DESIGN AND DEVELOPMENT OF A MULTI-COLOR SOOT EMISSION DIAGNOSTICS TO MEASURE SOOT TEMPERATURE AND CONCENTRATION

By

Ashwinraj Gnanavel,

Master of Engineering, Ryerson University (2019)

A project presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Engineering in the program of

Aerospace Engineering

Toronto, Ontario, Canada, 2020

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A PROJECT

I hereby declare that I am the sole author of this project. This is a true copy of the project, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this project to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this project by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my project may be made electronically available to the public.

ABSTRACT

In this paper, temperature of the soot particles from the flame was determined using line of sight attenuation setup. The emission from the soot particles of the flame will be filtered using a three color pyrometry, in which three slits of different wavelengths will be placed in front of the AP-3200T-USB camera. The author did not use spectrometer to filter the wavelengths because, a spectrometer would give a spectrum consisting of 15 different colors, which is not required in this experiment, since calculation of temperature of the soot only requires two colors to be filtered from the soot. Each wavelength corresponds to RED, BLUE and GREEN color respectively. After the soot emission images are captured in all the three wavelengths, three images will be obtained from the camera. For reading these images, MATLAB code was used, and the pixel intensity values were read from which temperature could be calculated.

ACKNOWLEDGEMENT

Presentation inspiration and motivation have always played a key role in the success of any venture.

I would like to express my special appreciation and thanks to my advisor Professor Dr. Emre Karatas, you have been a very great mentor for me. I would like to thank you for encouraging this research work. Your advice on my research work have been invaluable. I would also like to thank my committee members, professor Dr. JeffreyYokota, professor Dr. Seyed Hashemi and to my advisor Professor Dr. Emre Karatas who is also a part of this. I also thank my program administrator Leah Rogan for her support.

I am also immensely obliged to my friends for their elevating inspiration, encouraging guidance and kind of supervision in the completion of my project.

Last, but not the least, my parents are also an important inspiration for me. So, with due regards, I express my gratitude to them.

TABLE OF CONTENTS **PAGE NO.**

LIST OF FIGURES **PAGE NO.**

1. INTRODUCTION:

A flame is a visible, gaseous part of a fire. A highly exothermic reaction taking place in a thin zone. Before studying the concept behind soot formation, firstly, the origin of soot must be understood, which is nothing but flame **[1]**. Some flames are considered plasma when they are very hot due to the presence of ionized gas components. The type of fuel that is involved in the combustion influences the temperature and color of the flame. The fuel molecules of the candle wax vaporize due to applied heat. Therefore, they can readily react with oxygen in the atmosphere and this gives a consistent flame due to some heat escape in the exothermic reaction. The vaporized molecules from the fuel decomposes due to high temperature forming many different products and free radicals from incomplete combustion, after which reaction takes place between these products along with the oxidizer which is also a part of this reaction. Then the energy in the flame tends to excite the electrons in intermediates such as methylidyne radical and diatomic carbon. This leads to the emission of visible light as those substances' releases excess energy. The temperature of the flame is directly proportional to the average energy of the electromagnetic radiation emitted by the flame. The soot particles are present in the flame about which we will be looking into the next session. In this report, the author mainly focusses on the programming part of the spectral soot emission with a MATLAB code. However, to understand the theoretical concept behind the programming code, discussion about the soot formation and line of sight attenuation setup in brief is mandatory. **A program was previously developed for LOSA setup which was then initially tested by first running in .spe format and then in .tiff format to make sure both the file formats gives the same results of soot volume fraction in this project.**

2. LITERATURE SURVEY:

2.1 SOOT FORMATION AND ITS MECHANISM:

Soot is a mass of impure carbon particles resulting from the incomplete combustion of hydrocarbons **[2].** In simple words, when hydrocarbon fuels undergo diffusive combustion, soot and fullerenes are produced as by-products. Non optical burning conditions is indicated by the presence of soot. Soot is a product of the gas-phase combustion process. Soot is radiative in

nature which leads to the distribution of the thermal energy that gets created during combustion in unwanted fires followed by flame propagation and fire sustainment. When the soot particles get oxidized within the flame, then that is where soot particles become undesirable in furnaces and boilers. If any device emits soot particles to the outside atmosphere, then it leads to environmental pollution which is against living beings. The size of soot particles affects the level of risk in a way. For instance, ultrafine soot particles can enter the human body via the artery network and get transmitted to all parts of the body and might produce free radicals and damage the DNA strands. Ultrafine soot particles can also cause lung and heart failure.

Pyrolysis, which is nothing but the thermal decomposition of materials at elevated temperatures in an inert atmosphere, is the sole reason to produce soot through many different forms.

They include soot from coal burning, internal-combustion engines, power plant boilers, forest fires, house fires, hog-fuel boilers, central steam-heat boilers and furnaces. However, our objective is to find the temperature of the soot from flame. Therefore, in this project, the soot emission from flame is considered.

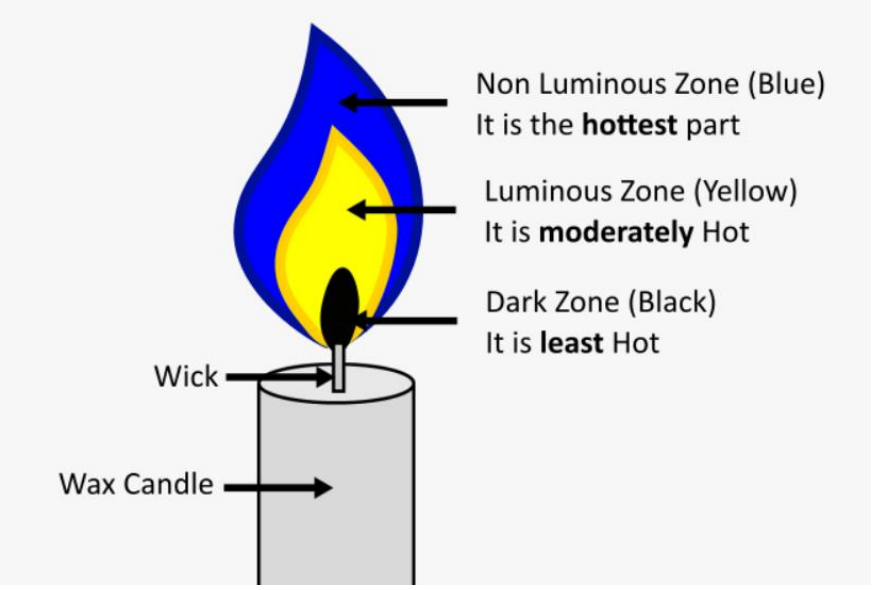

Figure 1: Zones in a candle flame [3]

It is soot which is the only reason for the luminescence in the flame. Most of the proposals say that soot originates inside the flame from polycyclic aromatic hydrocarbon (PAH) **[4]**. Then the

process of inception happens which is nothing but the growth of PAH into the first solid soot particles. The precursor produced from fuel structure highly affects the final soot yield. Initially the fuel decomposes chemically leading to the formation of aromatic rings. This decomposition can occur by two ways. One is through H-atom abstraction and the second is fission. Since we are going to use a simple fuel initially, then radical concentration would be less, and temperature will be high. Therefore, fission reaction will happen. After the formation of aromatic rings, these rings would start to grow into high PAH known as coagulation. Now, as every descendent PAH is thermodynamically more stable than its predecessor, this growth process of PAH would happen by recursive mechanism **[5]**. After coagulation, the process of soot inception starts to occur by accumulation of PAH clusters into solid soot particles. Following this, the final process of agglomeration occurs, where the surface reaction tends to grow the solid soot particles **[6]**. Agglomeration process provides most of the final mass.

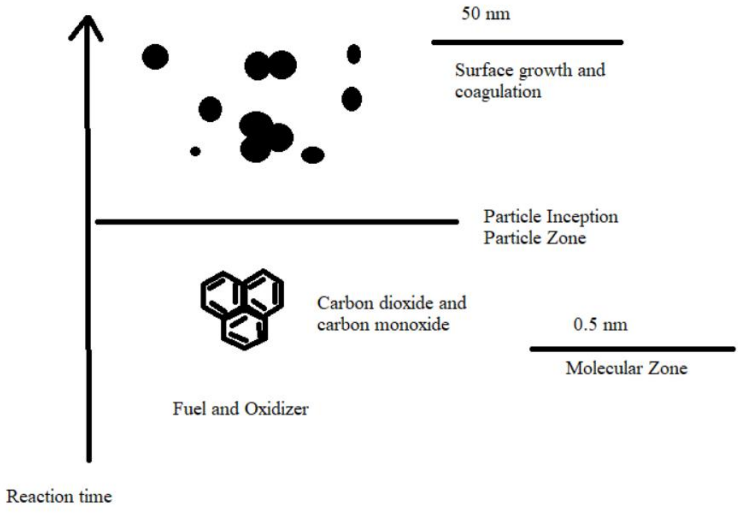

Figure 2: Mechanism of soot formation

Agglomeration is the accumulation of smaller particles into a cluster. Therefore, we have discussed a brief explanation of soot formation mechanism and let's discuss the factors affecting soot formation in brief followed by line of sight attenuation setup.

2.2 FACTORS AFFECTING SOOT FORMATION:

The environment around the flame highly influences the rate of soot formation and vice versa. However, each factor is considered as an independent variable. The major environmental factors consist of temperature and pressure. Apart from the environmental factors, an additive to the fuel that gives flame would also affect the rate of soot formation. These additives could affect the soot formation in various ways **[7]**. The carbon composition in the fuel could be altered known as the dilution effect, the temperature in the region just around the soot could change known as the thermal effect or there might be a slight change in the chemical reaction known as chemical effect. Burner nozzle material is also one of the factors that affects soot formation.

1. Fuel structure:

Different fuels have different structures, i.e., different number of carbon atoms. When the number of carbon atoms and complexity is more in the fuel, the tendency of the fuel to produce the amount of soot increases. Polyaromatics has the highest number of carbon atoms and the most complex structure followed by monoaromatics, alkynes, alkenes and then alkanes. Owing to the fact that complex fuel structure produces a relatively larger range of intermediate molecules, they also produce more soot. Moreover, some researchers also found that when the height of the flame increases, the amount of soot formation increases. Shorter flames were found to emit less soot.

2. Temperature:

Prior to studying the effect of temperature on soot formation, fuels were diluted with inert diluents **[8]**. The temperature of the fuel is directly proportional to the amount of soot formation. This is because of the increase in the fuel mole fraction when the temperature of the fuel is high.

3. Pressure:

Unlike other factors, the soot volume fraction is very sensitive to pressure. In other words, it could be said that, even for a small change in pressure, the soot volume fraction varies drastically. The amount of soot generally increases with increase in pressure. One of the researchers also proved that maximum percentage of fuel's carbon got converted into fuel with increasing pressure. However, when the pressure is approaching to the value of 40 atmospheric pressure, the change in soot volume fraction becomes less sensitive. This is because, at this level of pressure, it was observed that there was some heat loss from the flame thereby losing some

soot too. Some other study shows that when the pressure was increased to 60 atm, the height of the flame was the same for all the pressures. But, the width of the flame was found to be decreasing. When the flame is more thin or narrow, the heat transfer increases, and closely packed combustion gases could be observed. This leads to pyrolysis reaction which is the basic reason of the soot volume formation being less sensitive to when the pressure is high.

4. Additives:

In this section, the influence of chemicals (additives) to the fuel over the soot formation will be discussed briefly. Some of the soot suppressors are inert gases, steam, carbon dioxide, sulphur compounds and metals. The heat capacity and the diffusivity of the fuel decides their suppression efficiency. Preventing soot formation to some level by using additives can be due to any combination from thermal, chemical and dilution. But, the promotion of soot formation is because of the nucleation of soot formation after adding additives. The chemical substances which promote soot formation are halogens and their compounds which increases the decomposition of fuel when added even in small amounts.

3. EXPERIMENTAL SETUP:

3.1 LINE OF SIGHT ATTENUATION:

Before getting into the working principle of this setup, let us first discuss the theory behind it.

Theory:

The line of sight attenuation (LOSA) is technique which is non-intrusive and optical diagnostic used for measuring the soot volume fraction which depends on the line of sight extinction of light by the soot particles in a flame. In this technique, a noise level of maximum upto 0.0007 in extinction and a spatial resolution of 30-40 μm wavelengths for imaging can be achieved **[9]**. In LOSA method, calibration is not required for the soot measurement.

LOSA layout:

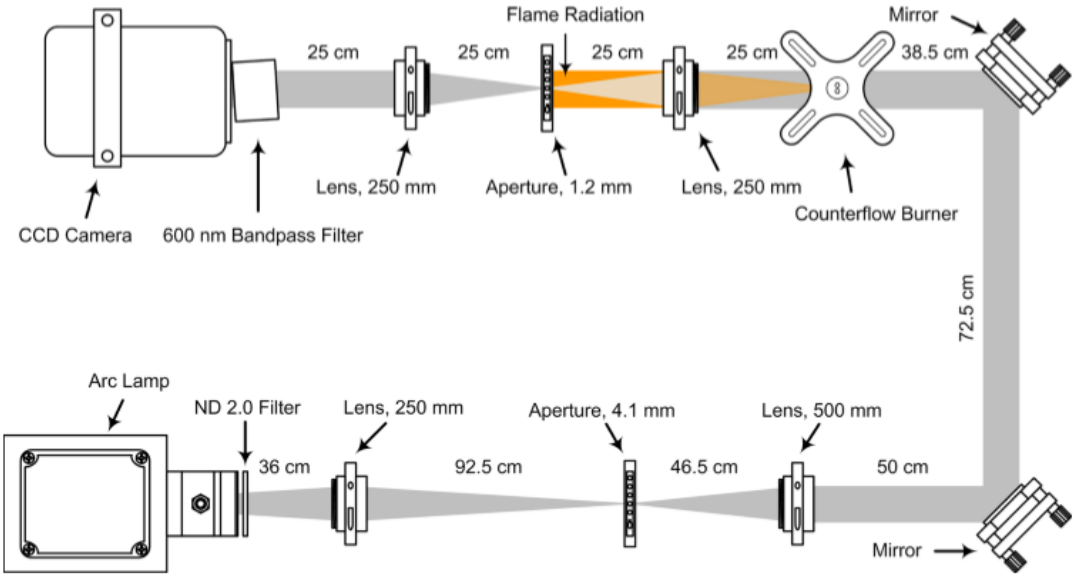

Figure 3: Line of sight attenuation apparatus [9]

The source of light is a 100 W mercury arc system which is used to produce the broadband light that is required for this experiment. This arc emits at wavelengths from UV (200 nm) to mid IR (2400 nm) and sometimes above this range. This light is made to pass through a filter of neutral density of 2 that gives 1 % transmittance. The filter is attached to the arc lamp (source) housing. The filter is used to prevent the oversaturation of the CCD camera as shown in the diagram below. The light is first focussed onto a pre-flame aperture with a 4.1 mm opening. This is done with the help of a collecting lens of 250 mm focal length by converging the parallel beam of light to a point in the aperture as shown below. Then this collimated light passes through the counter-flow burner. This LOSA system has more linear distance. Therefore, in order to make this system compact, two mirrors are placed. Then the reflected light from the second mirror is made to pass through an achromatic lens of 500 nm which collimates the light. The post flame light is made to pass through a 250 mm achromatic lens. This lens is placed at a distance from the burner equal to the focal length. The light that is diverging from the post flame aperture is collimated again by the 250 mm achromatic lens. Then the light is filtered by a 10 nm bandpass filter centered at 600 nm. This setup is used to find the transmittance which is nothing but the ratio of the amount of light that gets transmitted through the flame to the total amount of collimated light incident on the flame. Consequently, the soot volume fraction is determined.

3.2 SPECTRAL SOOT EMISSION:

In this project, spectral soot emission setup **[10]** will be used to achieve our final objective which is to calculate the temperature of the soot particles. This will be achieved by reading the pixel intensity values of three different wavelength flames (red, blue and green). To calculate the temperature, two colors (wavelengths) are more than enough, but in order to get more accurate values, we are considering three wavelength ranges. Now, let's investigate the experimental setup of the spectral soot emission.

3.2.1 EXPERIMENTAL APPARATUS:

The source of light is a flame produced by the combustion of ethylene. Initially ethylene reacts with air (oxygen) which produces flame containing soot through the mechanism that was discussed earlier in the section. As shown in the diagram below, the radiations are emitted from the flame soot, which is made to pass through an aperture that is adjustable and then through an achromatic lens pair. An anti-reflective coating is provided to the surface of the lens in order to reduce glare and reflections that are in the wavelength range of 650 mm to 1050 mm. A couple of changes are made to the actual experimental setup different from the diagram below that is to be conducted which will be discussed soon in this section. The lens is positioned exactly in between (equidistantly) the burner and the three slits. This lens produces lize size images. The spectrometer will be replaced by three slits. After the lens, the soot particles are directed and focussed onto the three slits one by one. We will be using three wavelength color (red, blue and green) filters (three slits) instead of a bandpass filter as we do not require the whole spectrum and we just need to read and display images of the three color wavelengths. Each color has its own range of wavelength. Red color filter allows light rays that has wavelength ranging from 620-750 nm. Blue color filter allows light rays that has wavelength ranging from 450-495 nm. Green color filter allows light rays that has wavelength ranging from 495 to 570 nm. There is one entrance slit placed before the soot particles pass onto the three slit and an exit slit after the three slits. This entrance and exit slits limit the soot particles radiation light to 290 μm in height to 20 μm in width. This is done to achieve a better spatial resolution of the collected images from the AU3200-TUSB camera. In this experiment we will be using AP-3200T-USB camera instead of a CCD camera. We are doing this since CCD (charged couple device) camera is way more expensive than AP-3200T-USB camera. Each of this camera reads the image in various formats.

Therefore, the programming codes are modified as per the file format which will be explained in the upcoming sections. If we had used the spectrometer as shown in the diagram below, it divides the beam into discrete wavelengths as a spectrum. But in this experiment to achieve the objective of determining the temperature of the soot particles, we just need two different ranges of wavelengths (corresponding to two colors) and we consider three ranges of wavelengths or three different colors (red, blue and green) as discussed before to get more accurate result. So, three slits are used replacing the spectrometer along with replacing CCD camera by AP-3200T-USB camera thereby reducing the overall cost of the experiment. The AP-3200T-USB camera will capture the images of different colors (wavelengths) after the soot particle radiation exits from the exit slit.

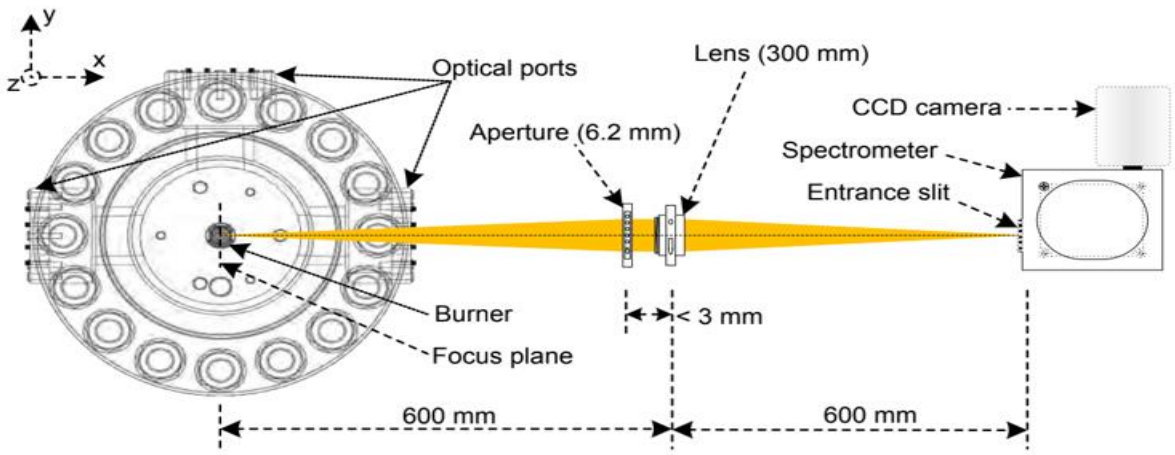

Figure 4: Schematic diagram of the spectral soot emission setup [10]

3.2.2 DESCRIPTION OF THE CAMERA:

The Apex AP-3200T-USB is a 3-CMOS prism-based industrial R-G-B area scan camera. It provides color fidelity and spatial precision than traditional Bayer color cameras **[11]**. The camera is built around the Pregius TM IMX265 1/1.8-inch sensor featuring 2064 X 1544 pixels and pixel sizes of 3.45 µm X 3.45 µm. The USB3 Vision interface offers excellent plug and play compatibility while delivering 3 X 3.2 megapixels at 38.3 frames/second.

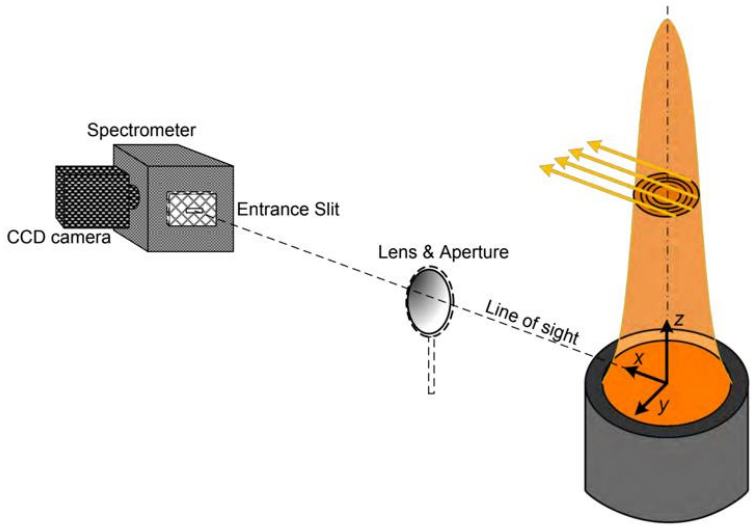

Figure 5: 3D view of Spectral Soot emission [10]

Figure 6: AP-3200T-USB camera used to replace CCD camera in the SSE setup [11]

The camera can be optionally ordered with the IR-cut filter removed (AP-3200T-USB-NF) for applications needing extended sensitivity in the red/NIR channel for applications that may need a combination of visible and NIR information. It is loaded with a range of advanced features and functions supporting the most challenging color imaging applications. Using a special prism-based imaging technique, the incoming light is separated into red, blue and green wavelengths, captured by three precisely aligned CMOS sensors. The result is better color accuracy and spatial precision than traditional color cameras using the Bayer mosaic technique. The AP-3200T-USB is built around three Sony Pregius IMX265 sensors. Combined with a USB3 Vision interface,

this machine vision camera can output as much as 3 x 3.2 megapixels at 38 frames/second in full resolution. The AP-3200T-USB provides great flexibility in on-board color space conversion including RGB to CIE-XYZ color space conversion.

4. MATLAB CODING PERFORMING LOSA SETUP:

The coding is almost the same for LOSA and SSE with a few changes, but the principle is the same. In this section, role of each segment of the program will be explained for reading tge images from line of sight attenuation (LOSA) and for reference, the main part of the MATLAB code is attached in the appendix A and a few user defined functions are attached in appendix B. The changes that have been made in the code are highlighted with yellow (refer to the appendices). The explanation behind each section of the code will be explained in this section. Then the soot volume fraction results were compared between .spe and .tiff format.

4.1 RUNNING IN SPE MODE:

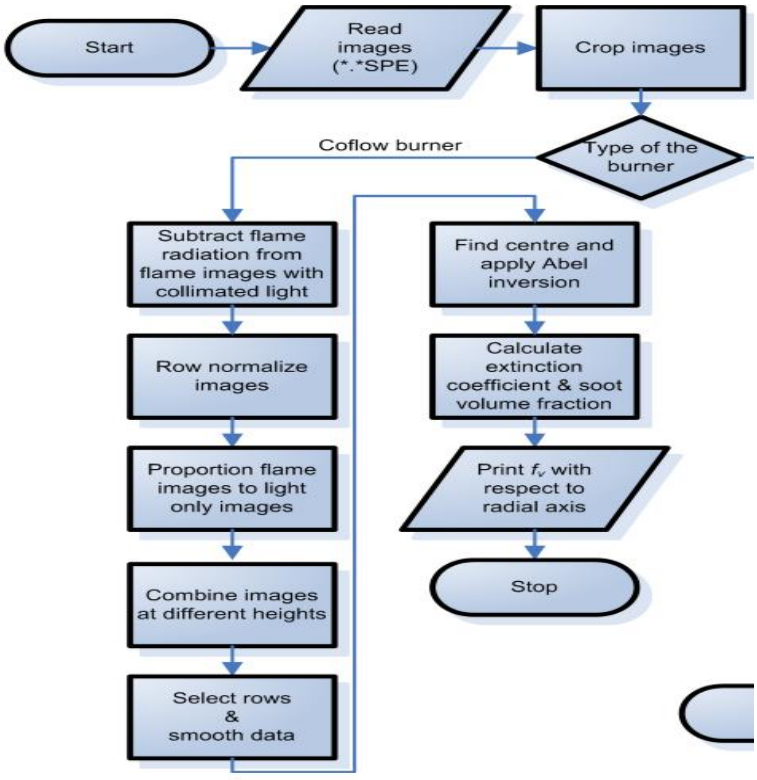

Figure 7: MATLAB coding process in a flowchart [10]

The author developed a flow chart using the paint software as shown above and explained each step in detail in this section. Firstly, the matlab code that was already developed was complied and checked for errors. The images of the flame that were captured using CCD camera from the LOSA setup were read into the matlab code as a .spe file format for various heights of the flame, starting from 0 mm from the burner tip to 60 mm with increments of 10mm. Initially the digital images were read using the Winspec/32 software and transformed into three-dimensional matrices. Spatially resolved images from the camera were indicated by the first two dimensions of the matrix. Each value in the matrix represents the intensity of the corresponding pixel in 16 bits. The order of the images is indicated by the third dimension of the matrix. The images are then averaged over the third dimension. This is what happens when reading the images in spectral soot emission too. The working of the program and process of the setup is the same between LOSA and SSE setup. The only difference is SSE theory is used to find the temperature of the soot particles and soot volume fraction which is to be done and LOSA setup was used to find the soot volume fraction which is already done. This is the reason we are considering LOSA setup to compare if the .spe and .tiff file images give the same result in order to make sure that .tiff images, when used in the SSE setup, will give the exact results of the pixel intensity values of the red, blue and green images (.tiff format) from the AP-3200T-USB camera. Now, before the images are read from LOSA setup in matlab, images were captured at different heights starting from 0 to 60 mm. The images were cropped in such a way that there is one translation between two adjacent heights (10 mm). Cropping was used to prevent the images from getting superimposed after combining the images at different heights to map the entire flame. The image was cropped so that the burner tip is not visible which provides a zero-reference point for the flame. Correspondingly, the flame only images were also cropped. Then the process of normalization was done where from the images with both flame and collimated light, the flame only images were subtracted. This difference gives the amount of light that passes through the flame at each height. After this, the resultant images were normalized. Normalization was done by considering lines that are parallel to the intensity gradients. Each array was divided by the value of the pixel with maximum intensity. Then transmissivity was then calculated which is nothing but the ratio of the amount of light that can pass through the flame to the overall incident light. In simple words, it is the ratio of image with both flame and collimated light to the light only images. Then using the catenation function, images for different

heights were joined to map the whole flame. As discussed before, the changes have been highlighted. Originally, the previous researchers used the **combine** function to map the entire flame which showed a few errors. Therefore, it was replaced by concatenate function. Then, rows of different flame heights were considered which had transmittance values in terms of pixel intensities which were smoothed using loess function. Then the process of **centre finding** was done using matlab. **Each process has its own user defined function (refer to appendix B).** For each height of the flame, the centre finding process was carried out. To each row, splines were fitted and average of the three values were calculated for each array. The half transmittance points on left and right sides of the image gave a basis for the centering algorithm. The centre point was the midpoint of these points. Then at each row, the image was divided into two from their centre point. Each pixel constitutes to 0.0129mm. A value of 0.5 was assigned to each empty column of the shorter half. A 3-point abel inversion technique was used to deconvolute the half images separately. A function was created to do the **abel inversion (refer to appendix B)**. In general, with respect to image analysis, abel transform is used to project an optically thin, axially symmetric emission function onto a plane, and the reverse abel transform (abel inversion) is used to calculate the emission function given a projection of that emission function. The tomographic transmittance images were inverted into radially resolved extinction coefficients from which the soot volume fraction values were calculated. Finally, the mean value of the soot volume fractions on the left and right side of the image is calculated and displayed as an output matrix. Now, upon running the program in spe mode, the following output (soot volume fraction) was obtained. Since it is a very huge matrix, the matrix in the form of excel has been attached below.

Book1.xlsx

4.2 CHANGES MADE BEFORE RUNNING IN TIFF (TAGGED IMAGE FILE FORMAT) MODE:

The images for all heights which were opened using Winspec/32 were converted into tagged image file format. For example, to read the image in spe format at 0mm, “f_0.SPE” was used while reading the same image in tiff format, “f_0.TIF” was used. Major changes were made in the get_TIF_raw_data2D() before reading the image in tiff mode and display their pixel intensity values. The corresponding matlab codes for both spe (get_SPE_raw_data2D()) and tiff

are displayed in the appendix B. However, all the other user defined functions were the same when opening the image in spe and tiff format. The changes that were made in the get_TIF_raw_data2D(), in the form of code are explained in detail in appendix B. While reading the image in spe, **fread(filename)** must be used whereas while reading the image in tiff, **imread(filename)** has to be used. After running the program in the tiff mode, the following soot volume results were obtained and upon comparing the values, the results between spe and tiff were exactly one and the same, indicating that .tiff format also gives correct and flawless results just like spe. Also, same result means .tiff format is going to give accurate results for spectral soot emission theory where we would be using the AP-3200T-USB camera which captures image in tiff format. The matrix in the form of excel (for tiff) has been attached below.

Book2.xlsx

4.3 DEVELOPING THE 3D CONTOUR PLOT:

A 3D contour plot was created to display the soot volume fraction value at each height from the tip and radial distance from the centreline of the flame. The code for developing this contour plot is displayed in appendix B. Radial distance is along the x-axis, height is along the y-axis and soot volume fraction is along the z-axis. Height increases by 10 mm (from the matrix table of soot volume fraction values, each row displays different height), radial distance increases by 0.1 from 0 to 6.6 mm, i.e., from the centreline to the left or right most point of the flame (from the matrix table of soot volume fraction values, each column displays different radial distance). After compiling the developed code, the following graph was displayed.

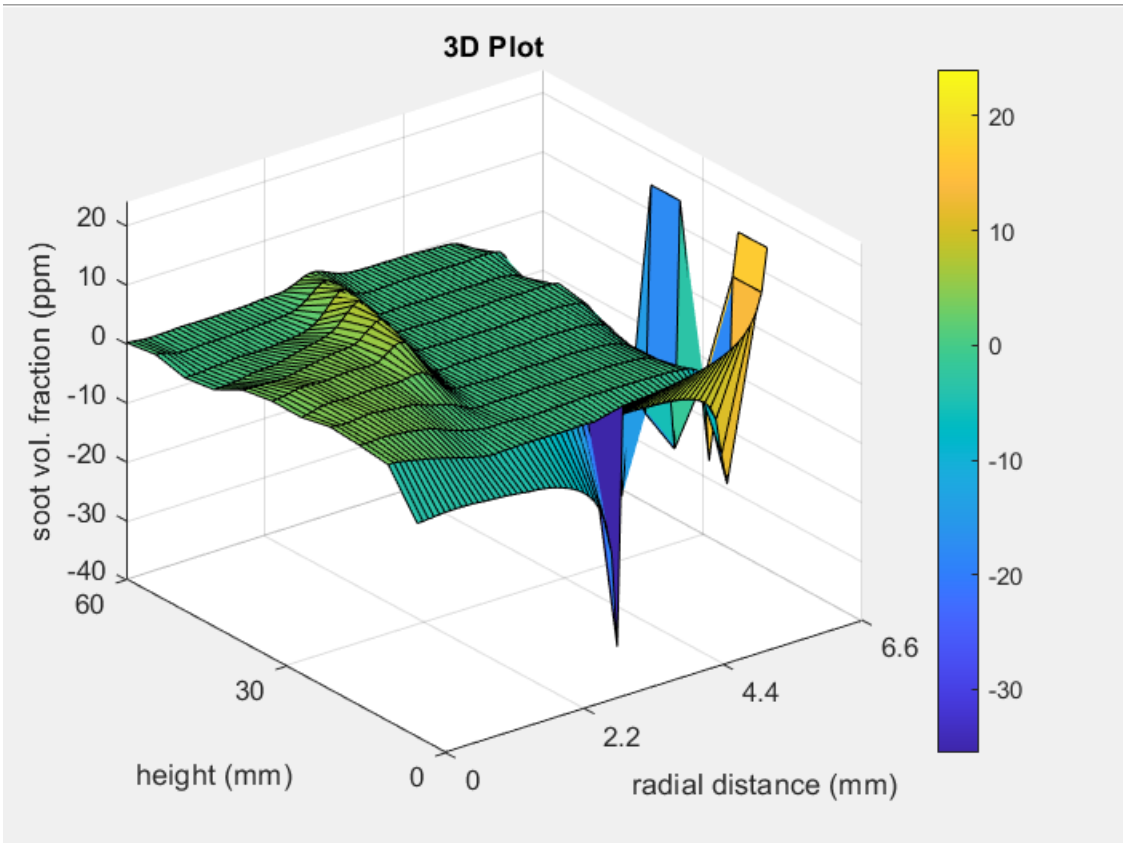

Figure 8: 3D contour plot for soot volume fraction for various heights and radial distance

Since the radial distance increases by 0.1 mm, the graph was plotted along radial distance for every 8 pixels once (8 pixels is almost equal to 0.1 mm and 1 pixel = 0.0129 mm).

5 CONCLUSION:

As of now, the conclusion cannot be drawn with respect to determining the temperature of the soot particles of the flame since the project is still on progress and only 60 percent of the project is completed due to time constraint. Developing a modified code for checking if the tagged image file format images gives correct results consumed a lot of time. Therefore, so far, it is proven that .tiff and .spe files gives exactly the same results conveying that AP-3200T-USB camera which is way cheaper when compared to CCD camera can be used for spectral soot emission setup to capture images with ranges of wavelength (red, blue and green). The activities that must be continued from here, in order to achieve the final objective of finding the temperature of soot particles, is discussed in the future scope below.

6 FUTURE SCOPE:

The spectral soot emission experimental setup must be done (devices required for this experiment have already been purchased). Then the flame images are captured in three different ranges of wavelength with the help of filters as explained already. After this, the modified code for the LOSA process is merged with a newly developed SSE code and make some alterations by removing the unnecessary parts of the main code and incorporating required codes for SSE. Almost, most of the code parts are the same for LOSA and SSE since the principle behind them is the same. Finally, the pixel intensities and soot volume fraction will be calculated (output of the new code) for the three images (red, blue and green). The temperature will be calculated using the physical formula that relates temperature and wavelength of the soot particles.

7 APPENDICES

7.1 APPENDIX A: Main program

```
clear all;
close all;
clc;

tic


%Collimated initial images without burner tip

Ci_temp = 'c_init_notip.TIF';
Ci = get_TIF_raw_data2D(Ci_temp);
clear Ci_temp;

Ci_d = double(Ci);
clear Ci;

Ci_avg = mean(Ci_d, 3);   % the 3 denotes that it is the images to be
% averaged, not rows or columns: Ci_d (1024, 1024, 30) array of 30 images
% with 1024 rows of intensities and 1024 columns of intensities (not sure
% which one is rows and which one is columns)

clear Ci_d;


% Collimated initial images (burner tip visible)

Cib_temp = 'c_init_tip.TIF';
```

```
Cib = get_TIF_raw_data2D(Cib_temp);
clear Cib_temp;

Cib_d = double(Cib);
clear Cib;

Cib_avg = mean(Cib_d, 3);

clear Cib_d;



% Collimated final images (no burner tip)

Cf_temp = 'c_fin_notip.TIF';
Cf = get_TIF_raw_data2D(Cf_temp);
clear Cf_temp;

Cf_d = double(Cf);
clear Cf;

Cf_avg = mean(Cf_d, 3);

clear Cf_d;

% Collimated final images (burner tip visible)

Cfb_temp = 'c_fin_tip.TIF';
Cfb = get_TIF_raw_data2D(Cfb_temp); //get_SPE_raw_data2D(Cfb_temp) in case of spe
clear Cfb_temp;
```

```
Cfb_d = double(Cfb);
clear Cfb;

Cfb_avg = mean(Cfb_d, 3);

clear Cfb_d;




%Flame images at 0 mm (with and without collimated light)

f_0_temp = 'f_0.TIF';    // f_0_temp= ‘f_0.SPE’ in case of opening the image in spe format//
                         //same format is repeated for all heights which have been highlighted
                         //below
f_0 = get_TIF_raw_data2D(f_0_temp);
clear f_0_temp;

f_0_d = double(f_0);
clear f_0;

f_0_avg = mean(f_0_d, 3);

clear f_0_d;
fc_0_temp = 'fc_0.TIF';
fc_0 = get_TIF_raw_data2D(fc_0_temp);
clear fc_0_temp;

fc_0_d = double(fc_0);
```

```
clear fc_0;

fc_0_avg = mean(fc_0_d, 3);

clear fc_0_d;


%Flame images at 10 mm

f_10_temp = 'f_10.TIF';
f_10 = get_TIF_raw_data2D(f_10_temp);
clear f_10_temp;

f_10_d = double(f_10);
clear f_10;

f_10_avg = mean(f_10_d, 3);
clear f_10_d;
fc_10_temp = 'fc_10.TIF';
fc_10 = get_TIF_raw_data2D(fc_10_temp);
clear fc_10_temp;
fc_10_d = double(fc_10);
clear fc_10;
fc_10_avg = mean(fc_10_d, 3);
clear fc_10_d;

%Flame images at 20 mm
f_20_temp = 'f_20.TIF';
f_20 = get_TIF_raw_data2D(f_20_temp);
clear f_20_temp;
```

```
f_20_d = double(f_20);
clear f_20;

f_20_avg = mean(f_20_d, 3);

clear f_20_d;



fc_20_temp = 'fc_20.TIF';
fc_20 = get_TIF_raw_data2D(fc_20_temp);
clear fc_20_temp;

fc_20_d = double(fc_20);
clear fc_20;

fc_20_avg = mean(fc_20_d, 3);

clear fc_20_d;


%Flame images at 30 mm

f_30_temp = 'f_30.TIF';
f_30 = get_TIF_raw_data2D(f_30_temp);
clear f_30_temp;

f_30_d = double(f_30);
clear f_30;
```

```
f_30_avg = mean(f_30_d, 3);

clear f_30_d;



fc_30_temp = 'fc_30.TIF';
fc_30 = get_TIF_raw_data2D(fc_30_temp);
clear fc_30_temp;

fc_30_d = double(fc_30);
clear fc_30;

fc_30_avg = mean(fc_30_d, 3);

clear fc_30_d;


%Flame images at 40 mm

f_40_temp = 'f_40.TIF';
f_40 = get_TIF_raw_data2D(f_40_temp);
clear f_40_temp;

f_40_d = double(f_40);
clear f_40;

f_40_avg = mean(f_40_d, 3);
```

```
clear f_40_d;
fc_40_temp = 'fc_40.TIF';
fc_40 = get_TIF_raw_data2D(fc_40_temp);
clear fc_40_temp;

fc_40_d = double(fc_40);
clear fc_40;

fc_40_avg = mean(fc_40_d, 3);

clear fc_40_d;


%Flame images at 50 mm

f_50_temp = 'f_50.TIF';
f_50 = get_TIF_raw_data2D(f_50_temp);
clear f_50_temp;

f_50_d = double(f_50);
clear f_50;

f_50_avg = mean(f_50_d, 3);

clear f_50_d;



fc_50_temp = 'fc_50.TIF';
fc_50 = get_TIF_raw_data2D(fc_50_temp);
```

```
clear fc_50_temp;

fc_50_d = double(fc_50);
clear fc_50;

fc_50_avg = mean(fc_50_d, 3);

clear fc_50_d;



%Flame images at 60 mm

f_60_temp = 'f_60.TIF';
f_60 = get_TIF_raw_data2D(f_60_temp);
clear f_60_temp;

f_60_d = double(f_60);
clear f_60;

f_60_avg = mean(f_60_d, 3);

clear f_60_d;



fc_60_temp = 'fc_60.TIF';
fc_60 = get_TIF_raw_data2D(fc_60_temp);
clear fc_60_temp;
```

```
fc_60_d = double(fc_60);
clear fc_60;

fc_60_avg = mean(fc_60_d, 3);

clear fc_60_d;




%*************************************************************************

% Averaging Initial and Final Collimated light images



% Images with no burner tip visible:

C_avg = (Ci_avg + Cf_avg)./2;

clear Ci_avg;
clear Cf_avg;

% figure,imshow(C_avg, []);



% Images with burner tip visible:

Cb_avg = (Cib_avg + Cfb_avg)./2;
```

```
clear Cib_avg;
clear Cfb_avg;

% figure,imshow(Cb_avg, []);

%%%%%%%%%%%%%%%%%%%%%Cropping:%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

startx = 1;
deltax = 1024;
starty = 231;
deltay1 = 784; %For burner tip level pictures
deltay2 = 793; %For all other pictures

%%%%%%%%%%Cropping Collimated images:

Cb_crop = do_crop_image(Cb_avg,startx,deltax,starty,deltay1);

clear Cb_avg;

C_crop = do_crop_image(C_avg,startx,deltax,starty,deltay2);

clear C_avg;
```

```
%%%%%%%%%%%%%%Cropping Flame images:


f_0_crop = do_crop_image(f_0_avg,startx,deltax,starty,deltay1);
clear f_0_avg;



f_10_crop = do_crop_image(f_10_avg,startx,deltax,starty,deltay2);
clear f_10_avg;
f_20_crop = do_crop_image(f_20_avg,startx,deltax,starty,deltay2);
clear f_20_avg;
f_30_crop = do_crop_image(f_30_avg,startx,deltax,starty,deltay2);
clear f_30_avg;
f_40_crop = do_crop_image(f_40_avg,startx,deltax,starty,deltay2);
clear f_40_avg;
f_50_crop = do_crop_image(f_50_avg,startx,deltax,starty,deltay2);
clear f_50_avg;
f_60_crop = do_crop_image(f_60_avg,startx,deltax,starty,deltay2);
clear f_60_avg;
%%%%%%%%%%%%%%

fc_0_crop = do_crop_image(fc_0_avg,startx,deltax,starty,deltay1);
clear fc_0_avg;



fc_10_crop = do_crop_image(fc_10_avg,startx,deltax,starty,deltay2);
clear fc_10_avg;
```

```
fc_20_crop = do_crop_image(fc_20_avg,startx,deltax,starty,deltay2);
clear fc_20_avg;
fc_30_crop = do_crop_image(fc_30_avg,startx,deltax,starty,deltay2);
clear fc_30_avg;
fc_40_crop = do_crop_image(fc_40_avg,startx,deltax,starty,deltay2);
clear fc_40_avg;
fc_50_crop = do_crop_image(fc_50_avg,startx,deltax,starty,deltay2);
clear fc_50_avg;
fc_60_crop = do_crop_image(fc_60_avg,startx,deltax,starty,deltay2);
clear fc_60_avg;



%Subtracting:

Diff1 = fc_0_crop - f_0_crop;
clear fc_0_crop;
clear f_0_crop;
Diff2 = fc_10_crop - f_10_crop;
clear fc_10_crop;
clear f_10_crop;
Diff3 = fc_20_crop - f_20_crop;
clear fc_20_crop;
clear f_20_crop;
Diff4 = fc_30_crop - f_30_crop;
clear fc_30_crop;
clear f_30_crop;
Diff5 = fc_40_crop - f_40_crop;
clear fc_40_crop;
clear f_40_crop;
```

```
Diff6 = fc_50_crop - f_50_crop;
clear fc_50_crop;
clear f_50_crop;
Diff7 = fc_60_crop - f_60_crop;
clear fc_60_crop;
clear f_60_crop;



% %%%%%%%%%%%%%%%%%%%%sri's
loop%%%%%%%%%%%%%%%%%%%%%
% [lenx,leny] = size(Diff5);
% colpixel = [];
% for i = 1:lenx
%     colpixel = [colpixel Diff5(i,:)];
% end
% figure,hist(colpixel,51);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Normalization%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%


C_n = NORMBYROW(C_crop);
clear C_crop;
Cb_n = NORMBYROW(Cb_crop);
clear Cb_crop;


Diff1_n = NORMBYROW(Diff1);
clear Diff1;
Diff2_n = NORMBYROW(Diff2);
clear Diff2;
Diff3_n = NORMBYROW(Diff3);
clear Diff3;
```

```
Diff4_n = NORMBYROW(Diff4);
clear Diff4;
Diff5_n = NORMBYROW(Diff5);
clear Diff5;
Diff6_n = NORMBYROW(Diff6);
clear Diff6;
Diff7_n = NORMBYROW(Diff7);
clear Diff7;


% C_n = NORM(C_crop);
% clear C_crop;
% Cb_n = NORM(Cb_crop);
% clear Cb_crop;
%
% Diff1_n = NORM(Diff1);
% clear Diff1;
% Diff2_n = NORM(Diff2);
% clear Diff2;
% Diff3_n = NORM(Diff3);
% clear Diff3;
% Diff4_n = NORM(Diff4);
% clear Diff4;
% Diff5_n = NORM(Diff5);
% clear Diff5;
% Diff6_n = NORM(Diff6);
% clear Diff6;
```

```
%Can make a new function (with selecrows created earlier) to only normalize
%the TIFcific rows I want. Will save time...


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


%Taking the Ratios separately:

Ratio1 = Diff1_n./Cb_n;
clear Diff1_n;
clear Cb_n;
Ratio2 = Diff2_n./C_n;
clear Diff2_n;
Ratio3 = Diff3_n./C_n;
clear Diff3_n;
Ratio4 = Diff4_n./C_n;
clear Diff4_n;
Ratio5 = Diff5_n./C_n;
clear Diff5_n;
Ratio6 = Diff6_n./C_n;
clear Diff6_n;
Ratio7 = Diff7_n./C_n;
clear Diff7_n;
clear C_n;
```

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Combining%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%

Ratioa = cat(1,Ratio7,Ratio6,Ratio5,Ratio4,Ratio3,Ratio2,Ratio1); // combine () was replaced by cat()//

clear Ratio7;

clear Ratio6;

clear Ratio5;

clear Ratio4;

clear Ratio3;

clear Ratio2;

clear Ratio1;

Ratiob = Ratioa;

%--%

%--%

%%%%%%Selecting TIFcific Rows

% %selectrows(1) = [106];

%

% %selectrows(1) = [490];

%

% %selectrows(1) = [875];

%

% selectrows(1) = [1259];

```
% selectrows(2) = [1644];
% selectrows(3) = [2029];
% % selectrows(4) = [2415];
% % selectrows(5) = [2798];
% %selectrows(6) = [3183];
% %selectrows(7) = [3567];
% %selectrows(8) = [3952];
% %selectrows(9) = [4336];



%selectrows(1) = [157];
selectrows(1) = [542];
selectrows(2) = [927];
selectrows(3) = [1311];
selectrows(4) = [1696];
selectrows(5) = [2080];
selectrows(6) = [2465];
selectrows(7) = [2850];
selectrows(8) = [3234];
selectrows(9) = [3619];
selectrows(10) = [4004];
selectrows(11) = [4388];
%selectrows(13) = [4773];
%selectrows(14) = [5157];


% figure, imshow(Ratioa, []);
% figure, imshow(Ratioa_n, []);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Smoothing%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Rrows Rcolumns] = size(Ratiob);


%Smoothing using a loess function:

span =0.1;

for a = selectrows(:,:)
    %Rrows:-1:1
    Ratio_n(a,:) = smooth(Ratiob(a,:),span,'rloess')';
end

% for a = Rrows:-500:1
%     Ratio_n(a,:) = smooth(Ratioa(a,:),span,'loess')';
% end


%plotting the smoothed versus mean raw transmition for centering purposes

% figure, subplot(2,1,1);
% plot(1:1:Rcolumns,Ratiob(4388,:),'o',1:1:Rcolumns,Ratio_n(4388,:));
% title(['Smoothed Transmission Curve at Height = ', num2str(15),'mm']);
% legend('Raw','Smooth',0);
% grid on;
%
% figure, subplot(2,1,1);
```

```
% plot(1:1:Rcolumns,Ratiob(4004,:),'o',1:1:Rcolumns,Ratio_n(4004,:));
% title(['Smoothed Transmission Curve at Height = ', num2str(20),'mm']);
% legend('Raw','Smooth',0);
% grid on;
%
% figure, subplot(2,1,1);
% plot(1:1:Rcolumns,Ratiob(3619,:),'o',1:1:Rcolumns,Ratio_n(3619,:));
% title(['Smoothed Transmission Curve at Height = ', num2str(25),'mm']);
% legend('Raw','Smooth',0);
% grid on;
%
% figure, subplot(2,1,1);
% plot(1:1:Rcolumns,Ratiob(3234,:),'o',1:1:Rcolumns,Ratio_n(3234,:));
% title(['Smoothed Transmission Curve at Height = ', num2str(30),'mm']);
% legend('Raw','Smooth',0);
% grid on;
%
% figure, subplot(2,1,1);
% plot(1:1:Rcolumns,Ratiob(2850,:),'o',1:1:Rcolumns,Ratio_n(2850,:));
% title(['Smoothed Transmission Curve at Height = ', num2str(35),'mm']);
% legend('Raw','Smooth',0);
% grid on;
%
% figure, subplot(2,1,1);
% plot(1:1:Rcolumns,Ratiob(2465,:),'o',1:1:Rcolumns,Ratio_n(2465,:));
% title(['Smoothed Transmission Curve at Height = ', num2str(40),'mm']);
% legend('Raw','Smooth',0);
% grid on;
%
% figure, subplot(2,1,1);
```

```
% plot(1:1:Rcolumns,Ratiob(2080,:),'o',1:1:Rcolumns,Ratio_n(2080,:));
% title(['Smoothed Transmission Curve at Height = ', num2str(45),'mm']);
% legend('Raw','Smooth',0);
% grid on;
%
% figure, subplot(2,1,1);
% plot(1:1:Rcolumns,Ratiob(1696,:),'o',1:1:Rcolumns,Ratio_n(1696,:));
% title(['Smoothed Transmission Curve at Height = ', num2str(50),'mm']);
% legend('Raw','Smooth',0);
% grid on;
%
% figure, subplot(2,1,1);
% plot(1:1:Rcolumns,Ratiob(1311,:),'o',1:1:Rcolumns,Ratio_n(1311,:));
% title(['Smoothed Transmission Curve at Height = ', num2str(55),'mm']);
% legend('Raw','Smooth',0);
% grid on;



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



%***********************************************************************%


%Finding the centre:
```

```
mid_tol = 0.05; %Can be changed

% size1 = 1024;
% size = double(size1);
% clear size1;

% [Rrows Rcolumns] = size(Ratio_n);

% for i = selectrows(:,:)
%     %Rrows:-1:1
%     %Rrows:-500:1
%     centre(i,:) = SYMAXIS_ver2(Ratio_n(i,:),Rcolumns,mid_tol);
% end

for b = 1:Rcolumns

   Rcol(b) = b;

end

for i = selectrows(:,:)
   %Rrows:-1:1
   %Rrows:-500:1

   Rtrans = 1-Ratio_n(i,:);
   %Rtrans = Ratio_n(i,:);
   %Rtrans = 1./Ratio_n(i,:);

   [centre_exact1(i) centre_exact2(i) centre_exact3(i)] = SYMAXIS_ver2(Rtrans,Rcol,mid_tol);
```

```
end


clear i;
clear mid_tol;
clear b;
clear Rcol;
clear Rtrans;



%%%%%%%%%%%%%%%%%%%%%%Compare Centre Finding
methods%%%%%%%%%%%%%%%%%%%%%%%%

centre1_condensed(1) = centre_exact1(selectrows(1));
centre1_condensed(2) = centre_exact1(selectrows(2));
centre1_condensed(3) = centre_exact1(selectrows(3));
centre1_condensed(4) = centre_exact1(selectrows(4));
centre1_condensed(5) = centre_exact1(selectrows(5));
centre1_condensed(6) = centre_exact1(selectrows(6));
centre1_condensed(7) = centre_exact1(selectrows(7));
centre1_condensed(8) = centre_exact1(selectrows(8));
centre1_condensed(9) = centre_exact1(selectrows(9));
centre1_condensed(10) = centre_exact1(selectrows(10));
centre1_condensed(11) = centre_exact1(selectrows(11));
% centre1_condensed(12) = centre_exact1(selectrows(12));
% centre1_condensed(13) = centre_exact1(selectrows(13));
% centre1_condensed(14) = centre_exact1(selectrows(14));



centre2_condensed(1) = centre_exact2(selectrows(1));
```

```
centre2_condensed(2) = centre_exact2(selectrows(2));
centre2_condensed(3) = centre_exact2(selectrows(3));
centre2_condensed(4) = centre_exact2(selectrows(4));
centre2_condensed(5) = centre_exact2(selectrows(5));
centre2_condensed(6) = centre_exact2(selectrows(6));
centre2_condensed(7) = centre_exact2(selectrows(7));
centre2_condensed(8) = centre_exact2(selectrows(8));
centre2_condensed(9) = centre_exact2(selectrows(9));
centre2_condensed(10) = centre_exact2(selectrows(10));
centre2_condensed(11) = centre_exact2(selectrows(11));
% centre2_condensed(12) = centre_exact2(selectrows(12));
% centre2_condensed(13) = centre_exact2(selectrows(13));
% centre2_condensed(14) = centre_exact2(selectrows(14));



centre3_condensed(1) = centre_exact3(selectrows(1));
centre3_condensed(2) = centre_exact3(selectrows(2));
centre3_condensed(3) = centre_exact3(selectrows(3));
centre3_condensed(4) = centre_exact3(selectrows(4));
centre3_condensed(5) = centre_exact3(selectrows(5));
centre3_condensed(6) = centre_exact3(selectrows(6));
centre3_condensed(7) = centre_exact3(selectrows(7));
centre3_condensed(8) = centre_exact3(selectrows(8));
centre3_condensed(9) = centre_exact3(selectrows(9));
centre3_condensed(10) = centre_exact3(selectrows(10));
centre3_condensed(11) = centre_exact3(selectrows(11));
% centre3_condensed(12) = centre_exact3(selectrows(12));
% centre3_condensed(13) = centre_exact3(selectrows(13));
% centre3_condensed(14) = centre_exact3(selectrows(14));
```

```
% for o = selectrows(:,:)
%     %w = 0:dr:6.643
%
%     q = o;
%
%     x_axis_values(o,1) = q;
%
% end
%
% clear o;
% clear q;
%

for e = selectrows(:,:)

centre_1(e) = (centre_exact1(e) + centre_exact2(e) + centre_exact3(e))./3;

centre(e) = round(centre_1(e));

end

clear e;

%-------------------------------------------------------------------------%
```

```
%Adjusting matrices:

for k = selectrows(:,:)
   %Rrows:-1:1
   %Rrows:-500:1
   if (centre(k) >= Rcolumns/2)
      Lradial = length(centre(k)+1:Rcolumns);

      LNtau_smooth(k,1:Lradial) = Ratio_n(k,centre(k):-1:centre(k)-(Lradial-1));
      RNtau_smooth(k,1:Lradial) = Ratio_n(k,centre(k)+1:Rcolumns);

      %might as well create the left and right position vectors

%        Lpos(k) = length(centre(k):-1:centre(k)-(Lradial-1));
%        Rpos(k) = length(centre(k)+1:Rcolumns);

      [LNrows LNcolumns] = size(LNtau_smooth); %%Need to force to 512!!!
      [RNrows RNcolumns] = size(RNtau_smooth);



      %%%%%%%%%%%%%%%%%%%%%%%%%%%Removing
Zeros%%%%%%%%%%%%%%%%%%%%%%%%%%%

      for h = 1:LNcolumns

         if (LNtau_smooth(k,h) == 0)
```

```
        LNtau_smooth(k,h) = 0.5;
    end
end

for s = 1:RNcolumns

    if (RNtau_smooth(k,s) == 0)

        RNtau_smooth(k,s) = 0.5;
    end

end

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

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Adding
Columns%%%%%%%%%%%%%%%%%%%%%%%%%%%

if LNcolumns < 512

    missing_colL = 512 - LNcolumns;

    for fL = 1:missing_colL

        vL = LNcolumns + fL;

        LNtau_smooth(k,vL) = 0.5;

    end
```

```
    end

    if RNcolumns < 512

       missing_colR = 512 - RNcolumns;

       for fR = 1:missing_colR

          vR = RNcolumns + fR;

          RNtau_smooth(k,vR) = 0.5;

       end

    end

    clear missing_colL;
    clear fL;
    clear vL;

    clear missing_colR;
    clear fR;
    clear vR;

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

    Lpos(k) = length(LNcolumns:-1:LNcolumns-(Lradial-1));
    Rpos(k) = length(RNcolumns+1:Rcolumns);

%       Lpos(k) = length(LNtau_smooth(k):-1:LNtau_smooth(k)-(Lradial-1));
```

```
%         Rpos(k) = length(RNtau_smooth(k)+1:Rcolumns);

%        disp(sprintf('LNtau_smooth is %i',LNtau_smooth));

       %k+1;

    elseif (centre(k) < Rcolumns/2)

       LNtau_smooth(k,1:centre(k)) = Ratio_n(k,centre(k):-1:1);
       RNtau_smooth(k,1:centre(k)) = Ratio_n(k,centre(k):2*centre(k)-1);

       %might as well create the left and right position vectors

%        Lpos(k) = length(centre(k):-1:1);
%        Rpos(k) = length(centre(k):2*centre(k)-1);

       [LNrows LNcolumns] = size(LNtau_smooth);
       [RNrows RNcolumns] = size(RNtau_smooth);

       %%%%%%%%%%%%%%%%%%%%%%%%%%%Removing
Zeros%%%%%%%%%%%%%%%%%%%%%%%%%%%

       for h = 1:LNcolumns

          if (LNtau_smooth(k,h) == 0)

             LNtau_smooth(k,h) = 0.5;
          end
       end
```

```
    for s = 1:RNcolumns

       if (RNtau_smooth(k,s) == 0)

          RNtau_smooth(k,s) = 0.5;
       end

    end

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

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%Adding
Columns%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if LNcolumns < 512

       missing_colL = 512 - LNcolumns;

       for fL = 1:missing_colL

          vL = LNcolumns + fL;

          LNtau_smooth(k,vL) = 0.5;

       end

    end

    if RNcolumns < 512
```

```
        missing_colR = 512 - RNcolumns;

        for fR = 1:missing_colR

            vR = RNcolumns + fR;

            RNtau_smooth(k,vR) = 0.5;

        end

    end

    clear missing_colL;
    clear fL;
    clear vL;

    clear missing_colR;
    clear fR;
    clear vR;

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

    Lpos(k) = length(LNcolumns:-1:1);
    Rpos(k) = length(RNcolumns:2*RNcolumns-1);

%       Lpos(k) = length(LNtau_smooth(k):-1:1);
%       Rpos(k) = length(RNtau_smooth(k):2*LNtau_smooth(k)-1);

%       disp(sprintf('LNtau_smooth is %i',LNtau_smooth));
```

```
        %k+1;

    end
end



% % figure, imshow(LNtau_smooth, []);
% % figure, imshow(RNtau_smooth, []);
%
%
%
%-----------------------------------------------------------------------------------------------------------------
---------------------------------

%1-DIMENSIONAL TOMOGRAPHY
%The one-dimensional tomography is performed using a three-point Abel
%inversion method.  The algorithm used was developed by
%Cameron J. Dasch (Dasch, Cameron J.  "One-dimensional tomography:
%a comparison of Abel, onion-peeling, and filtered backprojection methods."
%Applied Optics. Vol 31, No. 8.  10 March 1992).

%calling the ABEL function which will provide the D matrix needed for
%convolution NB: only a function of # of positions



for g = selectrows(:,:)
```

```
intervals = Lpos(g); %right and left position vectors are same size
%D = ABEL(intervals);

D = ABEL(512);

%Deconvoluted results NB: the equation fv = (-lambda/6*pi*E(m))*dln(tau)/dr

dr = 0.013;
%Size of pixel based on TIFcification --> calculated it's 0.0129 mm

%%This may not be the correct value for dr. This only gives the
%%distance between pixel centres if there is no space between the
%%pixels themselves. Should call Kevin or talk to Paul about this.

% May need to bin data - spacing of 100 micrometers was used by
% Thomson, here I'm using 12.9 micrometers. This may account for
% excessive noise(See Thomson p. 2483).!!!!!


%%Need to talk to Kevin before the end of the month when he leaves
%%NSERC - after that I won't be able to get ahold of him. Closer it
%%gets to the end of the month the busier he will be. Make list
%%tomorrow.

lambda = 600e-9;%in metres

Em = 0.258; %Check value again - see Decio's project, Kevin's project,
        %and the Thomson paper.

fv_conv = (lambda/(6*pi*Em))*1e9;%ppm
```

```
    tau_L(g,:) =(1/dr)*D*-log(LNtau_smooth(g,:))';
    tau_R(g,:) =(1/dr)*D*-log(RNtau_smooth(g,:))';



    %soot volume fraction
    fv_L(g,:) = fv_conv*tau_L(g,:);
    fv_R(g,:) = fv_conv*tau_R(g,:);
    fv_avg(g,:) = (fv_L(g,:)+ fv_R(g,:))/2;

%     g+1;

end


fv_L_condensed(1,:) = fv_L(selectrows(1),:);
fv_L_condensed(2,:) = fv_L(selectrows(2),:);
fv_L_condensed(3,:) = fv_L(selectrows(3),:);
fv_L_condensed(4,:) = fv_L(selectrows(4),:);
fv_L_condensed(5,:) = fv_L(selectrows(5),:);
fv_L_condensed(6,:) = fv_L(selectrows(6),:);
fv_L_condensed(7,:) = fv_L(selectrows(7),:);
fv_L_condensed(8,:) = fv_L(selectrows(8),:);
fv_L_condensed(9,:) = fv_L(selectrows(9),:);
fv_L_condensed(10,:) = fv_L(selectrows(10),:);
fv_L_condensed(11,:) = fv_L(selectrows(11),:);
% fv_L_condensed(12,:) = fv_L(selectrows(12),:);
% fv_L_condensed(13,:) = fv_L(selectrows(13),:);
% fv_L_condensed(14,:) = fv_L(selectrows(14),:);
```

```
fv_R_condensed(1,:) = fv_R(selectrows(1),:);
fv_R_condensed(2,:) = fv_R(selectrows(2),:);
fv_R_condensed(3,:) = fv_R(selectrows(3),:);
fv_R_condensed(4,:) = fv_R(selectrows(4),:);
fv_R_condensed(5,:) = fv_R(selectrows(5),:);
fv_R_condensed(6,:) = fv_R(selectrows(6),:);
fv_R_condensed(7,:) = fv_R(selectrows(7),:);
fv_R_condensed(8,:) = fv_R(selectrows(8),:);
fv_R_condensed(9,:) = fv_R(selectrows(9),:);
fv_R_condensed(10,:) = fv_R(selectrows(10),:);
fv_R_condensed(11,:) = fv_R(selectrows(11),:);
% fv_R_condensed(12,:) = fv_R(selectrows(12),:);
% fv_R_condensed(13,:) = fv_R(selectrows(13),:);
% fv_R_condensed(14,:) = fv_R(selectrows(14),:);
```

```
fv_avg_condensed(1,:) = fv_avg(selectrows(1),:);
fv_avg_condensed(2,:) = fv_avg(selectrows(2),:);
fv_avg_condensed(3,:) = fv_avg(selectrows(3),:);
fv_avg_condensed(4,:) = fv_avg(selectrows(4),:);
fv_avg_condensed(5,:) = fv_avg(selectrows(5),:);
fv_avg_condensed(6,:) = fv_avg(selectrows(6),:);
fv_avg_condensed(7,:) = fv_avg(selectrows(7),:);
fv_avg_condensed(8,:) = fv_avg(selectrows(8),:);
fv_avg_condensed(9,:) = fv_avg(selectrows(9),:);
fv_avg_condensed(10,:) = fv_avg(selectrows(10),:);
```

```
fv_avg_condensed(11,:) = fv_avg(selectrows(11),:);
% fv_avg_condensed(12,:) = fv_avg(selectrows(12),:);
% fv_avg_condensed(13,:) = fv_avg(selectrows(13),:);
% fv_avg_condensed(14,:) = fv_avg(selectrows(14),:);
for u = 1:513
   %w = 0:dr:6.643

   w = (u-1)*dr;

   x_axis_values(u,1) = w;

end
toc
```

7.2 APPENDIX B: (USER DEFINED FUNCTIONS TO RUN THE MAIN PROGRAM ATTACHED ABOVE IN APPENDIX A)

To read the image in tiff format

```
function [extracted_data header] = get_TIF_raw_data2D(filename)
raw_data = imread(filename); % 2050 uint16 = 4100 bytes = 32800 bits

Xdim = 1024;
Ydim = 1024;

Zdim = 30;

extracted_data = raw_data;
```

To read the image in spe format

```
function [extracted_data header] = get_spe_raw_data2D(filename)

fid=fopen(filename);
headersize=4100;
raw_data=[];

header = fread(fid,2050,'uint16=>uint16'); % 2050 uint16 = 4100 bytes = 32800 bits
```

```
Xdim = header(22);
Ydim = header(329);
Zdim = header(724);

DataType = header(55);

if Zdim == 0
   Zdim =1;
end

%Total_Size_XYZ = Xdim*Ydim*Zdim


%raw_data
switch DataType
   case 0  % FLOATING POINT (4 bytes / 32 bits)
      raw_data = fread(fid,inf,'float32=>float32');
   case 1  % LONG INTEGER (4 bytes / 32 bits)
      raw_data = fread(fid,inf,'int32=>int32');
   case 2  % INTEGER (2 bytes / 16 bits)
      raw_data = fread(fid,inf,'int16=>int16');
   case 3  % UNSIGNED INTEGER (2 bytes / 16 bits)
      raw_data = fread(fid,inf,'uint16=>uint16');
end

fclose(fid);

frame4view1 = reshape(raw_data,Xdim,Ydim,Zdim);
clear raw_data;
frame4view1 = permute(frame4view1,[2,1,3]);
extracted_data  = frame4view1;
```

do_show_image() :

```
function [nothing] = do_show_image(raw_image,style)

if style ==1

   figure


   imshow(raw_image,[min(min(raw_image)) max(max(raw_image))])
   %--------------------------
```

```
    impixelinfo
    imdisplayrange

    clear scrap display_this;

elseif style ==2


    imshow(raw_image,[min(min(raw_image)) max(max(raw_image))])
    %---------------------------

    impixelinfo
    imdisplayrange


elseif style ==3
    if new_figure ==1
        figure
    end
    [Ydim Xdim] = size (raw_image);
    [X,Y] = meshgrid([1:Xdim],[1:Ydim]);
    contour3(X,Y,raw_image,30)
    surface(X,Y,raw_image,'EdgeColor',[.8 .8 .8],'FaceColor','none')
    grid off
    view(-15,25)
    colormap cool

end
```

NORM():

```
function [I_norm] = NORM(I)

 I_double = double(I);

I_norm = I_double./max(max(I_double));
End
```

NORMBYROW():

```
function Ratiob = NORMBYROW(Ratioa)
[Nrows Ncolumns] = size(Ratioa);

for k = 1:Nrows

    Ratiob(k,:) = NORM(Ratioa(k,:));
```

```
End
```

SYMAXIS2():

```
function centre = SYMAXIS2(Ntau_smooth,pos,mid_tol)
 ini_guess = pos./2;
midvalue = 1-(1/3)*(1-Ntau_smooth(ini_guess));


for  m = ini_guess:-1:1 %left hand side of flame
   if abs((midvalue-Ntau_smooth(m))./midvalue)<= mid_tol
      mid_left = m;
%        disp('Obtained value for mid_left');
      break
   end
end

for n = ini_guess:pos  %right hand side of flame

   mid_right = mid_left + 512;

   if abs((midvalue-Ntau_smooth(n))./midvalue)<= mid_tol
      mid_right = n;
%        disp('Obtained value for mid_right');
      break
   end
end

centre = floor((mid_left+mid_right)./2);
```

do_crop_image():

```
function [cropped_image] = do_crop_image(raw_image,startx,deltax,starty,deltay)



cropped_image = raw_image(starty:(starty+deltay-1),startx:(startx+deltax-1),:);
```

SYMAXIS_ver2():

```
function [centre_exact, centre_exact2, centre_exact3] =
SYMAXIS_ver2(orig_data,xdata,mid_tol)



% Please note:  There are 3 different methods for finding the centre that
```

```
% are implimented here.



% METHOD 1:  This purpose of this function is to find the 2D axis of

% symmetry of the data. Note that the midpoint is defined as the position

% half way between the left and right positions for which the transmission

% value has dropped by 1/2



% In order to more accurately find the center, it is necessary to interpolate between the smoothed

% data. In order to accomplish this a suitable interpolation method is required. NB: the spline used

% is a not-a-knot spline



% Specify the increment

inc = mid_tol;
DATAspline = spline(xdata,orig_data);
finer_xdata = xdata(1):inc:xdata(end);
derv_DATAspline = fnder(DATAspline);
i = xdata(1)+((xdata(end)-xdata(1))/2);% Lefthand side

j = i; % Righthand side



neg_DATAspline = fncmb(DATAspline,-1);

max_DATAspline = fnmin(neg_DATAspline)*-1;

target = max_DATAspline/2;
```

```
height_vector = 0:max_DATAspline/100:max_DATAspline;
ones_vector = ones(1,length(height_vector));


figure
hold on
plot (xdata,orig_data,':bo')
plot (finer_xdata,ppval(DATAspline,finer_xdata),'-r')
plot (xdata, max_DATAspline*ones(1,length(xdata)),':',xdata, target*ones(1,length(xdata)),':')

shift_DATAspline = fncmb(DATAspline,'+',-target);
plot (finer_xdata,ppval(shift_DATAspline,finer_xdata),'-k')

zeros_DATAspline = fnzeros(shift_DATAspline);

cropped_xdata = zeros_DATAspline(1,1):inc:zeros_DATAspline(1,end);
plot (cropped_xdata,ppval(shift_DATAspline,cropped_xdata),'->k')

%%%%%%%%%%%%%%%% CENTERING METHOD #1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Left side
```

```
i=zeros_DATAspline(1,1);

fprintf('\n\n');

fprintf('Found left side centering positon %0.5g\n',i);


%Right side

%j=zeros_DATAspline(1,2);

j=zeros_DATAspline(1,end);

fprintf('Found right side centering position %0.5g\n',j);


centre_exact = i+((j-i)/2);

plot (centre_exact*ones_vector,height_vector,':')

plot (cropped_xdata,ppval(derv_DATAspline,cropped_xdata),'-k')




zeros_derv_DATAspline  =fnzeros(derv_DATAspline,[zeros_DATAspline(1,1)
zeros_DATAspline(1,end)]);


if size(zeros_derv_DATAspline,2)==3

   centre_exact2 = zeros_derv_DATAspline(1,2); % the valley

   centre_exact3 = (zeros_derv_DATAspline(1,1)+zeros_derv_DATAspline(1,3))/2; %half way
between both peaks

elseif size(zeros_derv_DATAspline,2) ==1

   %This is if there is only one peak
```

```
    centre_exact2 = zeros_derv_DATAspline(1,1); % the peak

    centre_exact3 = (centre_exact2+centre_exact)/2; %Provides a fake unbiased point since we
can't calculate it;

elseif size(zeros_derv_DATAspline,2) >3

    %This is for noisey data that provides multiple peaks....

    centre_exact3 = (zeros_derv_DATAspline(1,1)+zeros_derv_DATAspline(1,end))/2; %half
way between both peaks

    centre_exact2 = (centre_exact3+centre_exact)/2; %Provides a fake unbiased point since we
can't calculate it

end
plot (centre_exact2*ones_vector,height_vector,':g',centre_exact3*ones_vector,height_vector,':r')

legend ('Orig', 'spline',",",",",'Centre1',",'Centre2','Centre3')

title('Fitting a spline to the already orig_data data')

hold off

percent_more = max(orig_data)*0.1;

ylim([min(orig_data)-percent_more max(orig_data)+percent_more])



fprintf('Centre position is %0.5g or %0.5g or %0.5g\n',centre_exact,
centre_exact2,centre_exact3);

fprintf('\n\n');
```

7.3 APPENDIX C: Developing the 3D contour plot:

```
t=linspace(0,6.565,512);

figure, subplot(2,1,2);

plot(t,fv_L(4388,:),t,fv_R(4388,:));

title(['Transmittance Curves ']);

ylabel('Soot Volume Fraction (ppm)')

xlabel('Radial Position (mm)');
```

```
legend('46 mm L','46 mm R','48 mm L','48 mm R','50 mm L','50 mm R','52 mm L','52 mm R','54 mm L','54 mm R',0);

grid on;
```

8 REFERENCES:

[1] Law, C. K. (2006), "Laminar premixed flames". Combustion physics. Cambridge, England: Cambridge University Press. p. 300. ISBN 0-521-87052-6. Archived from the original on 8 December 2017.

[2] Omidvarborna et al. (2015), "Recent studies on soot modeling for diesel combustion". Renewable and Sustainable Energy Reviews. 48: 635–647. doi:10.1016/j.rser.2015.04.019.

[3] www.kindpng.com/imgv/iRmJJT_structure-of-candle-flame-zones-of-candle-flame/, retrieved from this website on 7th of October,2019.

[4] H. Gg. Wagner (1979), "Soot formation in combustion", Proceedings of the Combustion Institute, 17:3–19.

[5] M. Frenklach (2002), "Reaction mechanism of soot formation in flames". Physical Chemistry Chemical Physics, 4:2028–2037.

[6] Q. L. Zhang, S. C. O'Brien, J. R. Heath, Y. Liu, R. F. Curl, H. W. Kroto, and R. E. Smalley (1986), "Reactivity of large carbon clusters: Spheroidal carbon shells and their possible relevance to the formation and morphology of soot". The Journal of Physical Chemistry, 90:525–528.

[7] I. Glassman (1998), "Sooting laminar diffusion flames: Effect of dilution, additives pressure and microgravity". Proceedings of the Combustion Institute, 27:1589–1596.

[8] K. P. Schug, Y. Manheimer-Timnat, P. Yaccarino, and I. Glassman (1984), "Sooting be havior of gaseous hydrocarbon diffusion flames and the influence of additives". Combustion Science and Technology, 22:235–250.

[9] D. R. Snelling, K. A. Thomson, G. J. Smallwood, and ¨O. L. Giilder (1999), "Two-Dimensional imaging of soot volume fraction in laminar diffusion flames". Applied Optics, 38:2478–2485.

[10] Ahmet Emre Karatas (2014), "High-Pressure Soot Formation and Diffusion Flame

Extinction Characteristics of Gaseous and Liquid Fuels", Journal of Combustion Science.

[11] https://www.jai.com/products/ap-3200t-usb, retrieved from this website on 3rd of December,2019.