Non-uniform combustion temperature estimation using laser absorption spectroscopy and Multi Output Gaussian Process Regression

A project report submitted

in partial fulfillment for the award of the degree of

Bachelor of Technology

in

Electronics and Communication Engineering

by

Saumya Vilas Roy

Department of Avionics
Indian Institute of Space Science and Technology
Thiruvananthapuram, India

May 2024

Certificate

This is to certify that the project report titled ***Non-uniform combustion temperature estimation using laser absorption spectroscopy and Multi Output Gaussian Process Regression*** submitted by **Saumya Vilas Roy**, to the Indian Institute of Space Science and Technology, Thiruvananthapuram, in partial fulfillment for the award of the degree of **Bachelor of Technology** in **Electronics and Communication Engineering** is a bona fide record of the original work carried out by him/her under my supervision. The contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Rajesh Sadanandan	**Dr. Deepak Mishra**
Associate Professor	Professor
Department of Aerospace	Department of Avionics

Dr. N. Selvagnesan
Professor & HoD
Department of Avionics

Place: Thiruvananthapuram
Date: May 2024

Declaration

I declare that this project report titled ***Non-uniform combustion temperature estimation using laser absorption spectroscopy and Multi Output Gaussian Process Regression*** submitted in partial fulfillment for the award of the degree of **Bachelor of Technology** in **Electronics and Communication Engineering** is a record of the original work carried out by me under the supervision of **Dr. Rajesh Sadanandan** and **Dr. Deepak Mishra**, and has not formed the basis for the award of any degree, diploma, associateship, fellowship, or other titles in this or any other Institution or University of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Place: Thiruvananthapuram
Date: May 2024

Saumya Vilas Roy
(SC20B121)

Acknowledgements

I would like to express my sincere gratitude to Dr. Rajesh Sadanandan Associate Professor, Aerospace Department, IIST and Dr. Satheesh K. Associate Professor, Aerospace Department, IIST during this project has been an enriching experience. I am truly grateful for the knowledge and insights I gained from them, and this opportunity to work and enrich myself.

I would also like to extend my thanks to Dr. Deepak Mishra, Professor, Department of Avionics, IIST for consistently motivating me to set higher goals and for providing valuable feedback, guiding me in the right direction.

Also thanks to Mr. Gokul Krishnan for entertaining all my doubts and conducting the experiment to validate the model.

Finally, I am indebted to all my family, friends and seniors who supported and assisted me throughout this project. Their collective encouragement and guidance have been instrumental in shaping my learning experience.

Saumya Vilas Roy

Abstract

The estimation of temperature profile inside the combustion system is one of the most important factor for us to consider for combustion characterization. To infer that information about stability, efficiency etc. associated with the combustion system, current methods of temperature estimation is slow and less accurate. This project combines Laser Absorption Spectroscopy (LAS) & a Multi Output Gaussian Process Regression framework to estimate the non-uniform temperature profile inside the combustion system by parameterizing it as a Boltzmann distribution. The method uses using a single transition of molecular absorption spectra to estimate those parameters. We further test effects of noise on the GPR framework and try to validate the model with experimental results. The model performs consistently with average Root Mean Squared Loss (RMSE) of 19.9165 and maximum loss of 87.7101 in the simulated data testing whereas in the experimental data the error in maximum temperature prediction is 20.0593%.

Contents

List of Figures

List of Tables

Abbreviations

ML	Machine Learning
GPR	Gaussian Process Regression
GP	Gaussian Process
MOGPR	Multi Output Gaussian Process
LMC	Linear Model of Coregionalization
ICM	Intrinsic Coregionalization Model
TLDAS	Tunable Laser Diode Absorption Spectroscopy

Chapter 1

Introduction

Machine Learning (ML) has revolutionized various fields by enabling computers to learn from data and make predictions or decisions without being explicitly programmed. Within the realm of ML, Gaussian Process Regression (GPR) stands out as a flexible and powerful technique for modeling complex relationships in data. GPR is a Bayesian non-parametric approach that can capture uncertainties and provide rich probabilistic outputs, making it well-suited for applications such as regression, classification, and optimization tasks. This report delves into the fundamentals of ML and the intricacies of GPR, exploring their combined potential to unlock valuable insights from data and drive advancements in diverse domains. By harnessing the capabilities of ML and GPR, researchers and practitioners can tackle complex problems, make informed decisions, and pave the way for innovative solutions in the era of data-driven discovery. In this study we utilize the strengths of GPR to estimate the non-uniform temperature profile indide the model combustion system.

Gaussian Process Regression (GPR) has a wide range of applications across various fields due to its flexibility and ability to model complex relationships in data. Some common use cases of GPR include:

- Regression Analysis: GPR is commonly used for regression tasks where the goal is to predict continuous values based on input data. It can handle non-linear relationships and provide uncertainty estimates for the predictions.

- Time Series Forecasting: GPR is effective in modeling and forecasting time series data, such as stock prices, weather patterns, or sensor readings. It can capture trends, seasonality, and irregularities in the data.

- Optimization: GPR can be used in optimization problems to find the optimal values of parameters or inputs by modeling the objective function and its uncertainties. This is particularly useful in tuning hyperparameters of machine learning models.

- Anomaly Detection: GPR can be applied to detect anomalies or outliers in data by modeling the normal behavior and identifying deviations from it. This is useful in various domains, including cybersecurity and fault detection.

- Robotics and Control: GPR is utilized in robotics and control systems for tasks such as motion planning, sensor fusion, and adaptive control. It can help robots learn from data and make decisions in dynamic environments.

- Surrogate Modeling: GPR is often used as a surrogate model in expensive simulations or optimization problems to reduce computational costs. It can approximate the complex simulation models and guide the search for optimal solutions.

For some cases where we have to solve multiple output problems or model correlation between data points for that we look towards expanding it by using Multi Output GPR.

Multi-output Gaussian Process Regression (MOGPR) is an extension of Gaussian Process Regression (GPR) that allows for modeling multiple correlated outputs simultaneously. This technique is particularly useful when dealing with tasks where the outputs are interrelated or dependent on each other. Here are some key points about Multi-output GPR:

- Correlated Outputs: In many real-world scenarios, the outputs of a system or process are not independent but exhibit correlations or dependencies. MOGPR takes into account these correlations and jointly models the relationships between multiple outputs.

- Shared Information: MOGPR leverages the shared information among outputs to improve the modeling accuracy. By capturing the correlations between outputs, it can make more informed predictions and better handle complex relationships in the data.

- Dimensionality Reduction: MOGPR can also be used for dimensionality reduction by capturing the underlying structure of the outputs. It can identify common patterns or trends across multiple outputs, leading to a more compact representation of the data.

- Transfer Learning: MOGPR enables transfer learning between related tasks by leveraging the shared information among outputs. This can be beneficial in scenarios where data for one task is limited, but there is abundant data available for related tasks.

- Applications: MOGPR finds applications in various fields such as neuroscience, environmental modeling, bioinformatics, and robotics, where multiple outputs need to be modeled jointly. It is used for tasks like multi-target regression, multi-label classification, and multi-sensor fusion.

Estimating temperature profiles in flame diagnostics is crucial for several reasons:

- Understanding Combustion Processes: Temperature is a key parameter that influences the kinetics of chemical reactions in combustion processes. By accurately estimating temperature profiles, researchers can gain insights into the efficiency, stability, and pollutant emissions of combustion systems.

- Optimizing Combustion Efficiency: Temperature profiles provide information about the distribution of heat within a flame. By optimizing temperature distribution, engineers can enhance combustion efficiency, reduce fuel consumption, and minimize the formation of harmful byproducts like NOx and CO.

- Predicting Flame Behavior: Temperature profiles can help predict flame behavior, such as ignition, propagation, and extinction. Understanding how temperature varies spatially and temporally within a flame can aid in predicting flame characteristics and optimizing combustion systems.

- Diagnosing Flame Anomalies: Deviations in temperature profiles can indicate anomalies or inefficiencies in combustion processes. By monitoring temperature variations, researchers can diagnose issues such as incomplete combustion, flame instability, or heat losses, allowing for timely corrective actions.

- Validating Computational Models: Temperature profiles serve as valuable data for validating computational models of combustion. By comparing experimental temperature measurements with model predictions, researchers can improve the accuracy and reliability of simulation tools used in flame diagnostics.

In last few decades there has been a surge of laser diagnostics techniques to be developed for characterizing combustion systems for measuring the gas temperature, pressure, velocity and composition. The landmark work by Wolfrum [1] and Hanson [2] highlight the development of laser diagnostics for combustion studies. There are multiple optical (i.e., absorption, laser-induced fluorescence, Raman, emission and laser-induced incandescence) and non-optical diagnostics techniques (thermocouples, pressure transducers, gas chromatography and mass spectroscopy) employed for studying combustion. The advantages of using laser based diagnostics methods is that it allows us to have a high-bandwidth measurements of thermodynamic conditions and work on wide range of molecular species existing in combustion flow. In this project we use data from a tunable Taser Diode based Molecular Absorption Spectroscopy Technique (TDLAS) because it allows us to have a non invasive temperature measurements as well as a high temporal resolution and high sampling rates (KHz or MHz).

1.1 Problem definition

In this project we are trying to estimate the non-uniform temperature profile of a combustion system by parameterizing it as a Boltzmann distribution.

Gaussian process regression, has emerged as a valuable tool for modeling and predicting flame characteristics with high accuracy and efficiency. By leveraging the flexibility and non-parametric nature of Gaussian processes, researchers have been able to extract valuable insights from complex flame data, leading to improved understanding and control of combustion processes. This paper explores the combined application of TDLAS and Gaussian process regression in flame diagnostics, highlighting its potential to revolutionize the way we analyze and interpret flame behavior.

1.2 Advantages of Machine Learning (ML) and Deep learning (DL) based methods over the conventional methods in combustion diagnostics

Machine learning (ML), deep learning (DL), and Gaussian process regression (GPR) are increasingly being used in combustion research for various reasons:

- **Data Analysis:** ML/DL can analyze large datasets more efficiently than traditional methods, helping researchers extract valuable insights from complex combustion processes.

- **Modeling Complex Systems:** ML/DL can capture non-linear relationships within combustion systems, allowing for more accurate predictions and modeling of combustion behavior.

- **Optimization:** These techniques can be used to optimize combustion processes by identifying optimal parameters and configurations, leading to improved efficiency and reduced emissions.
- **Overcoming Limitations:** ML/DL can overcome limitations of traditional empirical methods by providing more accurate predictions without relying on simplified assumptions.
- **Better Predictions:** ML/DL can offer better predictions by learning from data patterns and adapting to new information, resulting in improved accuracy compared to empirical methods.

In summary, using ML/DL in combustion research can enhance data analysis, model complex systems more accurately, optimize processes, overcome limitations of traditional methods, and provide better predictions. It offers a more sophisticated and data-driven approach to understanding combustion processes.

Further more it it provides the following advantages over the conventional empirical techniques:

1. **Data-Driven Approach:** ML/DL use data to make predictions and model combustion processes, allowing for a more comprehensive and accurate representation of complex systems compared to simplified empirical models.
2. **Non-Linear Relationships:** ML/DL/GPR can capture non-linear relationships within combustion systems, which may be challenging for traditional empirical methods that rely on linear approximations.
3. **Adaptability:** These techniques can adapt to new data and information, continuously improving predictions and models over time, whereas empirical methods may struggle to incorporate new insights effectively.
4. **Optimization:** ML/DL/GPR can optimize combustion processes by identifying patterns and relationships in data that lead to more efficient and effective outcomes, surpassing the capabilities of empirical optimization methods.
5. **Reduced Bias:** ML/DL/GPR are less susceptible to bias and assumptions that can be inherent in empirical methods, providing a more objective and data-driven approach to combustion research.

Chapter 2

Theoretical Background

2.1 Theory behind the Machine Learning (ML) model

Selection of the model

For the problem statement we selected Gaussian Process Regression (GPR) to help us estimate the non-uniform temperature profile. **Definition:** A Gaussian process is a collection of random variables, where any finite number of variables have a joint Gaussian distribution, capturing the essence of uncertainty in a continuous manner.

- A Gaussian process is a powerful extension of the Gaussian probability distribution, enabling a flexible framework for modeling functions rather than just individual data points.

- It represents a distribution over functions, allowing for a rich representation of complex relationships in data without imposing specific functional forms.

- As a non-parametric approach to data modeling, Gaussian processes do not rely on predefined parameters or assumptions about the data distribution, offering a versatile tool for various applications.

- While conceptually existing in infinite dimensions, Gaussian processes are practically implemented using a finite subset of data points for computational efficiency.

- The covariance function, also known as the kernel function, plays a crucial role in defining the relationships and properties within the function space of a Gaussian process.

- By utilizing data points as reference locations, Gaussian processes effectively anchor the function to specific observations, allowing for interpolation and extrapolation with uncertainty quantification.

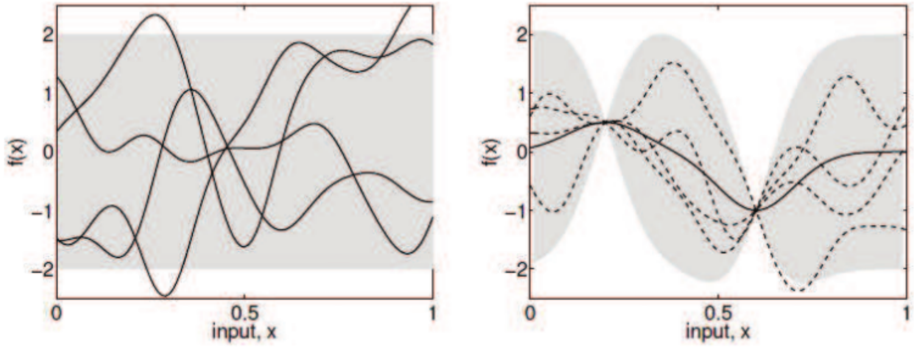

Figure 2.1: Left Samples drawn from the prior distribution of functions with no observed datapoints. **Right:** Samples drawn from the prior distribution of functions with two observed datapoints. Solid black line shows the mean and shaded region twice the standard deviation.

Introduction to Multi Output Gaussian Process (MOGPR)

The conventional Gaussian Process Regression (GPR) maps only a single distribution but in cases where we need to map two distributions together for example non uniform T and mole fraction of the absorbing species x_i we need to expand it so it can handle multiple outputs.

There are several ways it can be achieved by:

- Parallel single output GPRs

  It is just using multiple single output GPRs in parallel to estimate the each individual point in output distribution (i.e. Assuming all the points in output are independent.)

- MOGPR- Linear Model of Coregionalization (LMC)

- MOGPR - Intrinsic Coregionalization Model (IMC)

2.1.1 MOGPR - Linear Model of Coregionalization (LMC)

It works by allowing several independent samples from GPs with different covariances and talking their weighted sum to model the output distribution.

For an output $f_d(x)$

$$f_d(x) = \sum_{q=1}^{Q} \sum_{i=1}^{R_q} a_{d,q}^i u_q^i(x)$$

Here
$u_q^i(x) \sim \mathcal{GP}_q^i(0, k(x, x^{'}))$
Q number of groups.
R_q is number samples in a group.

2.1.2 MOGPR - Intrinsic Coregionalization Model

It works by take a single sample from GP and talking their weighted sum to model the output distribution.

For an output $f_d(x)$

$$f_d(x) = \sum_{i=1}^{R} a_d^i u^i(x)$$

Here
$u(x) \sim \mathcal{GP}(0, k(x, x^{'}))$
Q number of groups.
R_q is number samples in a group.

2.1.3 Theorem in GPR

Autokrigeability

- If the outputs are considered to be noise-free, prediction using the ICM under an isotopic data case is equivalent to independent prediction over each output.

2.2 Basics of Spectroscopy

- Spectroscopy is the study of properties of matter through its interaction with different frequency components of the electromagnetic spectrum.
- Spectroscopy is a general methodology that can be adapted in many ways to extract the information you need (energies of electronic, rotational states, structure and symmetry of molecules, dynamic information).

- With light, you aren't looking directly at the molecule—the matter—but its "ghost." You observe the light's interaction with different degrees of freedom of the molecule. Each type of spectroscopy—different light frequency—gives a different picture - the spectrum.

Interaction of light with a medium can influence the sample and/or the light. Method involves: (1) excitation and (2) detection.

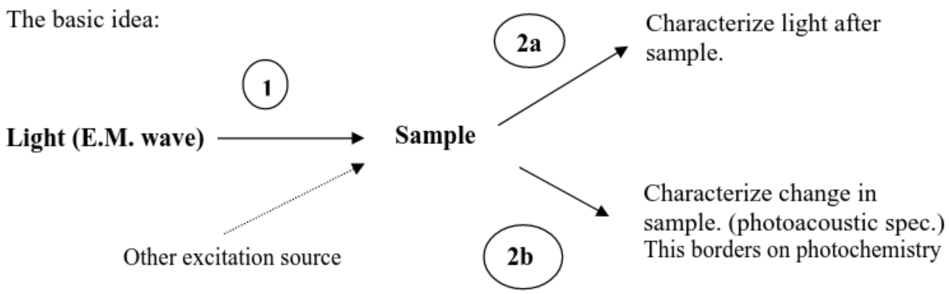
The basic idea:
1
Light (E.M. wave)
Sample
Other excitation source
2a
Characterize light after sample.
2b
Characterize change in sample. (photoacoustic spec.) This borders on photochemistry

Figure 2.2: Block diagram of Spectroscopy.

2.3 Theory behind laser absorption spectroscopy

Absorption: Change in intensity I_o of incident light as it passes through a medium [3].

Transmission $\tau_v = I/I_0$

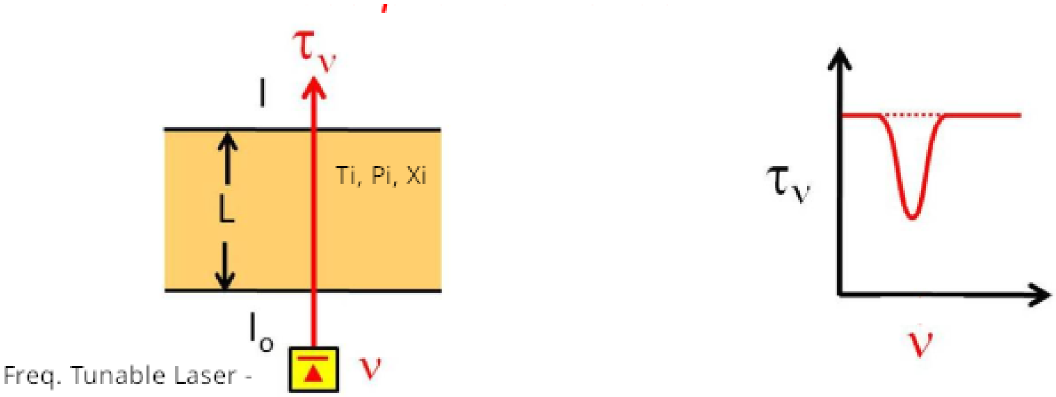

Figure 2.3: Absorption of monochromatic light, Molecular absorption spectrum (right) .

- Line-of-sight direct absorption spectroscopy (DAS)

- Line-of-sight direct absorption: This refers to the direct absorption of light along a specific path without any scattering or reflection. In this scenario, the Beer-Lambert relation is often used to quantify the attenuation of light as it passes through a medium. The relation is expressed as $\tau_v = \frac{I}{I_0} = \exp(-k_v \cdot L)$, where τ_v is the transmittance at a given wavelength (v), I is the intensity of light after passing through the medium, I_0 is the initial intensity, k_v is the spectral absorption coefficient, and L is the path length.

- Spectral absorption coefficient: The spectral absorption coefficient, k_v, is a measure of how a material absorbs light at a specific wavelength (v). It is influenced by factors such as temperature (T), pressure (P), and the concentration of the absorbing species (χ_i). The equation for the spectral absorption coefficient is given by $k_v = S(T) \cdot \Phi(T, P, \chi_i) \cdot \chi_i \cdot P$, where $S(T)$ and $\Phi(T, P, \chi_i)$ represent temperature-dependent functions.

- Uniform temperature profile: This assumption implies that the temperature within the medium remains constant along the path of light transmission. It simplifies the analysis by considering a consistent temperature distribution, which is often a reasonable approximation in certain experimental setups.

- In figure 2.4: absorption spectra of a molecular species of interest to combustion systems.

- Line-strength $s(T)$ depends on Temperature T, Pressure P,Mole fraction X_i and Pathlenght L.

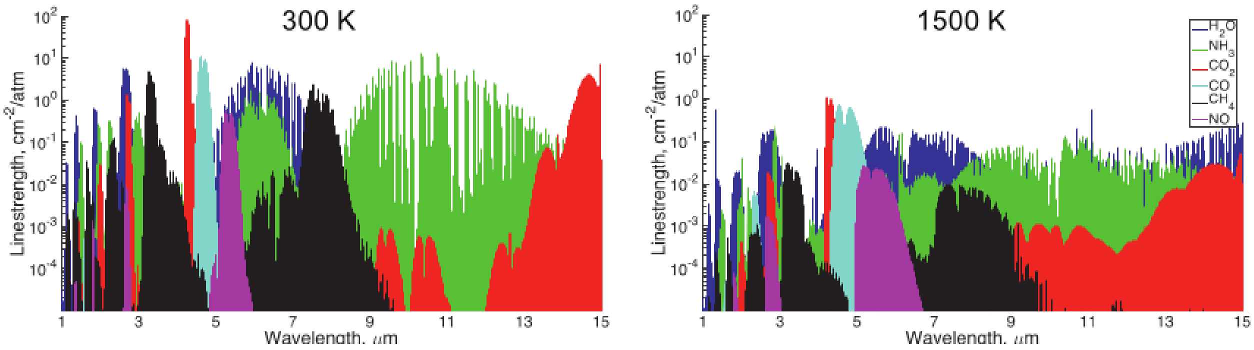

Figure 2.4: Linestrengths (i.e., absorption strength) of several molecular species of interest to combustion systems for a fixed X_i at two different temperatures.

- From the above absorption spectrum, for the molecule of interest (in the case water vapor H_2O), the absorption transition which is sensitive to temperature changes is selected for the laser diagnostics experiments using DAS.

- Experiments are performed to capture the molecular absorption spectrum through the medium of interest (Fig. 2.2, right).

- After selecting the absorption line. The objective is to find the non-uniform temperature profile inside the region of interest using the line absorbance information.

- The Beer-Lambert law above assumes a uniform temperature profile. However in real combustion systems profiles could be like as follows but they can take any shape or form (Non-uniform temperature profile inside the medium).

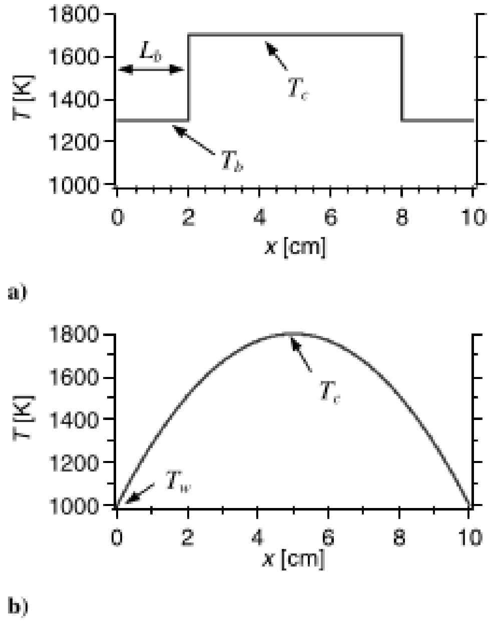

Figure 2.5: Postulated non-uniform temperature distribution profiles for confined combustion gases with cold walls.

In addtion the mole fraction of the absorbing species x_i can also have a non-uniform distribution though the medium. In the present study the goal is to estimate their non-uniform temperature profile from the measured molecular absorption spectrum using DAS assuming a constant x_i through the medium.

2.4 Methods to estimate temperature profile

- Fixed wavelength

  In it the laser's wavelength which is resonant with a molecules absorption transition of interest and constant in time (for Direct absorption (DA)) or sinusoidally modulated about a fixed wavelength (for Wavelength-modulation spectroscopy (WMS))

- Scanned wavelength

  In it the laser's wavelength is scanned across a portion of the molecules absorption transition of interest, and in WMS, this wavelength scanning is accompanied by simultaneous sinusoidal wavelength modulation performed at a much higher frequency (typically 50 to 100 times greater than scan rate).

One possibility to estimate the temperature profile from the DAS measurement data is using Radon transform.

- Radon transform:

  It is the integral transform which takes a function f defined on the plane to a function Rf defined on the (two-dimensional) space of lines in the plane, whose value at a particular line is equal to the line integral of the function over that line.

  Drawbacks:

  It required a lot of lines or data points to estimate the shape or the curve of underling profile.

Chapter 3

Related Work

3.1 Literature Review

In [4] Wang et al. introduced a novel approach in their study by employing Gaussian Process Regression (GPR) models, including both Single Output (SO)-GPR and Multi Output (MO)-GPR models, to estimate the line average temperature, as well as the concentrations of H_2O and CO_2 directly from absorption spectra. Their research demonstrated impressive results, showcasing an error rate of less than 3$\%$ relative Root Mean Square Error (RMSE) across various tests conducted on both SOGPR and MOGPR models using different configurations like LMC and ICM.

The findings of their study highlighted the effectiveness of both SOGPR and MOGPR models in accurately inferring temperature and gas concentrations from absorption spectra. While the MOGPR model offers the advantage of capturing correlations between multiple outputs, Wang et al. recommended the utilization of the SOGPR model due to its lower complexity and ease of implementation.

This research by Wang et al. not only contributes to the advancement of flame diagnostics through Gaussian Process Regression but also provides valuable insights into the practical considerations of choosing between SOGPR and MOGPR models based on factors such as complexity and implementation feasibility. Their study sets a strong foundation for further exploration and application of GPR techniques in flame diagnostics and related fields.

In [3] Goldenstein et al. conducted a comprehensive review focusing on the advancements in Infrared laser-absorption spectroscopy (IR-LAS) techniques, shedding light on the current capabilities, applications, and recent developments in this field. Their review paper delves into the diverse range of applications of IR-LAS and explores the design considerations and practical use cases of IR-LAS sensors for analyzing combustion gases.

By discussing the capabilities and advancements in IR-LAS, Goldenstein et al. provide a thorough overview of how this technique has evolved and diversified to address various objectives in combustion diagnostics and gas analysis. They highlight the versatility of IR-LAS in detecting and quantifying gases in combustion processes, environmental monitoring, and industrial applications, showcasing the potential for precise and real-time measurements.

Moreover, the review paper by Goldenstein et al. delves into the design aspects of IR-LAS sensors, elucidating the key considerations in sensor development and deployment for efficient gas analysis. By exploring the challenges faced in implementing IR-LAS sensors and discussing potential improvements, they pave the way for enhancing the accuracy, sensitivity, and reliability of IR-LAS technology in practical applications.

This review paper not only provides a comprehensive overview of the current landscape of IR-LAS techniques but also offers valuable insights into the future directions and opportunities for advancements in this field. By addressing the challenges and discussing potential avenues for improvement, Goldenstein et al. contribute significantly to the ongoing development and optimization of IR-LAS for combustion diagnostics and gas analysis applications.

In [5] Ma et al., studied on utilizing Boltzmann profile fitting for combustion temperature profiling through Tunable Diode Laser Absorption Spectroscopy (TDLAS) measurements, alongside experimental thermocouple data, presents a novel approach. Their comparison of results from a Computational Fluid Dynamics (CFD) model simulation, thermocouple measurements, and TDLAS profile fitting highlights the efficacy of the Boltzmann profile fitting method. This research can significantly contribute to enhancing our understanding of combustion processes and optimizing diagnostic techniques in this field.

In [6] Schulz et al.'s paper offers a comprehensive introduction to Gaussian Process Regression (GPR) for tackling unknown function problems. Starting with the fundamentals of Gaussian Processes (GPs), they progress towards regression, providing a detailed analysis and intuition behind the process. The discussion extends to optimizations, kernel selection, and prior encoding, offering valuable insights into the practical application of GPR. By concluding with a reflection on the limitations and restrictions of GPR, this paper equips readers with a well-rounded understanding of GPR methodology.

In [7] Álvarez et al.'s review delves into the diverse techniques for designing or learning valid kernel functions tailored for Mixture of Gaussian Process Regressors (MOGPR). They introduce coregionalization techniques such as LMC (Linear Model of Coregionalization) and ICM (Intrinsic Coregionalization Model) along with their extensions. Furthermore, they elaborate on non-separable kernel techniques like Process Convolutions (PC) to extend GPR for multi-output problems. By treating the multiple output problem as a regularization method that directly minimizes a loss function while constraining the parameters of the learned vector-valued function, they provide a comprehensive framework for addressing complex multi-output regression challenges.

3.2 Summary

The field of combustion diagnosis has seen a remarkable surge in the adoption of Tunable Diode Laser Absorption Spectroscopy (TDLAS), propelled by the evolution of advanced sensors and laser technologies. This uptick in usage has created an opportunity for the integration of cutting-edge Machine Learning (ML) techniques, enabling the efficient processing of vast datasets to derive meaningful insights swiftly and with exceptional precision. Addressing the complex task of estimating combustion temperature profiles has spurred the exploration of diverse methodologies, encompassing both traditional approaches and innovative ML-based solutions. Among these, Gaussian Process Regression (GPR) has emerged as a promising method gaining traction, complemented by the ongoing development and deployment of a spectrum of Multi-Output techniques aimed at enhancing its versatility and effectiveness across a broad spectrum of applications.

Chapter 4

Process Flow

4.1 Experimental Setup

The experimental setup is designed as such to get a non uniform temperature field while maintaining an uniform mole fraction (X_i) throughout the laser path. For this a stainless steel tube with a 1cm hole for passing the laser through was kept in a heated three zone furnace. Three K type thermocouples are used to monitor the temperature in the middle portion of the cell.

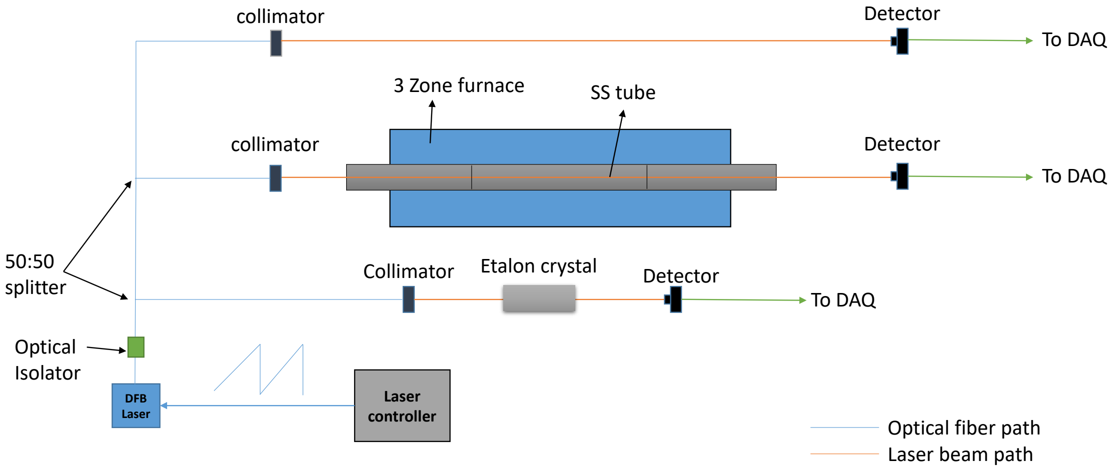

Figure 4.1: Schematic of tube in the furnace.

The furnace ensures that the middle zone of the cell maintains a constant temperature at a pre-set value. The temperature then gradually goes down to room temperature as it comes to the exit of the cell.

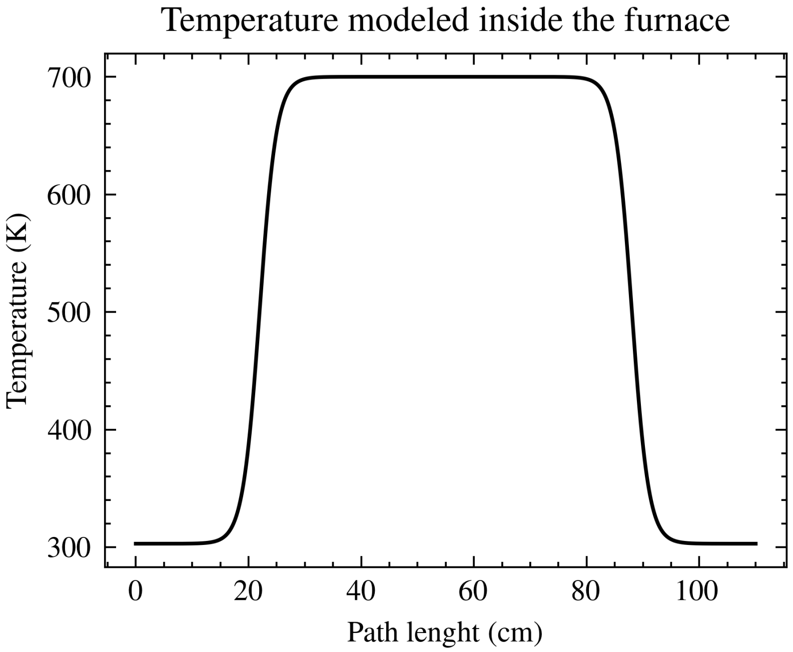

Figure 4.2: Temperature profile in the furnace.

A DFB laser centered at the molecular absorption transmission of H_2O molecule at 1343.3 nm is used in the experiment to generate the absorption spectrum. To capture the full line profile of the absorbing line; The laser current is modulated using a laser current controller at a frequency of 1Khz. The laser beam path is split into three. One path for wavelength calibration of the laser using an etalon crystal. One for getting a reference spectrum at ambient conditions and the third path passes through the heated cell. The resulting laser beam intensity after absorption by the H_2O molecule in the heated cell is collimated using a doublet and detected using a photodetector.The detector signal is post processed and absorption spectrum is obtained.

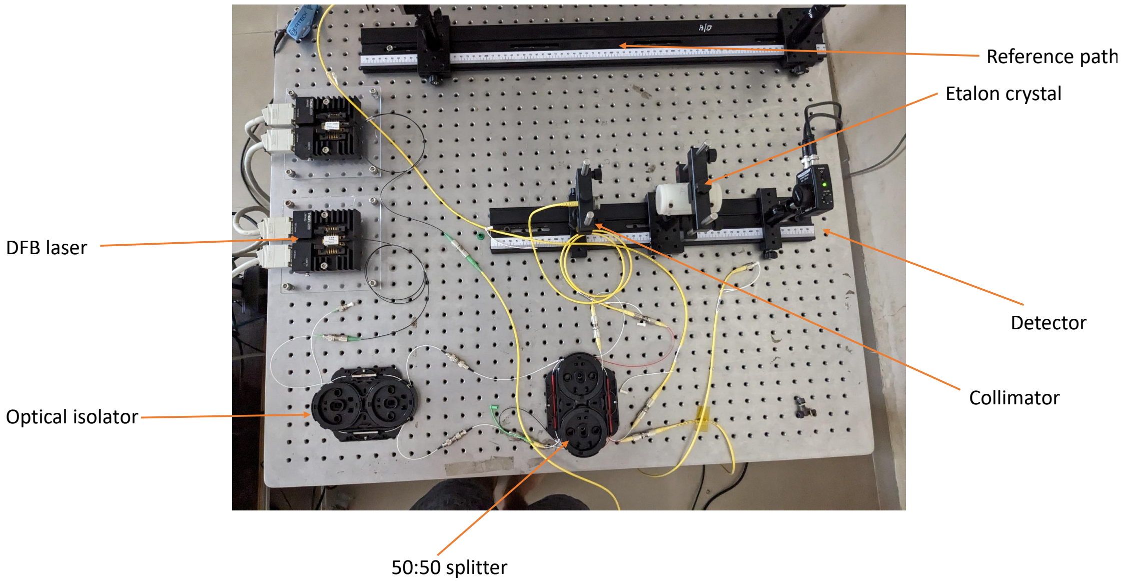

Figure 4.3: DFB laser setup.

4.2 Process flow chart

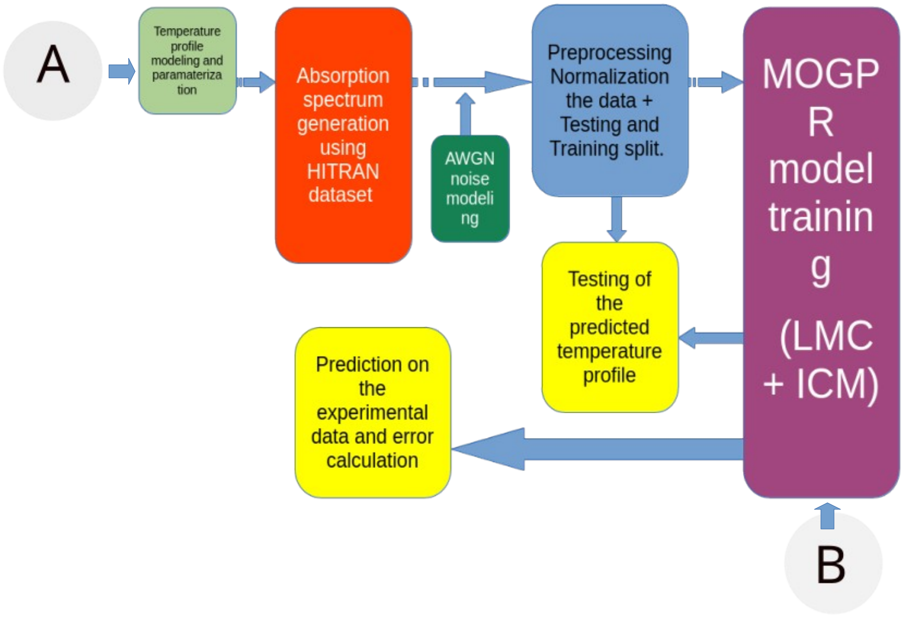
A
Temperature profile modeling and paramaterization
Absorption spectrum generation using HITRAN dataset
AWGN noise modeling
Preprocessing Normalization the data + Testing and Training split.
MOGPR model training (LMC + ICM)
Testing of the predicted temperature profile
Prediction on the experimental data and error calculation
B

Figure 4.4: Flow chart of MOGPR model.

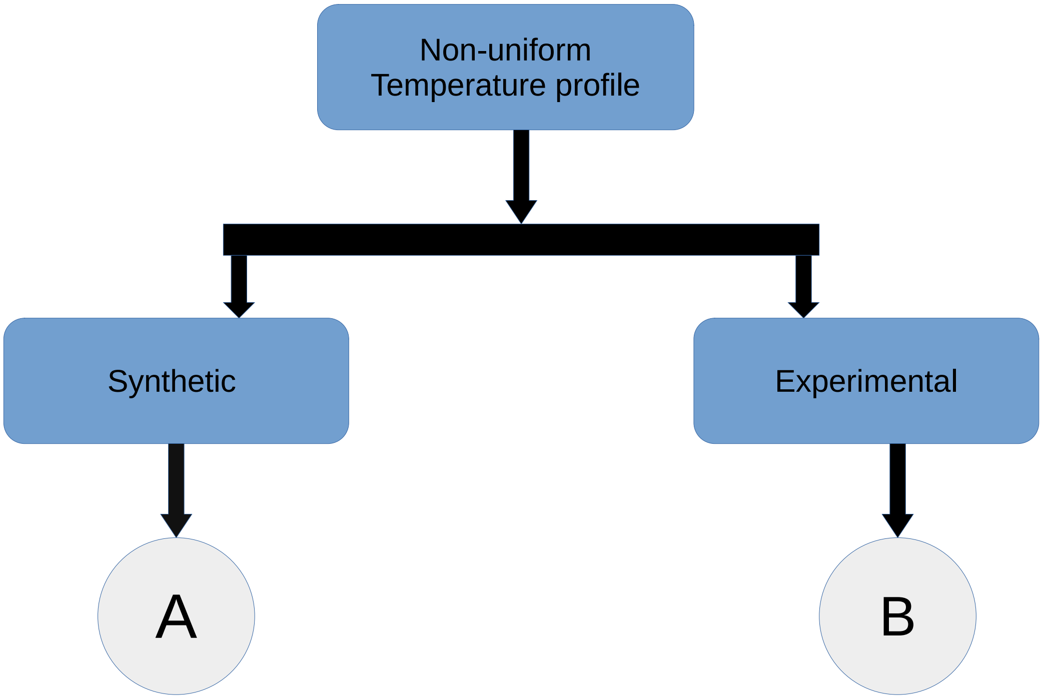

Figure 4.5: Data modeling.

- **Temperature profile modeling and parameterization**: This step involves creating a model to represent temperature profiles and determining the parameters that define these profiles.

- **Absorption spectrum generation using HITRAN dataset**: Utilizing the HITRAN dataset to generate absorption spectra, which is crucial for understanding how different gases interact with light at various wavelengths.

- **AWGN noise modeling**: Adding Additive White Gaussian Noise (AWGN) to the data to simulate noise that can occur in real-world scenarios.

- **Preprocessing - Normalization of data + Testing and Training split**: Preprocessing involves preparing the data for modeling by normalizing it and splitting it into training and testing sets to evaluate the model's performance.

- **MOGPR model training (LMC + ICM)**: Training a Gaussian Process Regression model using a combination of Linear Model of Coregionalization (LMC) and Independent Coregionalization Model (ICM) techniques to predict temperature profiles.

- **Testing of the predicted temperature profile**: Evaluating the accuracy and performance of the model by testing the predicted temperature profiles against known data.
- **Prediction on the experimental data and error calculation**: Applying the trained model to experimental data to make predictions and calculating the error between the predicted values and the actual observations.

4.3 Dataset

The spectral dataset for the H_2O molecule was sourced from HITRAN database [8] which contains spectroscopic parameters which are used to predict and simulate the transmission and emission of electromagnetic wave in the atmosphere.

For this project we have selected Water Vapor (H_2O) as species of interest and we are observing the absorption effect in the wavelength region 7443.9 to 7445.1 cm^{-1}.

To generate the training data we have simulated a Boltzmann temperature profile parameterized by 3 variables, which is used in tandem with HITRAN database to generate the features i.e. in this case is the absorption spectrum and the parameterized values are used as the labels.

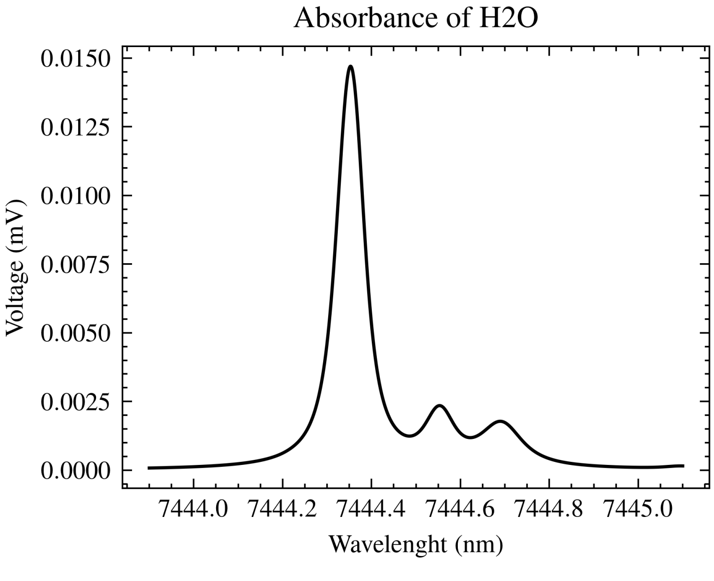

Figure 4.6: The absorbance of H2O in the given wavelength limits.

Further more we have assumed the mole fraction of the species to be constant though out the experiment and for all cases which is taken to be as: **0.026**

Pressure: **1 bar**

Ambient temperature (Room temperature): **303 K**

Path length: **110 cm**

Total dataset samples generated: **7000**

4.4 Noise Modeling

Further to simulate real world conditions simulated noise was added to the absorbance.

To model the noise we calculated the Signal to Noise Ration (SNR) of a test experimental signal and Additive White Gaussian Noise (AWGN) proportional to the SNR was added to the absorbance sigal.

Value of SNR: **33.5505**

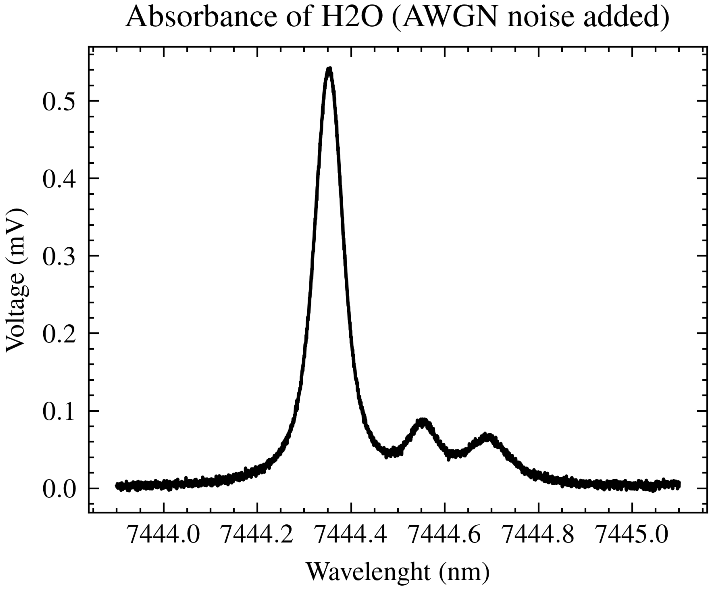

Figure 4.7: The absorbance of H2O in the given wavelength limits (With AWGN noise added).

4.5 Modeling of sample temperature profile

The modeling of the non-uniform temperature profile in the flame in a high-temperature system was done using Boltzmann fitting profile to reduce prediction overhead in the modeling of multiple points of the profile. Instead by using a fitting we are able to effectively parameterize the temperature profile of a with few parameters.

$$y = A_2 + \frac{A_1 - A_2}{1 + e^{x - x_0/A_3}}$$

y is the flame temperature, A_1 is the central uniform temperature, A_2 is the ambient temperature (Room temperature), A_3 is a describes the level of transition gradient, and x_0 indicates the radial position where the flame temperature is equal to $(A_1 - A_2)/2$.

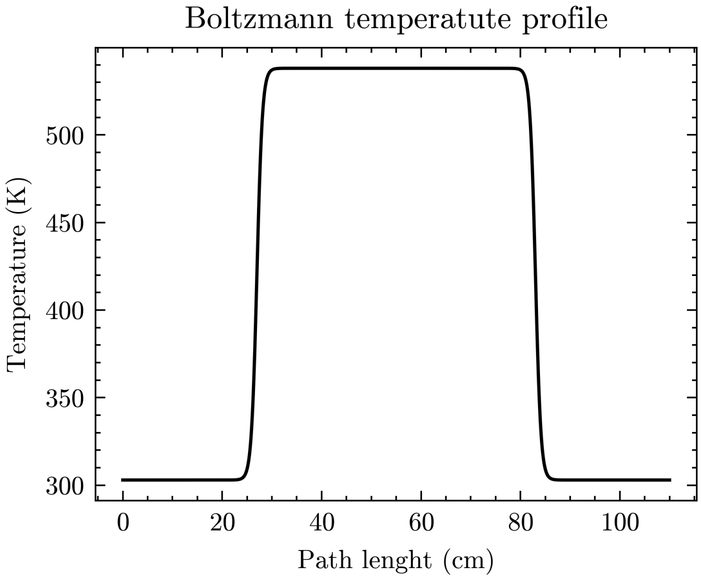

Figure 4.8: The modeled Boltzmann temperature profile.

Multiple profiles can be generated by changing values of parameters in equation above

A_1 is varied from 303 K to 773 K.

$A_2 = 303$ K as the ambient temperature.

A_3 is varied from 1 to 10.

x_0 is varied from 0 to 55 (simulating half of the pathlenght of the burner).

These three variables $A_1, A_3 \& x_0$ are varied to generate multiple test and training temperature profiles.

4.6 MOGPR Modeling

4.7 Dataset

The spectral dataset for H_2O was sourced from the HITRAN database [8]. ...

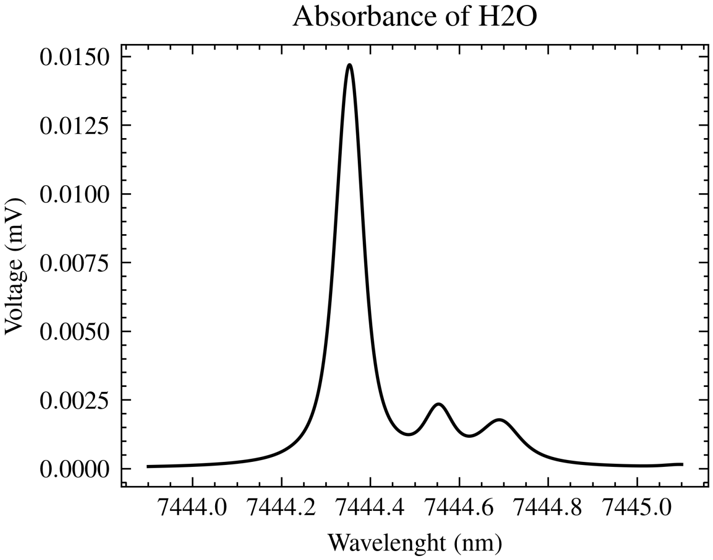

Figure 4.9: Absorbance of H2O in the given wavelength limits.

...

4.8 Noise Modeling

To simulate real-world conditions, simulated noise was added to the absorbance. ...

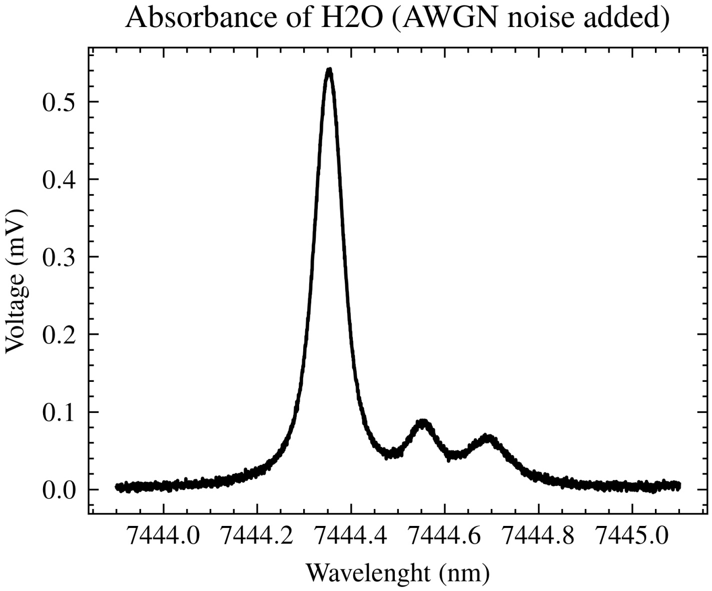

Figure 4.10: Absorbance of H2O in the given wavelength limits (with AWGN noise added).

...

4.9 Modeling of Sample Temperature Profile

The non-uniform temperature profile in the flame was modeled using a Boltzmann fitting profile to reduce prediction overhead. ...

$$y = A_2 + \frac{A_1 - A_2}{1 + e^{x - x_0/A_3}}$$

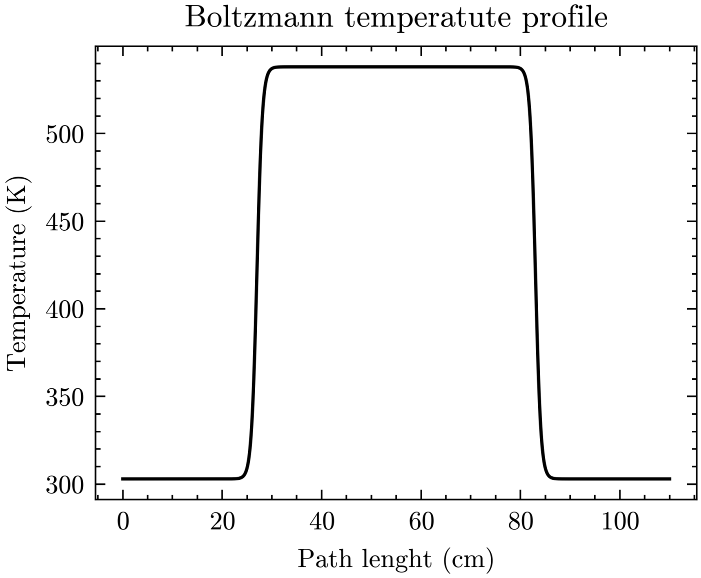

Figure 4.11: Modeled Boltzmann temperature profile.

To realize problem with MOGRP

- **Utilization of GPy**: GPy [9], a Python library developed by SheffieldML, was employed to address the challenges associated with MOGPR. This library provides functionalities for Gaussian processes, making it a valuable tool for implementing GPR models effectively.

- **Data Normalization**: The data used for modeling was normalized using the StandardScaler function from SciPy [10] . Normalizing data helps in standardizing the scale of features, ensuring that all variables contribute equally to the model's performance.

- **LMC and ICM Techniques**: The incorporation of LMC and ICM techniques allows for the modeling of correlations between multiple output parameters and input features. LMC enables the modeling of complex relationships between outputs, while ICM provides a flexible framework for capturing non-linearities and dependencies in the data.

- **Model Saving and Reloading**: The Pickle [11] module in Python was utilized for saving and reloading models and variables. This allows for the preservation of trained models and data, enabling easy access and reuse without the need to retrain the model each time.

- **Feature and Target Data**: The absorbance spectrum, with a size of [1x3000], was used as a feature for the MOGPR models. These models were tasked with predicting temperature profile parameters of size [1x3] post normalization, indicating the prediction of specific parameters based on the absorbance spectrum data.

- **Choice of Kernel**: In GPR modeling, the Radial Basis Function (RBF) or Gaussian Kernel was used as the kernel function. The kernel function determines the similarity between data points and plays a crucial role in the model's ability to capture complex patterns in the data.

- **Model Convergence**: The models were allowed to run until convergence, indicating that the training process continued until the models reached a stable state where further iterations did not significantly improve performance.

4.9.1 Loss Calculation

To assess the quality of the predicted temperature profile in the context provided, the Root Mean Squared Error (RMSE) function is used:

Total Points and Test Profiles: The total number of points in each temperature profile is denoted as D, and the total number of test temperature profiles is represented by H.

- **RMSE Maximum** (**$RMSE_{max}$**): This metric calculates the maximum RMSE value across all test temperature profiles. It is determined by finding the square root of the sum of squared differences between the predicted (x_i) and actual (y_i) values for each point in the temperature profile, for each test profile, and selecting the maximum value among them.

- **RMSE Minimum ($RMSE_{min}$)**: Conversely, the RMSE minimum assesses the minimum RMSE value across all test temperature profiles. It computes the square root of the sum of squared differences between predicted and actual values for each point in the temperature profile, for each test profile, and identifies the minimum RMSE value.

- **RMSE Average ($RMSE_{avg}$)**: The RMSE average calculates the average RMSE value across all test temperature profiles. It involves summing the RMSE values for each test profile (calculated similarly to the maximum and minimum RMSE) and dividing by the total number of test profiles (H) to obtain an average RMSE value.

$$RMSE_{max} = MAX_H(\sqrt{\sum_{i=1}^{D}(x_i - y_i)^2})$$

$$RMSE_{min} = MIN_H(\sqrt{\sum_{i=1}^{D}(x_i - y_i)^2})$$

$$RMSE_{avg} = \frac{1}{H}\sum_{i=1}^{H}(\sqrt{\sum_{i=1}^{D}(x_i - y_i)^2})$$

Chapter 5

Results and Analysis

5.1 Methodology

Both AWGN modeled noise added and noise free simulated data were used to train MOGPR models.

7000 total data points were generated for each case which was further split **80:20** for training and testing in MOGPR

Total training data samples: **5000**

Total testing data samples: **2000**

Both ICM and LMC MOGPR methods were used and with two noise variations total 4 models were trained as follows.

Table 5.1 shows the models trained.

Table 5.1: MOGPR Model List

Models	**Methods**	**Noise**
α MOGPR	LMC	Noise Free
β MOGPR	LMC	AWGN Noise
γ MOGPR	ICM	Noise FreeTemperature
θ MOGPR	ICM	AWGN Noise

5.2 Results

Table 5.2 shows the models tested on the generated dataset.

Table 5.3 shows the models tested on the experimental data point.

Table 5.2: MOGPR Model Testing Error

Models	$RMSE_{max}$	$RMSE_{min}$	$RMSE_{avg}$
α MOGPR	458.5093	0.1938	22.3659
β MOGPR	87.7101	4.4214e-04	19.9165
γ MOGPR	410.5743	0.0965	27.7380
θ MOGPR	82.8912	7.923e-05	20.2450

As measurement of the entire temperature profile inside the heated cell was not possible, only the peak flame temperature were compared and the percentage difference show in the table below

Table 5.3: MOGPR Model Testing Error

Models	**Error (%)**
α MOGPR	20.0594
β MOGPR	20.0593
γ MOGPR	20.0593
θ MOGPR	20.0594

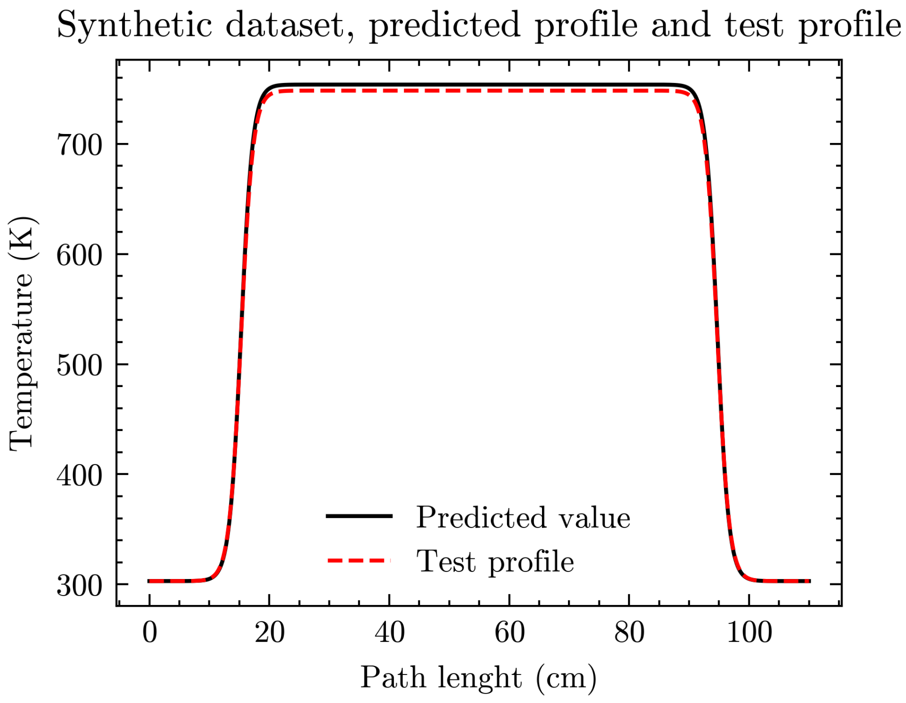

Figure 5.1: The non-uniform temperature profile predicted by the MOGPR model along with the test profile.

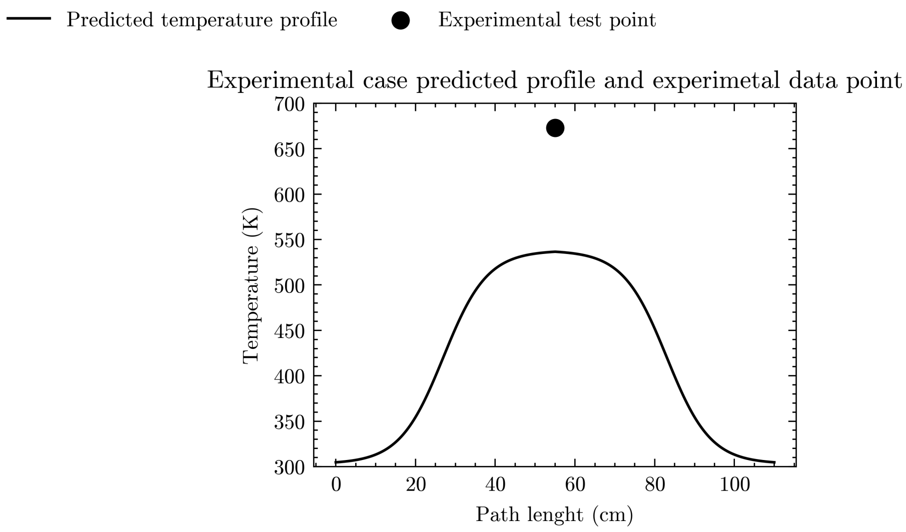

Figure 5.2: Predicted profile for the experimental test case along with the experimental test value of the maximum temperature in the heated tube.

5.3 Observations

From the information provided, it appears that in the simulated dataset, the β MOGPR (LMC + AWGN noise) model showed the best testing loss, followed closely by the β MOGPR (ICM + AWGN noise) model. It's interesting to note that the noise-modeled models outperformed the noise-free trained models in this scenario. This suggests that incorporating noise modeling into the training process can lead to more accurate predictions in simulated datasets.

The observation that ICM models are performing consistently while LMC models are more accurate but less consistent aligns with the strengths and weaknesses typically associated with these two modeling approaches. ICM's consistency across datasets can be advantageous in certain contexts where stability and reliability are paramount. On the other hand, LMC's ability to achieve higher accuracy, albeit with some variability, may be preferred in situations where maximizing predictive performance is the primary goal.

However, it's worth noting that in the real-world test case, all models exhibited very similar error values. This convergence of error values across different modeling approaches in a real-world setting highlights the complexities and uncertainties inherent in applying machine learning models to practical, uncontrolled environments. It underscores the importance of considering real-world factors, such as those mentioned earlier, and adapting modeling strategies accordingly to account for the inherent variability and unpredictability of real-world data.

5.4 Conclusions

LMC samples from different GPs it can fit data better but is very sensitive to the data and noise present where as ICM is only sampled from one GP is consistent across dataset and is resilient to noise.

In the real word testing can be affected by following factors:

- Modeling of the real world temperature profile.

  When it comes to modeling real-world scenarios, the temperature profile plays a crucial role in accurately representing the environment under study. Variations in temperature can impact the behavior of the system being analyzed, so it's essential to have an accurate model of temperature changes over time.

- Noise modeling in the simulated data.

  Noise modeling is another critical aspect that can significantly affect the results of simulations. Understanding and accounting for noise in the data is essential for obtaining reliable and meaningful outcomes. Different noise levels can introduce uncertainties and influence the performance of the models, so it's important to address this factor appropriately.

- Effect of external interference.

  External interference, such as electromagnetic signals or other environmental factors, can also impact the accuracy of experimental results. Shielding against external interference and considering its potential effects on the data are important steps in ensuring the reliability of the findings.

- Change in mole fraction (X) during the experiment.

  Changes in mole fraction (X) during the experiment can introduce complexities that need to be carefully monitored and accounted for in the analysis. Understanding how X evolves over time and its impact on the system is crucial for drawing accurate conclusions from the experiments.

- Amount of synthetic samples used.

The number of synthetic samples used in the analysis can affect the robustness and generalizability of the results. Using an appropriate amount of synthetic data to supplement the experimental data can help improve the reliability of the models and enhance their predictive capabilities.

- Low quantity of experimental data.

  Lastly, having a low quantity of experimental data can limit the insights that can be gained from the analysis. Increasing the amount of experimental data collected can provide a more comprehensive understanding of the system under study and lead to more robust conclusions.

Additionally, it's important to note that the experimental test was conducted on a single set of experimental data, limiting our ability to draw significant conclusions from the results. Conducting tests on a broader range of real-world data sets would provide a more comprehensive understanding of the model performance in practical scenarios.

5.5 Future Scope

Further refinement of the project can indeed enhance the accuracy and robustness of the models. Here are some suggestions to consider:

- Using specialized or optimized kernel for this problem.

  Tailoring the choice of kernel functions to the specific characteristics of the problem can improve model performance. Optimizing the kernel selection based on domain knowledge or through experimentation can lead to more accurate and efficient models.

- Usage of huge synthetic dataset.

  Increasing the size of the synthetic dataset can provide more diverse samples for training the models. A larger dataset can help capture a broader range of patterns and variations in the data, leading to more robust and generalizable models.

- Better noise modeling.

  Enhancing the representation of noise in the data can lead to more accurate modeling and better generalization to unseen data. Refining the noise modeling techniques to better reflect the underlying uncertainties in the data can help improve the model's performance.

- Usage of experimental data for training.

  Integrating real experimental data into the training process can enhance the model's ability to capture the complexities of real-world scenarios. Combining simulated data with experimental data can provide a more comprehensive and realistic training set for the models.

- Usage of Deep Learning (DL) techniques.

  Leveraging Deep Learning methods, such as neural networks, can offer more sophisticated modeling capabilities and potentially uncover complex patterns in the data. Deep Learning techniques can handle high-dimensional data and nonlinear relationships effectively, which may be beneficial for capturing intricate relationships in the dataset.

Bibliography

[1] J. Wolfrum, “Lasers in combustion: From basic theory to practical devices,” 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:96265219

[2] R. K. Hanson, “Applications of quantitative laser sensors to kinetics, propulsion and practical energy systems,” 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:111284091

[3] C. S. Goldenstein, R. M. Spearrin, J. B. Jeffries, and R. K. Hanson, “Infrared laser-absorption sensing for combustion gases,” *Progress in Energy and Combustion Science*, vol. 60, pp. 132–176, 2017.

[4] W. Wang, Z. Wang, and X. Chao, “Gaussian process regression for direct laser absorption spectroscopy in complex combustion environments,” *Optics Express*, vol. 29, no. 12, pp. 17 926–17 939, 2021.

[5] L. H. Ma, L. Y. Lau, and W. Ren, “Non-uniform temperature and species concentration measurements in a laminar flame using multi-band infrared absorption spectroscopy,” *Applied Physics B*, vol. 123, pp. 1–9, 2017.

[6] E. Schulz, M. Speekenbrink, and A. Krause, “A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions,” *Journal of Mathematical Psychology*, vol. 85, pp. 1–16, 2018.

[7] M. A. Alvarez, L. Rosasco, N. D. Lawrence *et al.*, “Kernels for vector-valued functions: A review,” *Foundations and Trends® in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.

[8] I. E. Gordon, L. S. Rothman, and H. et al., “The HITRAN2020 molecular spectroscopic database,” *jqsrt*, vol. 277, p. 107949, Jan. 2022.

[9] GPy, “GPy: A gaussian process framework in python,” http://github.com/SheffieldML/GPy, since 2012.

[10] P. Virtanen and G. et al.1 SciPy 1.0 Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.

[11] G. Van Rossum, *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

Appendix A

Data Generation Code

A.1 Absorption Spectra

```
# Python translation to the TDLAS MATLAB code with annotation and vali
The script version of code contains individual py files, for function
Here, they are combined in systematic order along with anootations on

The variable and function names used in this script are the same as th
## Input parameters definition
This section contains the definitions of parameters that are used in t
# HITRAN database filename

loc_file = '/home/saumya/Documents/Flame_dianostics-temperature_profile

HITRAN_filename          = loc_file + '01_7000-7500_HITEMP2010_7443.2-744

# partition filename
PART_filename            = "q1.txt"

# minimum and maximum wave numbers in cm^-1
nu_1_min                 = 7443.9
nu_1_max                 = 7445.1

# number of terms between min ans max wave numbers to be considered
N_nu_range               = 3000
```

```
# total radial distance in cm and radial step size
total_radial_distance = 3
delta_x               = 0.01

# parameters for temperature and mole fraction fields construction----------
# maximum flame temperature and ambient temperature
T_mag_uni_field       = A1 = 950.0
A2                    = 300.0

# location where flame temperature reaches half the maximum temperature
X0                    = 6.0*total_radial_distance/7.0

# gradient of the transition region
A3                    = 0.06

# maximum and ambient mole fraction of water vapour
x_mag_uni_field       = B1 = 0.095
B2                    = 0.028
Following are the constants used in the work
# reference temperature
T0 = 296.0

# Plank's constant
h  = 6.62607004e-34

# speed of light in cm/s in vacuum
c  = 29979245800

# Boltzmann constant
k  = 1.38064852e-23

# molar mass of water
M  = 18.01
```

```
P  = 1 #  <----------- DOUBT   /unit = ATM

# Avogadro number
Na = 6.02214076e23
---

## Importing libraries/modules
The needed modules and libraries are imported in this section
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.special import wofz     # Voigt function



from multiprocessing import Pool
from tqdm import *
---

## Function definitions
### HITRAN database reader function
This function was developed to read the Hi-resolution Tranmission datal
This will return the pandas dataframe with all the columns
def HITRAN_reader(fname):

# reading all file contents
fid = open(fname,"r")
lines = fid.readlines()

# preparing lists to store line values
molNo_list = []; isoNo_list = []; transitionWaveNo_list = []
lineIntensity_list = []; einsteinACoeff_list = []; airBroadenedWidth_l
selfBroadenedWidth_list = []; lowEState_list = []; tempCoeff_list = []
pressureShift_list = []; upperVibQuanta_list = []; lowerVibQuanta_list
upperLocalQuanta_list = []; lowerLocalQuanta_list = []; errorCodes_lis
```

```
referenceCodes_list = []; flagLineMixing_list = []
upperStatisticalWeight_list = []; lowerStatisticalWeight_list = []


# looping through the lines to read individual data
for line in lines:
# setting starting read position on the string
idx = 0

# reading molecule number
molNo = int(float(line[idx:idx+2]))
idx+=2

# isotopologue number
isoNo = int(float(line[idx:idx+1]))
idx+=1

# transition wavenumber / vaccum wavenumber
transitionWaveNo = float(line[idx:idx+12])
idx+=12

# line intensity
lineIntensity = float(line[idx:idx+10])
idx+=10

# einstein A-coefficient
einsteinACoeff = float(line[idx:idx+10])
idx+=10

# air-broadened half width / air-broadened width
airBroadenedWidth = float(line[idx:idx+5])
idx+=5

# self-broadened half width / self-broadened width
selfBroadenedWidth = float(line[idx:idx+5])
```

```
idx+=5

# lower energy state
lowEState = float(line[idx:idx+10])
idx+=10

# temperature dependence coefficient
tempCoeff = float(line[idx:idx+4])
idx+=4

# pressure shift / air-pressure-induced line shift
pressureShift = float(line[idx:idx+8])
idx+=8

# upper vibrational quanta / upper-state "global" quanta
upperVibQuanta = line[idx:idx+15]
idx+=15

# lower vibrational quanta / lower-state "global" quanta
lowerVibQuanta = line[idx:idx+15]
idx+=15

# upper local quanta / upper-state "local" quanta
upperLocalQuanta = line[idx:idx+15]
idx+=15

# lower local quanta / lower-state "local" quanta
lowerLocalQuanta = line[idx:idx+15]
idx+=15

# error codes / uncertainity indices
errorCodes = line[idx:idx+6]
idx+=6

# reference codes / reference indices
```

```
referenceCodes = line[idx:idx+12]
idx+=12

# flag for lineMixing / Flag
flagLineMixing = line[idx:idx+1]
idx+=1

# upper statistical weight / statistical weight for the upper state
upperStatisticalWeight = float(line[idx:idx+7])
idx+=7

# lower statistical weight / statistical weight for the lower state
lowerStatisticalWeight = float(line[idx:idx+7])
idx+=7

# appending read values to the lists
molNo_list.append(molNo)
isoNo_list.append(isoNo)
transitionWaveNo_list.append(transitionWaveNo)
lineIntensity_list.append(lineIntensity)
einsteinACoeff_list.append(einsteinACoeff)
airBroadenedWidth_list.append(airBroadenedWidth)
selfBroadenedWidth_list.append(selfBroadenedWidth)
lowEState_list.append(lowEState)
tempCoeff_list.append(tempCoeff)
pressureShift_list.append(pressureShift)
upperVibQuanta_list.append(upperVibQuanta)
lowerVibQuanta_list.append(lowerVibQuanta)
upperLocalQuanta_list.append(upperLocalQuanta)
lowerLocalQuanta_list.append(lowerLocalQuanta)
errorCodes_list.append(errorCodes)
referenceCodes_list.append(referenceCodes)
flagLineMixing_list.append(flagLineMixing)
upperStatisticalWeight_list.append(upperStatisticalWeight)
lowerStatisticalWeight_list.append(lowerStatisticalWeight)
```

```
#  break

# column names for pandas dataframe
colNames = ["moleculeNumber","isotopologueNumber","transisionWaveNumbe
"lineIntensity","einsteinACoefficient","airBroadenedWidth",
"selfBroadenedWidth","lowerStateEnergy","temperatureDependence",
"pressureShift","upperVibrationalQuanta","lowerVibrationalQuanta",
"upperLocalQuanta","lowerLocalQuanta","errorCodes",
"referenceCodes","flagForLineMixing","upperStatisticalWeight",
"lowerStatisticalWeight"]

# preparing dataframe
fid = pd.DataFrame(np.transpose([molNo_list,isoNo_list,transitionWaveN
lineIntensity_list,einsteinACoeff_list,
airBroadenedWidth_list,selfBroadenedWidth_list,
lowEState_list,tempCoeff_list,pressureShift_list,
upperVibQuanta_list,lowerVibQuanta_list,
upperLocalQuanta_list,lowerLocalQuanta_list,
errorCodes_list,referenceCodes_list,flagLineMixing_list,
upperStatisticalWeight_list,lowerStatisticalWeight_list]),
columns = colNames)

# writing dataframe to csv file
fid.to_csv("HITEMP_data.csv", index = None)

# returning dataframe
return fid
### Partition function interpolator
This function interpolates the partition function values over a range
def Q_func(T, T_series, Q_series):
"""
This function obtains the value of partition function for given temper
through linear interpolation
```

```
T        - given input temperature
T_series - series of temperature values from database
Q_series - series of partition function values corresponding to each
temperature in the database
"""

return np.interp(T,T_series,Q_series)
### Line strength function
This function computes the line strength using the following equation
$$ S(T) = S(T_0) \frac{Q(T_0)}{Q(T)}\frac{T_0}{T} exp\left(\frac{-h c E"}{k
def␣lineStrength(S0,␣nu0,␣Edd,␣T,␣fid_Q):
"""
This function computes the line strength of a particular transition
at a given temperature

equation has been split into 4 parts for ease of programming
"""

part_1 = Q_func(T0,fid_Q["T"],fid_Q["Q"])/Q_func(T,fid_Q["T"],fid_Q["Q"])
part_2 = T0/T*np.exp(-h*c/k*Edd*(1/T - 1/T0))
part_3 = 1 - np.exp(-h*c*nu0/k/T)
part_4 = 1 - np.exp(-h*c*nu0/k/T0)

S = S0*part_1*part_2*part_3/part_4

return S
### Spectral absorption coefficient function
This function computes the spectral absorption coefficient for a given x ov

Here,
the $a$  equation is given below, it describes the gausian half-width at ha
$$ a = 2 \sqrt{log(2)} \frac{\nu - \nu_0}{\alpha_G} $$

the $w$ equation which describes the lorenztian half-width at half maximum
$$ w = \sqrt{log(2)} \frac{\alpha_L}{\alpha_G} $$
```

```
Then the $\phi$ function value is computed with the equation described
$$ \phi(\nu) = \frac{2}{\alpha_G}\sqrt{\frac{log(2)}{\pi}}V(z) \; \; \

Finally, the value of spectral absorption coefficient is computed by t
$$ K(x) = \left(S(T) \times \chi \times P\right) . \phi(\nu,T) $$

The name of the function is given as per the MATLAB code,

def voigt(dat, nu0, s, delnu_g, delnu_l):
"""
this function computes value of K (spectral absorption coefficient)
at given X location for all wave numbers.

The name of this program function is given  as per the matlab code
"""

# computing difference between chosen wavenumber ranges and wave numbe
# from the HITRAN database
""" for this, the matrix is obtained by subtracting each element of on
arrau over each other elements of the other array"""
vv0 = nu0 - dat.reshape(dat.shape[0],1) #  it is (nu-nu0) in the equat

# computing "a" equation from problem definition
"""
DOUBT:
here a 2 is missing in the code but present in the pblm definition
"""
a = np.sqrt(np.log(2))*vv0/delnu_g

# computing "w" equation from problem definition
w = np.sqrt(np.log(2))*np.ones([dat.shape[0],1])*delnu_l/delnu_g

# combining w and a into a complex function to feed it to voigt functio
z = a+w*1j
```

```
# computing phi function
"""
DOUBT:
here a 2 is missing in the code but present in the pblm definition
"""
phi = 1/delnu_g*np.sqrt(np.log(2)/np.pi)*wofz(z)

# computing the K function in problem definition for current X value
K = np.matmul(np.real(phi),s)

return K


---


## Computing the Non-uniform Absorption Spectra
Reading the HITRAN database and the partition function table
# reading data from HITRAN file
fid_HITRAN = HITRAN_reader(HITRAN_filename)

# reading Partition function table
fid_Q = pd.read_csv(PART_filename, header = None, delim_whitespace = True,
names = ["T","Q"])
Extracting individual columns of data from the database and computing a ran
# reading data from the database file
# center wavenumber
nu0  = fid_HITRAN["transisionWaveNumber"].to_numpy().astype("float64")
# reference line strength
s0   = fid_HITRAN["lineIntensity"].to_numpy().astype("float64")
# air-broadened width
g_a0 = fid_HITRAN["airBroadenedWidth"].to_numpy().astype("float64")
# self-broadened width
g_s0 = fid_HITRAN["selfBroadenedWidth"].to_numpy().astype("float64")
# lower state energy
Edd0 = fid_HITRAN["lowerStateEnergy"].to_numpy().astype("float64")
```

```
# temperature dependence exponent
n      = fid_HITRAN["temperatureDependence"].to_numpy().astype("float64"
# pressure shift coefficient
d0     = fid_HITRAN["pressureShift"].to_numpy().astype("float64")

# computing a range of wave numbers
dat1 = np.linspace(nu_1_min, nu_1_max, N_nu_range)
Filtering the entries from database based on the following criteria
- line intensity should be greater than $10^{-30}$
- dropping extreme wave numbers
# selecting the indices of entries with nu0 falling b/w extreme nu val
# with line intensity <= 10^-30
index1 = (nu0 > nu_1_min) & (nu0 < nu_1_max)
index2 = s0 > 1e-35
index  = index1 & index2

nu1    = nu0[index]
s01    = s0[index]
g_a01  = g_a0[index]
g_s01  = g_s0[index]
Edd1   = Edd0[index]
n1     = n[index]
d01    = d0[index]
pressure is taken to be contstant, hence, multiplying the line strengt
S0 = s01*2.479371939e19*P # <--------------- DOUBT        UNIT CONVERSI
Constructing the temperature and mole fraction profiles along with dis
def calc_abs(loc=700, xon=.026, *args, **kwargs):
# temperature and mole fraction fields construction--------------------

# loc_pro = loc
# name = 'Temp_profile.npy'

# Temp_pofiles = np.load(loc_pro + name)

# discretized radial distance
```

```
X = np.arange(0,total_radial_distance+delta_x ,delta_x)

# non uniform temperature field
# Tnon = A2+(A1-A2)/(1+np.exp((X-X0)/A3))

Tnon =  np.ones((300))*loc

# Tnon = Temp_pofiles[100]

# non uniform mole fraction field
# xnon = B2+(B1-B2)/(1+np.exp((X-X0)/A3))

xnon = np.ones((300))*xon


# obtaining the size of s0 and Tnon for 2D array pre-allocations
ns = S0.shape[0]
nt = Tnon.shape[0] - 1 #  <---------------- DOUBT, nt+1 is actually nx,,
it is due to section of absorbance...

# preallocating 2d arrays
s       = np.zeros([nt,ns]) # line strength
d       = np.zeros([nt,ns]) # pressure shift coefficient
g_a     = np.zeros([nt,ns]) # air broadened width
g_s     = np.zeros([nt,ns]) # self broadened width
nu0_s   = np.zeros([nt,ns]) # shifted center-wavenumber
delnu_g = np.zeros([nt,ns]) # gaussian half-width at halft maximum
delnu_l = np.zeros([nt,ns]) # lorentzian half-width at halft maximum

# computing spatial step size
dx = np.diff(X)


# looping through all x-locations and given line intensities
for i in range(ns):
```

```
for j in range(nt):
# getting current mole fractionp.shape(Abs).shapen value
x = xnon[j]

# computing the pressure shift coefficient
d[j,i] = d01[i]*(T0/Tnon[j])**0.96

# air broadened width
g_a[j,i] = g_a01[i]*(T0/Tnon[j])**n1[i]

# self broadened width
g_s[j,i] = g_s01[i]*(T0/Tnon[j])**0.75

# shifted center wave nuber
nu0_s[j,i] = nu1[i] + P*(1-x)*d[j,i]

# line strength
s[j,i] = lineStrength(S0[i], nu0_s[j,i], Edd1[i], Tnon[j], fid_Q)*dx[j

# gaussian HWHM
delnu_g[j,i] = np.sqrt(2*k*np.log(2)*Na/c**2*1000*10**4)*nu0_s[j,i]*np
# delnu_g[j,i] = 0.5*7.162242257e-7*nu0_s[j,i]*np.sqrt(Tnon[j]/M)

# lorentian HWHM
delnu_l[j,i] = P*(x*g_s[j,i]+(1-x)*g_a[j,i])



# pre-allocating absorbance array for all wave numbers
"""
it is -ln(I(nu)/I_0)
"""
Absorbance = np.zeros([N_nu_range])

# computing absorbance coef
# ficient K for each x and suming it up for total
for i in range(X.shape[0]-2): # <--------------- DOUBT
Absorbance += voigt(dat1, nu0_s[i,:], s[i,:], delnu_g[i,:], delnu_l[i,:])

return Absorbance
def repeat(arr, count):
return np.ravel(np.stack([arr for _ in range(count)], axis=0))
def run_apply_async_multiprocessing(func, argument_list, num_processes):

pool = Pool(processes=num_processes)

jobs = [pool.apply_async(func=func, args=(*argument,)) if isinstance(argume
pool.close()
result_list_tqdm = []
for job in tqdm(jobs):
result_list_tqdm.append(job.get())

return result_list_tqdm

# data_ph = np.load(’/home/saumya/Documents/Flame_dianostics-temperature_pr
# data_ph = argument_list = [(random.randint(1500, 2200), random.randint(1,
spect = calc_abs()
import scienceplots

plt.style.use([’science’,’ieee’])
plt.plot(dat1,spect)
plt.title(’Absorbance of H2O’)
plt.ylabel(’Voltage (mV)’)
plt.xlabel(’Wavelenght (nm)’)
plt.savefig(’H20ABS’)
s
# print("Running apply_async multiprocessing for multi-argument functions .


# Abs = run_apply_async_multiprocessing(func=calc_abs, argument_list=data_p
# # assert result_list == data_ph
# for i in xor_pro:
#       np.append(Abs, imap_unordered_bar(calc_abs, [temp_pro, xor_pro])
# absorption = calc_abs([1700, 0.2])
# Tnon = np.ones_like(Tnon)*Tnon.max()

# xnon = np.ones_like(xnon)*xnon.max()
Creating 2D matrices with size being the number of line strength entit
and computing the spatial step size
# # obtaining the size of s0 and Tnon for 2D array pre-allocations
# ns = S0.shape[0]
# nt = Tnon.shape[0] - 1 # <----------------- DOUBT, nt+1 is actually
it is due to section of absorbance...

# # preallocating 2d arrays
# s       = np.zeros([nt,ns]) # line strength
# d       = np.zeros([nt,ns]) # pressure shift coefficient
# g_a     = np.zeros([nt,ns]) # air broadened width
# g_s     = np.zeros([nt,ns]) # self broadened width
# nu0_s   = np.zeros([nt,ns]) # shifted center-wavenumber
# delnu_g = np.zeros([nt,ns]) # gaussian half-width at halft maximum
# delnu_l = np.zeros([nt,ns]) # lorentzian half-width at halft maximum

# # computing spatial step size
# dx = np.diff(X)
Looping through all the filtered database entries and the x-locations
- Pressure shift coefficient
$$ \delta_{p} = \delta_0 \left(\frac{T_0}{T}\right)^{0.96} $$
- Air broadened width
$$ \gamma_{air}(T) = \gamma_{air}(T_0)\left(\frac{T_0}{T}\right)^{n_{a
- Self broadened width
$$ \gamma_{self}(T) = \gamma_{self}(T_0) \left(\frac{T_0}{T}\right)^{0.
- Shifted center wave number
$$ \nu_0 = \nu + P (1-\chi) \delta_{p} $$
- Line strength with product of mole fraction and spatial step size
```

```
$$ S(T) = S(T_0) \frac{Q(T_0)}{Q(T)}\frac{T_0}{T} exp\left(\frac{-h c E"}{k
$$ S(T) \chi dx $$
- Gaussian half width at half maximum (1000 is a unit conversion factor for
$$ \alpha_G = \nu_0\left(\frac{2 N_a k T log(2) \times 1000 \times 10^4}{m
- Lorentzian half width at half maximum
$$ \alpha_L = P \left(\chi \gamma_{self} + (1 - \chi) \gamma_{air}\right) $
# # looping through all x-locations and given line intensities
# for i in range(ns):
#     for j in range(nt):
#         # getting current mole fraction value
#         x = xnon[j]

#         # computing the pressure shift coefficient
#         d[j,i] = d01[i]*(T0/Tnon[j])**0.96

#         # air broadened width
#         g_a[j,i] = g_a01[i]*(T0/Tnon[j])**n1[i]

#         # self broadened width
#         g_s[j,i] = g_s01[i]*(T0/Tnon[j])**0.75

#         # shifted center wave nuber
#         nu0_s[j,i] = nu1[i] + P*(1-x)*d[j,i]

#         # line strength
#         s[j,i] = lineStrength(S0[i], nu0_s[j,i], Edd1[i], Tnon[j], fid_Q)

#         # gaussian HWHM
#         delnu_g[j,i] = np.sqrt(2*k*np.log(2)*Na/c**2*1000*10**4)*nu0_s[j,
#         # delnu_g[j,i] = 0.5*7.162242257e-7*nu0_s[j,i]*np.sqrt(Tnon[j]/M)

#         # lorentian HWHM
#         delnu_l[j,i] = P*(x*g_s[j,i]+(1-x)*g_a[j,i])
Computing the absorption spectra
$$ K(x) = P \chi(x) \sum_{i = 1}^{N_x} S(T_i) \phi(\nu_i,T_i) $$
```

```
# # pre-allocating absorbance array for all wave numbers
# """
# it is -ln(I(nu)/I_0)
# """
# Absorbance = np.zeros([N_nu_range])

# # computing absorbance coefficient K for each x and suming it up for
# for i in range(X.shape[0]-2): #  <-------------- DOUBT
#      Absorbance += voigt(dat1, nu0_s[i,:], s[i,:], delnu_g[i,:], deln
---

## Post-processing
Preparing pandas dataframe with computed values, and compute error per
# # reading matlab output file
# fid_mat = pd.read_csv("absorbance_matlab.csv", header = None,
#                         names = ["wavenumber", "absorbance"])

# # writing computed values to file along with matlab
# fid_ab = pd.DataFrame(np.transpose([dat1, Absorbance]), columns = ["
#
"absorbance"])
# fid_ab["absorbance_matlab"] = fid_mat["absorbance"]
# fid_ab["error_percentage"] = abs((fid_ab["absorbance_matlab"]-fid_ab
# fid_ab.to_csv("absorbance_pythonOutput.csv", index = None)
Plotting comparison and error graphs
# # plotting absorbance graph
# plt.figure()
# plt.plot(dat1, absorption, '-b', label = "Python")
# # plt.plot(fid_mat["wavenumber"], fid_mat["absorbance"], '-r', label
# plt.grid()
# plt.legend()
# plt.xlabel("wave numbers")
# plt.ylabel("Absorbance")
# plt.title("Absorbance comparison")
# plt.savefig("absorbance.png", dpi = 150)
# # # plotting error percentage between matlab and python
# # plt.figure()
# # plt.plot(fid_ab["wavenumber"],fid_ab["error_percentage"], '-b')
# # plt.grid()
# # plt.xlabel("wave number")
# # plt.ylabel("error percentage")
# # plt.title("Error percentage w.r.t. matlab output")
# # plt.savefig("error.png", dpi = 150)

# # plt.show()

# # plotting absorbance graph
# plt.figure()
# plt.plot(dat1, Abs[9], '-b', label = "Python")
# # plt.plot(fid_mat["wavenumber"], fid_mat["absorbance"], '-r', label = "M
# plt.grid()
# plt.legend()
# plt.xlabel("wave numbers")
# plt.ylabel("Absorbance")
# plt.title("Absorbance comparison")
# plt.savefig("absorbance.png", dpi = 150)
import os


os.chdir('/home/saumya/Documents/spec_data/Temp_uniform_Mol_uniform/')
np.save('x_axis', dat1)
np.save('abs', Abs)
np.save('temp', temp_pro)
np.save('conc', xor_pro)
# np.savetxt('Abosorb', absorption)
# np.savetxt('X_axis', dat1)
```

A.2 Noise Modeling

```
import os


os.chdir('/home/saumya/Documents/Spec_data/Temp_uniform_Mol_uniform/
np.save('abs_SNR_clean', Abs)
plt.plot(Abs[1])
def signalPower(x):
return np.average(x**2)
def SNR(signal, noise):
powS = signalPower(signal)
powN = signalPower(noise)
return 10*np.log10((powS-powN)/powN)
def add_adgn(sig, SNR):
# Set a target SNR
x_volts = sig
x_watts = x_volts ** 2
x_db = 10 * np.log10(x_watts)
target_snr_db = SNR
# Calculate signal power and convert to dB
sig_avg_watts = np.mean(x_watts)
sig_avg_db = 10 * np.log10(sig_avg_watts)
# Calculate noise according to [2] then convert to watts
noise_avg_db = sig_avg_db - target_snr_db
noise_avg_watts = 10 ** (noise_avg_db / 10)
# Generate an sample of white noise
mean_noise = 0
noise_volts = np.random.normal(mean_noise, np.sqrt(noise_avg_watts),
# Noise up the original signal
y_volts = x_volts + noise_volts
return y_volts
SNR = 33.5505
u = []
```

```
for i in Abs:
y = add_adgn(i, SNR)
u.append(y)

u = np.array(u)
np.save(’ABS_noisy_SNR’, u)
```

A.3 Profile Modeling

```
import numpy as np
import matplotlib.pyplot as plt

import decimal



# decimal.getcontext().prec = 5
def bolts(A2, A3, x, x_o):
A1 = 303
a = []
y = []
for i in A2:
for k in x_o:
for j in A3:
b = A1 + (i - A1)/(1+np.exp((x-k)/j))
a.append(b)
y.append([i , j, k])
return(np.array(a), np.array(y))
def bolts_int(A2, A3, x, x_o):
A1 = 303
b = A1 + (A2-A1)/(1+np.exp((x-x_o)/A3))
return b
A2 = 538.0080602192687
A3 = 0.5503972556122777
x_o = 28.00498305421301
```

```
A2 = np.linspace(303, 773, 50)
# A3 = np.linspace(0.1, 1, 5)
# x_o = np.linspace(1, 55, 50)
x = np.linspace(0, 55, 1000)

# mol = np.linspace(.1, 0.3, 10)
A2
b
a2 = bolts_int(A2, A3, x, x_o)
plt.plot(a2)
np.save('EXP_TEST', a2)
e1, e2 = a2
e1 = np.array(e1)
e2 = np.array(e2)
e1
e2
plt.plot(e1[2219])
from scipy import signal

# r1 = []
# for i in e1:
#     downsampled =  signal.resample(i, 1000)
#     r1.append(downsampled)

# r1 = np.array(r1).astype(np.half)
e1.shape
np.save('final_profile_300', e1)
np.save('final_param_300', e2)
test = bolts_int(7.73e+02, 1.00e-01, x, 1.00e+00)
plt.plot(test)
```