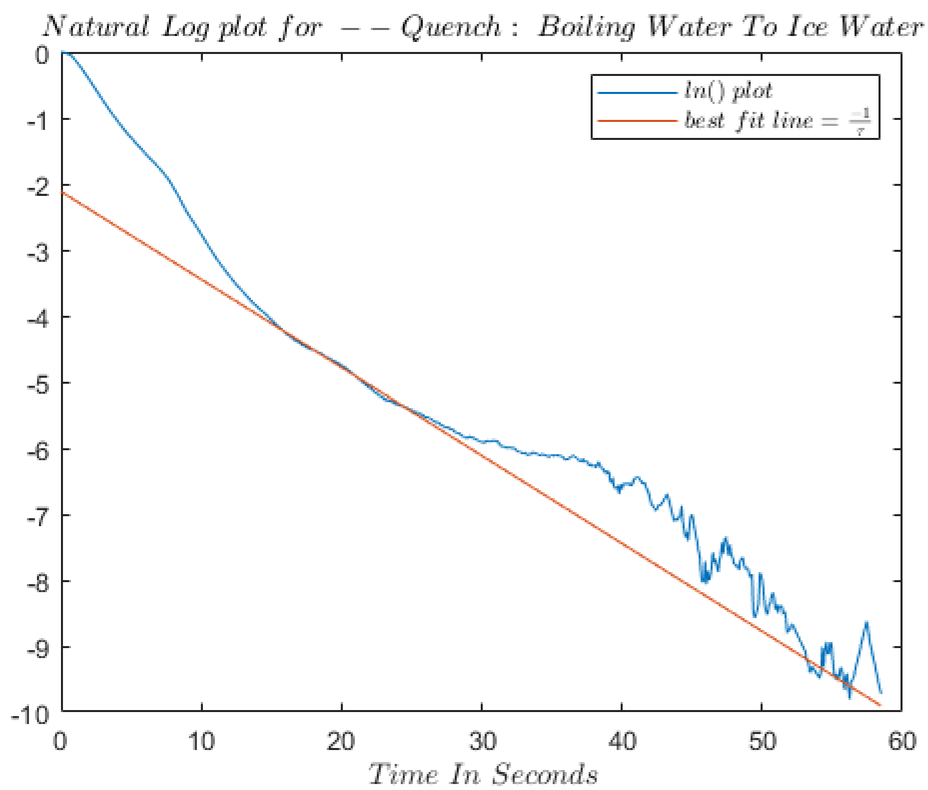
*Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating variable names for the table. The original column headers are saved in the VariableDescriptions property.*
*Set 'VariableNamingRule' to 'preserve' to use the original column headers as table variable names.*
*Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating variable names for the table. The original column headers are saved in the VariableDescriptions property.*
*Set 'VariableNamingRule' to 'preserve' to use the original column headers as table variable names.*
*Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating variable names for the table. The original column headers are saved in the VariableDescriptions property.*
*Set 'VariableNamingRule' to 'preserve' to use the original column headers as table variable names.*

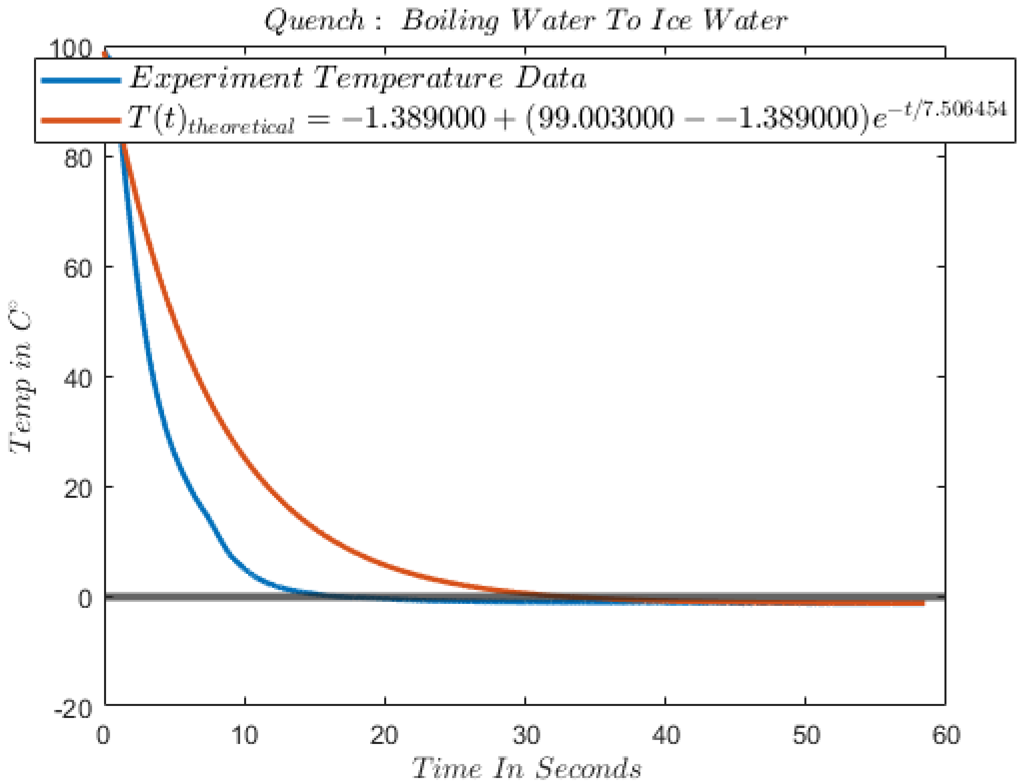
*startingTime =*

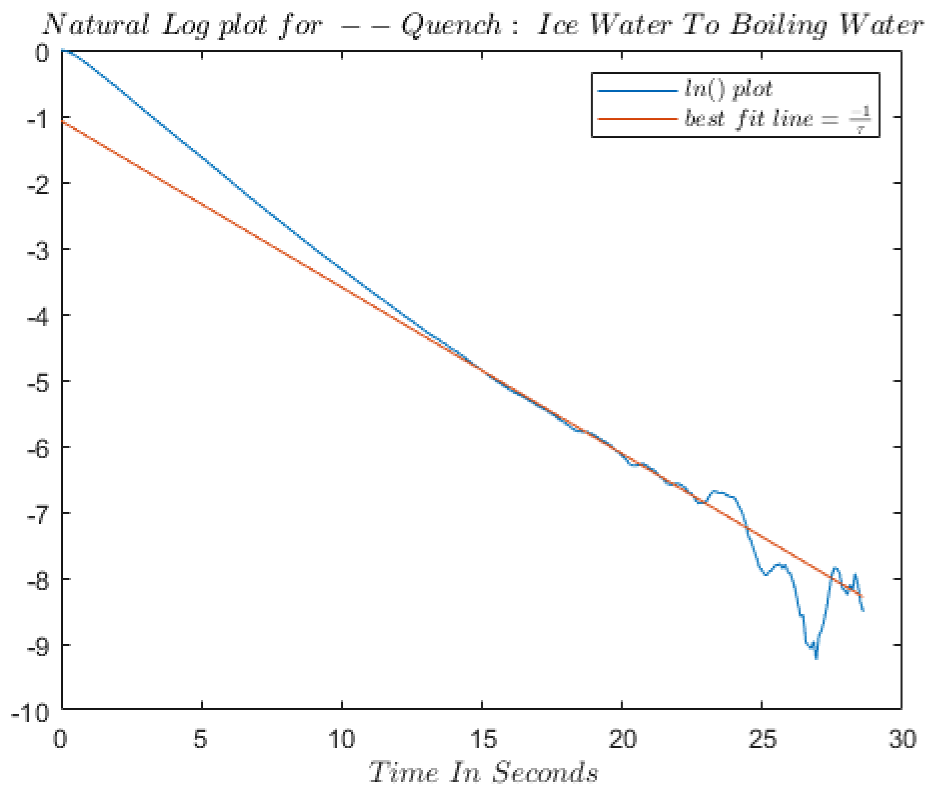
```
   15.5190

startingTime =

    5.4010

startingTime =

    6.8860
```

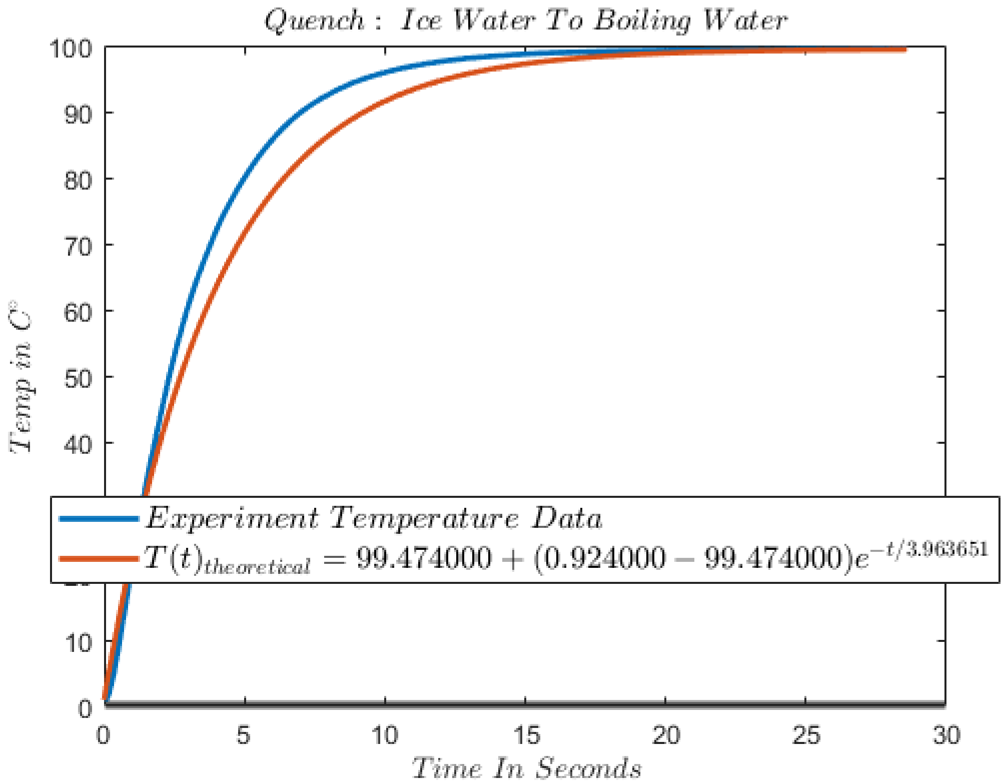



$Quench:\ Boiling\ Water\ To\ Air$
$Temp\ in\ C^\circ$
$Time\ In\ Seconds$
$Experiment\ Temperature\ Data$
$T(t)_{theoretical} = 23.713000 + (99.133000 - 23.713000)e^{-t/387.104180}$
0
20
30
40
50
60
70
80
90
100
0
200
400
600
800
1000
1200
1400
1600
1800


$Natural\ Log\ plot\ for\ --Quench:\ Boiling\ Water\ To\ Ice\ Water$
$ln()\ plot$
$best\ fit\ line = \frac{-1}{\tau}$
$Time\ In\ Seconds$
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10
0
10
20
30
40
50
60


$Quench: \ Boiling \ Water \ To \ Ice \ Water$
$Experiment \ Temperature \ Data$
$T(t)_{theoretical} = -1.389000 + (99.003000 - -1.389000)e^{-t/7.506454}$
$Temp \ in \ C^{\circ}$
$Time \ In \ Seconds$


$Natural \ Log \ plot \ for \ --Quench: \ Ice \ Water \ To \ Boiling \ Water$
$ln() \ plot$
$best \ fit \ line = \frac{-1}{\tau}$
$Time \ In \ Seconds$




*Published with MATLAB® R2022b*