Calorimetry Lab – Project 1

0dc91b091fd8[1]
University of Colorado, Boulder, CO, 80309

This report details the findings of a basic calorimetry expirement carried out using a calorimeter of known specific heat. The expirement was carried out in order to calculate the specific heat of an unknown sample, which was first heated in a boiling water bath to establish a known intial temperature. Thruough the use of the an NI DAQ system the temperature of the calorimter was monitored and logged. Using least squares regression and the first law of Thermodynamics, the specific heat of the sample could be calculated. Comparing the result of this analysis with a table of known specific heats for certain materials, the unknown sample was determined to be Tellurium Copper (Alloy 145).

Nomenclature

ΔE	=	*change in energy of a system*
ΔU	=	*change in internal energy of a system*
ΔKE	=	*change in kinetic energy of a system*
ΔPE	=	*change in potential energy of a system*
$Q_{net,in}$	=	*net heat transfer into system by surroundings*
$W_{net,out}$	=	*net work done by system on surrounds*
m_s	=	*mass of the sample*
c_s	=	*specific heat of the sample*
m_c	=	*mass of the calorimeter*
c_c	=	*specific heat of the calorimeter*
T_0	=	*initial temperature of the calorimeter*
T_1	=	*initial temperature of the sample*
T_2	=	*equilibrium temperature of the calorimeter and the sample*
K	=	*trailing-edge (TE) nondimensional angular deflection rate*

I. Introduction

CALORIMETRY is a thermodynamic technique for determining thermodynamic quantities and can successfully be used to calculate the specific heat of a material. The system used in the lab is an isolated system defined as a calorimeter with temperature sensors measuring the temperature inside the calorimeter.

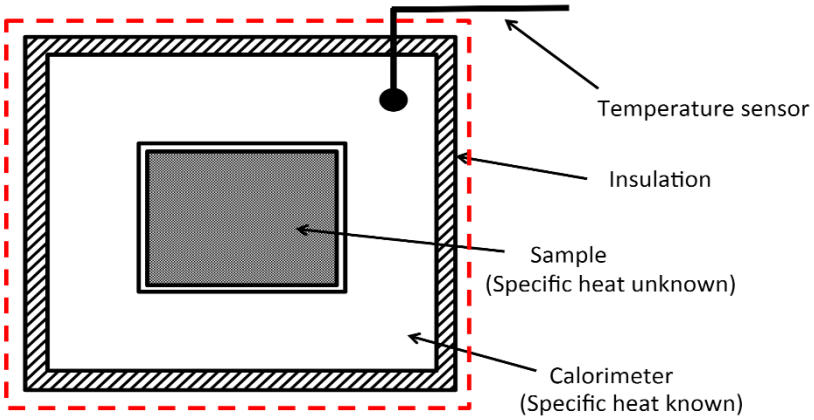

Figure 1. Calorimeter setup used in lab and Thermodynamic system definition

[1] AES

The calculation of the specific heat of a material is derived from the 1st law of Thermodynamics. This equation can be stated for an isolated system simply as:

$$\Delta E = Q_{net,in} - W_{net,out} = \Delta U + \Delta KE + \Delta PE$$

Under the *assumptions* that the system is both isolated ($Q_{net,in}$ & $W_{net,out}$ are 0) and is stationary (no changes in either KE or in PE), then the equation can be reduced to the following:

$$\begin{gathered} \Delta U = Q_{net,in} - W_{net,out} = 0 \\ \Delta U = U_2 - U_1 = 0 \\ U_2 = U_1 \end{gathered} \tag{1}$$

Assuming that the specific heat of the materials in the system is only dependent on temperature and that the temperature interval is small enough to approximate the internal energy-temperature gradient relation to be linear, the following relation can be used[1],

$$U = mc_{avg}\Delta T$$

where m is mass and c_{avg} is the average specific heat of the material. Substituting this into Eq. (1) yields the following:

$$m_s c_s (T_1 - T_2) = m_c c_c (T_2 - T_0)$$

$$c_s = \frac{m_c\, c_c\, (T_2 - T_0)}{m_s (T_1 - T_2)} \tag{2}$$

The non-temperature constants in Eqn. (2) are all defined in the nomenclature section above, and are all known values. The temperature in Eqn. (2) are all values that result from the analysis of the calorimeter's temperature profile. Therefore, a relationship between c_s and the rest of the system parameters can be obtained, and the thus the specific heat of a material can be calculated.

II. Experimental Method

The calorimeter in this lab is used to calculate the specific heat of any material placed inside it. It is designed out of material with a well-known specific heat, a crucial design element. In order to determine the specific heat of a sample, the specific heat of the calorimeter itself is needed, as per Eq. (2) above.

The experiment was carried out by first examining and preparing the for use the calorimeter, where the calorimeter material is aluminum with a known c_c = 0.214 cal/(g°C) and the mass of the calorimeter is then measured to be (313.50 ± 0.05)g. The sample is then selected and weighed several times to determine an average mass, measured to be (91.75 ± 0.05)g. Three thermocouples with software cold-junction compensation at the ITLL LabStation are then used to take temperature readings of the aluminum calorimeter through the use of an NI DAQ system and a LabVIEW VI. To obtain good temperature readings the thermocouple must maintain good contact with the aluminum calorimeter. The thermocouple is placed into the hole provided and secured with high temperature cotton before replacing the insulation cap (as shown in lab demonstration). While this is set up, the sample is immersed in boiling water for about 10 minutes so it can be assumed to be in equilibrium with the boiling water. A fourth thermocouples with software cold-junction compensation at the ITLL LabStation is also put in the boiling water to record its temperature with the sample in the water. Five minutes before removing the sample from the water, the *Temperature History VI* is initiated. It is set to take samples from these thermocouples every second. Just before removing the sample from the boiling water, the temperature of the water is recorded. Using tongs, the sample is removed and shaken to remove excess water. It is quickly placed in the calorimeter. The time at which the sample is placed in the calorimeter is recorded and the calorimeter is quickly sealed. The VI runs for approximately 10 more minutes before concluding the run. The program is terminated and the data saved.

III. Results

In order to calculate the specific heat of the sample, three temperatures need to be calculated from the experimental data, as can be seen from the three temperatures needed to compute c_s in Eqn. (2). T_1 is computed using the data taken from the fourth thermocouple, which is the thermocouple in the boiling water. By finding the average of the set of temperatures recorded before the sample is taken out of the water, a good estimate for the temperature of the boiling water – and thus of the sample (which is *assumed* to be in thermal equilibrium with the boiling water) – can be found. This is illustrated in the plot of the boiling water / sample over time as given below.

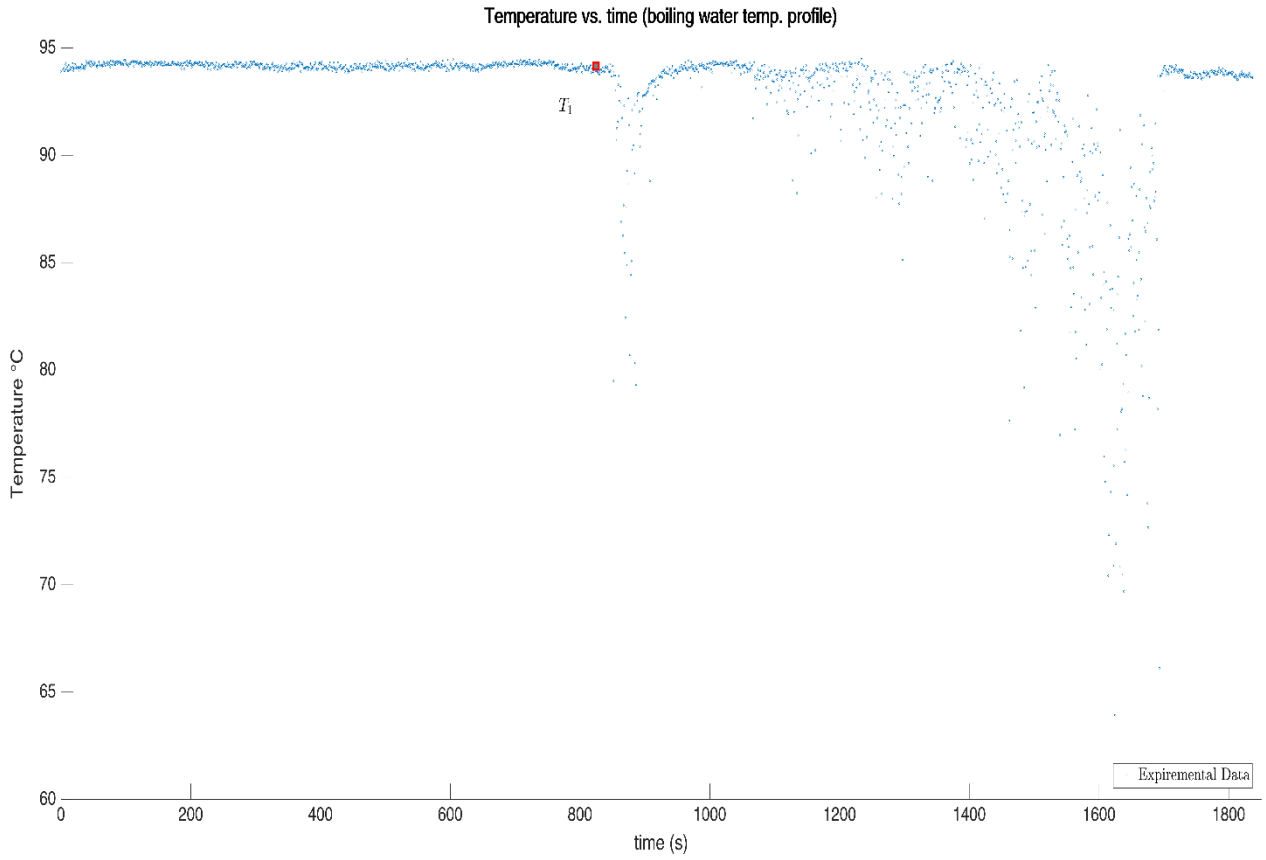

Figure 2. Temperature profile of the boiling water with the sample inside it, showing the time and temp. at which the sample was removed from the boiling water bath.

The uncertainty in the measurement of T_1 can be found by finding the standard deviation of the data set, instead of the standard deviation of the mean. The standard deviation is used to quantify the uncertainty in the mean of the data set, as the data for finding T_1 is recorded just once per experiment. A more detailed explanation of the algorithm for finding T_1 can be found in the appendix in the flow chart or in the header of the file "Project_1_calc_T_o.m" included in the appendix.

Next, T_0 needs to be calculated in order to find both T_2 and c_s. The data from the three thermocouples measuring the calorimeters was averaged together to get the most accurate representation of the temperature within the thermocouple. In order to find T_0, I first determined the time at which the sample was added to the calorimeter. This was done by analyzing the calorimeter data (from now on I am referring to the averaged data set) for "acceleration" changes in the data. When the acceleration of the data was found to be non-zero, indicating that the slope of the calorimeter was changing slopes. With this time, a linear regression was done on the data points from the initial time up until the time when the sample was added (can be seen as first regression line in the legend of Fig. 3 below). Plugging the time the sample was added into the line of best fit determined by the linear regression for this "lower" set of data yields the temperature termed T_L or T_0 which is 21.82˚C. This temperature represents the initial temperature of the calorimeter. To find the uncertainty in the measurement of T_0, the discrepancies between the experimental data

and the temperature value yielded by the line of best fit in each term in the "lower" data set were summed in quadrature, as defined by the following equation[2]:

$$\sigma_T = \sqrt{\frac{1}{N-2}\sum_{i=1}^{N}(T_i - (\beta_0 + \beta_1 t_i))^2} \quad (3)$$

where N is the number of samples in the linear regression, $T = \beta_0 + \beta_1 t$ is the line of best fit determined by the regression, and σ_T is σ_{T_o}, which is the uncertainty in T_0. This uncertainty was calculated to be 0.01˚C. For more information about how $T_0$was calculated, a detailed description can be found in the flow chart or in the file "Project_1_calc_T_1.m" both found in the appendix.

To calculate T_2, the data from the temperature profile of the calorimeter is used, as was used before for calculating T_0. This is done by determining the time at which the sample and the calorimeter reach thermal equilibrium. This is done by finding when the maximum temperature occurs in the temperature profile, as after this the sample would not have transferred any more heat to the calorimeter and there could be no more thermal gradient to promote heat transfer. Next, a linear regression was done on the data from the time when the sample and calorimeter were in thermal equilibrium ot the last data point to determine a line of best fit for this data set, shown below in Fig. 3 as the second regression line in the legend. Below is a plot of the overall temperature profile of the calorimeter used in the calculation of both T_0 and T_2:

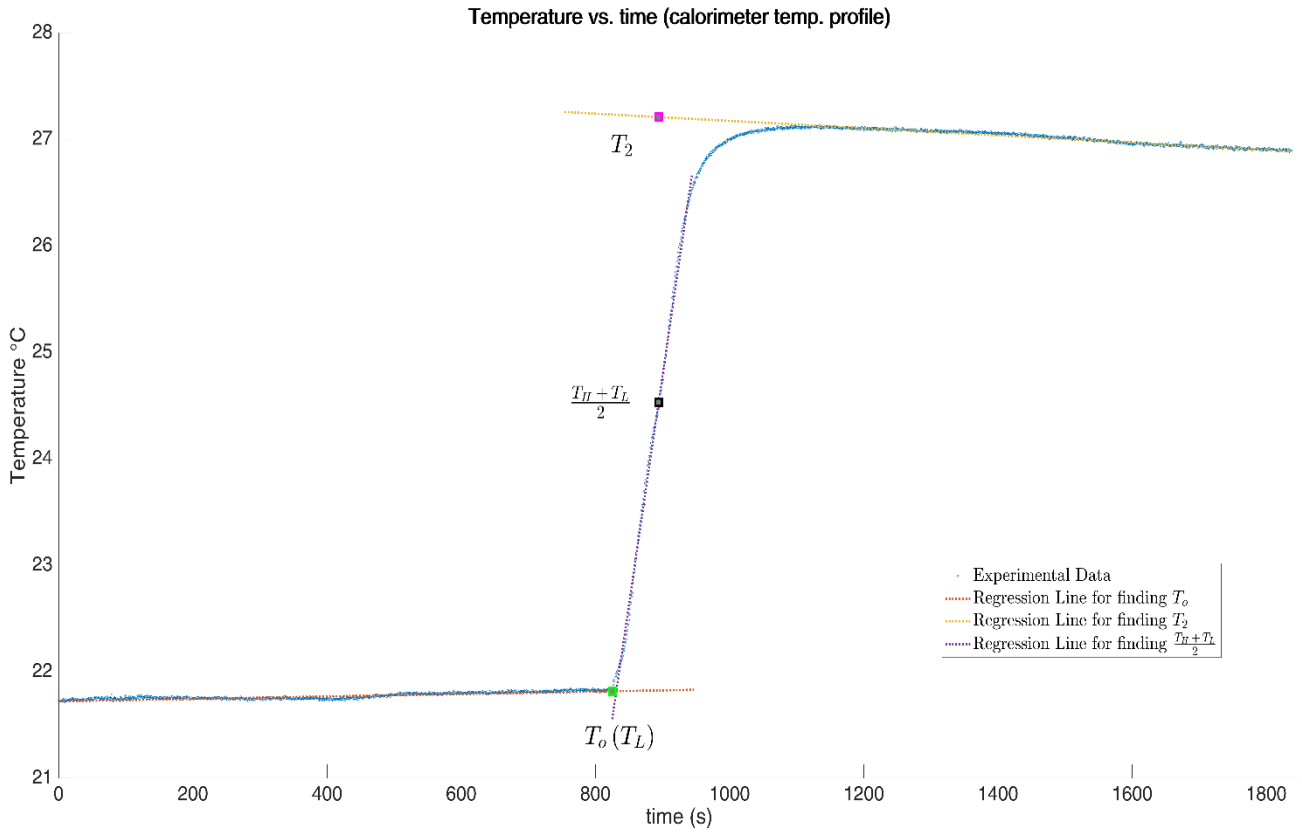

Figure 3. Plot of the temperature profile of the calorimeter over the duration of the test. *Labeled are the points used in the calculation of the specific heat of the sample*

Plugging in the time when the sample was added to the line of best fit yields the temperature T_H. The average temperature between T_L and T_H is then computed, shown on Fig. 3 above. Then a linear regression is done on the data from when the sample was added to just before the sample and calorimeter are in thermal equilibrium, which yields the line of best fit shown above in Fig. 3 as the third regression line. As per the calorimeter producers procedure, this line of best fit is used the time when the temperature is the average of $T_L$and T_H, which was calculated to be 24.53˚C. Then, this time is plugged into the line of best fit for the data of the sample and calorimter in thermal equilibrium to

yield T_2, which is shown above in Fig. 3 as occurring at the same time as the average temperature. T_2 was calculated to be 27.21˚C. To find the uncertainty in the measurement of T_2, the uncertainty in the data along the line of best fit for the equilibirum temperature data is first computed in exactly the same way as for T_0, using Eqn. (3). Then, this uncertainty was used to create a matrix, **Q**, conatinaing the uncertainties in the coefficients of the line of best fit. This matrix **Q** was then used along with a vector containing the time used to extrapolate the value of T_2, and with the scalar product then square rooted. This scalar value is the uncertainty in the value of T_2, which was calculated to be 0.0013˚C. For more information about this uncertainty calculation, or for the details of the algorithm behind finding the value of T_2, please see the appendix for a detailed breakdown of the algorithm in the flow chart, and a detailed description of my steps in the file "Project_1_calc_T_2.m".

To calculate c_s, the values calculated above, along with the given values for m_s, c_c, and m_c, were simply substituted into Eqn. (2) to find the value of c_s, which was found to be 0.247 J / g / K. The uncertainty in c_s, σ_{c_s}, was found using the general error propogation formula[3]:

$$\sigma_{c_s} = \sqrt{\left(\frac{\partial c_s}{\partial m_s}\sigma_{m_s}\right)^2 + \left(\frac{\partial c_s}{\partial m_c}\sigma_{m_c}\right)^2 + \left(\frac{\partial c_s}{\partial c_c}\sigma_{c_c}\right)^2 + \left(\frac{\partial c_s}{\partial T_0}\sigma_{T_0}\right)^2 + \left(\frac{\partial c_s}{\partial T_1}\sigma_{T_1}\right)^2 + \left(\frac{\partial c_s}{\partial T_2}\sigma_{T_2}\right)^2}$$

The results of plugging in all of finding all of the partial derivatives and plugging the known uncertainties for each term yield a value of uncertainty in c_s, σ_{c_s}, to be 0.001 j / g / K (8.2 E -04 unrounded).

These results seem to match best with the Tellurium Copper (Alloy 145) alloy, which has a specific heat of 0.261 j / g / K. While the value for c_s matches closely to the value for Tellurium Copper (Alloy 145), the two value do not overalp, even given the uncertainty in c_s. This means that I cannot say for absolute sure that the sample is Tellurium Copper, as ther uncertainty is c_s is too small to guarantee that.

IV. Discussion

The reults of the analysis seem to suggest that while the sample is likely Tellurium Copper, the relatively small uncertainty in the value for c_s means that the sample and Telluriurm Copper are not necessarily the same material. This result is likely more due to the fact that some of the parameters used in the calculation of c_s were not quite correct. The term with the greatest chance to be the influencing factor in the incorrect calculation of the sample's c_s is the T_2, as this value was the result of many calculations that could be erroneous in their assumptions. If T_2 were to be larger, perhaps due to a different assumption for the rate of heat loss from the calorimeter during thermal equilibirum, then the value for c_s may have been corrected enough to make it align better witht the known value of for Tellurium Copper.

Another source of possible error in the expirement relates to T_1. If the sample and the hot water were not quite in thermal equilibirum, such that the sample was still a little bit below the temperature of the boiling water, then correcting for this mistake would lead to a result for c_s that matched the value for Tellurium Copper.

V. Conclusion

Overall, the value of c_s matched well with the uncertainties seen in the rest of the values used in its calculation. This validifies the results of my algorithm, as the combined small uncertainties that the independent variables in Eqn. (2) had should result in a similarly small (in terms of fractional uncertainty) uncertainty in c_s. So while the value of c_s matched realtively well with the value given for Tullurium Copper, adjustments in expiremental procedure and/or in the execution of the experiment itself would need to be made to be more certain of the sample's identity. Other procedures for calculating heat loss that resulted in higher equilibirum tempratures, T_2, would probably be the first thing to attempt to improve the analytical procedure. With some slight adjustments, the other small uncertainties in all of the variables used in calculating c_s would yield both precise and accurate results for the calculation of c_s.

Appendix

A. Software Algorithm Flow Chart

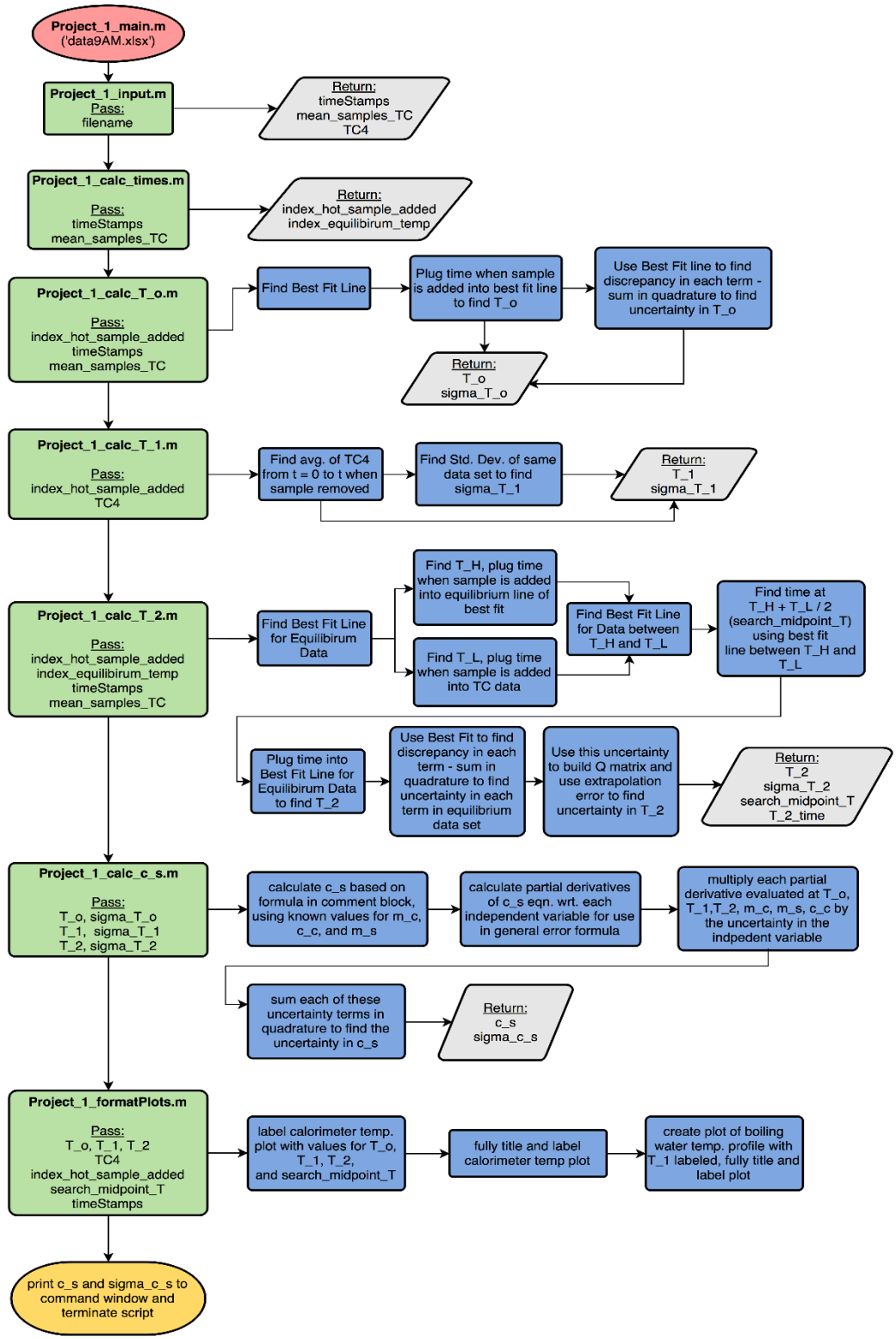

Figure 4. Flow Chart

B. Code Used

1. "Project_1_main.m"

```
%%% Purpose:
%%%       This script calls a set of routines used to proccess the excel data
%%%       in the file defined by the user below as "filename". The code
%%%       calculates the specific heat of the material tested and the
%%%       uncertainty in the specific heat of the material based on the
%%%       quality of the data inputted.
%%%
%%% Inputs:
%%%         - user defines filename to be analyzed by setting "filename" to
%%%         the name of the file to be analyzed.
%%% Outputs:
%%%         - calculates the specific heat of the sample, its uncertainty,
%%%         and generates plots of its temperature profile along with the
%%%         results of the regressions used to analyze this overall
%%%         temperature profile.
%%%
%%% Assumptions:
%%%             - none made specifically in this main script. see each
%%%               individual routine for the specific assumptions made for
%%%               each routine
%%%
%%% Author ID:  0dc91b091fd8
%%% Date Created:   10/15/2016
%%% Date Modified:  10/21/2016


%% Calculating Specific Heat of Sample

clear
clc

filename = 'data9AM.xlsx';

% reading in data from xlsx file
[timeStamps, mean_samples_TC, TC4] = Project_1_input(filename);

% finding the time when the sample was added to the calorimeter and the
% time when the sample is in equilibrium with the calorimeter
[index_hot_sample_added, index_equilibirum_temp] = Project_1_calc_times(...
                                                    timeStamps,...
                                                    mean_samples_TC);

% finding T_o
[T_o, sigma_T_o] = Project_1_calc_T_o(index_hot_sample_added,...
                                      timeStamps, mean_samples_TC);

% finding T_1
[T_1, sigma_T_1] = Project_1_calc_T_1(index_hot_sample_added, TC4);

% finding T_2
[T_2, sigma_T_2, T_2_time, search_midpoint_T] = Project_1_calc_T_2(...
                                                index_hot_sample_added,...
                                                index_equilibirum_temp,...
                                                timeStamps, mean_samples_TC);

% calculating specific heat of sample in and its uncertainty
[c_s, sigma_c_s] = Project_1_calc_c_s(T_o, sigma_T_o, T_1,...
                                      sigma_T_1, T_2, sigma_T_2);
```

```
% formatting / creating plots
Project_1_formatPlots(T_o, T_1, T_2, T_2_time, TC4, search_midpoint_T, ...
                      index_hot_sample_added, timeStamps);


%% Printing Results

fprintf('The specific heat of the sample is: %0.4g (J / g / K)\n', c_s);
fprintf('The uncertainty in the calculation of the specific heat is: %0.1g (J / g /
K)\n', sigma_c_s);
```

2. "Project_1_input.m"

```
%%% Purpose:
%%%         This function reads in the experimental data from an .xlsx file
%%%         into several arrays and returns them to the driver script for
%%%         processing.
%%%
%%% Inputs: filename - string containing .xlsx file to be read
%%%
%%% Outputs: timeStamps - array containing time stamps for each measurement
%%%          mean_samples_TC - array containing the average of all three TC
%%%                            that took data during the experiment
%%%          TC4 - array containing the TC measurements of the boiling water
%%%                bath
%%%
%%% Assumptions:
%%%              - the three calorimeter TCs can be averaged together
%%%                initially to produce one continuous curve most
%%%                representative of the overall state of the calorimeter.
%%%
%%% Author ID:  0dc91b091fd8
%%% Date Created:   10/15/2016
%%% Date Modified:  10/21/2016

function [timeStamps, mean_samples_TC, TC4] = Project_1_input(filename)

    % load .xlsx timestamp & TC data
    xlsxData = xlsread(filename);

    % save .xlsx data into arrays containing data from time stamp and each TC
    timeStamps = xlsxData(:, 1);
    TC1 = xlsxData(:, 2);
    TC2 = xlsxData(:, 3);
    TC3 = xlsxData(:, 4);
    TC4 = xlsxData(:, 5);

    % average the values from the 3 calorimeter
    mean_samples_TC = (TC1 + TC2 + TC3) / 3;

end
```

3. "Project_1_calc_times.m"

```
%%% Purpose:
%%%         This function takes the timestamp and mean TC value for
%%%         calorimeter arrays and calculates the indices for when the sample is
%%%         added to the calorimeter, and when the equilibrium regime of
%%%         the system is reached.
%%%
%%%         Algorithm for finding when sample is
```

```
%%%         added works by analyzing the second derivative of the temp vs
%%%         time data and detecting when slope is rapidly changing,
%%%         indicating the calorimeter is now being rapidly heated by the
%%%         introduction of a sample.
%%%
%%%         Algorithm for finding equilibrium index works as stated in the
%%%         lab procedure, by simply finding when the maximum temp. of the
%%%         equilibrium region occurs.
%%%
%%% Inputs: mean_samples_TC - array of average TC measurement of
%%%                           calorimeter
%%%         timeStamps - array of time stamps for each measurement taken
%%%
%%% Outputs: index_hot_sample_added - index in data arrays for when the
%%%                                   sample is added to the calorimeter
%%%
%%%          index_equilibirum_temp - index in data arrays for when the
%%%                                   sample and the calorimeter first are
%%%                                   in thermal equilibrium
%%%
%%% Assumptions:
%%%             - equilibrium begins when the maximum temperature is
%%%               achieved
%%%             - no significant thermal gradient is experienced by the
%%%               system until the sample is added. The only way the
%%%               temperature of the calorimeter changes is by the addition
%%%               of the sample
%%%
%%% Author ID:  0dc91b091fd8
%%% Date Created:   10/15/2016
%%% Date Modified:  10/21/2016

function [index_hot_sample_added, index_equilibirum_temp] =
Project_1_calc_times(timeStamps, mean_samples_TC)

    % initializations
    slope = zeros(1, length(mean_samples_TC));
    new_slope = slope;

    % build up vec of slopes to find when sample was added
    num = 100; % number of data points in best fit line interval
    hasFoundT_L = false;
    for i = num + 1:length(mean_samples_TC)

        % find instantaneous slope using best fit line over num data
        % points
        fit_params = polyfit(timeStamps(i - num:i), mean_samples_TC(i - num:i), 1);
        slope(i - num) = abs(fit_params(1));

        % finding "acceleration" of data set to find when slope changes
        if i > num + 1
            new_slope(i - num) = slope(i - num) - slope(i - num - 1);
        end

        if i > num + 1 && new_slope(i - num) > 0.25e-4 && ~hasFoundT_L
           index_hot_sample_added = i - 2; % this is the index of when the sample was
added to the calorimeter
           hasFoundT_L = true;
        end

    end

    % find the beginning index of the third region of calorimeter data
```

```
    max_T = max(mean_samples_TC(1:1100)); % exclude later data from consideration
(noisy)
    index_equilibirum_temp = find(mean_samples_TC == max_T);
    %plot(timeStamps, mean_samples_TC, index_equilibirum_temp(1), max_T, '+')
end
```

4. *"Project_1_calc_T_o.m"*

```
%%% Purpose:
%%%             This function takes the timestamp and mean TC values for
%%%         calorimeter arrays and the index for when the sample is first
%%%         added to the calorimeter and calculates T_o and its
%%%         uncertainty. This is done using a linear least-squares
%%%         regression on the mean TC data from the start time to the time
%%%         at which the sample was added. Then, using this linear
%%%         regression, the time at which the sample was introduced is
%%%         plugged into the regression line to find T_o.
%%%
%%%             The uncertainty in T_o is calculated using eqn. 8-15 from
%%%         textbook. This is just essentially finding the deviation of
%%%         each term and summing them in quadrature.
%%%
%%% Inputs: mean_samples_TC - array of average TC measurement of
%%%                           calorimeter
%%%         timeStamps - array of time stamps for each measurement taken
%%%
%%%         index_hot_sample_added  - index in data arrays for when the
%%%                                   sample is added to the calorimeter
%%%
%%%
%%% Outputs: T_o - initial temperature of the calorimeter
%%%          sigma_T_o - uncertainty in the measurement of T_o
%%%
%%% Assumptions:
%%%             - can use eqn. 8-15 in book to find uncertainty (errors are
%%%             all independent of each other and can be summed in
%%%             quadrature).
%%%
%%%             - data actually fits a linear relationship with time, and
%%%             as such can be fitted with a line of best fit by least
%%%             square linear regression.
%%%
%%% Author ID:  0dc91b091fd8
%%% Date Created:   10/15/2016
%%% Date Modified:  10/21/2016


%%
function [T_o, sigma_T_o] = Project_1_calc_T_o(index_hot_sample_added, timeStamps,
mean_samples_TC)

    %%% use unweighted least-squares to find best fit line and the
    %%% uncertainty in the measurement of T_o

    % create A from time values for each sample and the slope coefficient
    % m in T = mt + b
    numTimeStamps = length(timeStamps(1:index_hot_sample_added));
    bCoefficients = ones(numTimeStamps, 1);
    mCoefficients = timeStamps(1:index_hot_sample_added);

    A = cat(2, mCoefficients, bCoefficients); % make A from m and b coeffs.
```

```
    % find d matrix from A x = d matrix equation from the T values in the
    % range from t = 0 to the time when the sample was added
    d = mean_samples_TC(1:index_hot_sample_added);

    % find least squares linear fit line
    P_leastSquares = inv(A' * A) * A' * d; %#ok<MINV>
    m = P_leastSquares(1);
    b = P_leastSquares(2);

    linear_fit = @(t) m*t + b; % linear fit anonymous function

    % using line of best fit to find T_o
    time_hot_sample_added = timeStamps(index_hot_sample_added);
    T_o = linear_fit(time_hot_sample_added);

    %%% Computing Uncertainties
    sum_discrepency = 0; % start with no uncertainty

    % summing discrepancies
    for i = 1:numTimeStamps
        t_i = A(i, 1);
        sum_discrepency = sum_discrepency + ( d(i) - linear_fit(t_i) )^2 ;
    end

    % finding uncertainty in measurement of T_o
    sigma_T_o = sqrt( ( 1 / (numTimeStamps - 2) ) * sum_discrepency );

    %% Plotting
    LINEWIDTH = 2.5;
    FIT_STYLE = ':';
    MARKERSIZE = 2;
    FONTSIZE = 22;
    xExtent = 120;
    X_LOW = 0;
    X_HIGH = 1850;


    hFig = figure(1);
    set(gca,'FontSize', FONTSIZE)
    set(hFig, 'Position', [100 100 1600 900])

    xlim([X_LOW, X_HIGH])


    hold on
    plot(timeStamps, mean_samples_TC, 'o', 'MarkerSize', MARKERSIZE)
    plot(timeStamps(1:index_hot_sample_added + xExtent),
linear_fit(timeStamps(1:index_hot_sample_added + xExtent)),...
                    FIT_STYLE, 'LineWidth', LINEWIDTH)

End
```

5. *"Project_1_calc_T_1.m"*

```
%%% Purpose:
%%%        This function takes in TC4, the array of measurements of the
%%%        boiling water and calculates T_1 and its uncertainty.
%%%
%%%           By finding the mean of the data before the sample is removed,
%%%        a more accurate estimation of the temperature of the boiling
%%%        water and the sample can be found by averaging the relatively
```

```
%%%        constant boiling water temperatures from t = 0 up until the
%%%        sample is removed.
%%%
%%%         To find the uncertainty in T_1, take the standard deviation of
%%%         the set of data averaged to find T_1. This will give the best
%%%         estimate as to the average deviation of the temp. of the
%%%         boiling water / sample
%%%
%%%
%%% Inputs:
%%%         index_hot_sample_added  - index in data arrays for when the
%%%                                   sample is added to the calorimeter
%%%
%%%         TC4 - array containing the temperature profile of the boiling
%%%               water
%%%
%%% Outputs:  T_1 - initial temperature of the sample
%%%           sigma_T_1 - uncertainty in the measurement of T_1
%%%
%%% Assumptions:
%%%             - The sample and the water are in thermal equilibrium, so
%%%              the initial temp of the sample is the also the temperature
%%%              of the boiling water.
%%%
%%%              - while water boils, temp. does not change, so therefore a
%%%              linear regression is unnecessary here as the slope is just
%%%              = 0.
%%%
%%%              - can just use data from t = 0 to time when sample is
%%%              removed to find the temperature of the sample, as the
%%%              sample should be in thermal equilibrium with the boiling
%%%              water at that point.
%%%
%%% Author ID:  0dc91b091fd8
%%% Date Created:   10/15/2016
%%% Date Modified:  10/21/2016


function [T_1, sigma_T_1] = Project_1_calc_T_1(index_hot_sample_added, TC4)


    % make vector of water temp from t = 0 to when sample was removed
    boiling_water_temp_vec = TC4(1:index_hot_sample_added);

    % T_1 is the average of the boiling water temp from t = 0 up until the
    % sample is removed
    T_1 = mean(boiling_water_temp_vec);

    % error in T_1 is just the standard deviation of the water temp
    % measurements.
    sigma_T_1 = std(boiling_water_temp_vec);


end
```

6. "Project_1_calc_T_2.m"

```
%%% Purpose:
%%%             This function takes the timestamp and mean TC values for
%%%          calorimeter arrays, the index for when the sample is first
%%%          added to the calorimeter, and the index for when the sample and
%%%          the calorimeter reach thermal equilibrium.
%%%
%%%             This is done by using unweighted least-squares to find best
%%%          fit line for the equilibrium line and then plugging in the time
%%%          at which the sample was added to find a new theoretical maximum
%%%          temp. achieved - T_H. Then, by assuming that T_o as found
%%%          before is actually T_L, I calculated the average of T_H and T_L
%%%          as described in the lab document.
%%%
%%%             Next, I find the time at which this new average temperature
%%%          occurs by fitting a best fit line to the data between T_L and
%%%          T_H, and solving for the time at which it satisfies the eqn:
%%%                  T_avg = mt + b (m and b are regression coeffs.)
%%%
%%%             I then take this time and plug it back into the regression
%%%          equation for the equilibrium data, to find the temperature of
%%%          the system at the same time as when T_avg occurs, thus finding
%%%          the value for T_2 needed for calculation of the specific heat
%%%          of the sample.
%%%
%%%             To find the uncertainty in the measurement of T_2, one needs
%%%          to coefficients of the regression to compute the error in
%%%          extrapolating beyond the range of known data. This is done by
%%%          creating the weighting matrix using the formulation shown in
%%%          class and using eqn. 8-15 from the book as the uncertainty in
%%%          each data point used in the regression.
%%%
%%%
%%% Inputs:
%%%          index_hot_sample_added  - index in data arrays for when the
%%%                                    sample is added to the calorimeter
%%%
%%%          index_equilibirum_temp - index in data arrays for when the
%%%                                    sample and the calorimeter first are
%%%                                    in thermal equilibrium
%%%
%%%          timeStamps - array of time stamps for each measurement taken
%%%
%%%          mean_samples_TC - array of average TC measurement of
%%%                            calorimeter
%%%
%%%
%%%
%%% Outputs:  T_2 - equilibrium temperature of the calorimeter and sample
%%%           sigma_T_2 - uncertainty in the measurement of T_2
%%%           search_midpoint_T - avg of T_H and T_L
%%%
%%% Assumptions:
%%%             - can use eqn. 8-15 in book to find uncertainty (errors are
%%%             all independent of each other and can be summed in
%%%             quadrature).
%%%
%%%             - data actually fits a linear relationship with time, and
%%%             as such can be fitted with a line of best fit by least
%%%             square linear regression.
%%%
%%%             - can use the extrapolation error formula as the actual
```

```
%%%             uncertainty of T_2, as this encapsulates all of the
%%%             needed sources of error in its measurement
%%%
%%% Author ID:  0dc91b091fd8
%%% Date Created:   10/15/2016
%%% Date Modified:  10/21/2016



%%
function [T_2, sigma_T_2, T_2_time, search_midpoint_T] =
Project_1_calc_T_2(index_hot_sample_added, index_equilibirum_temp, timeStamps,
mean_samples_TC)


    %%% Calculate T_2 and its uncertainty using linear regression on
    %%% several regions of the data. Extrapolates the T_2 from the best fit
    %%% line of the equilibrium temperature profile of the calorimeter.

    %% use unweighted least-squares to find best fit line for the equilibrium line

    % create A from time values for each sample and the slope coefficient
    % m in T = mt + b
    numTimeStamps = length(timeStamps(index_equilibirum_temp:end));
    bCoefficients = ones(numTimeStamps, 1);
    mCoefficients = timeStamps(index_equilibirum_temp:end);

    A = cat(2, mCoefficients, bCoefficients); % make A from m and b coeffs.

    % find d matrix from A x = d matrix equation from the T values in the
    % range from t = 0 to the time when the sample was added
    d = mean_samples_TC(index_equilibirum_temp:end);

    % find least squares linear fit line
    P_leastSquares = inv(A' * A) * A' * d; %#ok<MINV>
    m = P_leastSquares(1);
    b = P_leastSquares(2);

    linear_fit = @(t) m*t + b; % linear fit anonymous function

    % find T_H
    time_hot_sample_added = timeStamps(index_hot_sample_added);
    T_L = mean_samples_TC(index_hot_sample_added);
    T_H = linear_fit(time_hot_sample_added);


    %% find least squares fit line between high and low T lines

    % create A from time values for each sample and the slope coefficient
    % m in T = mt + b
    index_middle_region_end = 925;
    numTimeStamps_2 =
length(timeStamps(index_hot_sample_added:index_middle_region_end));
    bCoefficients_2 = ones(numTimeStamps_2, 1);
    mCoefficients_2 = timeStamps(index_hot_sample_added:index_middle_region_end);

    A_2 = cat(2, mCoefficients_2, bCoefficients_2); % make A from m and b coeffs.

    % find d matrix from A x = d matrix equation from the T values in the
    % range from t = 0 to the time when the sample was added
    d_2 = mean_samples_TC(index_hot_sample_added:index_middle_region_end);

    % find least squares linear fit line
```

```
    P_leastSquares_2 = inv(A_2' * A_2) * A_2' * d_2; %#ok<MINV>
    m_2 = P_leastSquares_2(1);
    b_2 = P_leastSquares_2(2);

    linear_fit_2 = @(t) m_2*t + b_2; % linear fit anonymous function

    %% Finding T2 using linear model and extrapolating by finding t s.t. T(t) = T_H +
T_L / 2

    % developing midpoint value
    search_midpoint_T = (T_H + T_L) / 2;

    syms t
    T_2_time = double(solve(m_2*t + b_2 == search_midpoint_T, t));

    % plugging this value back into the linear fit for the high curve to
    % find T2 by extrapolating using t = t_avg_T_middle_line
    T_2 = linear_fit(T_2_time);

    %% Finding uncertainty in T_2 using extrapolation error

    %%% find uncertainty by first building Q to satisfy the following
    %%% formula:
    %%%           sigma_T2_extra = sqrt( [t_extra, 1] * Q * [t_extra; 1] )
    %%%
    %%%  such that Q = inv(A' * W * A)
    %%%
    %%%  where W is in this case:
    %%%
    %%%         sigma_T2 = sqrt( ((1 /(N - 2)) * SUM(1, N, (T_i - b - m*t_i)^2 )
    %%%         W = sigma_T * I (identity matrix)
    %%%
    %%%  Q (2x2 for linear regression):
    %%%
    %%%         Q =   |1 / sigma_m^2    1 / sigma_mb^2|
    %%%               |1 / sigma_mb^2    1 / sigma_b^2|

    % finding sigma_T2
    sum_discrepency = 0; % start with no uncertainty

    % summing discrepancies
    for i = 1:numTimeStamps
        t_i = A(i, 1);
        sum_discrepency = sum_discrepency + ( d(i) - linear_fit(t_i) )^2 ;
    end

    sigma_T = sqrt( ( 1 / (numTimeStamps - 2) ) * sum_discrepency );

    %%% Finding Error using matrix formulation
    I = eye(numTimeStamps); % create Identity matrix for W
    W = (1 / (sigma_T)^2) * I; % create weighting matrix

    % Finding Error using matrix formulation
    Q = inv(A' * W * A); % uncertainty matrix

    % finding uncertainty in T_2 at t = t_avg_T_middle_line
    % (uncertainty in extrapolation)

    sigma_T_2 = sqrt( [T_2_time, 1] * Q * [T_2_time; 1] );


    %% plot formatting
    LINEWIDTH = 2.5;
```

```
    FIT_STYLE = ':';
    xExtent = 360;


    hold on
    plot(timeStamps(index_equilibirum_temp - xExtent:end),
linear_fit(timeStamps(index_equilibirum_temp - xExtent:end)),  FIT_STYLE, 'LineWidth',
LINEWIDTH)
    plot(mCoefficients_2, linear_fit_2(mCoefficients_2), FIT_STYLE, 'LineWidth',
LINEWIDTH)

    hold off

end
```

7. "Project_1_calc_c_s.m"

```
%%% Purpose:
%%%            This function takes in the three calculated temperatures, along
%%%            with their uncertainties, and combines them with known
%%%            information about the calorimeter to calculate specific heat
%%%            of the sample measured in the experiment. This is done
%%%            through general thermodynamic analysis of the system:
%%%
%%%            calculates c_s based on the 1st law of Thermodynamics applied to
%%%            and isolated/stationary system s.t.
%%%
%%%             delta_U = 0
%%%             U_1 = U_2
%%%             (energy leaving sample = energy entering calorimeter)
%%%
%%%             m_s * c_s * (T_1 - T_2) = m_c * c_c * (T_2 - T_o)
%%%
%%%             c_s  = ( m_c * c_c * (T_2 - T_o) ) / ( m_s * (T_1 - T_2) )
%%%                  = specific heat of sample
%%%             where m_c = mass of calorimeter
%%%                   m_s = mass of sample
%%%                   c_c = specific heat of calorimeter
%%%             The uncertainty in the measurement of c_s is done using the
%%%             general error propagation formula on the eqn. above. By
%%%             finding the partial derivatives of c_s with respect to each
%%%             independent variable, plugging in their respective values,
%%%             multiplying them by their respective uncertainties, and
%%%             then summing in quadrature the final uncertainty in c_s is
%%%             obtained. All differentiation and substitution was carried
%%%             out by MATLAB, and is done in these same steps below.
%%%
%%% Inputs:
%%%         T_o - initial temperature of the calorimeter
%%%         sigma_T_o - uncertainty in measurement of T_o
%%%         T_1 - initial temperature of the sample
%%%         sigma_T_1 - uncertainty in measurement of T_1
%%%         T_2 - equilibrium temperature of the calorimeter and sample
%%%         sigma_T_2 - uncertainty in measurement of T_2
%%%
%%%
%%% Outputs:  T_2 - equilibrium temperature of the calorimeter and sample
%%%           sigma_T_2 - uncertainty in the measurement of T_2
%%%           search_midpoint_T - avg of T_H and T_L
%%%
%%% Assumptions:
```

```
%%%             - can use eqn. 8-15 in book to find uncertainty (errors are
%%%             all independent of each other and can be summed in
%%%             quadrature).
%%%
%%%             - data actually fits a linear relationship with time, and
%%%             as such can be fitted with a line of best fit by least
%%%             square linear regression.
%%%
%%%             - can use the extrapolation error formula as the actual
%%%             uncertainty of T_2, as this encapsulates all of the
%%%             needed sources of error in its measurement
%%%
%%%             - no uncertainty in the specific heat of calorimeter
%%%
%%% Author ID:  0dc91b091fd8
%%% Date Created:   10/15/2016
%%% Date Modified:  10/21/2016


%%
function [c_s, sigma_c_s] = Project_1_calc_c_s(T_o, sigma_T_o, T_1, sigma_T_1, T_2,
sigma_T_2)

    %% defining masses (in g) and spec. heat of calorimeter (in cal / g / C))

    % m_c = mass of calorimeter
    m_c = 313.50;
    sigma_m_c = 0.05;

    % c_c = specific heat of calorimeter
    c_c = 0.214;
    sigma_c_c = 0;

    %  m_s = mass of sample
    m_s = 91.75;
    sigma_m_s = 0.05;

    %% Calculating c_s
    c_s  = ( m_c * c_c * (T_2 - T_o) ) / ( m_s * (T_1 - T_2) );

    % converting to SI (j / kg / K)
    c_s = c_s * 4186.8;

    % convert to j / g / C
    c_s = c_s / 1000;


    %% finding uncertainty in measurement of c_s

    % find partial derivatives of eqn. for c_s for use in general error
    % propagation formula
    syms mc cc ms To T1 T2
    q = ( mc .* cc .* (T2 - To) ) ./ ( ms .* (T1 - T2) ); % q = c_s

    % taking partials
    dq_dmc = diff(q, mc);
    dq_dcc = diff(q, cc);
    dq_dms = diff(q, ms);
    dq_dTo = diff(q, To);
    dq_dT1 = diff(q, T1);
    dq_dT2 = diff(q, T2);

    % evaluating them at m_c, c_c, m_s, T_o, T_1, T_2
```

```
    sym_array = {mc cc ms To T1 T2};
    exact_array = {m_c, c_c, m_s, T_o, T_1, T_2};

    dq_dmc_exact = double( subs(dq_dmc, sym_array, exact_array) );
    dq_dcc_exact = double( subs(dq_dcc, sym_array, exact_array) );
    dq_dms_exact = double( subs(dq_dms, sym_array, exact_array) );
    dq_dTo_exact = double( subs(dq_dTo, sym_array, exact_array) );
    dq_dT1_exact = double( subs(dq_dT1, sym_array, exact_array) );
    dq_dT2_exact = double( subs(dq_dT2, sym_array, exact_array) );

    % calculate error terms to be added in quadrature
    dq_dmc_sigma_m_c = dq_dmc_exact * sigma_m_c;
    dq_dcc_sigma_c_c = dq_dcc_exact * sigma_c_c;
    dq_dms_sigma_m_s = dq_dms_exact * sigma_m_s;
    dq_dTo_sigma_T_o = dq_dTo_exact * sigma_T_o;
    dq_dT1_sigma_T_1 = dq_dT1_exact * sigma_T_1;
    dq_dT2_sigma_T_2 = dq_dT2_exact * sigma_T_2;

    % use general error formula to sum all of the above errors in
    % quadrature
    sigma_c_s = sqrt( dq_dmc_sigma_m_c^2 + dq_dcc_sigma_c_c^2 + ...
                      dq_dms_sigma_m_s^2 + dq_dTo_sigma_T_o^2 + ...
                      dq_dT1_sigma_T_1^2 + dq_dT2_sigma_T_2^2 );

    % converting to SI (j / kg / K)
    sigma_c_s = sigma_c_s * 4186.8;

    % convert to j / g / C
    sigma_c_s = sigma_c_s / 1000;

end
```

8. "Project_1_formatPlots.m"

```
%%% Purpose:
%%%         This function takes the processed data and formats the basic
%%%         plots produced in the other function. Adds labeling for the
%%%         relevant data points calculated from the calorimeter data.
%%%
%%% Inputs: T_o - initial temperature of the calorimeter
%%%         T_1 - initial temperature of the sample
%%%         T_2 - equilibrium temperature of the calorimeter and sample
%%%         T_2_time - time at which T_2 occurs on the extrapolated best
%%%                    fit line for the equilibrium state of system
%%%         TC4 - array containing the temperature profile of the boiling
%%%               water
%%%         search_midpoint_T - average temperature of T_H and T_L as
%%%                             defined in the lab document procedure
%%%         index_hot_sample_added - array index for time / temp when
%%%                                  sample is introduced to calorimeter
%%%         timeStamps - array of time stamps for each measurement taken
%%%
%%% Outputs: none - formats existing calorimeter plot and creates and
%%%                 formats plot of the boiling water profile
%%%
%%% Assumptions:
%%%             - none made here - just plotting
%%%
%%% Author ID:  0dc91b091fd8
%%% Date Created:   10/15/2016
%%% Date Modified:  10/21/2016

%%
```

```
function Project_1_formatPlots(T_o, T_1, T_2, T_2_time, TC4, search_midpoint_T, ...
                               index_hot_sample_added, timeStamps)

    %% plot for calorimeter TCs

    LINEWIDTH = 2.5;
    FONTSIZE = 22;
    LEGEND_STRING = {sprintf('Experimental Data'), sprintf('Regression Line for
finding $T_{o}$'),...
                     sprintf('Regression Line for finding $T_{2}$'),...
                     sprintf('Regression Line for finding $\\frac{T_{H} \\, + \\,
T_{L}}{2}$')};
    LEGEND_LOCATION = 'southeast';
    X_LABEL_NAME = sprintf('time');
    X_LABEL_UNITS = ' (s)';
    X_LABEL_STRING = cat(2, X_LABEL_NAME, X_LABEL_UNITS);
    Y_LABEL_NAME = sprintf('Temperature');
    Y_LABEL_UNITS = sprintf( ' %cC', char(176));
    Y_LABEL_STRING = cat(2, Y_LABEL_NAME, Y_LABEL_UNITS);
    TITLE_STRING = sprintf('%s vs. %s (calorimeter temp. profile)', Y_LABEL_NAME,
X_LABEL_NAME);

    timeSampleAdded = timeStamps(index_hot_sample_added);

    T_o_string = sprintf('$T_{o} \\, (T_{L})$');
    T_2_string = sprintf('$T_{2} \\, (T_{H})$');
    T_TH_TL_string = sprintf('$\\frac{T_{H} \\, + \\, T_{L}}{2}$');

    hold on

    % plotting points and graph formatting
    plot(timeSampleAdded, T_o, 'gs', 'MarkerFaceColor', ...
        [0.5, 0.5, 0.5], 'MarkerSize', 11, 'LineWidth', LINEWIDTH) % T_o

    plot(T_2_time, search_midpoint_T, 'ks', 'MarkerFaceColor', ...
        [0.5, 0.5, 0.5], 'MarkerSize', 11, 'LineWidth', LINEWIDTH) % search_midpoint_T

    plot(T_2_time, T_2, 'ms', 'MarkerFaceColor', ...
        [0.5, 0.5, 0.5], 'MarkerSize', 11, 'LineWidth', LINEWIDTH) % T_2

    % labeling points on graph
    text(timeSampleAdded * 0.95, T_o * 0.98, T_o_string, 'FontSize', FONTSIZE, ...
         'interpreter', 'latex'); % T_o

    text(T_2_time * 0.85, search_midpoint_T, T_TH_TL_string, ...
        'FontSize', FONTSIZE, 'interpreter', 'latex'); % search_midpoint_T

    text(T_2_time * 0.92, T_2 * 0.99, T_2_string, 'FontSize', FONTSIZE, ...
         'interpreter', 'latex'); % T_2

    hold off

    legend(LEGEND_STRING, 'location', LEGEND_LOCATION, 'interpreter', 'latex')
    xlabel(X_LABEL_STRING, 'interpreter', 'default')
    ylabel(Y_LABEL_STRING, 'interpreter', 'default')
    title(TITLE_STRING)


    %% plot for boiling water TC

    LINEWIDTH = 2.5;
    FONTSIZE = 22;
    MARKERSIZE = 2;
```

```
    LEGEND_STRING = {sprintf('Expiremental Data')};
    LEGEND_LOCATION = 'southeast';
    X_LABEL_NAME = sprintf('time');
    X_LABEL_UNITS = ' (s)';
    X_LABEL_STRING = cat(2, X_LABEL_NAME, X_LABEL_UNITS);
    Y_LABEL_NAME = sprintf('Temperature');
    Y_LABEL_UNITS = sprintf( ' %cC', char(176));
    Y_LABEL_STRING = cat(2, Y_LABEL_NAME, Y_LABEL_UNITS);
    TITLE_STRING = sprintf('%s vs. %s (boiling water temp. profile)', Y_LABEL_NAME,
X_LABEL_NAME);
    X_LOW = 0;
    X_HIGH = 1850;

    T_1_string = sprintf('$T_{1}$');

    hFig = figure(2);
    set(gca,'FontSize', FONTSIZE)
    set(hFig, 'Position', [100 100 1600 900])

    xlim([X_LOW, X_HIGH])

    hold on

    plot(timeStamps, TC4, 'o', 'MarkerSize', MARKERSIZE) % expiremental data
    plot(timeSampleAdded, T_1, 'rs', 'MarkerFaceColor', [0.5, 0.5, 0.5], ... % T_1
        'MarkerSize', 11, 'LineWidth', LINEWIDTH)

    text(timeSampleAdded * 0.93, T_1 * 0.98, T_1_string, 'FontSize', FONTSIZE, ...
         'interpreter', 'latex'); % T_1

    hold off

    legend(LEGEND_STRING, 'location', LEGEND_LOCATION, 'interpreter', 'latex')
    xlabel(X_LABEL_STRING, 'interpreter', 'default')
    ylabel(Y_LABEL_STRING, 'interpreter', 'default')
    title(TITLE_STRING)

end
```

References

The following pages are intended to provide examples of the different reference types, as used in the AIAA Style Guide. When using the Word version of this template to enter references, select the "references" style from the drop-down style menu to automatically format your references. If you are using a print or PDF version of this document, all references should be in 9-point font, with reference numbers inserted in superscript immediately before the corresponding reference. You are not required to indicate the type of reference; different types are shown here for illustrative purposes only.

Periodicals

[1]Vatistas, G. H., Lin, S., and Kwok, C. K., "Reverse Flow Radius in Vortex Chambers," *AIAA Journal*, Vol. 24, No. 11, 1986, pp. 1872, 1873.

[2]Dornheim, M. A., "Planetary Flight Surge Faces Budget Realities," *Aviation Week and Space Technology*, Vol. 145, No. 24, 9 Dec. 1996, pp. 44-46.

[3]Terster, W., "NASA Considers Switch to Delta 2," *Space News*, Vol. 8, No. 2, 13-19 Jan. 1997, pp., 1, 18.

All of the preceding information is required. The journal issue number ("No. 11" in Ref. 1) is preferred, but the month (Nov.) can be substituted if the issue number is not available. Use the complete date for daily and weekly publications. Transactions follow the same style as other journals; if punctuation is necessary, use a colon to separate the transactions title from the journal title.

Books

[4]Peyret, R., and Taylor, T. D., *Computational Methods in Fluid Flow*, 2nd ed., Springer-Verlag, New York, 1983, Chaps. 7, 14.

[5]Oates, G. C. (ed.), *Aerothermodynamics of Gas Turbine and Rocket Propulsion*, AIAA Education Series, AIAA, New York, 1984, pp. 19, 136.

[6]Volpe, R., "Techniques for Collision Prevention, Impact Stability, and Force Control by Space Manipulators," *Teleoperation and Robotics in Space*, edited by S. B. Skaar and C. F. Ruoff, Progress in Astronautics and Aeronautics, AIAA, Washington, DC, 1994, pp. 175-212.

Publisher, place, and date of publication are required for all books. No state or country is required for major cities: New York, London, Moscow, etc. A differentiation must always be made between Cambridge, MA, and Cambridge, England, UK. Note that series titles are in roman type.

Proceedings

[7]Thompson, C. M., "Spacecraft Thermal Control, Design, and Operation," *AIAA Guidance, Navigation, and Control Conference*, CP849, Vol. 1, AIAA, Washington, DC, 1989, pp. 103-115

[8]Chi, Y., (ed.), *Fluid Mechanics Proceedings*, SP-255, NASA, 1993.

[9]Morris, J. D. "Convective Heat Transfer in Radially Rotating Ducts," *Proceedings of the Annual Heat Transfer Conference*, edited by B. Corbell, Vol. 1, Inst. Of Mechanical Engineering, New York, 1992, pp. 227-234.

At a minimum, proceedings must have the same information as other book references: paper (chapter) and volume title, name and location of publisher, editor (if applicable), and pages or chapters cited. Do not include paper numbers in proceedings references, and delete the conference location so that it is not confused with the publisher's location (which is mandatory, except for government agencies). Frequently, CP or SP numbers (Conference Proceedings or Symposium Proceedings numbers) are also given. These elements are not necessary, but when provided, their places should be as shown in the preceding examples.

Reports, Theses, and Individual Papers

[10]Chapman, G. T., and Tobak, M., "Nonlinear Problems in Flight Dynamics," NASA TM-85940, 1984.

[11]Steger, J. L., Jr., Nietubicz, C. J., and Heavey, J. E., "A General Curvilinear Grid Generation Program for Projectile Configurations," U.S. Army Ballistic Research Lab., Rept. ARBRL-MR03142, Aberdeen Proving Ground, MD, Oct. 1981.

[12]Tseng, K., "Nonlinear Green's Function Method for Transonic Potential Flow," Ph.D. Dissertation, Aeronautics and Astronautics Dept., Boston Univ., Cambridge, MA, 1983.

Government agency reports do not require locations. For reports such as NASA TM-85940, neither insert nor delete dashes; leave them as provided by the author. Place of publication *should* be given, although it is not mandatory, for military and company reports. Always include a city and state for universities. Papers need only the name of the sponsor; neither the sponsor's location nor the conference name and location are required. *Do not confuse proceedings references with conference papers*.

Electronic Publications

CD-ROM publications and regularly issued, dated electronic journals are permitted as references. Archived data sets also may be referenced as long as the material is openly accessible and the repository is committed to archiving the data indefinitely. References to electronic data available only from personal Web sites or commercial, academic, or government ones where there is no commitment to archiving the data are not permitted (see Private Communications and Web sites).

[13]Richard, J. C., and Fralick, G. C., "Use of Drag Probe in Supersonic Flow," *AIAA Meeting Papers on Disc* [CD-ROM], Vol. 1, No. 2, AIAA, Reston, VA, 1996.

[14]Atkins, C. P., and Scantelbury, J. D., "The Activity Coefficient of Sodium Chloride in a Simulated Pore Solution Environment," *Journal of Corrosion Science and Engineering* [online journal], Vol. 1, No. 1, Paper 2, URL: http://www.cp/umist.ac.uk/JCSE/vol1/vol1.html [cited 13 April 1998].

[15]Vickers, A., "10-110 mm/hr Hypodermic Gravity Design A," *Rainfall Simulation Database* [online database], URL: http://www.geog.le.ac.uk/bgrg/lab.htm [cited 15 March 1998].

Always include the citation date for online references. Break Web site addresses after punctuation, and do not hyphenate at line breaks.

Computer Software

[16]TAPP, Thermochemical and Physical Properties, Software Package, Ver. 1.0, E. S. Microware, Hamilton, OH, 1992.

Include a version number and the company name and location of software packages.

Patents

Patents appear infrequently. Be sure to include the patent number and date.

[17]Scherrer, R., Overholster, D., and Watson, K., Lockheed Corp., Burbank, CA, U.S. Patent Application for a "Vehicle," Docket No. P-01-1532, filed 11 Feb. 1979.

Private Communications and Web Sites

References to private communications and personal Web site addresses are generally not permitted. Private communications can be defined as privately held unpublished letters or notes or conversations between an author and one or more individuals. They *may* be cited as references in some case studies, but only with permission of the AIAA staff. Depending on the circumstances, private communications and Web site addresses may be incorporated into the main text of a manuscript or may appear in footnotes.

Unpublished Papers and Books

Unpublished works can be used as references as long as they are being considered for publication or can be located by the reader (such as papers that are part of an archival collection). If a journal paper or a book is being considered for publication choose the format that reflects the status of the work (depending upon whether it has been accepted for publication):

[18]Doe, J., "Title of Paper," Conference Name, Publisher's name and location (submitted for publication)
[19]Doe, J., "Title of Paper," *Name of Journal* (to be published).
[20]Doe, J., "Title of Chapter," *Name of Book*, edited by… Publisher's name and location (to be published).
[21]Doe, J., "Title of Work," Name of Archive, Univ. (or organization) Name, City, State, Year (unpublished).

Unpublished works in an archive *must* include the name of the archive and the name and location of the university or other organization where the archive is held. Also include any cataloging information that may be provided. Always query for an update if a work is about to be published.