PHYSICAL SENSING AND PHYSICS-BASED MACHINE LEARNING

FOR ACTIONABLE ENVIRONMENTAL INSIGHTS

by

John Waczak

APPROVED BY SUPERVISORY COMMITTEE:

David J. Lary, Chair

Christopher Simmons

David Lumley

Lindsay King

Joseph Izen

Copyright © 2024

John Waczak

All rights reserved

UPDATE REQUIRED!

PHYSICAL SENSING AND PHYSICS-BASED MACHINE LEARNING FOR ACTIONABLE ENVIRONMENTAL INSIGHTS

by

JOHN WACZAK, BS, PhD

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

PHYSICS

THE UNIVERSITY OF TEXAS AT DALLAS

September 2024

ACKNOWLEDGMENTS

UPDATE REQUIRED

August 2024

PHYSICAL SENSING AND PHYSICS-BASED MACHINE LEARNING FOR ACTIONABLE ENVIRONMENTAL INSIGHTS

John Waczak, PhD
The University of Texas at Dallas, 2024

Supervising Professor: David J. Lary, Chair

The rapid pace of global change poses a significant and ever present threat to human well-being. To facilitate the development of remediation technologies and to enable effective mitigation strategies, we must make data-driven decisions. However, the limitations posed by the lack of highly available, highly resolved data coupled together with the computational difficulties posed by direct simulation of physics at scale severely constrains our ability to make the low uncertainty predictions needed to meaningfully address these challenges in real time. This dissertation presents novel machine learning strategies for combining physics knowledge with data driven methods in three key case studies. In the first, we demonstrate the ability for a coordinated robotic team to estimate the concentration of chemicals-of-concern *in real time* by using machine learning to map reflectance spectra captured by an autonomous aerial drone directly to chemical concentrations with associated uncertainty estimates. In the second study, we present a novel technique for using temporal variograms to estimate the intrinsic uncertainty of low cost air quality sensors directly from their time series. Additionally, we implement two physics informed machine learning methods to model these collected time series enabling the identification of acute pollution events by modeling them as the result of external forcing. Finally, in the third study, we present the most comprehensive analysis of indoor air quality to date, which includes multi-component ob-

servations, a detailed chemical reaction mechanism (including ion chemistry), an extensive evaluation of indoor photolysis, and full chemical data assimilation (both 4D Var and a full Kalman filter) with detailed multi-component error analysis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

We live in an era marked by the confluence of environmental challenges and technological innovation. The rapid pace of global change poses a significant and ever-present threat to human well-being. To accelerate the development of new remediation and mitigation strategies we must make *data-driven decisions*. Yet, the constraints imposed by the lack of widely accessible, fine-resolution data, in conjunction with the formidable computational complexities intrinsic to large-scale physics simulations, markedly limits our capacity to develop the accurate assessments needed to address these pressing issues in a timely and substantive manner. By leveraging the data driven techniques of machine learning, we can bridge the data-model gap to transform the data we do have into the quantities we need to produce actionable environmental insights.

1.1 Motivation: Global Change and the Need for Improved Sensing Capabilities

Global change encompasses significant and long-term alterations in the Earth's physical, chemical, and biological systems, arising from both natural processes and human activities (Edenhofer et al., 2014; Masson-Delmotte et al., 2018; United Nations, 2015). These alterations include changes in climate, land use, biodiversity, and biogeochemical cycles, as well as the intricate interactions among these systems. Global change exerts profound impacts on both natural and human systems, manifesting as shifts in weather patterns, rising sea levels, heightened frequency and severity of extreme events, loss of biodiversity and ecosystem services, and adverse effects on human health and well-being. The comprehension and effective management of global change present critical challenges for society today, necessitating interdisciplinary approaches and international collaboration. Comprehensive environmental sensing plays a pivotal role by providing real-time data across these crucial areas, aiding in the identification and resolution of environmental concerns.

Global change can profoundly impact human health, both through direct and indirect mechanisms. For instance, the rise in global temperatures escalates the susceptibility to heat-related ailments like heat exhaustion and heat stroke. This heightened risk is particularly concerning for vulnerable demographics, including the elderly, young children, and outdoor workers (World Health Organization, 2018; Costello et al., 2009; Haines et al., 2006). Furthermore, exposure to air pollution is associated with an increased likelihood of developing respiratory and cardiovascular diseases, such as asthma, chronic obstructive pulmonary disease (COPD), and heart disease. Shifts in temperature and precipitation patterns can influence the distribution and abundance of disease vectors like mosquitoes and ticks, thereby elevating the threat of vector-borne diseases, including dengue fever, malaria, and Lyme disease. In a similar vein, alterations in precipitation patterns and water quality can heighten the risk of waterborne diseases, notably cholera. Food availability and production can also be impacted, potentially resulting in food shortages and malnutrition, among vulnerable populations.

1.1.1 Air Quality

Taking air quality as a particularly poignant example, the World Health Organization (WHO) estimates that over 90% of the world's population breathes air that exceeds air quality guidelines for fine particulate matter (PM2.5) concentrations (Organization et al., 2021). Ambient air pollution is estimated to cause 4.2 million premature deaths worldwide each year. In comparison, the total number of global confirmed COVID-19 deaths in 2020 was 1.9 million, and 3.6 million in 2021. Even at the height of the COVID pandemic, deaths from poor air quality outnumbered those from COVID-19. The quality of outdoor ambient air, in turn, influences the quality of indoor air. In areas where outdoor air quality is already poor, increasing ventilation inevitably introduces pollutants indoors, even when filtration is employed (Srikrishna, 2022). While death is unquestionably the most severe consequence

HEALTH OUTCOMES	SHORT-TERM STUDIES			LONG-TERM STUDIES		
	PM_{10}	$PM_{2.5}$	UFP	PM_{10}	$PM_{2.5}$	UFP
Mortality						
All causes	XXX	XXX	X	XX	XX	X
Cardiovascular	XXX	XXX	X	XX	XX	X
Pulmonary	XXX	XXX	X	XX	XX	X
Pulmonary effects						
Lung function, eg, PEF	XXX	XXX	XX	XXX	XXX	
Lung function growth				XXX	XXX	
Asthma and COPD exacerbation						
Acute respiratory symptoms		XX	X	XXX	XXX	
Medication use			X			
Hospital admission	XX	XXX	X			
Lung cancer						
Cohort				XX	XX	X
Hospital admission				XX	XX	X
Cardiovascular effects						
Hospital admission	XXX	XXX		X	X	
ECG-related endpoints						
Autonomic nervous system	XXX	XXX	XX			
Myocardial substrate and vulnerability		XX	X			
Vascular function						
Blood pressure	XX	XXX	X			
Endothelial function	X	XX	X			
Blood markers						
Pro-inflammatory mediators	XX	XX	XX			
Coagulation blood markers	XX	XX	XX			
Diabetes	X	XX	X			
Endothelial function	X	X	XX			
Reproduction						
Premature birth	X	X				
Birth weight	XX	X				
IUR/SGA	X	X				
Fetal growth						
Birth defects	X					
Infant mortality	XX	X				
Sperm quality	X	X				
Neurotoxic effects						
Central nervous system		X	XX			

Notes: X, few studies. XX, many studies. XXX, large number of studies.
Abbreviations: UFP, ultrafine particle; PEF, peak expiratory flow; COPD, chronic obstructive pulmonary disease; IUG, intrauterine growth restriction; SGA, small for gestational age.

Table 1.1: Tabular literature review of particulate matter and health outcomes related to PM_{10}, $PM_{2.5}$, and ultrafine particulates (UFPs) exposure (based on the literature review of (Lary et al., 2015)).

of pollution, it is far from the only one. The biological wear and tear caused by poor air quality has been shown to degrade cognitive and physical performance, as well as a variety of human health outcomes (Krebs and Luechinger, 2021; Gao et al., 2021; Carneiro et al., 2021; Ni et al., 2021; Shehab and Pope, 2019) like those listed in Table 1.1. Degraded cognitive performance has a significant impact on society, ranging from poor learning outcomes in schools to impaired decision-making and productivity at work. As a result, the ability to adequately and comprehensively quantify the key markers of air quality is critical for guiding appropriate, science-based mitigation strategies.

1.1.2 Air Quality Indoors

Humans spend approximately 90% of their time indoors (Finewax et al., 2021). Consequently, the quality of indoor air has a dramatic effect on physical health, cognitive performance, and general well-being (Krebs and Luechinger, 2021; Gao et al., 2021; Carneiro et al., 2021; Ni et al., 2021; Shehab and Pope, 2019). Heating, ventilation, and air conditioning (HVAC) systems are typically designed to circulate air throughout a building, bringing in outside air, in order to achieve comfortable thermal conditions while consuming as little energy as possible (Memarzadeh and Xu, 2012). When assigned the secondary task of maintaining indoor air quality, ventilation operates by the same principles as a laboratory fume hood: airtightness and controlled airflow. To be effective, the ventilation system must move air along a short, unimpeded path toward the exhaust point while preventing unintended leakage (Ng et al., 2015). Despite these assumptions, most buildings are not airtight nor are they well-mixed (Emmerich et al., 2005). Further, the variable nature of indoor human interactions coupled with the plethora of possible room configurations introduces significant design challenges. Computational fluid dynamics (CFD) simulations of the airflow in hospital rooms found that "the most important contributing factor to contaminant transmission in enclosed and mechanically ventilated environment is the path between the contaminant

source and the exhaust, not the air changes per hour" (Memarzadeh and Xu, 2012). This observation is widely confirmed across the literature with multiple studies indicating that air changes per hour (ACH) can be safely lowered without increasing concentrations of chemicals of concern above tolerable levels (Ng et al., 2015; Lamping and Muller, Lamping and Muller; Burroughs et al., 2014). In concert with these findings, studies of the transmission of airborne pathogens suggest that increasing the air change rate removes contaminants from the source room faster but also increases the rate of exposure in connected rooms.

Buildings contain significant sources of volatile chemical products such as cleaners, disinfectants, personal care products (PCPs), and paints whose use introduces a variety of potentially harmful compounds into the indoor environment (Finewax et al., 2021). The growing body of pandemic case studies calls into question the long-held belief that increasing ventilation is the only and best way to control indoor pollution and reduce the spread of airborne pathogens (Ng et al., 2015; Zaatari et al., 2016; Pease et al., 2021; Zheng et al., 2021). Furthermore, such proposals to focus on increased ventilation rates can only be realized through significant construction and renovation, and once completed, have a significant increase in running costs due to the energy required, preventing our most vulnerable communities from accessing clean air disproportionately (Lamping and Muller, Lamping and Muller).

The COVID-19 pandemic has unequivocally demonstrated the need for improved testing and evaluation methodology that characterizes the various components of indoor air quality holistically. This is necessary for the critical evaluation and assessment of built-environment ventilation, filtration, and air-cleaning approaches. Furthermore, given the dual challenges of global pandemics and global change, the carbon footprint of various indoor air quality remediation strategies must be evaluated in order to optimize both indoor air quality and the carbon footprint. If we fail to consider both aspects, we risk inadvertently promoting an increase in health and economic disparities by failing to consider the very real costs

associated with increased air changes. Without a doubt, the composition of the ambient environment has a significant impact on human health as well as general human physical and cognitive performance. Even after a significant initial indoor exposure event may have passed, the health impact associated with prior significant and persistent exposure events may well contribute to long-term health quality. For example, chemical disinfectants such as hypochlorite bleach, have been linked to respiratory ailments and have been observed to produce harmful byproducts such as chloroform, carbon tetrachloride, and N-chloroaldimines, which can take up to 3 hours to be ventilated away after the use the bleach (Finewax et al., 2021; Odabasi, 2008).

If HVAC control is driven by an incomplete subset of metrics/contaminants of concern, buildings will clearly fail to adapt to unexpected threats. For instance, CO_2 is frequently used as a control metric for ventilation because it closely tracks with the number of people in a room. Five studies found that CO_2 control alone cannot control for pollutants generated outdoors or indoors independent of human activity (Zaatari et al., 2016). More recently, one COVID-19 case study discovered that the rapid spread of the disease through a nursing home ward was most likely caused by decreased ventilation by a CO_2 demand control ventilation system optimized solely for decreased energy-demand (Lamping and Muller, Lamping and Muller). Taken together, the consequences of poor air quality coupled with the lack of real time monitoring of key constituents demonstrates the need for an improved understanding of indoor air.

1.2 Physics Based Machine Learning

In recent years machine learning has exploded in popularity becoming entwined in almost every aspect of our lives from image recognition on social media sites, to the now ubiquitous large large language models like ChatGPT. Because the field is so wide, many of the most popular machine learning models must be generic enough so as to apply to *any* well defined

regression or classification task. For example, the problem of image classification is incredibly challenging if we do not impose any restrictions on the type of content represented in our data, as illustrated in Figure 1.1. As a consequence, we typically can not expect a machine

Figure 1.1: A classic machine learning meme comparing pictures of Chihuahuas with muffins illustrating the potential difficulties faced by generic image classification tasks.

learning model to generalize beyond the bounds of their training data.

In the realm of scientific applications, this is often *not* the case; the universe is well described by deep physical principles with underlying mathematical symmetries and structures. Therefore, rather than throwing away all of our prior scientific knowledge for each machine learning task, we can instead develop techniques which enable us to directly or indirectly impose physical requirements on our machine learning models. Similarly, we can utilize machine learning to gain insight into physical processes where a first principles relationship has not yet been established. Together these ideas are known as *Physics-Based Machine Learning* or more generally *Scientific Machine Learning* (SciML for short) and provides us the exact set of tools needed to marry scientific models together with data to produce actionable insights (Rackauckas et al., 2020).

1.3 Dissertation Goals

The goal of this dissertation is to advance physical sensing in service of society by demonstrating the use of new techniques from physics-based machine learning on a variety of real

world data sets. Towards this end, this work presents a collection of case studies utilizing both supervised and unsupervised machine learning techniques in a variety of contexts together with physical sensing to produce actionable environmental insights. The applications of this research apply widely across many scientific domains including remote sensing, time series analysis, air quality, chemical kinetics, and data assimilation.

1.4 Dissertation Overview

In chapter 2 we present the relevant sensing techniques employed in each of the three case studies: The first is a autonomous robot team designed to enable the direct estimation of concentrations of chemicals-of-concern in water utilizing an autonomous drone equipped with a hyperspectral imager. The second is a distributed network of low-cost air quality monitors designed to continuously measure key air quality criterion such as particulate matter (PM) and relevant meteorological variables (temperature, pressure, humidity, etc.). The third is a highly advanced measurement chamber designed to evaluate the chemical kinetics of indoor air.

In chapter 3 we present the relevant machine learning background for this work. We also describe open source implementations of Gaussian Process Regression, Self Organizing Maps, and Generative Topographic Mappings, which we developed for use in this dissertation. Machine learning uncertainty quantification methods are discussed including the technique of Conformal Prediction which we now use for all of our supervised regression tasks. Finally, we provide an in depth discussion of data assimilation including both 4D-variational (4d-Var) data assimilation and extended Kalman Filtering (EKF) techniques which we utilize in modeling the chemical reaction kinetics of indoor air.

In chapter 4 we present results for the Robotic Team which consist of three main contributions. The first is the development of an original processing and georectification code which enables the *real time* generation of accurately georectified reflectance datacubes in

the field. In the second case study, we demonstrate the use of this robotic team to produce machine learning models mapping reflectance spectra directly to chemical concentrations in a North Texas lake. Finally, we present results of applying our unsupervised learning strategies, namely the SOM and GTM, to deduce the spectral signatures of key chemical constituents without the need for a predetermined spectral data-base.

In chapter 5 we present time series techniques designed for use with our low cost sensing network. The first is a method for using temporal variograms (a statistical technique) to generate reasonable estimates for the intrinsic uncertainty of low cost sensors. The second is an extension of the Hankel Alternative View Of Koopman (HAVOK) method to enable physics-based modeling of our gathered time series. Finally, I train a Hamiltonian Neural Network with a generalized coordinate autoencoder to learn representations of generalized position and momenta for a time series embedding of particulate matter concentrations together with an associated Hamiltonian. Together, the augmented HAVOK model and HNN provide interesting ways to identify transient pollution events from time series.

In chapter 6 we present a novel chemical data assimilation framework for the assessment of indoor air quality. The first major component involves performing a detailed characterization of indoor photolysis using GPR to generate relevant cross section and quantum yield fits from data scraped from a variety of databases. Next we use this photolysis data together with our advanced air quality measurement chamber to evaluate the chemical kinetics of indoor air. Using a combination of 4d-var and the EKF together with a detailed chemical reaction mechanism, we are able to infer the concentrations of reactive species that otherwise well below detectable limits.

CHAPTER 2

PHYSICAL SENSING

The successful application of machine learning methods demands comprehensive, carefully curated data sets. In order for our modeling efforts to be successful, it is critical that we capture the subtle nuances of the phenomena we wish to describe. In this chapter, we outline the various physical sensing approaches used throughout this dissertation, all of which are part of a broader effort in the MINTS-AI laboratory at the University of Texas at Dallas. MINTS-AI is an acronym for Multi-Scale Multi-Use Integrated Intelligent Interactive Sensing in Service of Society for Actionable Insights. An graphical overview of the MINTS-AI sensing paradigm is outlined in Figure 2.1.

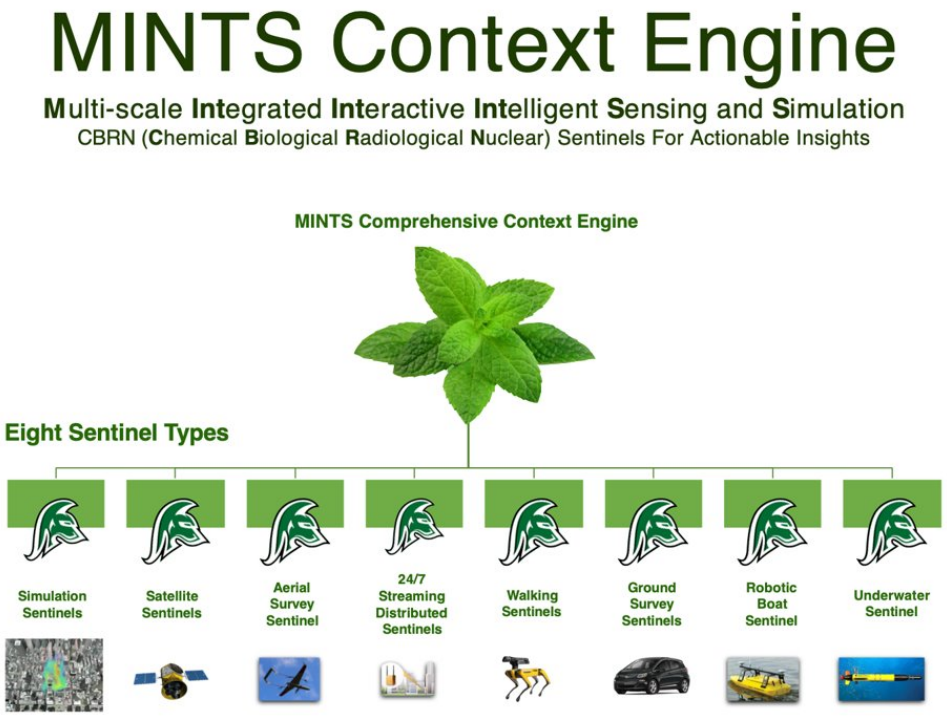


Figure 2.1: The MINTS-AI context engine is a sensing paradigm composed of flexible sensing sentinels spanning from remote sensing data products to autonomous robots and ground survey vehicles.

Of the variety of sensing sentinels listed above, this dissertation is primarily concerned with three key applications. The first is a team of autonomous robotic vehicles which we refer to as the *robotic team*. The second is a network of distributed streaming sentinels comprised of low-cost air quality sensors. The third describes a reference sensor chamber for air quality evaluation which we use to develop a *simulation sentinel*.

2.1 Coordinated Robotic Teams

Recent developments in hyperspectral imaging technology have led to dramatic reductions in both size and weight of imaging platforms. Due to these improvements, it is now possible to incorporate the technology as the payload of highly mobile autonomous aerial vehicles such as drones. However, the massive volume of hyperperspectral datacubes poses significant computational challenges to their adoption in real-time applications. For decades, multi-spectral imagers have seen wide spread use in the remote sensing community as a means to take advantage of the wealth of information contained in the reflectance spectra of materials. In addition to the three color filters of traditional cameras, *multi-spectral* imagers, like those deployed on MODIS, Sentinel 2, and other satellite missions, capture many additional features by utilizing wavelength bands ranging from the near-UV, through the visible spectrum, and into the Infrared. With this additional information, multi-spectral remote sensing platforms are able to aid in a variety of domains from tracking land change, characterizing deforestation, monitoring erosion, evaluating crop health, and many others (Curran, 1989; Navalgund et al., 2007; Thenkabail et al., 2000).

These uses are justified by the reflectance features of materials across the electromagnetic spectrum; water has vibrational modes in the infrared while pigments tend to have absorption peaks in the visible region (Van Der Meer, 2004; Burns, 1993; Hunt, 1977). Many currently used spectral indices like the normalized difference vegatation index (NDVI) take advantage of these spectral regions by comparing ratios of pigment sensitive passbands to

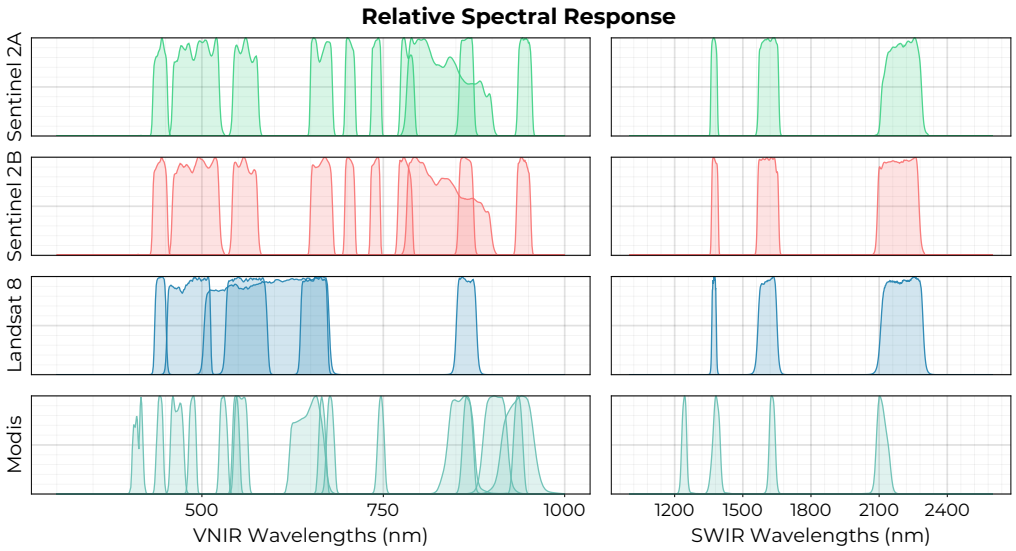


Figure 2.2: A comparison of the relative spectral response for the passbands of popular multi-spectral imaging remote sensing platforms.

the stable signals in the infrared to infer the abundance of chlorophyll, and consequently, the health of plants (Boiarskii and Hasegawa, 2019). However, despite the plethora of successful applications of multi-spectral imaging, more can be accomplished with the additional information provided by fully resolved spectra. For example, in the laboratory, spectrophotometry allows the direct determination of the concentrations of chemical constituents in solution by deconvolution of a sample spectrum against libraries of carefully collected of reference spectra. Individual chemicals be uniquely identified by the characteristic location and shape of their absorption features.

At the terrestrial level, drones (i.e. quadcopters, octocopters, and other similar multi-rotor craft) equipped with cameras are able to utlize techniques of photogrametry together with continuously sampled imagery to produce high quality digital elevation maps, high quality mosaics, 3-dimensional reconstructions, etc (Vacca, 2019). These capabilities provide significant aid for structural analysis and smart agriculture to name a few applications. Today, kilogram-scale HSI can be comfortably mounted to the payload of drones and ad-

vancements in spectral sensing such as (Yoon et al., 2022) suggest that sizes of HSI will continue to shrink, further expanding their application in this domain.

The inspiration for this autonomous robotic team is the automation of what is currently done manually in the production of remote sensing satellite data products. The typical timescale from starting work on a new remote sensing data product to its operational readiness is at least a couple of years, but, more typically, a decade or more. A key part of this substantial time delay is due to the time that is taken for the collection of the relevant training data. Hence, our goal is to reduce this timescale to be near real time by utilizing an autonomous robotic team that can both collect the training data, and then in real time process and stream the remote sensing data products. The fully autonomous team includes a robotic boat that carries a suite of sensors to measure in-situ water composition in real time as well as a sonar, and an autonomous UAV equipped with a down-welling irradiance spectrometer, hyper-spectral, and thermal imagers, together with an onboard Machine Learning (ML) capability. Figure 2.3 shows photographs of the robot team during a deployment in North Texas.

Figure 2.3: The MINTS robotic team

The autonomous boat used is a Maritime Robotics Otter . With a footprint of only 200 $\times$ 108 $\times$ 81.5 cm, a weight of 55 kg, and dual electrical fixed thrusters, it is an easily deployable asset that can be transported in a van or even within normal airliners to a survey site. With a cruise speed of two knots, it has a duration of 20 h from one charge of the batteries. It can use WiFi, cellular, and an optional AIS receiver for communication to the

control station. Our drone is a Freefly Alta-X professional quad-copter. It was specifically designed to carry cameras, with a payload capacity of up to 35 lb, a long range data link, and autonomy provided by the Open PX4 flight stack. The open source QGroundControl software is used to control the autonomous operations.

All of the robotic team members carry a high-accuracy GPS and inertial navigation system (INS) so that every data point can be geo-located and time stamped. Each of the robots can also join the same network which connects the robots and their ground-control stations. Our robots use long-range Ubiquiti 5 GHz LiteBeam airMAX WiFi. This network also includes a local Synology network-attached storage (NAS) device in the robot team control trailer, which, in real-time, syncs the data that are collected to the NAS in our home laboratory at the university.

The robotic boat payload includes a BioSonics MX Aquatic Habitat Echosounder sonar for rapid assessment and mapping of aquatic vegetation, substrate and bathymetry. Three Eureka Manta-40 multi-probes, a Sequoia Scientific LISST-ABS acoustic backscatter sediment sensor, and an Airmar Technology Corporation 220 WX ultra-sonic weather monitoring sensor.

The first Manta-40 multi-probe includes sensors for temperature and turbidity and Turner Designs Cyclops-7 submersible Titanium body fluorometers for Chlorophyll A, Chlorophyll A with Red Excitation, Blue-Green Algae for fresh water (Phycocyanin), Blue-Green Algae for salt water (Phycoerythrin), and CDOM/FDOM. The second Manta-40 multi-probe includes sensors for temperature, conductivity (with specific conductance, salinity, and total dissolved solids, TDS), pH (with separate ref- erence electrode), optical dissolved-oxygen, turbidity, and Ion Selective Electrodes by Analytical Sensors and Instruments (http://www.asi-sensors.com/, accessed 05/01/2021) for ammonium (NH_4^+), bromide (Br), calcium (Ca^{2+}), chloride (Cl), nitrate (NO_3^-), and sodium (Na^+). The third Manta-40 multi-probe includes sensors for temperature, turbidity, a total dissolved gas sensor, and Turner Designs Cyclops-

7 submersible Titanium body fluorometers for optical brighteners, crude oil, refined fuels, and tryptophan.

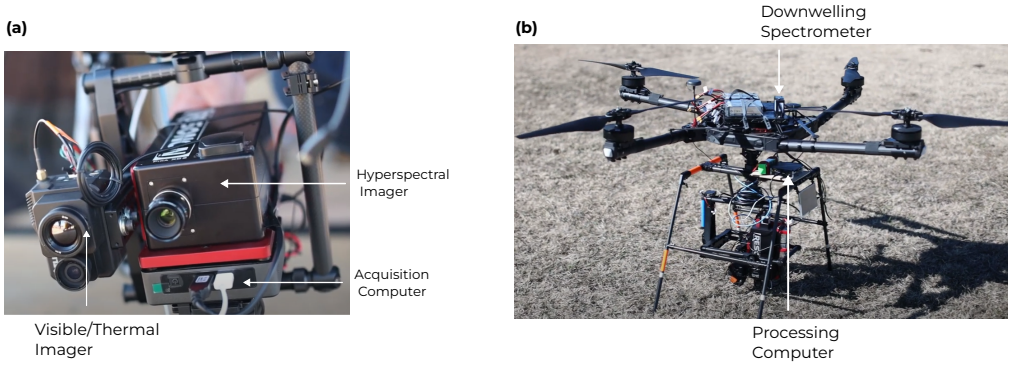


Figure 2.4: An annotated view of the autonomous drone showcasing the hyperspectral imager and onboard compute.

The aerial vehicle uses a custom built landing gear and mount made with aircraft grade aluminum and carbon fiber to carry a Resonon Visible+Near-Infrared (VNIR) Pika XC2 hyper-spectral camera (391–1011 nm) with a Schneider Xenoplan 1.4/17 mm lens, and a FLIR Duo Pro R, (640 × 512, 25 mm, 30 Hz) combining a high resolution, radiometric thermal imager, 4K color camera, and a full suite of onboard sensors. On the top of the quad copter there is a sky facing Ocean Optics UV-Vis-NIR spectrometer measuring the incident down-welling irradiance with a 180 degree cosine corrector, allowing us to convert the raw radiance measurements of each pixel to reflectance.

2.2 A Low Cost Sensor Network For Air Quality Monitoring

The highly expensive cost to acquire, calibrate, and maintain reference grade air quality monitors makes it challenging to assess the importance of spatial and temporal variability on local air quality. Since factors such as weather, terrain, traffic, and the distribution of other sources can all effect local air quality, the development of low-cost sensing solutions is vital to address risks of poor air quality on local communities. To address this gap, we have

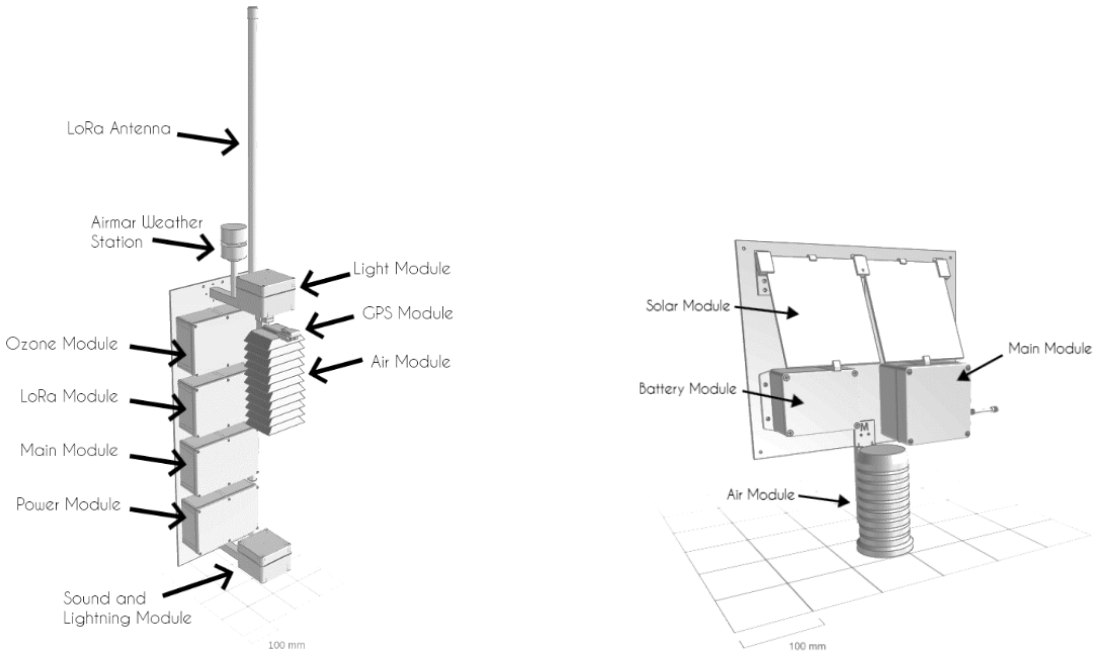
developed a hierarchy of low-cost air quality monitors which we have deployed throughout the Dallas Fort-Worth (DFW) metroplex. In this section, we describe the relevant sensor types as well as a robust data processing and visualization pipeline developed to enable open access to high quality data.

The sensor network is comprised of a combination of two types of nodes: *Central Nodes* and *LoRa Nodes*. The central nodes are designed to be deployed in locations with where dedicated power is available. Each contains a variety of sensors including Particulate Matter, VOCs, CO_2, NO_X, ionizing radiation, incident light intensity, sound levels, as well as meteorological variables including temperature, pressure, relative humidity, and dew point. The powered Central Nodes are equipped with a cellular modem to facilitate data transfer from the field.

Each Central Node supports a collection of 10 LoRa nodes (named for the LoRa long rage wireless transmission protocol) which can be separated by distances up to 5 km or more if line of sight is established. These smaller sensors are self powered using a combination of battery and solar cells, and each measures a similar array of relevant air quality metrics including particulate matter concentrations, gas concentrations, and meteorological parameters. Designs for the two node types are illustrated in Figure 2.5.

Using the LoRaWAN protocol, the Central and LoRa nodes form a low power, wide area network by which data packets from each LoRa node are transmitted in real time to the nearest Central Node. Using their inbuilt cellular connection, the central nodes are then able to pass sampled data to a data processing backend using an MQTT publish-subscribe model. As the primary goal of this network is to provide detailed, real-time air quality data, we have developed a public facing website, `http://SharedAirDFW.com`, to visualize the current and historic measurement at each sensor location together with 6-hour wind forecasts from NOAA and weather radar. Figure 2.6 shows a sample screenshot of the site.

In order to automate data collection, processing, analysis, and long-term storage, a containerized pipeline was developed as illustrated in Figure 2.7. This pipeline is composed of



(a) A 3d model of a Central Node (b) A 3d model of a LoRa Node

Figure 2.5: Two types of nodes from the MINTS Air Quality network.

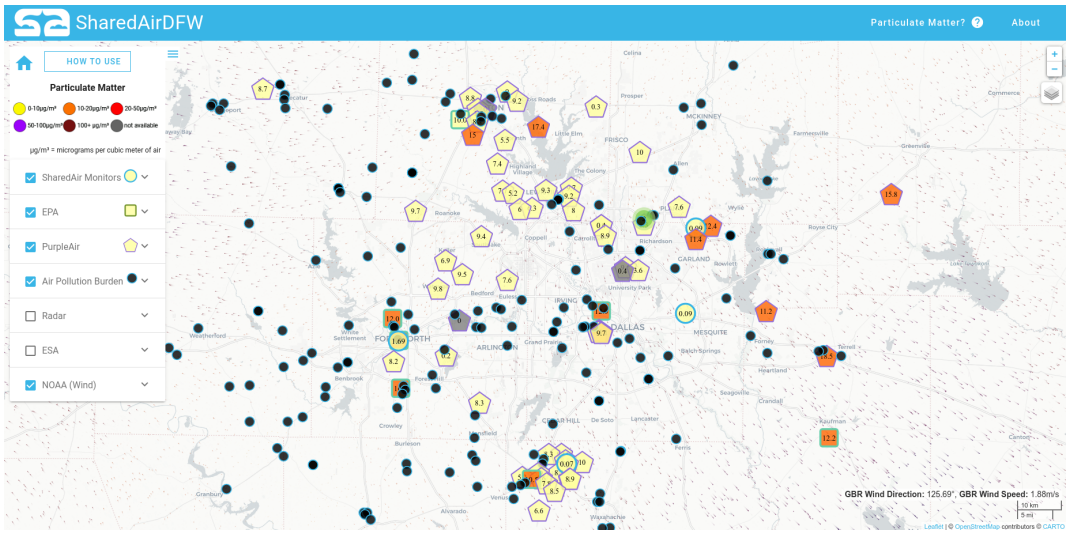
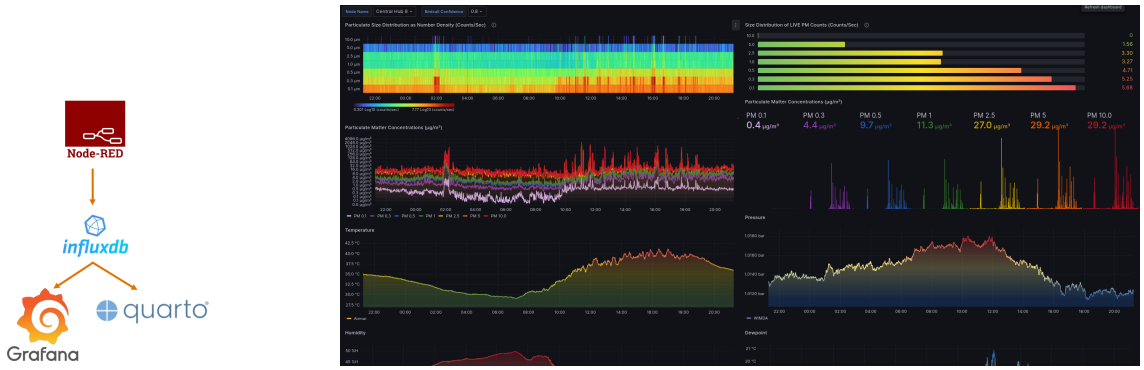


Figure 2.6: The interactive SharedAirDFW website show casing real time data streams from the MINTS sensor network.

5 tools that utilize containerization for reliable development and deployment.



(a) (b)

Figure 2.7: The backend infrastructure developed to support the MINTS sensor network.

- **NodeRed**: This tool developed by IBM allows the creation of complex data processing pipelines by defining directed acyclic graphs (DAGs) composed of individual processing nodes. We utlize this tool to subscribe to each sensor's relevant MQTT topic and decode the data packets into the relevant sensor measurements. At the end of each sensor pipeline, the data are then injected into the InfluxDB time series database. A key advantage of NodeRed is that the wide array of pre-existing nodes enables a no-code solution that is easy to maintain.

- **InfluxDB**: This is a open source time series database optimized for large cardinality datasets. By storing processed data in InfluxDB, we are able to provide easy queryable access to real time and historic data.

- **Grafana**: This tool is an open source visualization platform for creating interactive dashboards. We utilize Grafana to create detailed, real-time displays for each sensor in the network that we can use to observe *all* incoming measurements as well as identify pollution events and monitor sensor status.

- **Quarto**: This tool allows the generation of automated analysis reports by using a literate programming paradigm built on top of Jupyter Notebooks. By using quarto

we are able to automatically perform daily, weekly, monthly, and annual analyses for each sensor in the network and collect the results into formatted PDFs or a static website. We are developing analysis using quarto together with Julia, Python, and R to facilitate detailed report generation and provide actionable insights.

- **Open Storage Network**: With help from Dr. Chris Simmons, we have developed a pipeline for long term data storage utilizing the Open Storage Network to provide open access to historic sensor data stored in the popular S3 format.

Individual sensor dashboards are made publicly available at `http://mdash.circ.utdallas.edu:3000`. Historic data are updated daily at `https://portal.osn.xsede.org/s3browser//?bucket_path=https://ncsa.osn.xsede.org/ees230012-bucket01`.

2.3 Reference Grade Assesment of Indoor Air: The Heart Chamber

The coordinated Robot Team allows us to rapidly characterize novel outdoor environments with a particular focus on assessing concentrations of chemicals-of-concern in bodies of water. Our low-cost sensing network enables high spatio-temporal resolution of air quality dynamics for outdoor air. In this final section, we describe a comprehensive testing chamber for the assessment of *indoor* air. This chamber, called *HEART* for holistic environmental aerosol and reactive component testing, is being developed to enable assessment of the detailed chemical and aerosol dynamics which effect indoor air quality. Figure 2.8 provides visual overview of the design.

The HEART test chamber is designed for comprehensive characterization of air quality including the chemical composition (characterizing hundreds of chemicals), positive and negative ion density, aerosol size (from 5 nm to 100 microns), detailed morphology (shape), and composition, and individual bio-aerosol particle recognition, along with the accurately characterized illumination environment at more than 4,000 wavelengths from the UV to

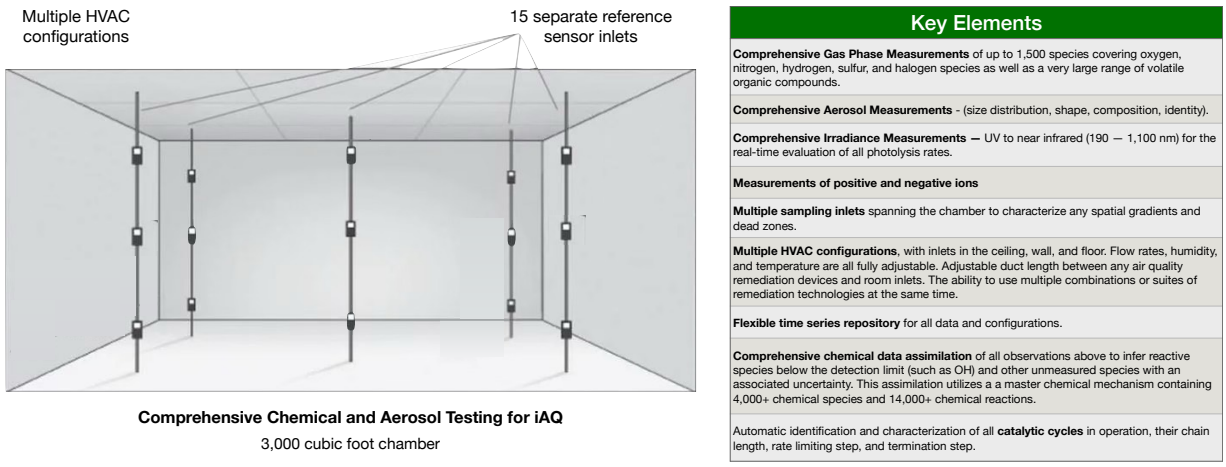


Figure 2.8: Design of comprehensive chemical and aerosol testing for IAQ using HEART: Holistic Environmental Aerosol and Reactive Component Auto Response Testing.

the near-infrared (190 — 1,100 nm) using a NIST calibrated spectrometer. The lighting environment is important, as, after all, air quality is atmospheric photochemistry in the indoor built environment. Photolysis is a key driver. Furthermore, some of the frequently advocated families of remediation technologies make use of various parts of the ultraviolet spectrum. This high-energy region of the spectrum has important implications for indoor air quality.

As shown in figure 2.9, 15 inlets in the HEART chamber are sampled using a combination of instruments including an Aerodyne Aerosol Mass Spectrometer (AMS), a Selected Ion Flow Tube (SIFT) mass spectrometer, a collection of gas concentration analyzers, as well as a an Ocean Optics HR4000 UV-NIR spectrometer, and a GRIMM wide range aerosol spectrometer. In the spirit of *multi-use*, the NodeRed+InfluxDB+Grafana software stack are also utilized to log sensor measurements directly into a time series database with real time dashboards for monitoring of experiments.

Figure 2.9: Visual overview of the various infrastructure components of the ActivePure lab. Note that the lab uses multiple instruments of those shown to accommodate the stated range of analytes.

It is inevitable that some chemical species, for example, reactive components such as OH, will typically be below the detection limit of the measurement systems. Information on these unmeasured reactive components is naturally embedded in the shape and partitioning of the time-series observations of the chemical species they are reacting with. This information can be effectively extracted, together with associated uncertainty estimates, using chemical data assimilation (Fisher and Lary, 1995; Lary, 1999; Lary et al., 2003). Hundreds of chemicals measured in real-time in the HEART chamber, along with high-resolution irradiance spectra used to calculate real-time photolysis rates and real-time aerosol measurements, are assimilated in a full chemical data assimilation system with a detailed chemical mechanism.

CHAPTER 3

MACHINE LEARNING METHODS

In this chapter, we outline the relevant techniques from machine learning that we will employ throughout the rest of this dissertation. Additionally, we present original software implementations developed in the Julia language for Gaussian Process Regression, Self Organizing Maps, and Generative Topographic Mapping.

3.1 Adjoint Methods for Optimization

The fundamental task of machine learning is to build data-driven models. To do this we must *fit* model parameters by taking advantage of the available training data. In many cases, this is straight forward: the sensitivities of model outputs to changes in model parameters can be backpropagated by application of the chain rule through each layer of our model. However, this naive propagation can become prohibitively expensive as we increase the size and complexity of our models. To efficiently determine these sensitivities and thereby enable the optimization of complicated models, we can instead turn the problem on it's side and consider a parallel *adjoint* system. Due to the utility of this technique for enabling the optimization of models involving implicit linear and nonlinear relationships and even differential equations, we shall begin our discussion of machine learning with a handful of these techniques which will be utilized in the remainder of this work.

3.1.1 Adjoint Methods for Linear Systems

To begin, suppose a component of our model is given by the θ-parameterized linear system

$$A(\theta)u = b(\theta). \tag{3.1}$$

If we wish to find the ideal θ subject to some loss function $J(u)$ we must obtain the gradients $dJ/d\theta$ which upon expansion yields

$$\frac{dJ}{d\theta} = \frac{\partial J}{\partial \theta} + \frac{\partial J}{\partial u}\frac{du}{d\theta}. \tag{3.2}$$

The Jacobian $du/d\theta$ will be challenging to compute and depends on solving the above implicit system:

$$\frac{d}{d\theta}(Au) = \frac{d}{d\theta}b$$
$$\left(\frac{dA}{d\theta}\right)u + A\left(\frac{du}{d\theta}\right) = \frac{db}{d\theta}$$

which with some slight abuse of notation in the, yields

$$\frac{du}{d\theta} = A^{-1}\left(\frac{db}{d\theta} - \frac{dA}{d\theta}u\right) \tag{3.3}$$

and therefore

$$\frac{dJ}{d\theta} = \frac{\partial J}{\partial \theta} + \frac{\partial J}{\partial u}A^{-1}\left(\frac{db}{d\theta} - \frac{dA}{d\theta}u\right) \tag{3.4}$$

must require p linear solves if there are p parameters in θ. Can we do better?

If we perform a clever *re-bracketing* so that

$$\frac{dJ}{d\theta} = \frac{\partial J}{\partial \theta} + \left(\frac{\partial J}{\partial u}A^{-1}\right)\left(\frac{db}{d\theta} - \frac{dA}{d\theta}u\right) \tag{3.5}$$

then we can see that the vector $\frac{\partial J}{\partial u}A^{-1}$ is the solution to *some* problem, say

$$\lambda^T A = \frac{\partial J}{\partial u} \tag{3.6}$$

then, we need only solve

$$A^T\lambda = \left(\frac{\partial J}{\partial u}\right)^T \tag{3.7}$$

once! This leaves us with the following procedure:

1. Solve the system $Au = b$ for u
2. Solve the adjoint system $A^T\lambda = \left(\frac{\partial J}{\partial u}\right)^T$ for λ
3. compute $\frac{dJ}{d\theta} = \frac{\partial J}{\partial \theta} + \lambda^T\left(\frac{db}{d\theta} - \frac{dA}{d\theta}u\right)$

3.1.2 Adjoint Methods for Nonlinear Systems

Having demonstrated an effective strategy for computing the gradients of cost functions with respect to underlying linear systems, a natural next step is to ask: can we do the same for nonlinear systems? Suppose now that we instead we find a component of our model to be of the form

$$f(u;\theta) = 0 \tag{3.8}$$

Then the gradient of some loss function $J(u;\theta)$ which depends indirectly on the solution of this system will be

$$\frac{dJ}{d\theta} = \frac{\partial J}{\partial \theta} + \frac{\partial J}{\partial u}\frac{du}{d\theta} \tag{3.9}$$

while the nonlinear system satisfies

$$\frac{df}{d\theta} = \frac{\partial f}{\partial u}\frac{du}{d\theta} + \frac{\partial f}{\partial \theta} = 0. \tag{3.10}$$

Therefore, the linear system that resulting from the differentiation procedure leads to

$$\frac{du}{d\theta} = -\left(\frac{\partial f}{\partial u}\right)^{-1}\frac{\partial f}{\partial \theta} \tag{3.11}$$

which we may substitute to find

$$\frac{dJ}{d\theta} = \frac{\partial J}{\partial \theta} - \left(\frac{\partial J}{\partial u}\left(\frac{\partial f}{\partial u}\right)^{-1}\right)\frac{\partial f}{\partial \theta} \tag{3.12}$$

so that again, by cleverly re-bracketing the above equations we my identify that

$$\frac{\partial J}{\partial u}\left(\frac{\partial f}{\partial u}\right)^{-1}$$

is the solution to *some* linear system of the form

$$\lambda^T\left(\frac{\partial f}{\partial u}\right) = \frac{\partial J}{\partial u}. \tag{3.13}$$

Therefore, we have arrived at a procedure similar to the linear case which is as follows:

1. Solve $f(u\ \theta) = 0$ for u by your favorite algorithm.
2. Solve $\left(\frac{\partial f}{\partial u}\right)^T \lambda = \left(\frac{\partial J}{\partial u}\right)^T$ for the adjoints λ.
3. Compute the gradient $\frac{dJ}{d\theta} = \frac{\partial J}{\partial \theta} - \lambda^T\left(\frac{\partial f}{\partial \theta}\right)$.

3.1.3 Adjoint Methods for ODEs

To complete the discussion, suppose that we instead seek to fit a model described by an Ordinary Differential Equations (ODE) initial-value problem (IVP) of the form

$$\begin{cases} \dfrac{du}{dt} = f(u, t; \theta) \\ u_0 = u(t = 0) \end{cases} \tag{3.14}$$

where $u \in \mathbb{R}^n$ denotes the time-dependent state vector, and f is a function of the state u, time t, and some number of parameters θ.

To *fit* such a model to data, we need to be able to compute the sensitivity of some cost function $J(u)$ to the parameters θ, that is, we need the gradient $dJ/d\theta$. Obtaining these sensitivities by direct forward-mode automatic differentiation or with tape/tracer based backpropagation demands we be able to establish the derivatives for any class of ODE integrator we want to choose. This task is further complicated if we desire to use modern differential equation solver suites, like those provided by the `DifferentialEquations.jl` Julia library, which employ complicated adaptive stepping schemes. How could we possibly know *a priori* how many function evaluations will be needed to produce the output for each ODE step, and thereby how to compute the partial derivatives? One approach is to establish language-wide automatic differentiation capability for generic computer code (dubbed ∂P) (Innes et al., 2019). An alternative approach, which we shall describe here, is to develop a set of adjoint differential equations whose solution we can use to compute the desired derivatives.

Let us suppose our loss function can be written in the form

$$J(u; \theta) = \int_0^T g(u; \theta) dt \tag{3.15}$$

where g is some function like the quadratic loss $u^T R u$ of 4d-var. To find our desired sensitivities, we can utilize the method of Lagrange multipliers by introducing a Lagrangian of

the form

$$\begin{aligned}\mathcal{L}(u,\lambda;\theta) &:= J(u;\theta) + \int_0^T \lambda^T \left(f - \frac{du}{dt}\right) dt \\ &= \int_0^T \left[g(u;\theta) + \lambda^T(t)\left(f - \frac{du}{dt}\right)\right] dt\end{aligned} \tag{3.16}$$

The λ are the so-called Lagrange multipliers, and the integrand, $f - du/dt$, is a clever way of adding 0 so that $d\mathcal{L}/d\theta = dJ/d\theta$. Evaluating this derivative, we find

$$\frac{d\mathcal{L}}{d\theta} = \int_0^T \left\{\left(\frac{\partial g}{\partial \theta} + \frac{\partial g}{\partial u}\frac{du}{d\theta}\right) + \lambda^T(t)\left[\frac{\partial f}{\partial \theta} + \frac{\partial f}{\partial u}\frac{du}{d\theta} - \frac{d}{dt}\frac{du}{d\theta}\right]\right\} dt. \tag{3.17}$$

The difficult term to compute is $du/d\theta$ and so we group the terms so that

$$\frac{d\mathcal{L}}{d\theta} = \int_0^T \left[\frac{\partial g}{\partial \theta} + \lambda^T(t)\frac{\partial f}{\partial \theta} + \left(\frac{\partial g}{\partial u} + \lambda^T(t)\frac{\partial f}{\partial u} - \lambda^T(t)\frac{d}{dt}\right)\frac{du}{d\theta}\right] dt. \tag{3.18}$$

We now observe that if we pick the $\lambda^T(t)$ such that

$$\lambda^T(t)\frac{\partial f}{\partial \theta} + \left(\frac{\partial g}{\partial u} + \lambda^T(t)\frac{\partial f}{\partial u} - \lambda^T(t)\frac{d}{dt}\right)\frac{du}{d\theta} = 0, \tag{3.19}$$

we will not have to compute this pesky term at all! To proceed, let's apply integration by parts to move the time derivative of the final term and simplify the matter:

$$\begin{aligned}\int_0^T -\lambda^T \frac{d}{dt}\frac{du}{d\theta} dt &= -\lambda^T \frac{du}{d\theta}\bigg|_0^T + \int_0^T \frac{d\lambda^T}{dt}\frac{du}{d\theta} dt \\ &= \lambda^T(0)\frac{du}{d\theta}(0) - \lambda^T(T)\frac{du}{d\theta}(T) + \int_0^T \dot{\lambda}^T \frac{du}{d\theta} dt.\end{aligned} \tag{3.20}$$

Plugging this back in to our previous expression yields

$$\frac{d\mathcal{L}}{d\theta} = \int_0^T \left[\frac{\partial g}{\partial \theta} + \lambda^T\frac{\partial f}{\partial \theta} + \left(\frac{\partial g}{\partial u} + \lambda^T\frac{\partial f}{\partial u} + \dot{\lambda}^T\right)\frac{du}{d\theta}\right] dt + \lambda^T(0)\frac{du}{d\theta}(0) - \lambda^T(T)\frac{du}{d\theta}(T). \tag{3.21}$$

Great! In order to make the pesky term disappear, it suffices to find the $\lambda^T(t)$ such that

$$\begin{cases}\dfrac{\partial g}{\partial u} + \lambda^T\dfrac{\partial f}{\partial u} + \dfrac{d\lambda^T}{dt} = 0 \\ \lambda(T) = 0\end{cases} \tag{3.22}$$

which amounts simply solving a second differential equation from the ending time T back to $t = 0$. Transposing the above, we can obtain the new ODE in the standard form:

$$\boxed{\begin{cases} \dfrac{d\lambda}{dt} = -\left(\dfrac{\partial f}{\partial u}\right)^T \lambda - \left(\dfrac{\partial g}{\partial u}\right)^T \\ \lambda(T) = 0 \end{cases}} \tag{3.23}$$

Better yet, this system is linear in λ even though the original differential equation may be fully nonlinear with no restrictions on f! To summarize, the procedure for determining the desired gradients is as follows:

1. Solve the ODE system given by $du/dt = f(u, t; \theta)$ forward in time from $t = 0$ to $t = T$ to obtain the solution $u(t)$.

2. Solve the Adjoint ODE system given by $d\lambda/dt = -(\partial f/\partial u)^T \lambda - (\partial g/\partial u)^T$ from $t = T$ to $t = 0$ to obtain the adjoints $\lambda^T(t)$

3. Compute the gradient $dJ/d\theta = d\mathcal{L}/d\theta = \int_0^T [(\partial g/\partial \theta) + \lambda^T(t)(\partial f/\partial \theta)]dt + \lambda(0)du\_0/d\theta$ by quadrature.

3.2 Supervised Regression Techniques

3.2.1 Neural Networks

3.2.2 Gaussian Process Regression

Gaussian Process Regression (GPR) is a powerful technique for non-linear, non-parametric supervised machine learning. In this section we present a pedagogical overview of the method (the kind the author wishes he had access when getting started) as we will use the technique for both the Robot Team and to model absorption cross sections and quantum yields for photolysis reactions in Chapter 6. To begin, let's first consider a motivating example: **Linear Regression**. We will use this guiding example to derive GPR from a *weight space view*. After

this derivation, we will suggest a simpler, but more abstract derivation using a *function space view*.

The Weight Space View

We consider a dataset $\mathcal{D}$ with n observations

$$\mathcal{D} = \left\{ (\mathbf{x}_i, y_i) \;\middle|\; i = 1, ..., n \right\} \tag{3.24}$$

- $\mathbf{x}_i$ is the i^{th} D-dimensional input (feature) vector
- y_i is the i^{th} target

Linear regression is easily understood in terms of *linear algebra*. We therefore collect our dataset $\mathcal{D}$ into a $D \times n$ dimensional **Design Matrix**.

$$X := \begin{pmatrix} \vdots & \vdots & & \vdots \\ \mathbf{x}_1 & \mathbf{x}_2 & ... & \mathbf{x}_n \\ \vdots & \vdots & & \vdots \end{pmatrix} \tag{3.25}$$

and our targets into a target vector

$$\mathbf{y} := (y_1, ..., y_n) \tag{3.26}$$

so that the full training set becomes

$$\mathcal{D} := (X, \mathbf{y}) \tag{3.27}$$

Note that we have used a transposed definition (features are rows, records are columns) as Julia is a column-major language (like Matlab and Fortran).

Standard linear regression is a model of the form

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{w} \tag{3.28}$$

where $\mathbf{w}$ is the D-dimensional vector of weights. By minimizing the mean-squared-error between our model and targets, one can show that the optimal weights are given by

$$\mathbf{w} = (XX^T)^{-1}X\mathbf{y} \tag{3.29}$$

This can also be easily obtained geometrically by finding the vector with the shortest distance to the hyperplane defined by the column space of X. This corresponds to solving the normal equations

$$XX^T\mathbf{w} = X\mathbf{y}. \tag{3.30}$$

The following demonstrates this procedure on a simple dataset We can also fit a y-

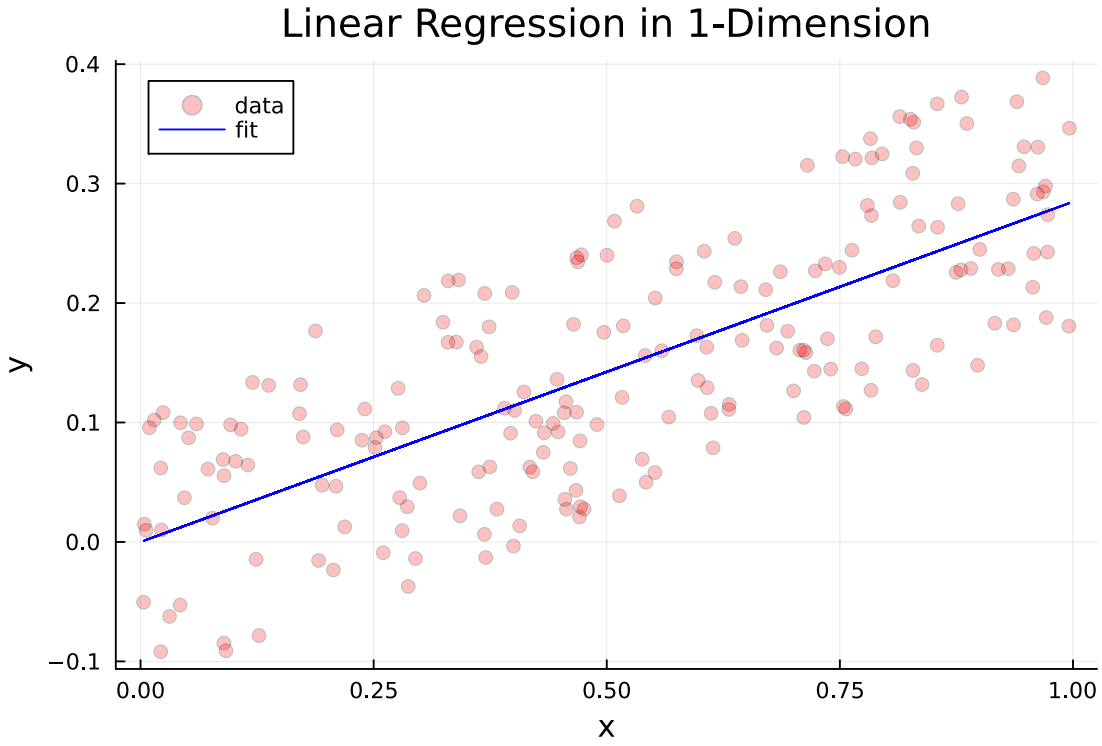


Figure 3.1: A standard linear regression fit on some noisy data.

intercept (aka *bias*) by augmenting the design matrix X to contain an extra row with all 1's, i.e.

$$X[D+1,:] = (1,...,1) \tag{3.31}$$

Standard linear regression assumes that our data, $\mathcal{D}$, are perfect, but we can clearly see that the above data are noisy. To account for this, we need to make our model *Bayesian* by augmenting it to consider the measurement error. We define

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} \tag{3.32}$$

$$\mathbf{y} = f(\mathbf{x}) + \epsilon \tag{3.33}$$

$$\epsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{3.34}$$

or, in words, our observed values differ from the *truth* by identically, independently, distributed Gaussian noise with mean 0 and variance σ_n^2. The assumption that the noise is i.i.d. is critical because it allows us to simplify the *likelihood* function by separating out each individual contribution by our datapoints:

$$p(\mathbf{y}|X, \mathbf{w}) := \prod_i^n p(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}) \tag{3.35}$$

$$= \prod_i^n \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{(\mathbf{y}_i - \mathbf{x}_i^T\mathbf{w})^2}{2\sigma_n^2}\right) \tag{3.36}$$

$$= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\left(-\frac{1}{2\sigma_n^2}|\mathbf{y} - X^T\mathbf{w}|^2\right) \tag{3.37}$$

$$= \mathcal{N}\left(X^T\mathbf{w}, \sigma_n^2 I\right) \tag{3.38}$$

To perform inference with this updated model, we apply Baye's Rule, that is:

$$p(\mathbf{w}|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)} \tag{3.39}$$

where

- $p(\mathbf{w}|\mathbf{y}, X)$ is the *posterior distribution*
- $p(\mathbf{y}|X, \mathbf{w})$ is the *likelihood*
- $p(\mathbf{w})$ is the *prior distribution*

- $p(\mathbf{y}|X)$ is the *marginal likelihood*, i.e. the normalization constant

It is now that the utility of choosing gaussian distributions for our likelihood and prior becomes clear. We have

$$p(\mathbf{w}|\mathbf{y}, X) \propto \exp\left(-\frac{1}{2\sigma_n^2}(\mathbf{y} - X^T\mathbf{w})^T(\mathbf{y} - X^T\mathbf{w})\right)\exp\left(-\frac{1}{2}\mathbf{w}^T\Sigma_p^{-1}\mathbf{w}\right) \tag{3.40}$$

Taking the log and expanding leads to

$$\log(p(\mathbf{w}|\mathbf{y}, X)) = \frac{1}{2}\left[\frac{1}{\sigma_n^2}\mathbf{y}^T\mathbf{y} - \frac{1}{\sigma_n^2}\mathbf{y}^T X^T\mathbf{w} - \frac{1}{\sigma_n^2}\mathbf{w}^T X\mathbf{y} + \frac{1}{\sigma_n^2}\mathbf{w}^T XX^T\mathbf{w} + \mathbf{w}^T\Sigma_p^{-1}\mathbf{w}\right] \tag{3.41}$$

$$= \frac{1}{2}\left[\mathbf{w}^T\left(\frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1}\right)\mathbf{w} - \left(\frac{1}{\sigma_n^2}\mathbf{y}^T X^T\right)\mathbf{w} - \mathbf{w}^T\left(\frac{1}{\sigma_n^2}X\mathbf{y}\right) + \mathbf{y}^T\frac{1}{\sigma_n^2}\mathbf{y}\right] \tag{3.42}$$

$$= \mathbf{w}^T A\mathbf{w} - B^T\mathbf{w} - \mathbf{w}^T B + C \tag{3.43}$$

where we have defined

$$A := \frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1} \tag{3.44}$$

$$B := \frac{1}{\sigma_n^2}X\mathbf{y} \tag{3.45}$$

$$C := \mathbf{y}^T\frac{1}{\sigma_n^2}\mathbf{y} \tag{3.46}$$

Now we can complete the square so that

$$\mathbf{w}^T A\mathbf{w} - B^T\mathbf{w} - \mathbf{w}^T B + C = (\mathbf{w} - \bar{\mathbf{w}})^T A\,(\mathbf{w} - \bar{\mathbf{w}}) + K \tag{3.47}$$

leading to

$$\bar{\mathbf{w}} = A^{-1}B = \frac{1}{\sigma_n^2}\left(\frac{1}{\sigma_n^2}XX^T + \Sigma_p^{-1}\right)^{-1}X\mathbf{y} \tag{3.48}$$

$$K = C - \bar{\mathbf{w}}^T A\bar{\mathbf{w}} \tag{3.49}$$

Since K does not depend on $\mathbf{w}$ directly, it may be absorbed into the normalization of $p(\mathbf{w}|\mathbf{y}, X)$. Thus we are left with

$$p(\mathbf{w}|\mathbf{y}, X) = \mathcal{N}\left(\bar{\mathbf{w}} = \frac{1}{\sigma_n^2} A^{-1} X\mathbf{y}, \Sigma = A^{-1}\right) \tag{3.50}$$

$$A = \frac{1}{\sigma_n^2} XX^T + \Sigma_p^{-1} \tag{3.51}$$

This result gives us the gaussian distriubtion over the space of possible parameter vectors $\mathbf{w}$. To use this distribution to make predictions, consider a newly supplied testpoint $\mathbf{x}_*$. We want to find

$$p(y_*|\mathbf{x}_*, \mathbf{y}, X) \tag{3.52}$$

We do this by marginalizing over our weight distribution, i.e.

$$p(y_*|\mathbf{x}_*, \mathbf{y}, X) = \int_{\mathbf{w}} p(y_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|\mathbf{y}, X) d\mathbf{w} \tag{3.53}$$

If we make the further assumption that testing points are i.i.d. guassian distriubted, we see that this integral is the product of two gaussians and therefore is also a guassian. To find the mean and covariance of the predictive distribution, we check

$$\bar{y}_* = \mathbb{E}[y_*] = \mathbb{E}[\mathbf{x}_*^T \mathbf{w}] = \mathbf{x}_*^T \mathbb{E}[\mathbf{w}] = \mathbf{x}_*^T \bar{\mathbf{w}} \tag{3.54}$$

$$\mathrm{Cov}(y_*) = \mathbb{E}[(y_* - \bar{y}_*)(y_* - \bar{y}_*)^T] \tag{3.55}$$

$$= \mathbb{E}[(\mathbf{x}_*^T \mathbf{w} - \mathbf{x}_*^T \bar{\mathbf{w}})(\mathbf{x}_*^T \mathbf{w} - \mathbf{x}_*^T \bar{\mathbf{w}})^T] \tag{3.56}$$

$$= \mathbb{E}[\mathbf{x}_*^T (\mathbf{w} - \bar{\mathbf{w}})(\mathbf{w} - \bar{\mathbf{w}})^T \mathbf{x}_*] \tag{3.57}$$

$$= \mathbf{x}_*^T \mathbb{E}[(\mathbf{w} - \bar{\mathbf{w}})(\mathbf{w} - \bar{\mathbf{w}})^T] \mathbf{x}_* \tag{3.58}$$

$$= \mathbf{x}_*^T \mathrm{Cov}(\mathbf{w}) \mathbf{x}_* \tag{3.59}$$

$$= \mathbf{x}_*^T A^{-1} \mathbf{x}_* \tag{3.60}$$

so that

$$\boxed{p(y_*|\mathbf{x}_*, \mathbf{y}, X) = \mathcal{N}\left(\mathbf{x}_*^T \mathbf{w},\ \mathbf{x}_*^T A^{-1} \mathbf{x}_*\right)} \tag{3.61}$$

Let's now take a break from our Bayesian regression discussion and return to the standard linear regression model for a moment. The key drawback of linear models like this is, of course, that they're *linear*! Considering that many (most) *interesting* relationships are non-linear, how can we extend our simple linear model to enable us to perform complicated non-linear fits?

In the parlance of machine learning, the simple solution is to do feature engineering. If our inital feature vector is

$$\mathbf{x} = (x_1, ..., x_n) \tag{3.62}$$

we can use our *expertise* to concoct new combinations of these features to produce the agumented vector

$$\tilde{\mathbf{x}} = (x_1, ..., x_n, x_1^2,\ sin(x_2),\ x_5 x_7 / x_4,\ ...) \tag{3.63}$$

As an example, a linear classifier is unable to distinguish points inside a circle from those outside just from the (x, y) coordinates alone. Augmenting the feature vector to include the squared radius $x^2 + y^2$ as a new feature removes this obstacle. This works because the *linear* part of linear regression only refers to the fact that our model takes *linear combinations* of feature variables to produce it's output. There is no restriction that the features themselves need to be independent variables! This same idea is what makes methods like SINDy work which use libraries of polynomial combinations of base features.

Constructing new features is often more art than science. To standardize the process, let's abstract the mapping from the original feature vector $\mathbf{x}$ to the augmented vector $\tilde{\mathbf{x}}$. This is accomplished via the projection map $\phi : \mathbb{R}^D \to \mathbb{R}^N$ where

$$\mathbf{x} \mapsto \tilde{\mathbf{x}} = \phi(\mathbf{x}) \tag{3.64}$$

The result is that our linear model updates to become

$$f(\mathbf{x}) := \phi(\mathbf{x})^T \mathbf{w} \tag{3.65}$$

where the weight vector has gone from D dimensional to N dimensional. Similarly, the normal equations for $\mathbf{w}$ update to become

$$\mathbf{w} = (\Phi\Phi^T)^{-1}\Phi\mathbf{y} \tag{3.66}$$

where $\Phi = \phi(X)$ is the $N \times n$ matrix resulting from applying ϕ columnwise to X.

The following example shows how to use such a mapping to produce a quadratic polynomial fit.

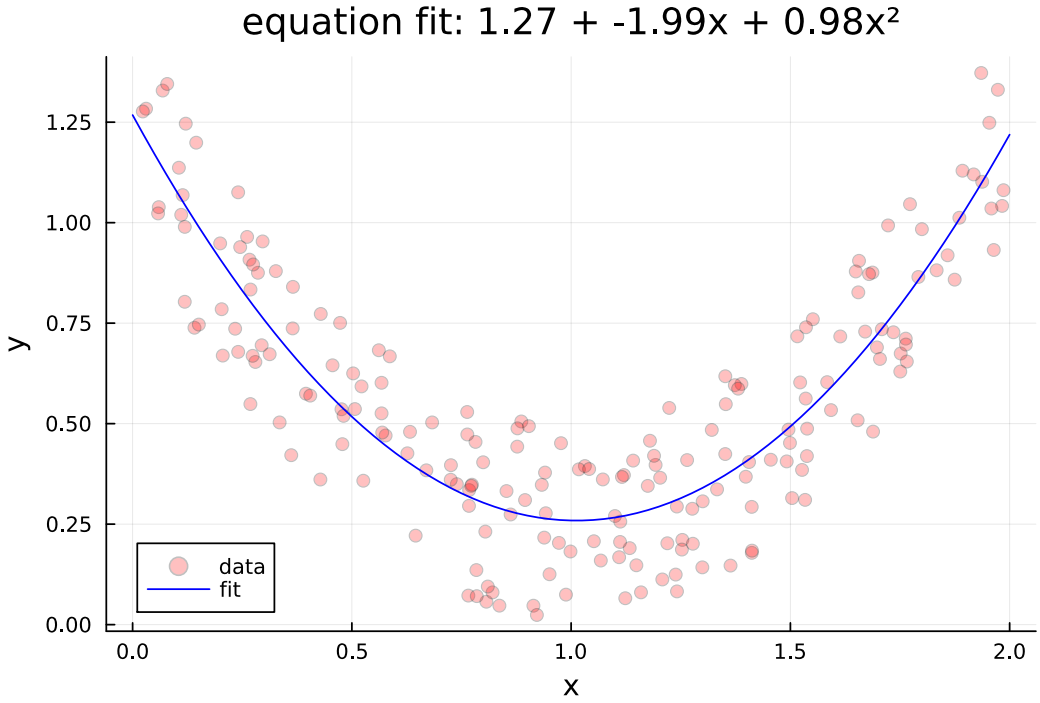


Figure 3.2: An example of polynomial regression for which we fit a quadratic polynomial to some noisy data.

We see from Figure 3.2 that our linear regression model found a great fit for a 2nd order polynomial when supplied with polynomial features. Let's update our Bayesian regression scheme to reflect the use of our feature projection map ϕ. First we define

$$\Phi := \phi(X) \tag{3.67}$$

$$\phi_* := \phi(\mathbf{x}_*) \tag{3.68}$$

Our predictive distribution therefore becomes

$$p(y_*|\mathbf{x}_*, X, \mathbf{y}) = \mathcal{N}\left(\frac{1}{\sigma_n^2}\phi_*^T A^{-1}\Phi\mathbf{y},\ \phi_*^T A^{-1}\phi_*\right) \tag{3.69}$$

$$A = \frac{1}{\sigma_n^2}\Phi\Phi^T + \Sigma_p^{-1} \tag{3.70}$$

Great! Now we can do our Bayesian inference with non-linear features given by ϕ. There is a *massive* problem with this approach, however. Our order 2 polynomial map ϕ takes us from a D dimenional feature vector to $(D+1)!$ many. This means that as we add more features to our feature map, the dimension of the resulting vector will quickly become prohibitively large. Looking at our current model equations, we see that the bottleneck is the matrix inversion of A which requires we invert an $N \times N$ matrix. Our prediction (i.e. the mean) involves multiplication on the right by the n dimensional vector $\mathbf{y}$. With that in mind, perhaps we can reformulate the above into an equivalent form using at most an $n \times n$ dimensional matrix.

Let $K := \Phi^T\Sigma_p\Phi$. Observe the following:

$$\frac{1}{\sigma_n^2}\Phi(K + \sigma_n^2 I) = \frac{1}{\sigma_n^2}\Phi\left(\Phi^T\Sigma_p\Phi + \sigma_n^2 I\right) \tag{3.71}$$

$$= \frac{1}{\sigma_n^2}\Phi\Phi^T\Sigma_p\Phi + \Phi I \tag{3.72}$$

$$= \left(\frac{1}{\sigma_n^2}\Phi\Phi^T\right)\Sigma_p\Phi + \left(\Phi I\Phi^{-1}\Sigma_p^{-1}\right)\Sigma_p\Phi \tag{3.73}$$

$$= \left(\frac{1}{\sigma_n^2}\Phi\Phi^T + \Sigma_p^{-1}\right)\Sigma_p\Phi \tag{3.74}$$

$$= A\Sigma_p\Phi \tag{3.75}$$

From there we see that

$$A^{-1}\frac{1}{\sigma_n^2}\Phi\left(K + \sigma_n^2 I\right) = \Sigma_p\Phi \tag{3.76}$$

$$\Rightarrow \frac{1}{\sigma_n^2}A^{-1}\Phi = \Sigma_p\Phi\left(K + \sigma_n^2 I\right)^{-1} \tag{3.77}$$

$$\Rightarrow \frac{1}{\sigma_n^2}\phi_*^T A^{-1}\Phi = \phi_*^T\Sigma_p\Phi\left(K + \sigma_n^2 I\right)^{-1} \tag{3.78}$$

For the covariance, we utilize the matrix inversion lemma (ADD REFERENCE HERE) which states

$$(Z + UWV^T)^{-1} = Z^{-1} - Z^{-1}U(W^{-1} + V^T Z^{-1} U)^{-1} V^T Z^{-1} \tag{3.79}$$

With the identification

$$Z^{-1} \to \Sigma_p \tag{3.80}$$

$$W^{-1} \to \sigma_n^2 I \tag{3.81}$$

$$V \to \Phi \tag{3.82}$$

$$U \to \Phi \tag{3.83}$$

we find

$$\Sigma_p - \Sigma_p \Phi \left(\Sigma_p + \Phi^T \Sigma_p \Phi\right)^{-1} \Phi^T \Sigma_p = \left(\Sigma_p^{-1} + \Phi \frac{1}{\sigma_n^2} I \Phi^T\right)^{-1} \tag{3.84}$$

$$= \left(\frac{1}{\sigma_n^2} \Phi \Phi^T + \Sigma_p^{-1}\right)^{-1} \tag{3.85}$$

$$= A^{-1} \tag{3.86}$$

Thus, we have the equivalent form for our predictive distribution:

$$\boxed{p(y_*|\mathbf{x}_*, X, \mathbf{y}) = \mathcal{N}\left(\phi_*^T \Sigma_p \Phi (K + \sigma_n^2 I)^{-1} \mathbf{y},\ \phi_*^T \Sigma_p \phi_* - \phi_*^T \Sigma_p \Phi (K + \sigma_n^2 I)^{-1} \Phi^T \Sigma_p \phi_*\right)} \tag{3.87}$$

where the pesky $N \times N$ term has been replaced by the $n \times n$ matrix $\Phi^T \Sigma_p \Phi$.

We now make the the *key* observation that the only matrices that appear in the above expression are

$$\Phi^T \Sigma_p \Phi, \qquad \phi_*^T \Sigma_p \phi_* \tag{3.88}$$

$$\phi_*^T \Sigma_p \Phi, \qquad \Phi^T \Sigma_p \phi_* \tag{3.89}$$

whose matrix elements we can write abstractly as

$$\phi(\mathbf{x})^T \Sigma_p \phi(\mathbf{x}') \tag{3.90}$$

To fit our model, we must determine appropriate values for the symmetric, positive semi-definite covariance matrix Σ_p (and σ_n too, technically). Instead, we observe that this matrix product is a quadratic form which we can think of as representing an inner product on our transformed vectors:

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{3.91}$$

We call the function $k(\mathbf{x}, \mathbf{x}')$ the *kernel function* or the *covariance function*.

All we need to perform the above calculations are the matrix elements of K applied to our data $\mathcal{D}$ and any test points $\mathbf{x}_*$ we wish to apply our model to. In effect, this means we are free to use feature vectors of any dimension, including ∞. The idea here is that the kernel function represents an inner product over *some* vector space. As it turns out, the RBF kernel corresponds to a an infinite dimensional feature vector.

There are many choices for the kernel function. One of the most popular is the RBF (radial basis function) kernel, also commonly referred to as the *squared exponential kernel*:

$$k_{\text{rbf}}(\mathbf{x}, \mathbf{x}') := \sigma_f^2 \exp(-\frac{1}{2\ell^2}|\mathbf{x} - \mathbf{x}'|^2) \tag{3.92}$$

where σ_f^2 is the *signal variance* and ℓ denotes the similarity length scale.

For notational convenience, let's define

$$K := k(X, X) \tag{3.93}$$

$$K_{**} := k(X_*, X_*) \tag{3.94}$$

$$K_* := k(X, X_*) \tag{3.95}$$

then, our predictive distribution takes the final, *clean* form

$$\boxed{p(\mathbf{y}_*|X_*, X, \mathbf{y}) = \mathcal{N}\left(K_*^T(K + \sigma_n^2 I)^{-1}\mathbf{y},\ K_{**} - K_*^T(K + \sigma_n^2 I)^{-1}K_*\right)} \tag{3.96}$$

This is the *end-result* of Gaussian Process Regression acheived via the *weight-space view*.

The Function-space View

So far our approach has been to generalize the standard linear regression model to allow for fitting over a (possibly infinite) basis of features with consideration for measurement and model uncertainty (our Bayesian priors). In essence, the idea was to fit the distribution of all possible weights conditioned on the available training data, $p(\mathbf{w}|X, \mathbf{y})$. A second *equivalent* approach is to instead consider the distribution of all possible model function $f(\mathbf{x})$. By constructing a Bayesian prior over this space, we constrain the space of possible model functions and learn a *distribution* over all allowed model functions, $p(f|X, \mathbf{y})$. To do so we will need to develop the abstract machinery of distributions over function spaces. When these distributions are Gaussian, the result is called a **Gaussian process**.

By this point, we are very familiar with the Gaussian distribution, a.k.a. the Normal distribtion $\mathcal{N}(\mu, \sigma^2)$. This distribution is defined by a mean value μ and a variance σ^2. It's *big brother* is the **Multivariate Normal Distribution**, $\mathcal{N}(\mu, \Sigma)$, described be a vector of means μ and a covariance matrix Σ. A natural question, then, is can we generalize the concept of the Gaussian distribution from N dimensions to being defined over a continuous field? This leads us naturally to the development of a so-called **Gaussian Process**.

> **Definition**: A *Gaussian Process*, $\mathcal{GP}$, is a collection of random variables for which any finite subset are described by a joint Gaussian distribution.

To see where this comes from, recall that in our previous derivation, we already made the assumption that all our our data points $\mathcal{D}$ are i.i.d. Gaussian distributed. A gaussian process is the natural extension of this and makes the assumption that the continuous set from which are data are sampled are so Guassian that any finite sample will be jointly Gaussian distributed. The term process is used to distinguish between finite collections of random variables (distributions) and their continuous counterparts described here.

Because each finite subset of this continuous collection is jointly gaussian, we can completely specify a Gaussian Process with two functions: the mean function $m(\S)$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$. To denote this, we typically write

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3.97}$$

To see this in action, recall our Bayesian regression model

$$f(\mathbf{x} = \phi(\mathbf{x})^T \mathbf{w} \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p) \tag{3.98}$$

where we have set the prior on $\sqsupseteq$ to have zero mean. The mean function is given by the expectation value of our model:

$$\mathbb{E}[f(\mathbf{x})] = \phi(\mathbf{x})^T \mathbb{E}[\mathbf{w}] = 0 \tag{3.99}$$

and the covariance function is given by

$$\mathbb{E}[f(\mathbf{x}) f(\mathbf{x}')] = \phi(\mathbf{x})^T \mathbb{E}[\mathbf{w}\mathbf{w}^T] \phi(\mathbf{x}') = \phi(\mathbf{x})^T \Sigma_p \phi(\mathbf{x}') \tag{3.100}$$

To repeat the point, the key feature of Gaussian processes is that finite subsets are jointly Gaussian distributed. Thus we can we can split our data into the testpoints $\mathcal{D} = (X, \mathbf{y})$ and testpoints X_* t and treat each collection as joint distributions with the following priors:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right) \tag{3.101}$$

where $\mathbf{f} := f(X)$ and $\mathbf{f}_* = f(X_*)$.

To obtain our predictive distribution, $p(\mathbf{f}_* | X_*, X, \mathbf{y})$, we *condition the joint prior distribution* on the observations. To see how this works, consider a general joint gaussian distribution given by

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right) \tag{3.102}$$

define the centered values $\tilde{x} := x - \mu_x$ and $\tilde{y} := x - \mu_y$. Define the intermediate variable

$$z := \tilde{x} - A\tilde{y} \tag{3.103}$$

Note that since we've subtracted out the mean we have $\mathbb{E}[\tilde{x}] = \mathbb{E}[\tilde{y}] = \mathbb{E}[z] = 0$. Let's now find A.

$$\mathbb{E}[z\tilde{y}^T] = \mathbb{E}[(\tilde{x} - A\tilde{y})\tilde{y}^T] \tag{3.104}$$

$$= \mathbb{E}[\tilde{x}\tilde{y}^T - A\tilde{y}\tilde{y}] \tag{3.105}$$

$$= \mathbb{E}[\tilde{x}\tilde{y}^T] - \mathbb{E}[A\tilde{y}\tilde{y}^T] \tag{3.106}$$

$$= \Sigma_{xy} - A\mathbb{E}[\tilde{y}\tilde{y}^T] \tag{3.107}$$

$$= \Sigma_{xy} - A\Sigma_{yy} \tag{3.108}$$

Therefore if we choose A so that z and $\tilde{y}$ are independent and uncorrelated, then $\Sigma_{zy} = \mathbb{E}[z\tilde{y}^T] = 0$. Using this assumption, we find

$$0 = \mathbb{E}[z\tilde{y}^T] = \Sigma_{xy} - A\Sigma_{yy} \Rightarrow \boxed{A = \Sigma_{xy}\Sigma_{yy}^{-1}} \tag{3.109}$$

If we now condition $\tilde{x}$ on $\tilde{y}$ (i.e. look at $\tilde{x}$ when $\tilde{y}$ is constant), we find

$$\mathbb{E}[\tilde{x}|\tilde{y}] = A\tilde{y} + \mathbb{E}[z] \tag{3.110}$$

$$= A\tilde{y} + 0 \tag{3.111}$$

$$= \Sigma_{xy}\Sigma_{yy}^{-1} \tag{3.112}$$

$$\tag{3.113}$$

By manipulating this expression, we can now derive $\mathbb{E}[x|y]$ as follows:

$$\mathbb{E}[x|\tilde{y}] = \mathbb{E}[\tilde{x}|\tilde{y}] + \mu_x \tag{3.114}$$

$$= \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}\tilde{y} \tag{3.115}$$

$$\tag{3.116}$$

$$\boxed{\mathbb{E}[x|y] = \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y)} \tag{3.117}$$

Similarly for the covariance, we have

$$\begin{aligned}
\mathrm{Cov}(x|y) &= \mathrm{Cov}(\tilde{x} + \mu_x|\tilde{y}) && (3.118)\\
&= \mathrm{Cov}(\tilde{x} + \mu_x|\tilde{y} + \mu_y) && (3.119)\\
&= \mathrm{Cov}(\tilde{x}|(\tilde{y} + \mu_y)) && (3.120)\\
&= \mathrm{Cov}(\tilde{x}|\tilde{y}) && (3.121)\\
&= \mathrm{Cov}((z + A\tilde{y})|\tilde{y}) && (3.122)\\
&= \mathrm{Cov}(z) + A\mathrm{Cov}(\tilde{y}) && (3.123)\\
&= \mathrm{Cov}(z) + 0 && (3.124)\\
&= \mathbb{E}[zz^T] && (3.125)\\
&= \mathbb{E}[(\tilde{x} - A\tilde{y})(\tilde{x} - A\tilde{y})^T] && (3.126)\\
&= \mathbb{E}[\tilde{x}\tilde{x}^T - A\tilde{y}\tilde{x}^T - x(A\tilde{y})^T + A\tilde{y}\tilde{y}^T A^T] && (3.127)\\
&= \Sigma_{xx} - A\Sigma_{yx} - \Sigma_{xy}A^T + A\Sigma_{yy}A^T && (3.128)\\
&= \Sigma_{xx} - (\Sigma_{xy}\Sigma_{yy}^{-1})\Sigma_{yx} - \Sigma_{xy}(\Sigma_{yy}^{-1})^T\Sigma_{xy}^T + \Sigma_{xy}\Sigma_y^{-1}\Sigma_y(\Sigma_y^{-1})^T\Sigma_{xy}^T && (3.129)\\
&= \Sigma_{xx} - \Sigma_{xy}\Sigma yy^{-1}\Sigma_{xy}^T - \Sigma_{xy}(\Sigma_{yy}^{-1})^T\Sigma_{xy}^T + \Sigma_{xy}(\Sigma_{yy}^{-1})^T\Sigma_{xy}^T && (3.130)\\
&= \Sigma_{xx} - \Sigma_{xy}\left[\Sigma_{yy}^{-1} - (\Sigma_{yy}^{-1})^T + (\Sigma_{yy}^{-1})^T\right]\Sigma_{xy}^T && (3.131)\\
&= \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx} && (3.132)
\end{aligned}$$

$$\boxed{\mathrm{Cov}(x|y) = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}} \tag{3.133}$$

Armed with this identity for joint Guassian distributions, we are ready to derive the predictive distribution for Gaussian Process Regression. We find:

$$p(\mathbf{f}_*|X_*, X, \mathbf{y} = \mathcal{N}\left(K_*^T K^{-1}\mathbf{f},\ K_{**} - K_*^T K^{-1}K_*\right). \tag{3.134}$$

To account for noisy observations, we can augment our correlation function to include a noise offset. The joint distrubtion then becomes:

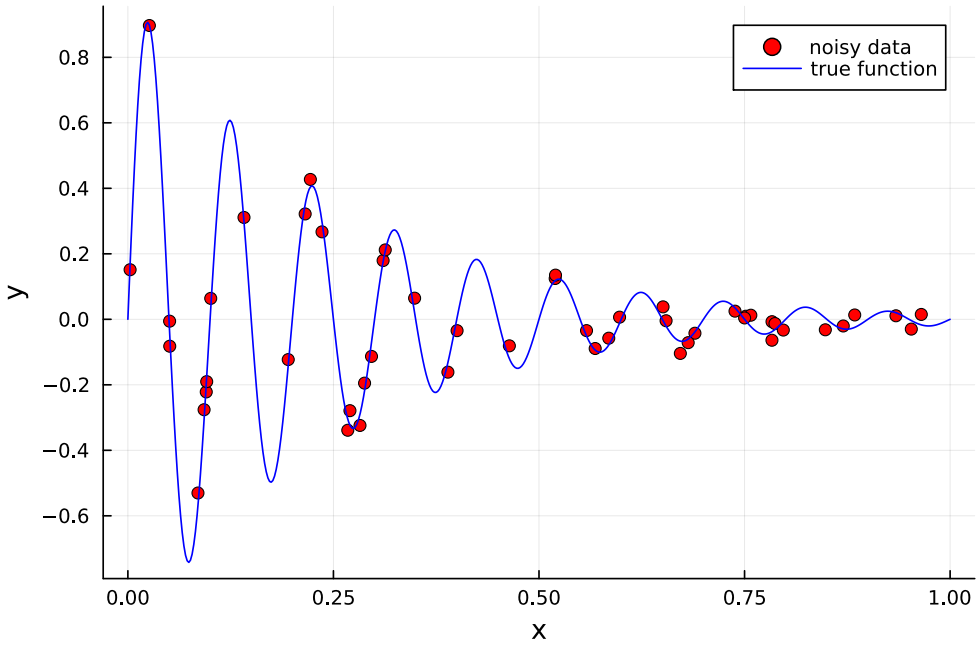
$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) - \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{3.135}$$

which leads to the predictive distribution

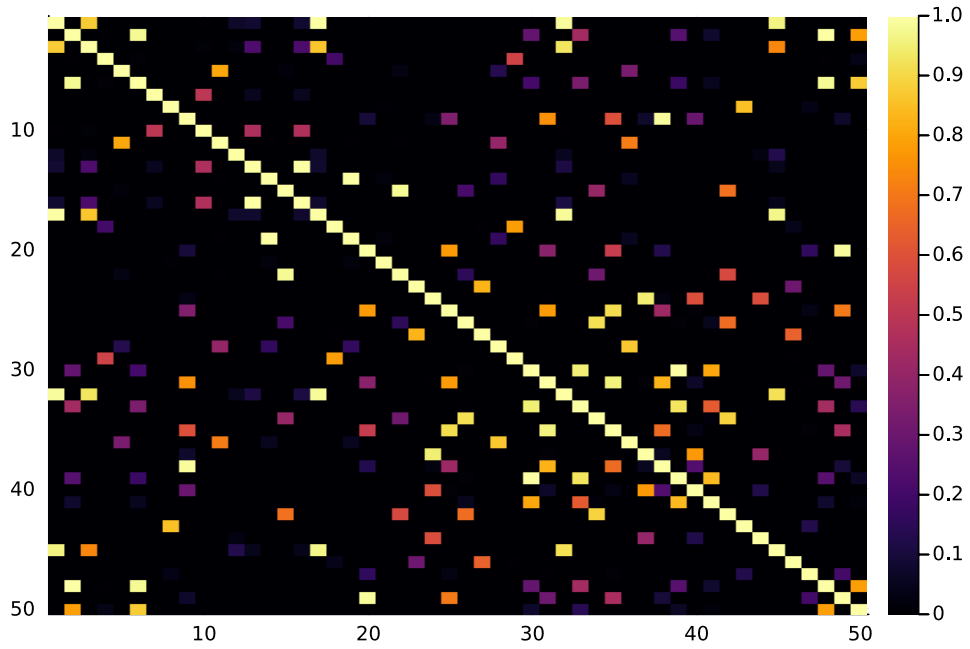
$$\boxed{p(\mathbf{f}_*|X_*,X,\mathbf{y}) = \mathcal{N}\left(K_*^T\left[K+\sigma_n^2 I\right]^{-1}\mathbf{f},\ K_{**} - K_*^T\left[K+\sigma_n^2 I\right]^{-1}K_*\right)} \tag{3.136}$$

Doing it in Julia

To see how we can use this in practice, let's first construct some sample data which we seek to model as a Gaussian Process



The excellent package KernelFunctions.jl provides a clean interface to create various kernel functions and apply them to data to create our K-matrices. Due to the fact that kernel functions obey composition laws, we can easily build up complicated Kernels from basic pieces via function composition with $\circ$



Unsurprisingly, there is a lot of activation on the diagonal as for a single datapoint $\mathbf{x}$, we have

$$k(\mathbf{x}, \mathbf{x}) = \exp\left(-\frac{0}{2\ell^2}\right) = 1.0 \tag{3.137}$$

The package AbstractGPs.jl provides an excellent way to define Gaussian Processes by supplying mean and kernel functions. We can then sample from our GPs with a simple interface designed to extend the basic functions from `Statistics.jl`. From an `AbstractGP` we can construct a `FiniteGP` by *indexing* into our datasets. The procedure can be summarized as follows

1. Build a kernel function $k(\cdot, \cdot)$ via composition using `KernelFunctions.jl`
2. Construct an a Gaussian Process $f \sim \mathcal{GP}$ abstractly using `AbstractGPs.jl`
3. Construct a finite representation of our GP, f_x, over our training data
4. Construct a posterior Gaussian Process from f_x and our training targets $\mathbf{y}$.
5. Construct a finite representation of the posterior GP applied to our prediction data

6. Sample this final distribution to obatin a prediction via `mean()` and variances via `var()`. Alternatively, we can obtain a multivariate normal distribution for each point by calling `marginals()`.

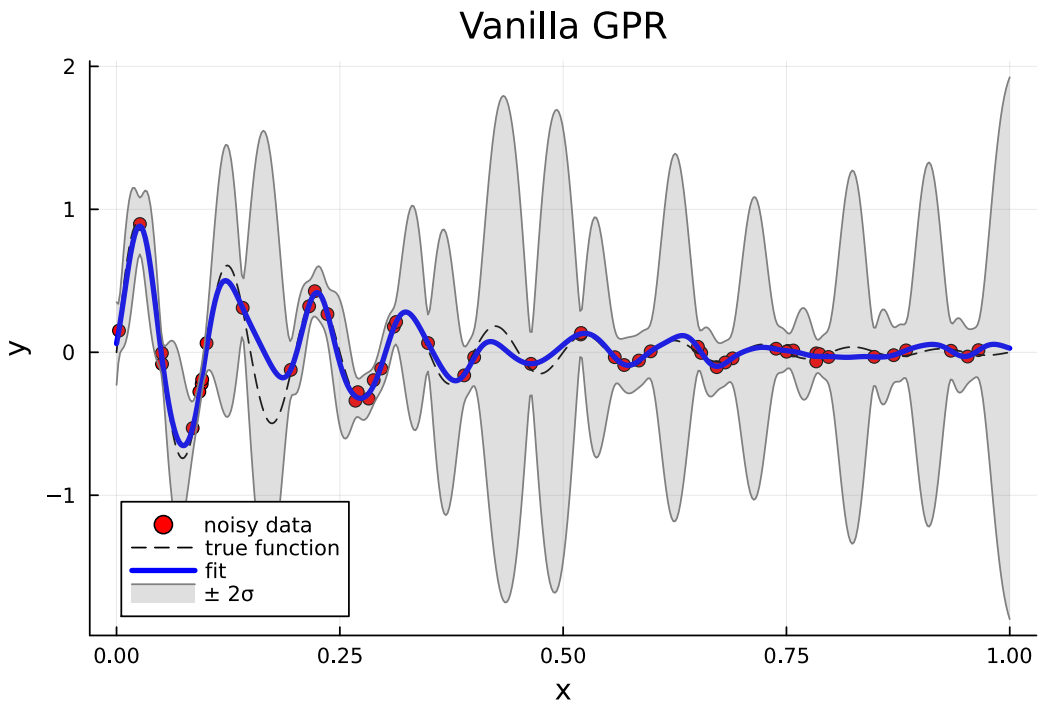


Figure 3.3: A Gaussian Process fit to the training data illustrating the predication (means) and a $\pm 2\sigma$ uncertainty interval. We can clearly see how the uncertainty is larger for inputs far away from supplied training data.

Fitting the Kernel Hyperparameters

At this point, it is easy to think we are finished; we *already* fit the Guassian process. However, we were forced to choose values for both ℓ and σ^2 kernel parameters. How can we optimally select the ideal hyperparameters for our Gaussian Process? This question leads us into the realm of Bayesian Model Selection. Rather than focusing specifically on our Gaussian Process model, let's take a step back and think about the process of model selection from a Bayesian perspective.

There are several levels of parameters in machine learning. At the lowest level, we have the model weights $\mathbf{w}$. Above that, we have model hyperparameters, θ. At the top we have model structure $\mathcal{H}$. In our Bayesian framework, we can consider prior distributions defined at each of these levels which codify credences for how much we trust a particular model, hyperparameter, etc. At the bottom, we have

$$p(\mathbf{w}|X, \mathbf{y}, \theta, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \mathbf{w}, \theta, \mathcal{H}_i)p(\mathbf{w}|\theta, \mathcal{H}_i)}{p(\mathbf{y}|X, \theta, \mathcal{H}_i)} \tag{3.138}$$

If this looks confusing, consider Bayes rule for 3 events R, H, S. We have:

$$P(R|H, S) = \frac{P(R, H, S)}{P(H, S)} \tag{3.139}$$

$$= \frac{P(H|R, S)P(R, S)}{P(H, S)} \tag{3.140}$$

$$= \frac{P(H|R, S)P(R|S)P(S)}{P(H|S)P(S)} \tag{3.141}$$

$$= \frac{P(H|R, S)P(R|S)}{P(H|S)} \tag{3.142}$$

To get the result, just think of θ and $\mathcal{H}_i$ as a single *event* and translate the above to distribution functions.

The prior $p(\mathbf{w}|\theta, \mathcal{H}_i)$ encodes any knowledge we have about the parameters prior to seeing the data. The denominator is the *marginal likelihood* and is given by

$$p(\mathbf{y}|X, \theta, \mathcal{H}_i) = \int d\mathbf{w}\ p(\mathbf{y}|X, \mathbf{w}, \theta, \mathcal{H}_i)p(\mathbf{w}|\theta, \mathcal{H}_i) \tag{3.143}$$

The next level up is to express the distribution of hyper-parameters θ:

$$p(\theta|X, \mathbf{y}, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \theta, \mathcal{H}_i)p(\theta|\mathcal{H}_i)}{p(\mathbf{y}|X, \mathcal{H}_i)}. \tag{3.144}$$

Here $p(\theta|\mathcal{H}_i)$ is called the *hyper-prior*. Similarly, the normalization constant is given by

$$p(\mathbf{y}|X, \mathcal{H}_i) = \int d\theta\ p(\mathbf{y}|X, \theta, \mathcal{H}_i)p(\theta|\mathcal{H}_i). \tag{3.145}$$

Finally, at the top level we have the set of possible model structures $\{\mathcal{H}_i\}$. This leads to

$$p(\mathcal{H}_i|X,\mathbf{y}) = \frac{p(\mathbf{y}|X,\mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y}|X)} \tag{3.146}$$

with normlization constant

$$p(\mathbf{y}|X) = \sum_i p(\mathbf{y}|X,\mathcal{H}_i)p(\mathcal{H}_i). \tag{3.147}$$

Depending on the model details, these integrals may be intractable without approximations or Monte Carlo methods. Since we rarely have sufficient knowledge to form a hyperparameter prior, one often attempts to maximize the marginal likelihood $p(\mathbf{y}|X,\theta,\mathcal{H}_i)$ with respect to the hyperparameters θ instead. This is known as Type II Maximium Likelihood Estimation. In the case of Gaussian Process Regression, we are once again saved by the fact that every piece has a convenient functional from resulting in analytically tractible integrals for the marginal likelihood function. We find

$$\ln p(\mathbf{y}|X,\theta) = -\frac{1}{2}\mathbf{y}^T(K_f + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\ln|K_f + \sigma_n^2 I| - \frac{n}{2}\ln(2\pi) \tag{3.148}$$

where we have employed the natural logarithm remove the pesky exponentials and arrive at a very convenient form. Note that because the lograithm is monotonically increasing, the maximum of the log-marginal-likelihood will be the same as the marginal-likelihood itself. Provided this functional form, we can use our favorite optimization routine to obtain the kernel hyperparameters which maximize the likelihood of obtaining our data as illustrated in the following comparison figure.

To enable easy application of Gaussian Process Regression in Julia's machine learning framework `MLJ.jl`, we have created an open-source implementation which is made freely available at `https://github.com/john-waczak/MLJGaussianProcesses.jl/tree/main`

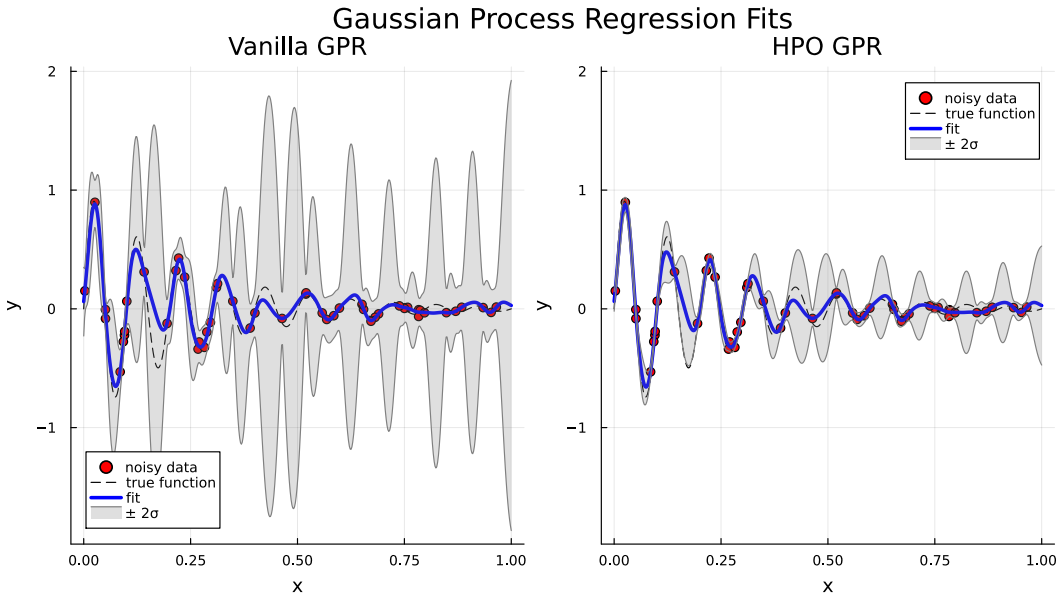


Figure 3.4: Left: the original Gaussian Process obtained with our default choice of kernel parameters. Right: A much better Gaussian Process obtained after performing hyperparameter optimization on the kernel parameters. Note how the mean function still does an excellent job fitting the data points while *also* minimizing the uncertainty of the fit.

3.2.3 Decision Trees

3.3 Unsupervised Classification

3.3.1 Self Organizing Maps

Self organizing maps (SOMs) are an unsupervised machine learning technique developed by Kohonen (Kohonen, 1982) based on the simple biological principle that *neurons near each other fire together*. This observation that the toplogical *closeness* of similar computational units is a critical feature of intelligent systems leads to a natural reinterpretation of the familiar perceptron model into a new form amenable for a variety of clustering and dimensionality reduction tasks. In particular, the SOM enables a rapid unsupervised classification of multidimensional data into a (typically) one or two dimensional *simplicial complex*, the discrete realization of a topological manifold, whose vertices correspond to representative points in the original data space $\mathcal{D}$. While a tad esoteric compared to other popular un-

supervised methods like KMeans clustering or DBSCAN, the SOM distinguishes itself with the added benefit that it's training procedure guarantees nodes (i.e. classes) close to each other in the feature manifold share similar weights. This additional structure makes the SOM particularly attractive when an interpretation of the discovered clusters as well as the relationships between them is desired.

The original treatment of the SOM by Kohonen was made in terms of processing units, sensory signals, and relaying networks (Kohonen, 1982), however, in the modern era of deep learning, a more easily digestible derivation can be obtained by re-interpreting the weights of a simple perceptron model to provide the foundation for a clustering approach. As described in previously, a perceptron is a function of the form

$$\mathbf{y} = \sigma.\left(W\mathbf{x}\right) \tag{3.149}$$

where $W \in \mathbb{R}^{n\times m}$ is a matrix of weights which transform the input $\mathbf{x} \in \mathbb{R}^m$ into $\mathbb{R}^n$ and σ is a nonlinear *activation function* applied element-wise to the outputs of the matrix multiplication (indicated by the . syntax). If we instead think of the weight matrix as an ordered collection of vectors $\{\mathbf{w}_i\}_{i=1}^n$, then this formula can be further decomposed into

$$(\mathbf{y})_i = \sigma(\mathbf{w}_i^T\mathbf{x}) = \sigma(\langle \mathbf{w}_i, \mathbf{x}\rangle) \tag{3.150}$$

The function of the perceptron is now clear: given an input vector $\mathbf{x}$ and a collection of n-many weight vectors $\mathbf{w}_i$, compute the n-many inner products of $\mathbf{x}$ with each weight vector $\mathbf{w}_i$, apply the nonlinear activation function σ, and concatenate the results.

If we now allow ourselves to imagine the weight vectors $\mathbf{w}_i$ as members of the same vector space as the input $\mathbf{x}$, a reasonable question to ask is: *how similar is the input* $\mathbf{x}$ *to each* $\mathbf{w}_i$. Further, the application of the inner product $\langle\cdot,\cdot\rangle$ suggests we may answer this question in terms of the distance

$$\langle \mathbf{w}_i - \mathbf{x}, \mathbf{w}_i - \mathbf{x}\rangle = d(\mathbf{w}_i, \mathbf{x})^2. \tag{3.151}$$

In other words, given a set of weight vectors $\mathbf{w}_i$ which we may now think of as the cluster centers for our unsupervised model, we can measure the similarity between a given datum $\mathbf{x}_j$ and each cluster by computing the distance

$$d_{ij} = d\left(\mathbf{w}_i, \mathbf{x}_j\right). \tag{3.152}$$

To make this mapping useful, we should prescribe a *training procedure* which updates the vectors $\mathbf{w}_i$ based on all of the available data points. Further, our goal is to establish some kind of interpretable relationship between the various weights $\mathbf{w}_i$ so that they *more than* disjoint classes. The SOM achieves this by providing a lower-dimensional grid (typically 2-dimensional) whose vertices are understood to be the location of each of the weight vectors. The procedure is as follows:

First, initialize a grid of weight vectors $\mathbf{w}_i \in \mathbb{R}^m$ which we collect into a matrix $W \in \mathbb{R}^{m\times n}$. Assign coordinates to each weight vector by mapping each $\mathbf{w}_i$ to a vertex in the grid. The *topology* of the grid is a hyperparameter which dictates the spatial relationship between neighboring nodes. Popular choices are: flat rectangular, rectangular on a cylinder (one periodic boundary condition), rectangular on a torus (two periodic boundary conditions), hexagonal (more equidistant neighbors per node than rectangular), and spherical. One can initialize the weights in a variety of ways. Some common choices are

- Randomly initialize them as in figure 3.5.
- Initialize them to the value of n-many randomly selected data samples $\mathbf{x}_i$.
- Initilize them to the first n-many principal components of the dataset.

Next, we proceed to update the weights according to the following steps for each $\mathbf{x}_k$ datum:

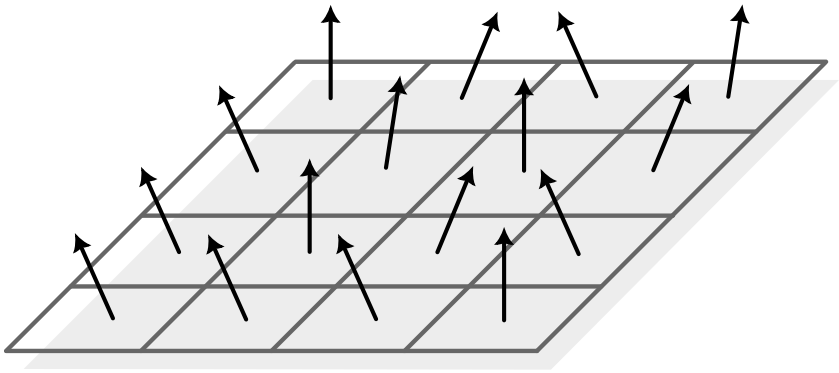


Figure 3.5: A Self Organizing Map with nodes configured in a rectangular grid topology with weight vectors randomly initialized.

1. Compute the *best match unit* (BMU) as

$$\mathbf{w}_i^{\text{best}} = \arg\min_{\mathbf{w}_j} \left(d(\mathbf{x}_k, \mathbf{w}_j)^2 \right) \tag{3.153}$$

   where $d(\cdot, \cdot)$ is some suitably chosen distance function. One often defaults to the Euclidean metric.

2. Choose a radius σ_t which we will use to identify the correct update for to apply to all nodes relative to $\mathbf{w}_i^{\text{best}}$.

3. Update *all* the weights according to the equation

$$w_m^{t+1} = w_m^t + \eta(t) f(x_m, y_n, \sigma_t) \left(\mathbf{x}_k - \mathbf{w}_n^t \right) \tag{3.154}$$

   where $\eta(t)$ is the learning rate, and $f(x_m, y_m, \sigma_t)$ is the *neighborhood* function which defines the relative amount each node $\mathbf{w}_m$ is updated according to its coordinate distance from the best match unit (i.e. the distance between the points (x_m, y_m) and $(x_{\text{best}}, y_{\text{best}})$ in the two dimensional case). Typically one chooses a the learning rate to

evolve according to the schedule

$$\eta(t) = \eta_0 \exp(^{-}\lambda t) \tag{3.155}$$

with the radius σ_t evolving similarly according to

$$\sigma_t = \sigma_0 \exp(-\beta t). \tag{3.156}$$

Popular choices for the neighborhood function are the Gaussian function, Mexican-hat function, a cone function, and a cylinder function.

A simple example: partitioning color spaces

As an illustrative example, consider the minimal dataset formed by the colors red, green, and blue which as vectors may be written as

$$\text{Red} = (1,\ 0,\ 0) \tag{3.157}$$

$$\text{Green} = (0,\ 1,\ 0) \tag{3.158}$$

$$\text{Blue} = (0,\ 0,\ 1)\,. \tag{3.159}$$

Now let us suppose we want to train an SOM to generate a classification for colors using only these three as the supplied training data. Figure 3.6 shows exactly this. Here we have trained an SOM of size 25×25 cells in a flat hexagonal topology. As the weight vectors are themselves vectors of length 3, we can then visualize the trained SOM by coloring each corresponding node by the color learned in its weight vector. What we find is that the resulting map has clearly separated red, green, and blue into distinct regions in the grid, and more importantly, there is a smooth gradient between neighboring cells reflecting the fact that neighboring classes are more *similar* than cells with a large separation distance. This is not the case for many other common clustering techniques like k-nearest neighbors or DBSCAN for which relationship between learned classes is challenging to interpret.

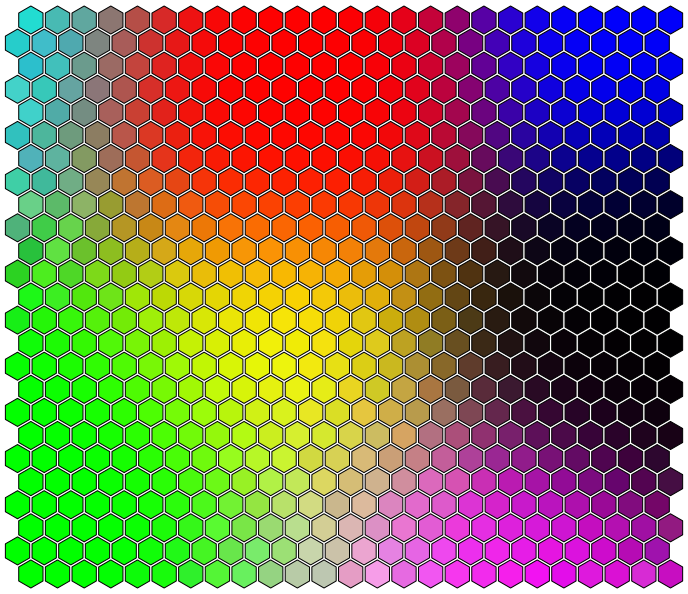


Figure 3.6: A trained Self Organizing Map of 25×25 cells in a hexagonal topology trained on a dataset consisting of the three colors red, green, and blue.

To enable this demonstration and its subsequent use throughout the rest of this dissertation, I have created a new open-source implementation of the SOM algorithm for the Julia programming language called `SelfOrganizingMaps.jl` which has been added to the general registry and can by freely downloaded for use by anyone. The repository for the code as well as the associated documentation can be found at this site.

3.3.2 Generative Topographic Maps

Now that we've developed the SOM algorithm, we see upon reflection that there are a few drawbacks to the methods, namely

- There is no probabilistic interpretation of the fitted SOM? Are we to trust the fit results with 100% certainty?

- There is no clearly stopping criteria for the learning process? How many epochs should be performed? Do you simply stop the training once the neighbor radius σ_t is less than the minimum neighbor distance?

With this in mind, we now seek to introduce a new method based on the SOM which will allow us to address the shortcomings by taking a principled, probabilistic approach, which will be called the *Generative Topographic Mapping* (GTM). This technique was first introduced by Bishop et al. in (Bishop et al., 1998) and our derivation of the method will follow their presentation closely.

Both the SOM and GTM rely on the assumption that our high dimensional dataset $\mathcal{D} \subseteq \mathbb{R}^D$ can be accurately represented as being constrained to a lower dimensional submanifold. In the SOM this is acheived by assigning coordinates to each of the weight vectors according to some predetermined grid. In the GTM, however, we make a slight change in our perspective and say that the high dimensional data we have are *generated* from some lower dimensional set of *latent vectors* living in $\mathbb{R}^L$ with $L < D$. For ease of visualization, it is standard to chose $L = 2$. Our goal, then, is to learn a mapping from the latent space to the data space which *maximizes the likelihood* of obtaining our dataset given the set of latent variables.

In the coming derivation, we shall assume the following notations:

$$\begin{aligned} \mathbf{x} \in \mathbb{R}^L &\quad \text{A latent vector} \\ \mathbf{t} \in \mathbb{R}^D &\quad \text{A data vector} \\ \mathbf{y} = \mathbf{y}(\mathbf{x}; W) &\quad \text{The transformation } \mathbf{y} : \mathbb{R}^L \to \mathbb{R}^D \\ W &\quad \text{The model weights} \end{aligned} \tag{3.160}$$

To get started, let us assume that our data can be reasonably described by a multivariate normal distrubtion such that

$$p(\mathbf{t}|\mathbf{x}; W, \beta) = \mathcal{N}\left(\mathbf{y}(\mathbf{x}; W), \frac{1}{\beta}\right), \tag{3.161}$$

that is, our data follow a normal distribution with mean given by the transformation function $\mathbf{y}$ applied to the latent variables $\mathbf{x}$ with covariance $\mathbf{1}\beta$. Suppose that we have some model for the prior distribution of our latent variables, $p(\mathbf{x})$, then by integration would find

$$p(\mathbf{t}|W,\beta) = \int d\mathbf{x} p(\mathbf{t}|\mathbf{x};W,\beta)p(\mathbf{x}) \tag{3.162}$$

which is the distribution we really care about, namely, the distribution of data $\mathbf{t} \in \mathcal{D}$ given our choice of model weights W and covariance β^{-1}. To make the problem tractible, we need an integrable distribution for $p(\mathbf{x})$. We therefore take inspiration from the SOM and choose $\mathbf{x}$ to be precisely distributed over a regular grid of points with equal weights. That is

$$p(\mathbf{x}) = \frac{1}{K}\sum_{k}^{K} \delta(\mathbf{x} - \mathbf{x}_k) \tag{3.163}$$

where $\mathbf{x}_k$ are the K-many grid points such that for any $\mathbf{x}$, there is a probability of precisely $\mathbf{1}K$ that it *came from* one of the nodes $\mathbf{x}_k$.

NOTE: Add figure of GTM grid here. (See pg 27 of notebook)

As mathematicians (or physicists, or data scientists, etc...) we rejoice at the wise decision to incorporate the integral-zapping Dirac-delta distributions which enables us to simplify the data distribution to become

$$\begin{aligned} p(\mathbf{t}|W,\beta) &= \int d\mathbf{x} p(\mathbf{t}|\mathbf{x};W,\beta)\frac{1}{K}\sum_{k}^{K}\delta(\mathbf{x}-\mathbf{x}_k) \\ &= \frac{1}{K}\sum_{k}^{K} p(\mathbf{t}|\mathbf{x}_k;W,\beta). \end{aligned} \tag{3.164}$$

Now provided we have in supply a dataset with N-many records, i.e. $\mathcal{D}^N \subseteq \mathcal{D} = \{\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_N\}$, how do we chose the *optimal* parameters W and β? We need an objective to optimize!

If we assume the $\mathbf{t}_i$ are independently, identically distributed, then the *likelihood* of obtaining the data given a choice of parameters (i.e. $p(\mathcal{D}^N|W,\beta)$) is

$$\begin{aligned}\mathcal{L}(W,\beta) &= \prod_n^N p(\mathbf{t}_n|W,\beta) \\ &= \prod_n^N \left(\frac{1}{K}\sum_k^K p(\mathbf{t}_n|\mathbf{x}_k;W,\beta)\right).\end{aligned} \tag{3.165}$$

Taking the logarithm (which must share the same optimum as it is a monotonically increasing function) yields the simpler expression

$$\ell(W,\beta) = \log(\mathcal{L}(W,\beta)) = \sum_n^N \log\left(\frac{1}{K}\sum_k^K p(\mathbf{t}_n|\mathbf{x}_k;W,\beta\right) \tag{3.166}$$

Maximizing this log-likelihood function is known as *maximum likelihood estimation* (MLE). But how should we optimize this function? We could use techniques like gradient descent or ADAM, however the probabilistic nature of the GTM suggests we may be able to obtain an *expectation-maximization* (EM) routine for a suitable choice of transformation $\mathbf{y}(\mathbf{x},W)$.

Suppose we already have some guesses W_o and β_o for the parameters (o for *old*). To simplify the derivation, we define $\theta_o = (W_o..., \beta_o)$. Then we can compute the responsibilities

r_{kn} as

$$\begin{aligned} r_{kn} &:= p(\mathbf{x}_k|\mathbf{t}_n, \theta_o) \\ &= \frac{p(\mathbf{t}_n|\mathbf{x}_k, \theta_o)p(\mathbf{x}_k|\theta_o)}{p(\mathbf{t}_n|\theta_o)} \\ &= \frac{p(\mathbf{t}_n|\mathbf{x}_k, \theta_o)p(\mathbf{x}_k|\theta_o)}{\sum_{k'}^{K} p(\mathbf{t}_n|\mathbf{x}_{k'}, \theta_o)p(\mathbf{x}_{k'}|\theta_o)} \\ &= \frac{p(\mathbf{t}_n|\mathbf{x}_k, \theta_o)\frac{p(\mathbf{x}_k, \theta_o)}{p(\theta_o)}}{\sum_{k'}^{K} p(\mathbf{t}_n|\mathbf{x}_{k'}, \theta_o)\frac{p(\mathbf{x}_{k'}, \theta_o)}{p(\theta_o)}} \\ &= \frac{p(\mathbf{t}_n|\mathbf{x}_k, \theta_o)p(\mathbf{x}_k, \theta_o)}{\sum_{k'}^{K} p(\mathbf{t}_n|\mathbf{x}_{k'}, \theta_o)p(\mathbf{x}_{k'}, \theta_o)} \\ &= \frac{p(\mathbf{t}_n|\mathbf{x}_k, \theta_o)p(\mathbf{x}_k)p(\theta_o)}{\sum_{k'}^{K} p(\mathbf{t}_n|\mathbf{x}_{k'}, \theta_o)p(\mathbf{x}_{k'})p(\theta_o)} \\ &= \frac{p(\mathbf{t}_n|\mathbf{x}_k, \theta_o)p(\mathbf{x}_k)}{\sum_{k'}^{K} p(\mathbf{t}_n|\mathbf{x}_{k'}, \theta_o)p(\mathbf{x}_{k'})} \\ &= \frac{p(\mathbf{t}_n|\mathbf{x}_k, \theta_o)\frac{1}{K}}{\sum_{k'}^{K} \mathbf{t}_n|\mathbf{x}_{k'}, \theta_o)\frac{1}{K}} \\ &= \frac{p(\mathbf{t}_n|\mathbf{x}_k, \theta_o)}{\sum_{k'}^{K} \mathbf{t}_n|\mathbf{x}_{k'}, \theta_o)} \end{aligned} \tag{3.167}$$

which we will soon utilize in our EM procedure.

Now, we must chose a form for the transformation $\mathbf{y}(\mathbf{x}, W)$ after which we will have all of the information we need to perform the computation. A convenient choice is to use a kernelized regression strategy so that

$$\mathbf{y} := \phi^T(\mathbf{x})W \tag{3.168}$$

where $W \in \mathbb{R}^{M\times D}$ are some constant parameters and $\phi : \mathbb{R}^L \to \mathbb{R}^M$ is a transformation employing M-many basis functions ϕ_m. To capture linear and nonlinear effects, we chose

$$\phi_m(\mathbf{x}) = \begin{cases} \exp(-\frac{|\mathbf{x}-\mu_m|^2}{2\sigma}), & m < M \\ 1, & m = M \end{cases} \tag{3.169}$$

that is, $M-1$ Gaussians and a term for a bias offset. If we apply the transformation to each of the K-many latent nodes, then we may collect the resulting vectors into a matrix,

$$Y = \Phi W, \tag{3.170}$$

where $\Phi \in \mathbb{R}^{K \times M}$ with $\Phi_{km} = \phi_m(\mathbf{x}_k)$.

Now that we are able to compute the responsibilities of each latent node to the observed data (the expectation step), we need to compute the relevant derivatives which we will use in the maximization step. For this, we treat the responsibilities r_{kn} as fixed, and solve for the relevant updates to W and β which make there derivatives 0 and therefore optimize ℓ. Let's begin by differentiating ℓ with respect to the weight matrix W. We have

$$\begin{aligned}
0 &= \frac{\partial}{\partial W_{md}} \ell(W, \beta) && (3.171)\\
&= \frac{\partial}{\partial W_{md}} \sum_n^N \log\left(\frac{1}{K}\sum_k^K p(\mathbf{t}_n|\mathbf{x}_k; W, \beta\right) && (3.172)\\
&= \sum_n^N \frac{1}{\frac{1}{K}\sum_{k'}^K p(\mathbf{t}_n|\mathbf{x}_{k'}; W, \beta)} \frac{1}{K} \frac{\partial}{\partial W_{md}} \sum_k^K p(\mathbf{t}_n|\mathbf{x}_k; W, \beta) && (3.173)
\end{aligned}$$

Noting that $p(\mathbf{t}_n|\mathbf{x}_k; W, \beta) = \mathcal{N}(\mathbf{y}_k, \beta^{-1})$, then the derivative of the exponential yields

$$\begin{aligned}
0 &= \sum_n^N \sum_k^K \frac{p(\mathbf{t}_n|\mathbf{x}_k; W, \beta)}{\sum_{k'}^K p(\mathbf{t}_n|\mathbf{x}_{k'}; W, \beta)} \frac{\partial}{\partial W_{md}} \left(-\frac{\beta}{2}|\mathbf{t}_n - \mathbf{y}_k|^2\right) && (3.174)\\
&= \sum_n^N \sum_k^K r_{kn} \frac{\partial}{\partial W_{md}} \left(-\frac{\beta}{2}|\mathbf{t}_n - \mathbf{y}_k|^2\right) && (3.175)\\
&= \sum_n^N \sum_k^K (-\beta) r_{kn} \sum_q^D (y_k^q - t_n^q) \frac{\partial}{\partial W_{md}} \sum_s^M \phi_s(\mathbf{x}_k) W_{sq} && (3.176)\\
&= \sum_n^N \sum_k^K \sum_q^D (-\beta) r_{kn} (y_k^q - t_k^q) \sum_s^M \phi_s(\mathbf{x}_k) \delta_{sm} \delta_{qd} && (3.177)\\
&= \sum_n^N \sum_k^K (-\beta) r_{kn} (y_k^d - t_k^d) \phi_m(\mathbf{x}_k) && (3.178)
\end{aligned}$$

Looking closely at the above expression, we see that there are two free indices suggesting we may write the above as a matrix expression. To do so, let's introduce a diagonal matrix G such that $G_{kk} = \sum_n r_{kn}$. Then upon rearrangement, we find

$$\sum_n^N \sum_k^K r_{kn} y_k^d \phi_m(\mathbf{x}_k) = \sum_n^N \sum_k^K r_{kn} t_n^d \phi_m(\mathbf{x}_k) \tag{3.179}$$

$$\sum_n \sum_k r_{kn} \left(\sum_s W_{sd} \phi_s(\mathbf{x}_k) \right) \phi_m(\mathbf{x}_k) = \sum_n \sum_k r_{kn} t_n^d \phi_m(\mathbf{x}_k) \tag{3.180}$$

$$\sum_n \sum_k \sum_s r_{kn} \Phi_{ks} W_{sd} \Phi_{km} = \sum_n \sum_k r_{kn} t_n^d \Phi_{km} \tag{3.181}$$

$$\sum_k \sum_s \left(\sum_n r_{kn} \right) \Phi_{ks} W_{sd} \Phi_{km} = \sum_n \sum_k r_{kn} t_n^d \Phi_{km} \tag{3.182}$$

$$\sum_k \sum_s G_{kk} \Phi_{ks} W_{sd} \Phi_{km} = \sum_n \sum_k r_{kn} t_n^d \Phi_{km} \tag{3.183}$$

$$\sum_k \sum_s (\Phi_{km})^T G_{kk} \Phi_{ks} W_{sd} = \sum_n \sum_k (\Phi_{km})^T r_{kn} t_n^d \tag{3.184}$$

$$\left(\Phi^T G \Phi W \right)_{md} = \left(\Phi^T R T \right)_{md} \tag{3.185}$$

where we have defined $T_{nd} = t_n^d$ and $R_{kn} = r_{kn}$.

By the same procedure, we find that differentiation with respect to β leads to

$$\frac{1}{\beta} = \frac{1}{ND} \sum_n^N \sum_k^K r_{kn} |\mathbf{y}_k - \mathbf{t}_n|^2 \tag{3.186}$$

so that together the maximization step amounts to solving

$$\begin{cases} \Phi^T G_{\text{old}} \Phi W_{\text{new}} = \Phi^T R_{\text{old}} T \\ \dfrac{1}{\beta_{\text{new}}} = \dfrac{1}{ND} \sum\limits_n^N \sum\limits_k^K R_{kn}^{(\text{old})} |\mathbf{y}_k - \mathbf{t}_n|^2 \end{cases} \tag{3.187}$$

The last piece we need for the training algorithm is a method to initialize the parameters W and β. As before with the SOM, there are a few different options. In particular we might

- Randomly initialize the weight matrix W

- Use PCA to initialize the weights to a linear model. To do this, one computes the data covariance matrix U keeping only the first two columns. We then initialize W so that

$$W\Phi^T \approx UX^T \tag{3.188}$$

and then set β^{-1} to the third principal component variance.

In summary, the GTM training procedure is as follows

1. Generate a grid of latent points $\{\mathbf{x}_k\}_{k=1}^K$.
2. Generate basis function centers $\{\mu_m\}_{m=1}^M$.
3. Select an basis function width σ.
4. Compute the matrix of activations Φ where

$$\Phi_{mk} = \Phi_m(\mathbf{x}_k) \tag{3.189}$$

5. Initialize the weight matrix W randomly or with PCA
6. Initialize β randomly or with PCA
7. (Optional) select a value for α to enable weight regularization, i.e. $(\Phi^T G\Phi + \frac{\alpha}{\beta}I)W = \Phi^T RT$. This corresponds to specifying a prior distribution on the weights W such that $p(W) \propto \exp(-\alpha||W||^2/2)$.
8. Compute the difference matrix $\Delta := |T - \Phi W|^2$
9. Repeat the following until convergence:
   (a) (Expectation step) Compute the responsibility matrix R using Δ and β.
   (b) Compute G from R

(c) (Maximization step) Compute the update to the weight matrix W with $W_{\text{new}} = \left(\Phi^T G \Phi\right)^{-1} \Phi^T R T$

(d) Compute the updated difference matrix Δ

(e) Compute the updated β

Excellent! We now have a robust procedure to fit the GTM model. With a fitted GTM in hand, how do we utilize the results? The main idea is that the fitted GTM provides us all of the relevant probability distributions to explain how we obtained our current data $\mathbf{t}_n$ from the latent vectors $\mathbf{x}_k$. Using Baye's rule, we can invert this relationship using our responsibility matrix R to understand the contributions of each latent vector $\mathbf{x}_k$ to each data. Since this would lead to a different R matrix for each datapoint, one often computes a handful of summary statistics using R rather then using the entire result. For example, we find the mean of the distribution to be

$$\begin{aligned}
\langle \mathbf{x} | \mathbf{t}_n; W, \beta \rangle &= \int d\mathbf{x} p(\mathbf{x}|\mathbf{t}_n; W, \beta)\mathbf{x} \\
&= \int d\mathbf{x} \frac{p(\mathbf{t}_n|\mathbf{x}; W, \beta)p(\mathbf{x})}{\int d\xi p(\mathbf{t}_n|\xi; W, \beta)p(\xi)}\mathbf{x} \\
&= \int \sum_k \frac{p(\mathbf{t}_n|\mathbf{x}, W, \beta)\mathbf{x}\delta(\mathbf{x} - \mathbf{x}_k)}{\sum_{k'} p(\mathbf{t}_n|\mathbf{x}_{k'}; W, \beta)} d\mathbf{x} \\
&= \sum_k \frac{p(\mathbf{t}_n|\mathbf{x}_k; W, \beta)\mathbf{x}_k}{\sum_{k'} p(\mathbf{t}_n|\mathbf{x}_{k'}; W, \beta)} \\
&= \sum_k^K R_{kn}\mathbf{x}_k
\end{aligned} \tag{3.190}$$

However, in cases where the distribution is multi-modal, the mean can be misleading. It can therefore also be useful to compute the mode of the distribution which is given by

$$\mathbf{x}_{\text{mode}} = \mathbf{x}_{k_{\max}} \tag{3.191}$$

where $k_{\max} = \arg\max_k(R_{kn})$.

To provide an easy to use open-source implementation of the GTM algorithm, I have created a publicly accessible github repo with a Julia implementation that comports with the `MLJ.jl` machine learning framework. The repo can be found here and makes it easy to incorporate GTMs into complicated machine learning pipelines.

UPDATE REQUIRED: Add demo use for GTM like we did for the SOM.

3.4 Uncertainty Quantification via Conformal Prediction

Having established a variety of machine learning methods which we will use throughout the rest of this dissertation, a natural next question is: How can we evaluate our confidence in the predictions of machine learning models? For some methods like Gaussian Process Regression, our models naturally output distributions which we can evaluate to provide predictions via the mean, and uncertainty estimates via the standard deviation or some similar statistic. For models like Neural Networks and Decision Tress which are not inherently probabilistic, there is no obvious way to extract uncertainty estimates purely from the model's predictions. One standard approach is to attempt to evaluate our confidence in a particular model by comparing the predictions of many copies of the same model trained on complementary cross validation folds of the original training set. By examining the prediction variance due to variation in the supplied training data, we can establish some expectation for uncertainty in our model's output.

Clearly this approach is far from perfect. Therefore, in order to prevent ourselves from biasing towards using only those methods like Gaussian Process Regression which output distributions by default, we would like to develop a robust procedure for augmenting *any* regression model with the capability to simultaneously estimate target values and *confidence intervals* by taking advantage of our (often) large datasets to establish sufficient statistics. Further, we do not wish to make any assumptions about the particular target distributions; without sufficient reason to expect a variable to be normally distributed why should we make

this *strong* assumption? This is the approach taken by *Conformal Prediction* which we will develop in this section and utilize throughout the rest of this dissertation.

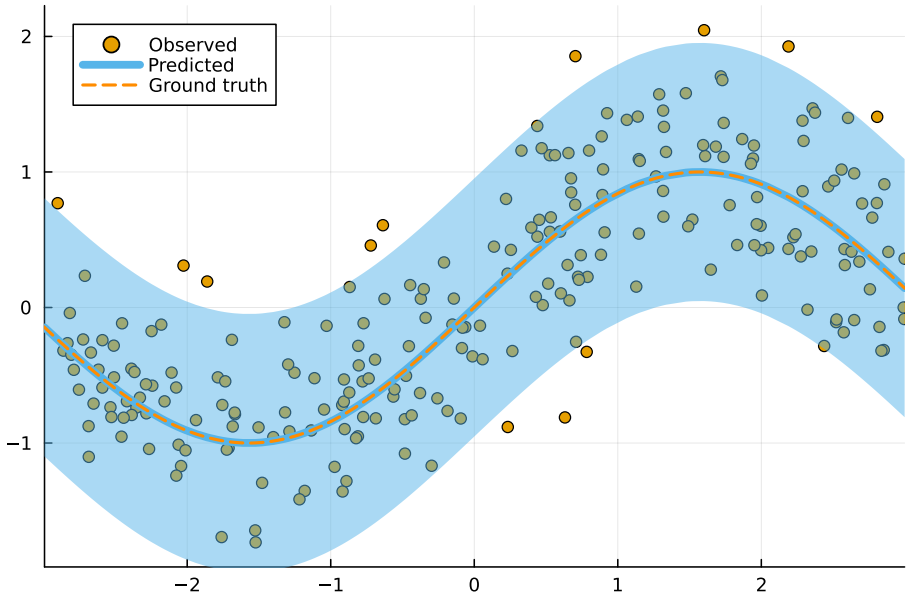


Figure 3.7: An example of conformal prediction for a model estimating a noisy one-dimensional target. The blue shaded region illustrates the learned confidence interval. Image taken from (Altmeyer, 2023)

As discussed in the previous section, the typical machine learning procedure involves partitioning our dataset into independent training and testing batches (or k-many cross-validation folds) so that we can evaluate a trained model's performance and ensure we have not run afoul of over/under-fitting. In conformal prediction, we introduce an additional split so that our original dataset $\mathcal{D}$ can be decomposed into $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{cal}} \cup \mathcal{D}_{\text{test}}$. We then start with *any* heuristic notion of model uncertainty, and make the estimated confidence interval rigorous by adjusting the output to *guarantee* the desired coverage is satisfied on the calibration set $\mathcal{D}_{\text{cal}}$.

A high level outline of the procedure is as follows

1. Define a score function $s(x, y) \in \mathbb{R}$ using some heuristic notion of uncertainty so that large s means *worse* agreement between $\hat{f}(x)$ and the target y.

2. Compute $\hat{q}$ as the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile of the scores $s_i = s(x_i, y_i)$ generated from the calibration set $\mathcal{D}_{\text{cal}}$.

3. Use $\hat{q}$ to form valid prediction intervals.

The question then becomes: How can we obtain accurate uncertainty estimates if the score function s need not be accurate? So long as the scores s_i accurately rank estimates from best to worst model error, then all we need to do is *calibrate* the scores to achieve a confidence interval with the desired coverage, i.e. so that the interval is always large enough to guarantee the coverage we require. It may be that our predicted interval is *too large* for a poor scoring function. As an example, let's demonstrate how we can form correct estimates for the specific problem of quantile regression.

3.4.1 Conformalizing Quantile Regression

In quantile regression the goal is to predict the γ^{th} quantile of the targets y given the input features x. Let us denote $t_\gamma(x)$ as the true value for the γ^{th} quantile and $\hat{t}_\gamma(x)$ as our predicted value. In this framework, we could then expect the interval $[t_{0.05}, t_{0.95}]$ to constitute a 90% coverage confidence interval. However, our fitted values $\hat{t}_\gamma$ will certainly be imperfect so that the actual interval obtained will not conform to the desired 90%. Suppose we have a holdout dataset $\mathcal{D}_{\text{cal}}$ as previously mentioned, and for notational consistency, let α be the coverage so that we may write our predicted interval as $[\hat{t}_{\alpha/2}, \hat{t}_{1-\alpha/2}]$. A reasonable score function (i.e. uncertainty heuristic) would be

$$s(x, y) = \max \left\{ \hat{t}_{\alpha/2}(x) - y, y - \hat{t}_{1-\alpha/2}(x) \right\} \tag{3.192}$$

so that the score is the distance to the nearest of our predicted quantiles. From this score, we may then form

$$\hat{q} = \text{Quantile}\left(\{ s(x_i, y_i) | (x_i, y_i) \in \mathcal{D}_{\text{cal}} \}, \frac{\lceil (n+1)(1-\alpha) \rceil}{n} \right), \tag{3.193}$$

that is, $\hat{q}$ is the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$-th quantile of the scores from our calibration set. Using these corrections, we then output the *conformalized* interval

$$C(x) = \left[\hat{t}_{\alpha/2}(x) - \hat{q}, \hat{t}_{1-\alpha/2}(x) + \hat{q}\right] \tag{3.194}$$

which we have guaranteed achieves the desired coverage on the holdout set. If $\mathcal{D}_{\text{cal}}$ is sufficiently representative, then we will have achieved a rigorous interval for a coverage of α. The effect is that $\hat{q}$ adjusts our previously estimated interval so that a positive $\hat{q}$ widens the interval and a negative $\hat{q}$ shrinks it in order to obtain the appropriate coverage.

How do we implement this in practice? Any model that we fit by minimizing the mean squared error (e.g. a Neural Network) can be agumented to instead fit the so-called *pinball loss*

$$L_\gamma(\hat{t}_\gamma, y) = \begin{cases} (y - \hat{t}_\gamma)\gamma; & \hat{t}_\gamma < y \\ (\hat{t}_\gamma - y)(1-\gamma); & \hat{t}_\gamma \geq y \end{cases} \tag{3.195}$$

For $\gamma = 0.5$, this reduces to the standard ℓ_1 loss $\ell_1(\hat{t}, y) = |\hat{t} - y|/2$ which encourages models to learn the median (i.e. the 0.5 quantile).

NOTE: Add plot of pinball loss for reference

3.4.2 Conformalizing Scalar Uncertainty Estimates

Another common method for producing uncertainty estimates it to simultaneously predict the mean and standard deviation of the target distribution by assuming a Gaussian shape. More generally, we may seek to estimate a model uncertainty directly by first training a model, $\hat{f}(x)$, to predict the target y, and then train a second model $\hat{r}(x)$ to predict the residual error, that is

$$\hat{r} = |y - \hat{f}(x)|. \tag{3.196}$$

In either case, one could then form a confidence interval as

$$\left[\hat{f}(x) - \hat{r}(x), \hat{f}(x) + \hat{r}(x)\right]. \tag{3.197}$$

Other reasonable approaches to extract a heuristic uncertainty from a trained model might include

- Compute the output variance of an ensemble model $\hat{f}$
- Compute the prediction variance obtained when randomly dropping out nodes from a neural network.
- Estimate the variance of $\hat{f}$ resulting from small perturbations of the input x.

For all of these cases, we may call the uncertainty estimate $u(x)$ and form the score function

$$s(x, y) = \frac{|y - \hat{f}(x)|}{u(x)} \tag{3.198}$$

so that the score function s measures how well our uncertainty estimate $u(x)$ matches the actual error $|y - \hat{f}(x)$. If the error is large our predicted uncertainty is small, then s will be large. Consequently, the score function tells us the correction we must apply to our estimate $u(x)$ in order to predict the actual model error, that is,

$$|y - \hat{f}(x)| = s(x, y)u(x). \tag{3.199}$$

If we now take $\hat{q}$ to be the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$-th quantile as before, we are guaranteed that

$$p(s(x_{\text{cal}}, y_{\text{call}}) \leq \hat{q}) \geq 1 - \alpha \tag{3.200}$$

An easy-to-use implementation of these methods is provided by the open-source Julia package `ConformalPrediction.jl` which is designed to integrate into the larger `MLJ.jl` machine learning framework. We will utilize these techniques throughout the remainder of this dissertation in order to establish uncertainty estimates for *all* of our regression models.

3.5 Data Assimilation

The proper application of scientific models to make real-world predictions requires that we commit ourselves to a full accounting of all possible sources of uncertainty when reporting results. Further, the explosion of *big data* across scientific fields provieds a plethora observational data that our models are typically unequipped to incorporate when making predictions. The field of *Data Assimilation* addresses this problem by providing a family of techniques engineered to combine model output together with observational data whilst enabling a complete accounting the sources of uncertainty. For chaotic systems in particular, data assimilation enables integration on long time scales that would be impossible via models alone. In this overview, we will follow the examples from (Ahmed et al., 2020).

Data assimilation can be cleanly developed in the framework of discrete dynamical systems. Since, at the end of the day, all of our scientific models must be discretized to be evaluated numerically, this is a reasonable course of action. Our goal is to find the best prediction for the system state vector u that combines our model predictions, also known as *forecasts*, with observational data. Model predictions are described via the discrete update equation:

$$u_{k+1} = \mathcal{M}(u_k; \theta) \tag{3.201}$$

For ODE systems, $\mathcal{M}$ represents the time integration scheme for a models like

$$\frac{du}{dt} = f(u, t; \theta), \tag{3.202}$$

in other words, a particular choice of ODE integration scheme (Runge Kutta for example) used to evolve the state vector$^{\smile}$ from time u_k to u_{k+1}.

To measure the performance of our assimilation scheme, we denote the *true* value of the state vector as $u^{(t)}$. The output of our model is denoted $u^{(b)}$ (b superscript for *background*). The discrepancy between the true value and our forecast is denoted $\xi^{(b)} = u^{(t)} - u^{(b)}$ characterizing the extent to which our model prediction is imperfect. The observations of our system

are denoted by $w_k = w(t_k)$. These observations do not necessarily need to be components of the state vector u, but rather, are related to it via the *observation function*, $h : u_k \mapsto w_k$. For example, one may attempt to predict surface sea temperatures using data assimilation with data from satellite observations. The function h would then be the Stefan-Boltzmann relating the measured spectra to temperature. However, real world data is *also* imperfect, which we must take into account let we over constrain our model with poor quality data. We write

$$w_k = h(u_k) + \xi_k^{(m)} \tag{3.203}$$

where $\xi_k^{(m)}$ denotes this measurement error.

Given our model predictions $u_k^{(b)}$ and observations w_k, we seek to obtain the *optimal* or best-possible prediction called the **analysis**, $u^{(a)}$. Despite our care, this analysis will still be imperfect, so we further define the analysis error as

$$\xi^{(a)} = u^{(t)} - u^{(a)} \tag{3.204}$$

In summary, we have defined the following relevant quantities (at time t_k):

$$u_k^{(t)} \in \mathbb{R}^n \quad \text{the true state vector} \tag{3.205}$$

$$u_k^{(b)} \in \mathbb{R}^n \quad \text{the model forecast} \tag{3.206}$$

$$u_k^{(a)} \in \mathbb{R}^n \quad \text{the analysis} \tag{3.207}$$

$$w_k \in \mathbb{R}^m \quad \text{the observation vector} \tag{3.208}$$

$$\xi^{(b)} \in \mathbb{R}^n \quad \text{the model forecast error} \tag{3.209}$$

$$\xi^{(m)} \in \mathbb{R}^m \quad \text{the observation noise vector} \tag{3.210}$$

$$\xi^{(a)} \in \mathbb{R}^n \quad \text{the analysis error} \tag{3.211}$$

$$\xi^{(p)} \in \mathbb{R}^n \quad \text{the process noise if we used our model on the true state} \tag{3.212}$$

$$\mathcal{M} : \mathbb{R}^n \to \mathbb{R}^n \quad \text{the discrete model evolution function} \tag{3.213}$$

$$f : \mathbb{R}^n \to \mathbb{R}^n \quad \text{differential equation model (the right hand side)} \tag{3.214}$$

$$h : \mathbb{R}^n \to \mathbb{R}^m \quad \text{observation function} \tag{3.215}$$

To move forward, we now establish the following (prior) assumptions about the distribution of errors in each component in order to make it possible to derive a closed form for the final analysis. We require

$$\mathbb{E}[\xi_k^{(b)}] = 0 \qquad \mathbb{E}[\xi_k^{(b)}(\xi_j^{(b)})^T] = 0 \text{ for } k \neq j \tag{3.216}$$

$$\mathbb{E}[\xi_k^{(m)}] = 0 \qquad \mathbb{E}[\xi_k^{(m)}(\xi_j^{(m)})^T] = 0 \text{ for } k \neq j \tag{3.217}$$

$$\mathbb{E}[\xi_k^{(b)}(u_0)^T] = 0 \qquad \mathbb{E}[\xi_k^{(m)}(u_0)^T] = 0 \tag{3.218}$$

$$\mathbb{E}[\xi_k^{(b)}\xi_j^{(m)}] = 0 \tag{3.219}$$

or in words, the errors in our model and observation are unbiased (e.g. mean zero) and uncorrelated.

We also define the error covariance matrices

$$Q_k := \mathbb{E}[\xi_k^{(p)}(\xi_k^{(p)})^T] \tag{3.220}$$

$$R_k := \mathbb{E}[\xi_k^{(m)}(\xi_k^{(m)})^T] \tag{3.221}$$

$$B_k := \mathbb{E}[\xi_k^{(b)}(\xi_k^{(b)})^T] \tag{3.222}$$

which we will use in our consideration of the final error of our analysis.

3.5.1 Kalman Filter

Given some model for the error covariance matrices Q_k and R_k, we would like a method that propagates *both* our model *and* the errors forward. This way we may guarantee that the accuracy of our analysis doesn't come at the cost of higher uncertainty.

The original implementation of the Kalman filter was for strictly linear systems, in particular as a filter for signal processing. We will first develop the analysis for this simplified case adn then will generalize to the *Extended Kalman Filter* (EKF) that can handle fully nonlinear situations.

In the linear case, our system may be written as

$$u_{k+1}^{(t)} = M_k u_k^{(t)} + \xi_{k+1}^{(p)} \tag{3.223}$$

$$w_k = H_k u_k^{(t)} + \xi_k^{(m)} \tag{3.224}$$

where M_k and H_k are now matrices defining the linear problem.

The goal of the Kalman filter is to derive the analysis $u^{(a)}$ which minimizes the trace of the analysis error covariance matrix (i.e. sum of squared errors):

$$\mathrm{Tr}\,(P_k) := \mathbb{E}[(u_k^{(t)} - u_k^{(a)})^T(u_k^{(t)} - u_k^{(a)})] \tag{3.225}$$

Finding the analysis consists of two steps: the forecast step and the assimilation step.

Beginning with the forecasting step, assume we have the analysis at time t_k denoted $u_k^{(a)}$. Then the forecast for time t_{k+1} is

$$u_{k+1}^{(b)} = M_k u_k^{(a)}, \tag{3.226}$$

or in words, we obtain the next prediction by evolving the previous analysis forward. The background error is therefore

$$\begin{aligned}
\xi_{k+1}^{(b)} &= u_{k+1}^{(t)} - u_{k+1}^{(b)} && (3.227)\\
&= M_k u_k^{(t)} + \xi_{k+1}^{(p)} - M_k u_k^{(a)} && (3.228)\\
&= M_k \left(u_k^{(t)} - u_k^{(a)} \right) + \xi_{k+1}^{(p)} && (3.229)\\
&= M_k \xi_k^{(a)} + \xi_{k+1}^{(p)} && (3.230)
\end{aligned}$$

We may now evaluate the covariance matrix of our background estimate as:

$$\begin{aligned}
B_{k+1} &= \mathbb{E}[\xi_{k+1}^{(b)} (\xi_{k+1}^{(b)})^T] && (3.231)\\
&= \mathbb{E}\left[\left(M_k \xi_k^{(a)} + \xi_{k+1}^{p} \right) \left(M_k \xi_k^{(a)} + \xi_{k+1}^{p} \right)^T \right] && (3.232)\\
& && (3.233)
\end{aligned}$$

If we presume that $\mathbb{E}[\xi_k^{(b)} (\xi_{k+1}^{(p)})^T] = 0$, then the cross terms vanish and we are left with

$$\boxed{B_{k+1} = M_k P_k M_k^T + Q_{k+1}} \tag{3.234}$$

Thus we now have the background (i.e forecast) estimate of the state at t_{k+1} and its covariance matrix. Given a measurement w_{k+1} at the same time with covariance matrix R_{k+1}, we may now perform the assimilation step where we fuse the two sources of information to obtain $u_{k+1}^{(a)}$ and P_{k+1}.

Let's suppose that the analysis has the form

$$u_{k+1}^{(a)} = \nu + K_{k+1} w_{k+1} \tag{3.235}$$

for some vector $\nu \in \mathbb{R}^n$ and matrix $K_{k+1} \in \mathbb{R}^{m \times n}$. In a perfect world, we would have $\mathbb{E}[u_k^{(t)} - u_k^{(a)}] = 0$. Therefore,

$$0 = \mathbb{E}[u_k^{(t)} - u_k^{(a)}] \tag{3.236}$$

$$= \mathbb{E}[(u_k^{(b)} + \xi_k^{(b)}) - (\nu + K_k w_k)] \tag{3.237}$$

$$= \mathbb{E}[(u_k^{(b)} + \xi_k^{(b)}) - (\nu + K_k H_k u_k^{(t)} + K_k \xi_k^{(m)})] \tag{3.238}$$

$$= \mathbb{E}[u_k^{(b)}] + \mathbb{E}[\xi_k^{(b)}] - \mathbb{E}[\nu] - K_k H_k \mathbb{E}[u_k^{(t)}] - K_k \mathbb{E}[\xi_k^{(m)}] \tag{3.239}$$

$$= u_k^{(b)} + 0 - \nu - K_k H_k u_k^{(b)} - 0 \tag{3.240}$$

$$= u_k^{(b)} - \nu - K_k H_k u_k^{(b)} \tag{3.241}$$

$$\Rightarrow \nu = u_k^{(b)} - K_k H_k u_k^{(b)} \tag{3.242}$$

which we now substitute to obtain

$$\boxed{u_k^{(a)} = u_k^{(b)} + K_k(w_k - H_k u_k^{(b)})} \tag{3.243}$$

Now that we know the form for the analysis we may derive the optimal matrix K_k by optimization of P_k. We have

$$\xi_k^{(a)} = u_k^{(t)} - u_k^{(a)} \tag{3.244}$$

$$= M_{k-1}u_{k-1}^{(t)} + \xi_k^{(p)} - u_k^{(b)} - K_k\left(w_k - H_k u_k^{(b)}\right) \tag{3.245}$$

$$= M_{k-1}u_{k-1}^{(t)} + \xi_k^{(p)} - M_{k-1}u_{k-1}^{(a)} - K_k\left(H_k u_k^{(t)} + \xi_k^{(m)} - H_k u_k^{(b)}\right) \tag{3.246}$$

$$= M_{k-1}u_{k-1}^{(t)} + \xi_k^{(p)} - M_{k-1}u_{k-1}^{(a)} - K_k H_k u_k^{(t)} - K_k\xi_k^{(m)} + K_k H_k u_k^{(b)} \tag{3.247}$$

$$= M_{k-1}u_{k-1}^{(t)} + \xi_k^{(p)} - M_{k-1}u_{k-1}^{(a)} - K_k H_k u_k^{(t)} - K_k\xi_k^{(m)} + K_k H_k u_k^{(b)} \tag{3.248}$$

$$= \left\{M_{k-1}(\xi_{k-1}^{(a)} + u_{k-1}^{(a)}) + \xi_k^{(p)} - M_{k-1}u_{k-1}^{(a)}\right\} - K_k H_k u_k^{(t)} - K_k\xi_k^{(m)} + K_k H_k u_k^{(b)} \tag{3.249}$$

$$= \left\{M_{k-1}\xi_{k-1}^{(a)} + \xi_k^{(p)}\right\} - K_k H_k u_k^{(t)} + K_k H_k u_k^{(b)} - K_k\xi_k^{(m)} \tag{3.250}$$

$$= M_{k-1}\xi_{k-1}^{(a)} + \xi_k^{(p)} - K_k H_k(M_{k-1}u_{k-1}^{(t)} + \xi_k^{(b)}) + K_k H_k u_k^{(b)} - K_k\xi_k^{(m)} \tag{3.251}$$

$$= M_{k-1}\xi_{k-1}^{(a)} + \xi_k^{(p)} - K_k H_k M_{k-1}(\xi_{k-1}^{(a)} + u_{k-1}^{a}) - K_k H_k\xi_k^{(b)} + K_k H_k u_k^{(b)} - K_k\xi_k^{(m)} \tag{3.252}$$

$$= M_{k-1}\xi_{k-1}^{(a)} + \xi_k^{(p)} - K_k H_k M_{k-1}(\xi_{k-1}^{(a)} + u_{k-1}^{a}) - K_k H_k\xi_k^{(b)} + K_k H_k M_{k-1}u_{k-1}^{(a)} - K_k\xi_k^{(m)} \tag{3.253}$$

$$= M_{k-1}\xi_{k-1}^{(a)} + \xi_k^{(p)} - K_k H_k M_{k-1}\xi_{k-1}^{(a)} - K_k H_k\xi_k^{(b)} - K_k\xi_k^{(m)} \tag{3.254}$$

$$= \big(I - K_k H_k\big)(M_{k-1}\xi_{k-1}^{(a)} - \xi_k^{p}) - K_k\xi_k^{(m)} \tag{3.255}$$

$$\tag{3.256}$$

and therefore the covariance matrix is

$$P_k = \mathbb{E}[\xi_k^{(a)}(\xi_k^{(a)})^T] \tag{3.257}$$

$$= \mathbb{E}\Big[\Big(\big(I - K_k H_k\big)(M_{k-1}\xi_{k-1}^{(a)} - \xi_k^p) - K_k \xi_k^{(m)} \Big) \Big(\big(I - K_k H_k\big)(M_{k-1}\xi_{k-1}^{(a)} - \xi_k^p) - K_k \xi_k^{(m)} \Big)^T \Big] \tag{3.258}$$

$$= \big(I - K_k H_k\big) M_{k-1} \mathbb{E}[\xi_{k-1}^{(a)}(\xi_{k-1}^{(a)})^T] M_{k-1}^T \big(I - K_k H_k\big)^T \tag{3.259}$$

$$+ \big(I - K_k H_k\big) \mathbb{E}[\xi_k^{(p)}(\xi_k^{(p)})^T] \big(I - K_k H_k\big)^T$$

$$- K_k \mathbb{E}[\xi_k^{(m)}(\xi_k^{(m)})^T] K_k^T$$

$$= \big(I - K_k H_k\big) B_k \big(I - K_k H_k\big)^T - K_k R_k K_k^T \tag{3.260}$$

Now all that remains is to derive the specific form for K_k. The Kalman filter is defined as that K_k which which minimizes the sum of squared analysis errors, i.e. the trace of the analysis error covariance matrix. Therefore, the following identities from matrix calculus will be useful:

$$\underset{A}{\nabla} \operatorname{tr}(AB) = B^T \tag{3.261}$$

$$\underset{A}{\nabla} \operatorname{tr}(BA^T) = B \tag{3.262}$$

$$\underset{A}{\nabla} \operatorname{tr}(ABA^T) = AB^T + AB \tag{3.263}$$

$$\tag{3.264}$$

from which we obtain

$$0 = \underset{K_k}{\nabla} \operatorname{tr}(P_k) \tag{3.265}$$

$$= \underset{K_k}{\nabla} \Big\{ B_k - B_k H_k^T K_k^T - K_k H_k B_k + K_k H_k B_k H_k^T B_k^T - K_k R_k K_k^T \Big\} \tag{3.266}$$

$$= -B_k H_k^T - (H_k B_k)^T + K_k \left[H_k B_k H_k^T + (H_k B_k H_k^T)^T - R_k + R_k^T \right] \tag{3.267}$$

$$= -2B_k H_k^T + 2K_k \left(H_k B_k H_k^2 - R_k \right) \tag{3.268}$$

$$\Rightarrow K_k = B_k H_k^T \Big[H_k B_k H_k^T - R_k \Big]^{-1} \tag{3.269}$$

we now substitute this result to obtain a simplified form for P_k:

$$P_k = (I - K_k H_k) B_k (I - K_k H_k)^T + K_k R_k K_k^T \tag{3.270}$$

$$= (I - K_k H_k) B_k - (I - K_k H_k) B_k (K_k H_k)^T + K_k R_k K_k^T \tag{3.271}$$

$$= (I - K_k H_k) B_k - \left\{ (I - K_k H_k) B_k (K_k H_k)^T + K_k R_k K_k^T \right\} \tag{3.272}$$

$$= (I - K_k H_k) B_k - \left\{ (I - K_k H_k) B_k H_k^T K_k^T + K_k R_k K_k^T \right\} \tag{3.273}$$

$$= (I - K_k H_k) B_k - \left\{ (I - K_k H_k) B_k H_k^T + K_k R_k \right\} K_k^T \tag{3.274}$$

$$= (I - K_k H_k) B_k - \left\{ B_k H_k^T - K_k \left(H_k B_k H_k^T + R_k \right) \right\} K_k^T \tag{3.275}$$

$$= (I - K_k H_k) B_k - \left\{ B_k H_k^T - B_k H_k^T \right\} K_k^T \tag{3.276}$$

$$= (I - K_k H_k) B_k \tag{3.277}$$

where we have used the fact that covariance matrices are symmetric.

Now that we have all of the pieces, let's summarize the procedure:

1. **Initialization**: We must set the system to some initial condition. This means we must define u_0^a and P_0. We must also come up with a model for the process noise covariance Q_k and measurement error covariance R_k.

2. **Forecast Step**: Starting with the analysis state vector $u_{k-1}^{(a)}$ and covariance P_{k-1}, we apply the model time evolution operator M_{k-1} to obtain the predicted state $u_k^{(b)}$ and covariance B_k at the next time step.

$$u_k^{(b)} = M_{k-1} u_{k-1}^{(a)} \tag{3.278}$$

$$B_k = M_{k-1} P_{k-1} M_{k-1}^T + Q_k \tag{3.279}$$

3. **Assimilation Step**: Combine our observations w_k and their associated uncertainties contained in R_k together with the background prediction $u_k^{(b)}$ and it's error covariance

matrix B_k to obtain the analysis state $u_k^{(a)}$ and error covariance P_k.

$$K_k = B_k H_k^T \Big[H_k B_k H_k^T - R_k \Big]^{-1} \tag{3.280}$$

$$u_k^{(a)} = u_k^{(b)} + K_k (w_k - H_k u_k^{(b)}) \tag{3.281}$$

$$P_k = (I - K_k H_k) B_k \tag{3.282}$$

3.5.2 Extended Kalman Filter

Given the nonlinear nature of many scientific models, it is desirable to extend out notion of the *Kalman Filter* to be able to handle nonlinear models $f(\cdot)$ (and by extension, their update function $\mathcal{M}(\cdot)$), and nonlinear observation functions $h(\cdot)$. This can be accomplished so long as these functions are sufficiently smooth (C^1 to be precise) so as to admit valid Taylor approximations to first order. That is,

$$\mathcal{M}(u_k) \approx \mathcal{M}(u_k^{(a)}) + D_M(u_k^{(a)})\xi_k^{(a)} \qquad h(u_k) \approx h(u_k^{(b)}) + D_h(u_k^{(b)})\xi_k^{(b)} \tag{3.283}$$

$$D_M := \left[\frac{\partial \mathcal{M}_i}{\partial u_j} \right] \qquad D_h := \left[\frac{\partial h_i}{\partial u_j} \right] \tag{3.284}$$

where $\mathcal{M}_i$ and h_i denote the ith component functions of $\mathcal{M}$ and h respectively. For the so-called *Extended Kalman Filter* (EKF), approach. We assume that the analysis errors can be reasonably approximated to evolve linearly such that we can use the above *linearized* approximations to our model update and observation functions to reasonably model our uncertainties.

We therefore replace the (previously) linear functions M_k and H_k with their nonlinear Jacobian derived counterparts to obtain the following procedure

1. **Initialization**: To begin we must choose values for $u_0^{(a)}$ and P_0. We must also provide models for Q_k and R_k.

2. **Forecast Step**:

$$u_k^{(b)} = \mathcal{M}(u_{k-1}^{(a)}) \tag{3.285}$$

$$B_k = D_M(u_{k-1}^{(a)}) P_{k-1} D_M^T(u_{k-1}^{(a)}) + Q_k \tag{3.286}$$

3. **Assimilation Step**:

$$K_k = B_k D_h^T(u_k^{(b)}) \left[D_h(u_k^{(b)}) B_k D_h^T(u_k^{(b)}) + R_k \right]^{-1} \tag{3.287}$$

$$u_k^{(a)} = u_k^{(b)} + K_k(w_k - h(u_k^{(b)})) \tag{3.288}$$

$$P_k = \left(I - K_k D_h(u_k^{(b)}) \right) B_k \tag{3.289}$$

Note that the most challenging piece in this implementation is to stably compute the model update Jacobian D_M, as this is not just the Jacobian of our continuous model $f(\cdot)$. For example, if we are modeling the state of the system via a set of ODEs, then D_M is the Jacobian of the output of a single integrator step with respect to the initial starting state. This can pose significant computational challenges where adaptive solvers are desired as we must be able to back-propagate out derivative information for all evaluations of the function $f(\cdot)$ taken during the step.

3.5.3 Continuous-discrete Extended Kalman Filter

So far in our discussion of data assimilation techniques we have limited ourselves to discretized realizations of continuous models and measurements. This allowed us to derive reasonable update equations for the error covariance matrix and analysis by considering the changes that occur by application of our model evolution operator $\mathcal{M}$. In practice though the case tends to be slightly different: the underlying dynamics are understood to evolve continuously despite our access to finite time measurements at some frequency. For this reason, it would be preferable to utilize our knowledge of the model covariance Q together with the continuous dynamics of the model itself to allow the background covariance matrix to evolve continuously between measurements. Then, only when we have access to new data to we need to perform a discrete update by computing the Kalman gain and generating a new analysis as before. As we shall see, this strategy yields an augmented dynamical system

for which

$$\begin{cases} \dfrac{du}{dt} = f(u,t;\theta) + \xi^{(b)}(t) \\ \dfrac{P}{dt} = J_u(t)P(t) + P(t)J_u^T(t) + Q(t) \end{cases} \tag{3.290}$$

where $J_u = \frac{\partial f}{\partial u}$ is the Jacobian of the system with respect to the state vector u. To derive this new form for the Extended Kalman filter, let us first begin with a linear system for which *both* the underlying dynamics continuously, that is

$$\begin{cases} \dot{u} = Mu + \xi^{(b)} \\ w_k = Hu_k + \xi_k^{(m)} \end{cases} \tag{3.291}$$

where $\xi^{(b)}$ now represents a Weiner process (e.g. Gaussian white noise process) of covariance $Q(t)$. Presuming the model covariance is still uncorrelated with the measurement means that we now have:

$$\mathbb{E}[\xi^{(b)}(t)(\xi^{(b)}(\tau))^T] = Q(t)\delta(t-\tau) \tag{3.292}$$

$$\mathbb{E}[\xi^{(m)}(\xi^{(b)}(t))^T] = 0 \tag{3.293}$$

To move forward, we now need to establish the relationship between our previous Q_k and Q so that in the limit as $\Delta t \to 0$, we may obtain the new dynamical equations. If we understand Q_k to be the model error *accumulated* over the integration window, then it follows that

$$\begin{aligned} Q_k &\approx \int\int d\zeta d\eta \mathbb{E}[\xi^{(b)}(\zeta)(\xi^{(b)}(\eta))^T] \\ &= \int\int d\zeta d\eta Q(\zeta)\delta(\zeta-\eta) \\ &= \int_{t_{k-1}}^{t_k} Q(\zeta)d\zeta \\ &\approx Q(t_k)\Delta t \end{aligned} \tag{3.294}$$

This tells us all that we need! For small enough Δt, we may approximate our continuous dynamics by

$$\frac{u_{k+1}-u_k}{\Delta t} = Mu_k + \xi_k^{(b)} \tag{3.295}$$

and therefore

$$u_{k+1} = (I - \Delta t M)u_k + \xi_k^{(b)} \tag{3.296}$$

from which we can identify our linear M_k matrix from the original discrete Kalman filter as $(I + \Delta t M)$ in the discretized version of this new continuous model. Substituting this in to our expression for the forecast step yields

$$B_{k+1} = (I + \Delta t M)\, P_k\, (I + \Delta t M)^T + Q\Delta t. \tag{3.297}$$

Expanding this expression and keeping only terms of $\mathcal{O}(\Delta t)$, we find

$$\begin{aligned} B_{k+1} &= P_k + \Delta t M P_k + P_k M^T \Delta t + M P_k M^T \Delta t^2 + Q\Delta t \\ &\approx P_k + \Delta t M P_k + \Delta t P_k M^T + Q\Delta t \end{aligned} \tag{3.298}$$

As we shrink Δt to zero between observations, there is no longer a distinction between the background covariance B_k and the analysis covariance P_k so that we may write rearrange and take the limit

$$\lim_{\Delta t \to 0} \frac{P_{k+1} - P_k}{\Delta t} = \lim_{\Delta t \to 0} M P_k + P_k M^T + Q \tag{3.299}$$

which evaluates to

$$\boxed{\frac{d}{dt}P = MP(t) + P(t)M^T + Q(t)} \tag{3.300}$$

To *extend* this linear version to the case of nonlinear model and observation functions, we linearize our nonlinear model so that $M \mapsto J_u = \partial f / \partial u$. This then leads to the continuous-discrete Extended Kalman Filter for which the procedure is as follows

1. **Initialization**: To begin we choose initial values for $u_0^{(a)}$ and P_0. We must also provide initial covariance values R_0 and $Q(0)$.

2. **Forecast Step**: Integrate the dynamical system

$$\begin{cases} \dfrac{du}{dt} = f(u, t; \theta) \\ \dfrac{dP}{dt} = J_u P(t) + P(t) J_u^T + Q(t) \end{cases} \tag{3.301}$$

   from t_{k-1} to t_k to obtain u_k and P_k

3. **Assimilation Step**: Compute the Kalman gain and update the state vector and associated covariance matrix according to

$$K_k = P_k D_h^T(u_k) \left[D_h(u_k) P_k D_h^T(u_k) + R_k \right]^{-1} \tag{3.302}$$

$$u_k \mapsto u_k + K_k(w_k - h(u_k)) \tag{3.303}$$

$$P_k \mapsto (I - K_k D_h(u_k)) P_k \tag{3.304}$$

3.5.4 3d-Var

For the *Kalman Filter* and the *EKF*, we derived an optimal way to combine observation with simulation so as to minimize the trace of the analysis error covariance matrix, P_k. An alternative approach is to recast the problem as a pure optimzation problem where rather than finding a filter K_k that will add an innovation to $u_k^{(b)}$ to obtain the analysis $u_k^{(a)}$, we obtain the analysis by directly optimizing a cost function

$$J(u) = \frac{1}{2} \left(w - h(u) \right)^T R^{-1} \left(w - h(u) \right) + \frac{1}{2} \left(u - u^{(b)} \right)^T B^{-1} \frac{1}{2} \left(u - u^{(b)} \right) \tag{3.305}$$

which we can justify as coming from the joint probability distribution (assuming Gaussian errors)

$$\mathcal{P}(u|w) = C \exp \left(-\frac{1}{2} \left(u - u^{(b)} \right)^T B^{-1} \frac{1}{2} \left(u - u^{(b)} \right) \right) \cdot \exp \left(-\frac{1}{2} \left(w - h(u) \right)^T R^{-1} \left(w - h(u) \right) \right) \tag{3.306}$$

with model error covariance B and measurement error covariance R as before. This is clearly a *very strong assumption.*

To optimize $J(u)$, we begin by taking it's gradient.

$$\nabla_u J(u) = -D_h^T R^{-1}(w - h(u)) + B^{-1}(u - u^{(b)}) \tag{3.307}$$

Thus, finding the analysis $u^{(a)}$ ammounts to solving the system

$$a - D_h^T R^{-1}(w - h(u^{(a)})) + B^{-1}(u^{(a)} - u^{(b)}) = 0 \tag{3.308}$$

As for Kalman filtering, let's begin with the assumption that our model and observation function are linear. Suppose that we have $h(u) = Hu$ so that $D_h(u) = H$. It follows that

$$D_h^T R^{-1}(w - Hu^{(a)}) = B^{-1}(u^{(a)} - u^{(b)}) \tag{3.309}$$

$$D_h^T R^{-1} w - D_h^T R^{-1} H u^{(a)} = B^{-1} u^{(a)} - B^{-1} u^{(b)} \tag{3.310}$$

$$\left(D_h^T R^{-1} H + B^{-1}\right) u^{(a)} = D_h^T R^{-1} + B^{-1} u^{(b)} \tag{3.311}$$

$$\left(H^T R^{-1} H + B^{-1}\right) u^{(a)} = H^T R^{-1} + B^{-1} u^{(b)} \tag{3.312}$$

Thus we see that the analysis is given by

$$u^{(a)} = u^{(b)} + BH^T \left(R + HB^T H\right)^{-1} (w - Hu^{(b)}) \tag{3.313}$$

which agrees with what we found for the Linear Kalman Filter.

Moving on to the non-linear case, we can expand h about an initial guess $u^{(c)}$ which we will later choose to be $u^{(b)}$ for convenience.

$$h(u^{(a)}) \approx h(u^{(c)}) + D_h(u^{(c)})\Delta u \tag{3.314}$$

Using this, we have

$$D_h^T(u^{(a)}) R^{-1}(w - h(u^{(a)})) = B^{-1}(u^{(a)} - u^{(b)}) \tag{3.315}$$

$$D_h^T(u^{(a)}) R^{-1}(w - h(u^{(c)}) - D_h(u^{(c)})\Delta u) \approx B^{-1}(u^{(c)} + \Delta u - u^{(b)}) \tag{3.316}$$

$$D_h^T(u^{(c)}) R^{-1}(w - h(u^{(c)}) - D_h(u^{(c)})\Delta u) \approx B^{-1}(u^{(c)} + \Delta u - u^{(b)}) \tag{3.317}$$

which we now solve for the update Δu to obtain the linear system

$$\left(B^{-1} + D_h^T(u^{(c)}) R^{-1} D_h(u^{(c)})\right) \Delta u = B^{-1}(u^{(b)} - u^{(c)}) + D_h^T(u^{(c)}) R^{-1}(w - h(u^{(c)})) \tag{3.318}$$

Thus we have the following prescription

1. To begin, take $u^{(c)} == u^{(b)}$.

2. Solve the system

$$\left(B^{-1} + D_h^T(u^{(c)})R^{-1}D_h(u^{(c)})\right)\Delta u = B^{-1}(u^{(b)} - u^{(c)}) + D_h^T(u^{(c)})R^{-1}(w - h(u^{(c)})) \tag{3.319}$$

   to obtain Δu

3. Update your guess using your favorite optimization algorithm. For example, in steppest descent, choose a learning rate η and set

$$u^{(c)}_{\text{new}} = u^{(c)}_{\text{prev}} - \eta\Delta u \tag{3.320}$$

4. Repeat the procedure until $|u^{(c)}_{\text{new}} - u^{(c)}_{\text{prev}}|$ converges to a desired tolerance.

In both the linear and nonlinear case, it should be noted that we have not added time indices to our state vectors. This is an indication that the 3d-var procedure is performed *at every time where you have observation data.*

3.5.5 4d-Var

The *3d-Var* algorithm attempts to optimize a cost function point-by-point to obtain the ideal analysis at each time where we have observation data. This can become computationally expensive as we require model evaluations *and* an optimization routine for every observation point. An alternative approach is to simultaneously optimize across all observations in order to obtain the ideal *initial condition* which achieves the best model fit. In other words, this amounts to parameter fitting for our model where we think of the initial conditions as the parameters. The difference is two-fold: We extend the usual ℓ_2 function into a quadratic form utilizing the observation error covariance matrix R_k. We also typically only have a small number of observables related to the state vector (potentially non-linearly) and therefore

must transform the state vector u_k into the observation space via h before computing the loss against our observations w_k. This results in a loss function of the form

$$J(u_0) = \frac{1}{2}\left(u_0 - u_0^{(b)}\right)^T B^{-1}\left(u_0 - u_0^{(b)}\right) + \frac{1}{2}\sum_k \left(w_k - h(u_k)\right)^T R_k^{-1}\left(w_k - h(u_k)\right) \quad (3.321)$$

$$= J_b(u_0) + J_m(u_0).$$

Note that the first term is usefull if we already have an initial guess $u_0^{(b)}$ for the inital condition in mind. If we do not have one, we may ommit this term.

As before, we now want to optimize this cost function. To do so, we first observe that

$$u_k = \mathcal{M}^{(k)}(u_0;\theta) \quad (3.322)$$

It is easy to obtain the gradient of J_b so we shall focus on the second term. We find that

$$\nabla_{u_0} J_m = \nabla_{u_0}\Big\{\sum_k \frac{1}{2}\left(w_k - h(u_k)\right)^T R_k^{-1}\left(w_k - h(u_k)\right)\Big\} \quad (3.323)$$

$$= -\sum_k \left[\frac{\partial}{\partial u_0} h\left(\mathcal{M}^{(k-1)}(u_0)\right)\right]^T R_k^{-1}\left(w_k - h(u_k)\right) \quad (3.324)$$

$$= -\sum_k \left[D_h(u_k)D_M(u_{k-1})D_M(u_{k-2})\cdots D_M(u_0)\right]^T R_k^{-1}\left(w_k - h(u_k)\right) \quad (3.325)$$

$$= -\sum_k \left[D_M^T(u_0)D_M^T(u_1)\cdots D_M^T(u_{k-1})D_h^T(u_k)\right] R_k^{-1}\left(w_k - h(u_k)\right) \quad (3.326)$$

With this in hand, the procedure is nearly identical to 3d-var:

1. Integrate your model forward to obtain $\{u_k\}$
2. Evaluate each of the $D_M^T(u_{k-1:0})$ and $D_h(u_k)$.
3. Using these values, compute $\nabla J_m(u)$
4. Set $u_0^{(new)} = u_0^{(prev)} - \eta\nabla J(u_0^{(prev)})$
5. Stop when $|u_0^{(new)} - u_0^{(prev)}|$ converges to your desired tolerance.

You can of course substitute another optimzation scheme after step 3.

There are a few further observations that we can make about this procedure. The first, is that should we feel our model $f(\cdot)$ is not sufficient to fit the entire time series without the periodic corrections of the Kalman filter (perhaps there is some missing physics we haven't captured in our model), then it may be ideal to restrict the optimization to a subset of the full time series of observations. The second observation is that the above formulation has assumed the adjoint D_M^T can be computed easily. We can take advantage of the methods from the *Adjoint Sensativity Analysis* of ODEs to take advantage of modern methods for automatic differentiation of ODE solutions. Finally, it should be noted that the $4d-var$ algorithm yields the optimal starting conditions assuming our model is perfect. At the sacrifice of smoothness, it is often desirable to first obtain the ideal initial condition with $4d-var$ and *then* employ a filtering method like the EKF to obtain the final analysis.

CHAPTER 4

AN AUTONOMOUS ROBOTIC TEAM FOR THE RAPID CHARACTERIZATION OF NOVEL ENVIRONMENTS

In this chapter, we present preliminary results from our autonomous robotic team. First we detail the development of a real time processing and georectification pipeline for the hyperspectral images collected by our drone. This is vital to enable real time application of the robotic team in realistic field settings. Next we discuss results of applying machine learning to directly model concentrations of of chemicals-of-concern form the captured reflectance spectra. Finally, we provide discuss the application of unsupervised methods including the SOM and GTM developed in chapter 3 to enable a physics-based unsupervised learning framework for the identification of novel constituents. Associated code are made freely accessible at `https://github.com/john-waczak/RobotTeam.jl`

4.1 Rapid Processing and Georectification of Hyperspectral Data Cubes

The increased spectral resolution of HSI systems poses unique challenges to their adoption for real time applications primarily stemming from the considerable size of generated data files. Current data collection workflows see researchers first perform the aerial survey (data collection) and then transfer data to ground based computers for post processing. This workflow is well established in the remote sensing community where, as an example, compressed raw imagery from Sentinel-2 are transferred to the ground and then subsequently post processed into their final L1C (top of atmosphere) and L2A (bottom of atmosphere) data products (Main-Knorn et al., 2017; Baillarin et al., 2012). Drone based applications often operate in a similar manner: images or video are collected by a survey and then post-processed and analyzed with software such as Open Drone Map to produce the desired data products (tile mosaics, 3d reconstructions, etc.) (Vacca, 2019). For an HSI platform to function in real time, three key tasks are critical:

1. **FileIO**: Raw imagery need to be quickly read by the on-board processing computer.

2. **Post-processing**: Raw imagery need to be rapidly converted to the chosen data product (typically, Reflectance), and importantly, must be georeferenced so that each image pixel can be located on the ground.

3. **Ground Transfer**: Sufficient wireless communication capabilities must be avaiable to transmit the final data products.

The first can be readily accomplished by means of light-weight, high volume solid state drives incorporated into the imaging system. To address the second, we need both sufficient compute and optimized processing software. Finally, ground transfer of final post-processed data products can be accomplished in a variety of ways. As we rarely need the full hyperspectral datacube immediately, we can generate the desired data products on-board (NDVI for example) and transfer only the relevant information to a ground station.To enable this workflow, our drone is outfitted with two additional components critical for this application. The first is a pair of light-weight processing computers (Intel NUCs). One is attached directly to the imager and manages data acquisition and saving of raw data files. The second NUC is mounted above the payload and serves as the onboard data processing unit. As illustrated in Figure 4.1, we also equip the drone with an upward facing irradiance spectrometer outfitted with a cosine corrector (to integrate over a half-sphere of solid angle). This spectrometer captures the *downwelling* irradiance spectrum which is necessary to enable conversion from radiance units to the desired reflectance spectra. To this end we present a procedure based on the methods described in (Muller et al., 2002; Bäumker and Heimes, 2001; Mostafa and Schwarz, 2000) for the rapid processing and georeferencing of imagery captured by a pushbroom HSI mounted on an autonomous drone. The processing steps are as follows:

1. Raw imagery are continuously captured by the hyperspectral imager (Radiance) and stored in binary ENVI format.

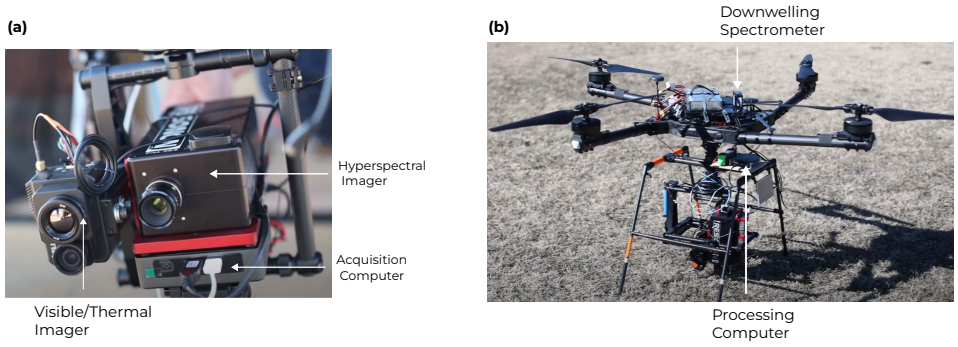


Figure 4.1: Components of the Autonomous Drone HSI platform.

2. The processing computer reads ENVI files into tensors as they become available.

3. Associated flight data (GPS and IMU) are read.

4. Flight data are interpolated to match the sample times for each scan-line.

5. The HSI is georeferenced to obtain coordinates (lat,lon) for each pixel.

6. The georeferenced HSI is then resampled to a regular grid.

7. The downwelling irradiance spectrum associated with the HSI is read.

8. The downwelling spectrum is interpolated to match the wavelength bins of the HSI

9. The HSI is converted o Reflectance under the assumption of a Lambertian surface (perfectly diffuse).

10. Any desired derived λ-metrics (NDVI, etc...) are computed.

11. Data products are selected and the result is transferred to the ground station.

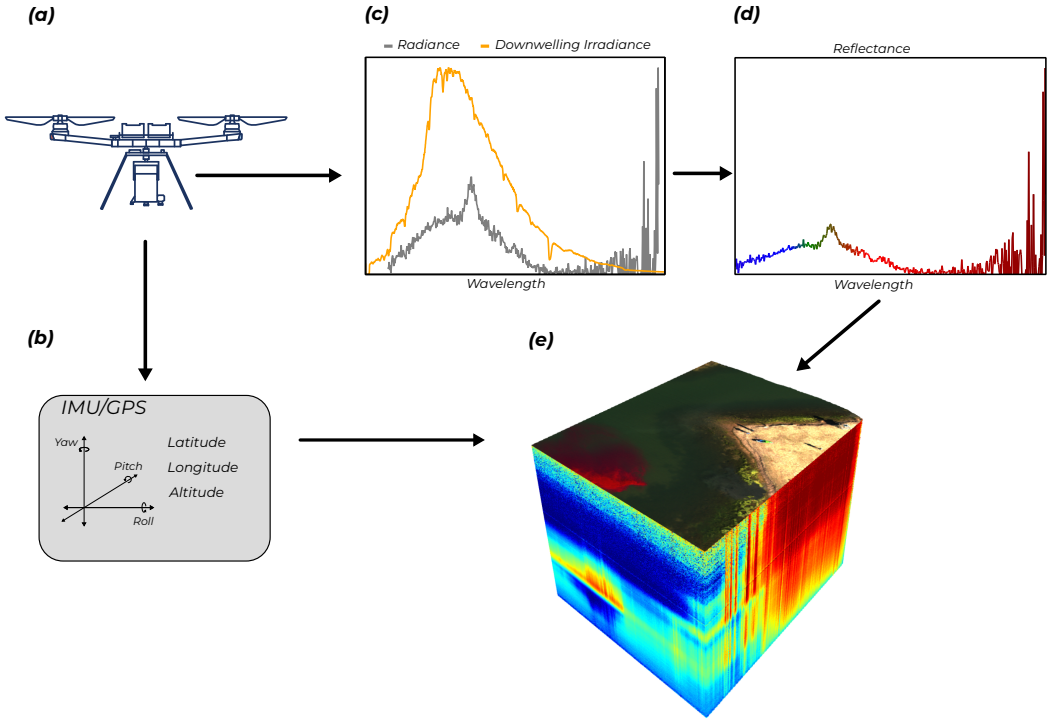


Figure 4.2: Visual representation of the processing pipeline

4.1.1 Georeferencing and Resampling

The hyperspectral imager used on our autonomous drone is different from a typical camera; instead of capturing an image by collecting light onto an array of sensors, our HSI is configured as a pushbroom imager. In this configuration, a single scanline of pixels is captured by the sensor and an image is formed as the drone flies along its route. This poses an interesting challenge for georeferencing as each individual scanline needs to be transformed independently. If the drone's path winds and turns, each subsequent scanline will be stretched and rotated independently based on the relative orientation of the drone to the ground at the time of capture. This sampling geometry as well as the relevant orientation angles are illustrated in Figure 4.3.

The GPS and IMU on the hyperspectral imager capture the location $(\lambda, \Phi, z)^T$ (longitude, latitude, altitude) of our drone as well it's orientation defined by the angles ϕ, θ, and ψ (roll,

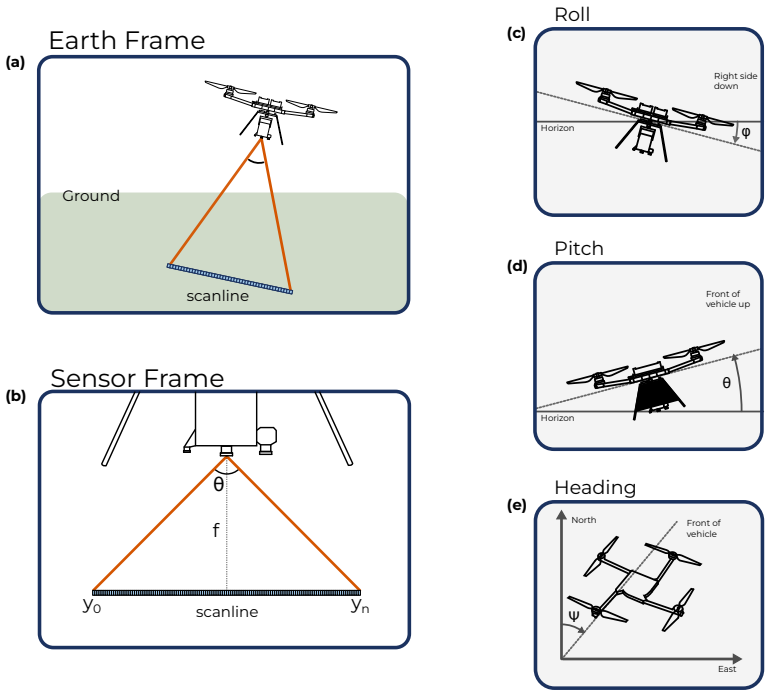


Figure 4.3: Visual representation of scan-line geometry for the drone based hyperspectral imaging platform.

pitch, and heading) at the time of capture of each scanline. To georeference each pixel, we therefore must transform its position from the frame of the sensor (measured in pixels relative to the HSI) first to the frame of the IMU used to measure the orientation, then to the navigation frame (east, north, up), and finally to the ground frame. Let us define the *sensor* frame so that the scanline falls upon the y axis. As each scanline must be transformed independently, we can assign it an x-coordinate of 0 pixels. Lastly, if the viewing angle of the HSI is θ_{view}, then the coordinates of the i$^{\text{th}}$ pixel, $\mathbf{r}_i^{\text{sensor}}$, in the sensor frame are defined as in panel (b) of Figure 4.3, namely

$$\mathbf{r}_i^{\text{sensor}} = (0, y_i, f)^T \tag{4.1}$$

where $y_i \in \left\{-\frac{(N-1)}{2}, ..., \frac{N-1}{2}\right\}$ and $f = \frac{(N-1)/2}{\tan(\theta_{\text{view}}/2)}$ for N-total pixels per scanline. To align the sensor frame with the axes of the IMU, we apply a sequence of rotation matrices defined

using the measured orientation angles (panels (c), (d), and (e) of Figure 4.3), which we call

$$R^{\text{IMU}}_{\text{sensor}}(\phi,\theta,\psi) = \begin{bmatrix} \cos(\psi)\cos(\theta) & \cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi) & \cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) \\ \sin(\psi)\cos(\theta) & \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)\cos(\phi) & \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi) \\ -\sin(\theta) & \cos(\theta)\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{4.2}$$

Next, we apply an orthogonal transformation to transform from the IMU frame to the navigation frame of the drone. This is important as the axes of the IMU and the drone itself are not necessarily identical depending on how the IMU is oriented on the HSI. For us, this amounts to

$$T^{\text{DRONE}}_{\text{IMU}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{4.3}$$

At this stage, we have now transformed the pixel coordinates into the frame of the drone. Next using the similar triangles defined by the sensor and navigation frames (panels (a) and (b) of Figure 4.3), we can rescale the coordinates into units of meters by introducing the scale factor

$$s = \frac{z - z_{\text{ground}}}{f\cos(\theta)}. \tag{4.4}$$

Finally, we translate the coordinates of each pixel using the position of the drone in the chosen coordinate system. As the geometric transformations are scaled to units of meters (by s), we apply a coordinate transformation f to obtain the position of the drone with respect to a local coordinate system such as the Universal Transverse Mercator (UTM). Written all together, this takes the form

$$\mathbf{r}^{\text{UTM}}_i = f^{\text{UTM}}_{\text{geo}} \begin{bmatrix} \lambda \\ \Phi \\ z_{\text{drone}} \end{bmatrix}^{\text{geo}}_{\text{GPS}} + sT^{\text{DRONE}}_{\text{IMU}} R^{\text{IMU}}_{\text{sensor}}(\theta,\phi,\psi) \begin{bmatrix} 0 \\ y_i \\ f \end{bmatrix}^{\text{sensor}} \tag{4.5}$$

For each image, the above transformation can be applied *in parallel* across all scanlines to obtain the ground coordinates $\mathbf{r}_i = (x_i, y_i, z_i)^T$ of each pixel. To aid in further analysis and enable the generation of data products, it is often desirable to resample the georeferenced datacube to a regular grid. To accomplish this rapidly, we can define a bounding box by the extrema of the pixel coordinates. Then, for a desired resolution, say Δx in meters, we truncate each pixels coordinates to the nearest Δx and then average all pixels that have the same position.

4.1.2 Reflectance Conversion

Once we have georeferenced and resampled the hyperspectral datacube we can use the measured downwelling irradiance spectrum to convert the raw data from radiance to reflectance. This allows us to normalize for the incident light so that spectra captured under variable lighting conditions can be compared. First, the downwelling irraidance spectrum is loaded and interpolated to match the wavelengths of the spectral bins for our HSI. The reflectance at pixel (i, j) and wavelength λ_k) is then obtained by

$$R_{ijk} = \frac{\pi L_{ijk}}{E_k} \tag{4.6}$$

where L_{ijk} denotes the original radiance pixel and E_k denotes the downwelling irradiance at wavelength λ_k. This conversion makes the assumption that the surface can be treated as *Lambertian*, that is, that the surface is perfectly diffuse and scatters light according to the cosine emission law which when integrated over a half sphere of solid angle introduces the factor of π.

Figure 4.2 illustrates the reflectance spectra for a variety of constituents in a sample hyperspectral data cube. In panel (a) we show a typical downwelling irradiance spectrum. Panels (c)-(f) show a selection of reflectance spectra for open water, a rhodamine dye plume, algae near the shore, and dead grass. Panel (b) is a visual representation of the hyperspectral

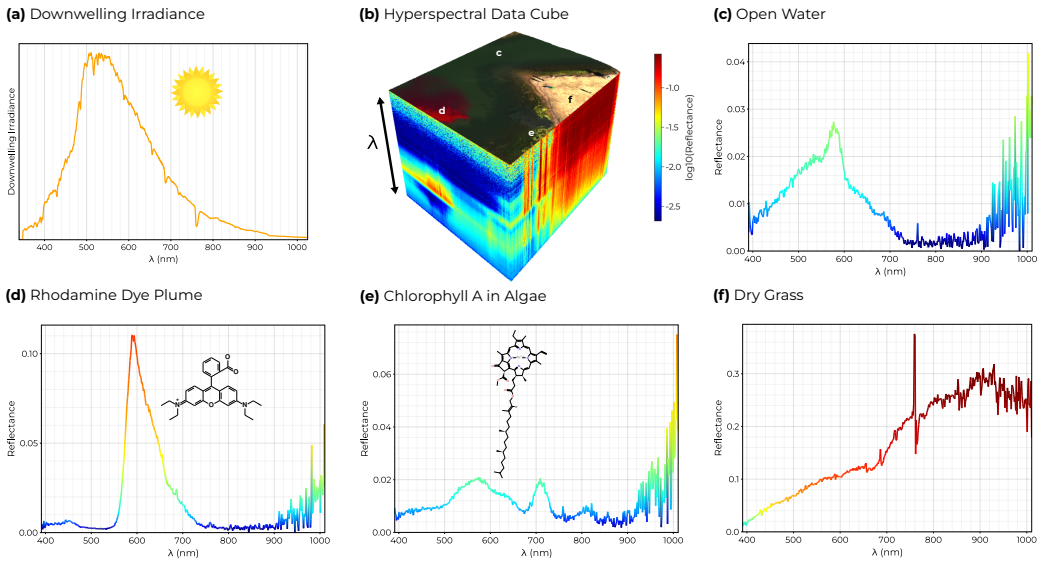


Figure 4.4: Annotated view of a hyperspectral data cube showcasing sampled spectra for a variety of constituents.

data cube with reflectance at each wavelength bin along the z axis and a pseudo color image showing the imaging scene on the top.

4.2 Timing Results and Discussion

The above procedure was implemented in the Julia programming language. Using the package `BenchmarkTools.jl` the following timings were obtained for each step of the pipeline with results indicating that a hyperspectral datacube of size $1000 \times 1600 \times 462$ can be georeferenced, resampled, and converted to reflectance in less than 5 seconds at a final spatial resolution of 25 cm.

Number of scan lines	Execution time (s)
371	0.548
462	0.702
1000	1.478

Table 4.1: Loading and georeferencing time as a function of number of scanlines

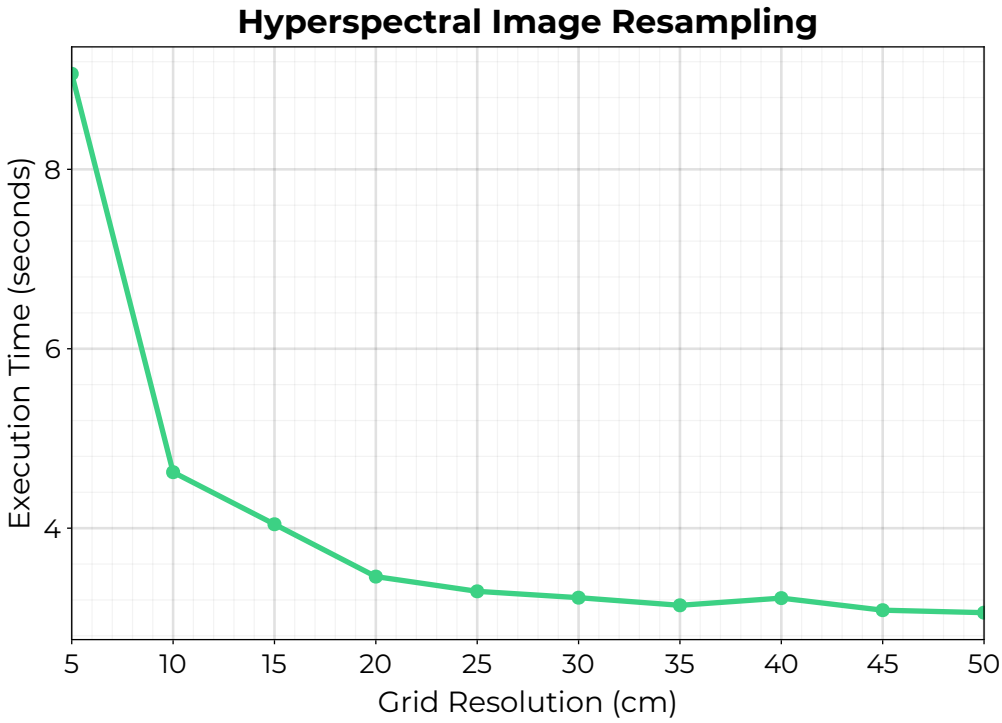


Figure 4.5: Timing results (in seconds) for resampling a 1000 scanline HSI as a function of output grid resolution.

Number of scan lines	Execution time (s)
371	0.161
462	0.197
1000	0.195

Table 4.2: Loading and georeferencing time as a function of number of scanlines

4.3 Supervised Regression with Uncertainty Quantification

The primary modality of our autonomous robotic team is to rapidly characterize the concentrations of chemicals-of-concern in novel aquatic environments. To demonstrate this capability, we performed a sequence of data collections throughout 2020 and 2021 at a ranch in North Texas. Over 3 Tb of hyperspectral imagery were collected across three separate deployments together with 10,00 in-situ measurements from our autonomous boat. To-

gether these data allow us to train machine learning models whereby direct concentration measurements are utilized to train a family of machine learning models to predict a variety of chemicals of concern. In the rest of this section we present preliminary results demonstrating machine learning models trained to estimate Colored Dissolved Organic Matter (CDOM) concentrations using these datasets.

4.4 Results

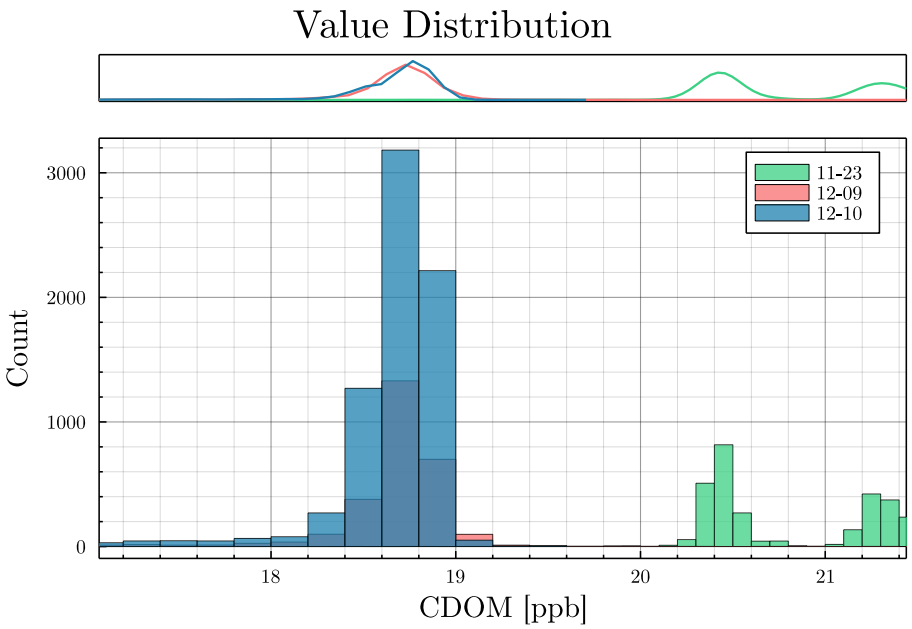


Figure 4.6: Stratified train-test split for CDOM

Figure 4.6 visualizes a sample distribution of in-situ crude oil concentrations collected by the autonomous boat during our deployments. To build models, each of these records is collocated with associated reflectance spectrum from the HSIs forming a large tabular dataset with spectra as features and concentrations as targets. Additionally, we include a number of derived spectral indices like the Normalized Difference Vegetation Index (NDVI) as well as the drone orientation angles and the relevant solar geometry to provide the model important contextual information about the illumination of the water. The solar zenith, azimuth,

and elevation are determined for each pixel using an open source code we implemented in Julia based on a matlab script by Darin Koblick. To train our machine learning models we partition the entire dataset into a train/test split. The bulk of the data is used to train our machine learning models with a representative subsample used as a holdout dataset to evaluate model performance.

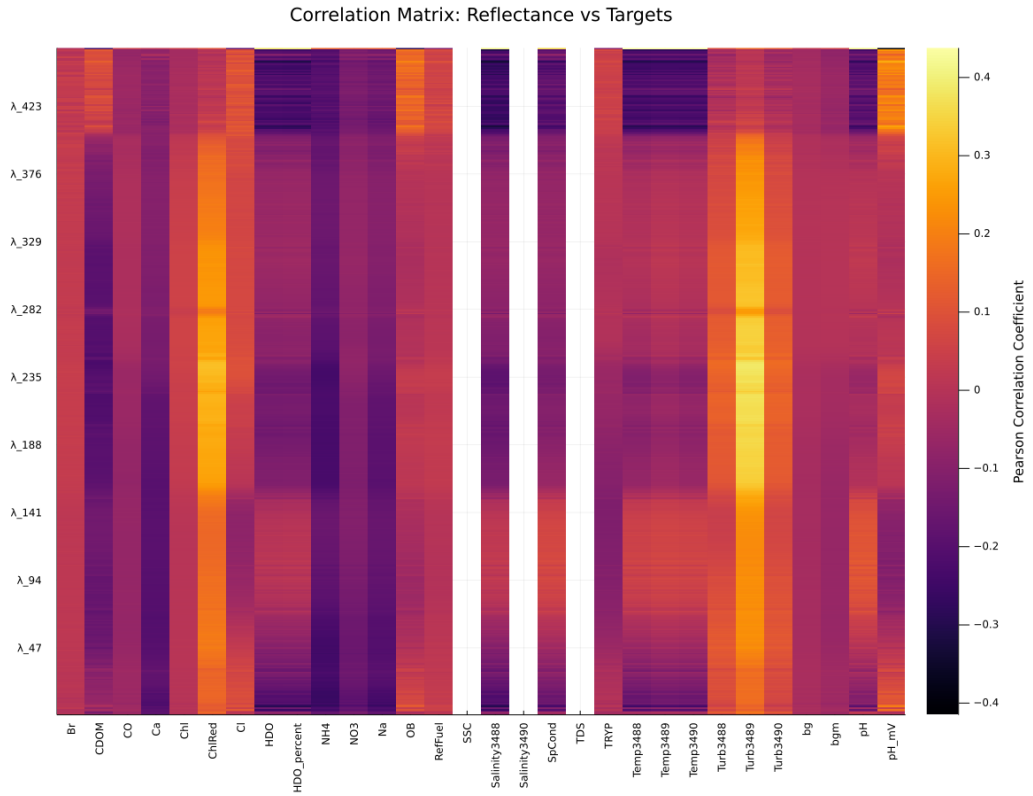


Figure 4.7: Correlation Matrix for Reflectance measurements versus Target Variables.

Figure 4.7 illustrates the correlation matrix between reflectances (as a function of wavelength) and each of the measured targets of the boat. In particular we can see that for CDOM, there is positive correlation between reflectances at long wavelengths and negative correlation at short wavelengths. This makes sense as crude CDOM tends to absorb broadly in the visible portion of the spectrum and reflects strongly in the infrared leaving water redish-brown. Figure 4.7 also demonstrates how different chemicals of concern can be discerned by absorption features in different wavelength bins.

Results So Far

For our machine learning approach, we've employed a an ensembling strategy called *model stacking* (also known as a *super-learner*) (Wolpert, 1992). Rather than arbitrarily choosing a single model such as a neural network or decision tree, we form a *stack* of many models including linear regression, neural networks, decision trees, gaussian process regression, random forest, and boosted trees (XGBoost and EvoTrees). A discriminator model (i.e. the super learner) is then trained on the output of each base model in order to effectively combine the results to generate a robust prediction (Wolpert, 1992). To prevent the super learner from simply siding with the best *average* model, the training dataset is carefully partitioned into a set of N cross validation folds. For each fold, every base model is trained on the $N-1$ fold complements and produces a set of out-of-bag predictions for the chosen fold. These predictions are the stitched together and used to train the super learner. Once the super learner is trained, the original base models are retrained on the entire dataset with the resulting stack of base learners plus discriminator forming the final predictive model.

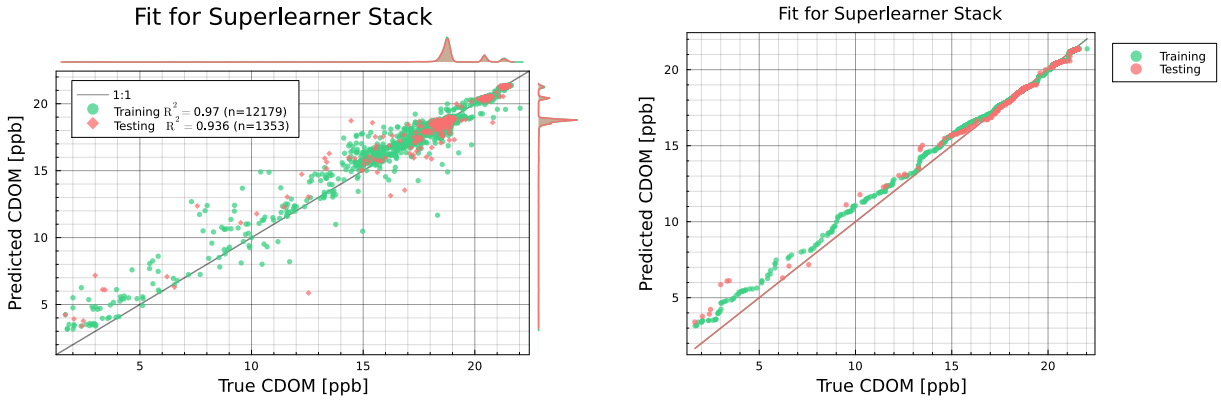


Figure 4.8: (**a**) Scatterplot for the trained CDOM super-learner model. (**b**) Quantile-Quantile plot for the same fit.

Figure 4.8 illustrates the quality of the resulting super-learner fit. The combined dataset is largely tri-modal with three large peaks. Consequently, the model performs best near these values with the worst performance occurring at low concentrations with sparse samples.

A number of the models involved in our super-learner employ a decision-tree based learner. For these base learners we can evaluate the relative importance of each input feature to the predictions of the model. These feature importances are ranked in Figure 4.9. This

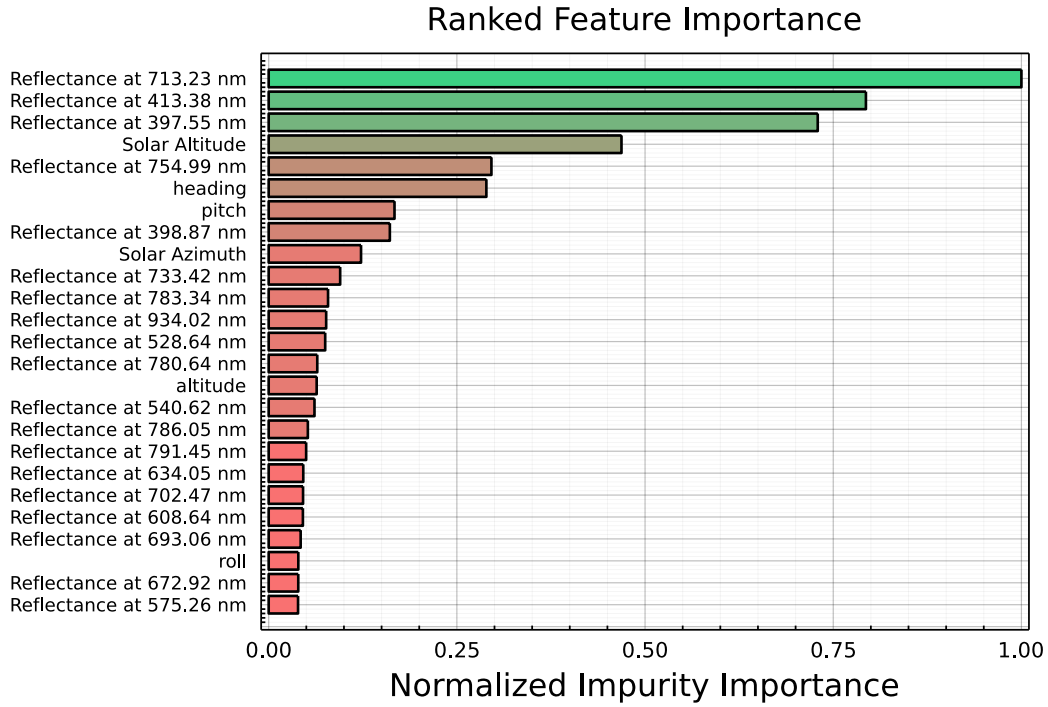


Figure 4.9: Ranked feature importance for CDOM.

feature ranking reveals that wavelengths in key portions of the visible spectrum have the largest impact on model predictions. Similarly, the Solar Altitude and Drone pitch/heading are ranked highly due to the important impact of the solar illumination and viewing geometry on the captured spectra. Using these models, we can rapidly generate wide area maps that estimate concentrations at all HSI pixel location. Sample maps for both CDOM and Crude Oil are shown in Figure 4.10.

Figure 4.10 generates precisely how this Robotic Team can be used to generate actionable environmental insights. Both of these chemicals-of-concern are observed to collect in a region of the pond that is largely isolated with little flow. However, because we were able to acquire samples from this region we are also able to identify accumulation occurring near the shores.

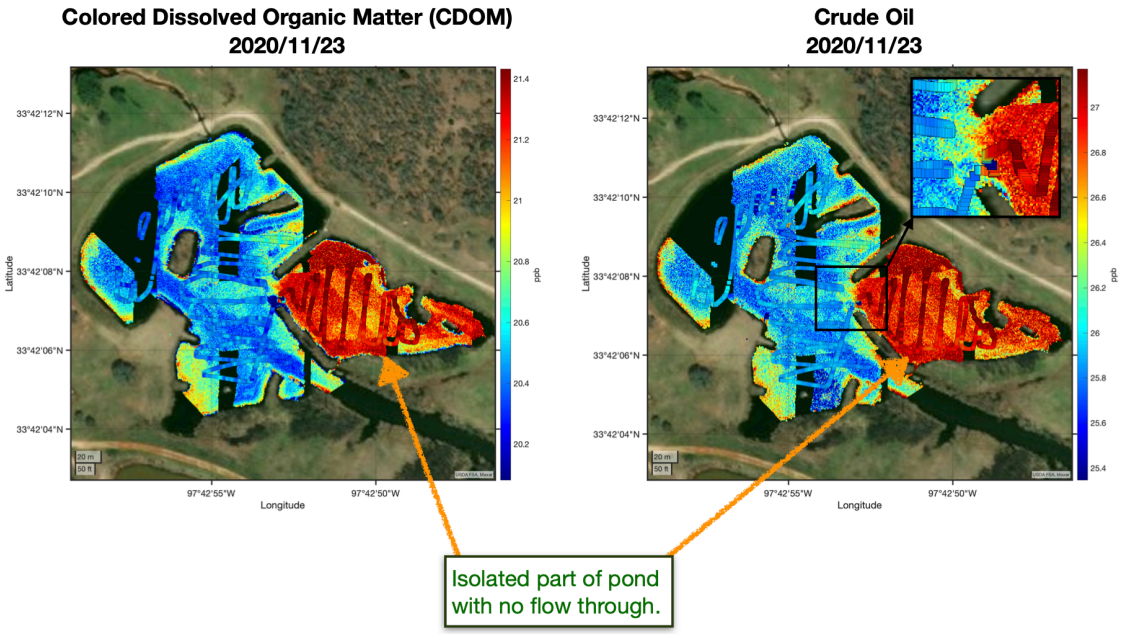


Figure 4.10: Maps of Crude Oil and CDOM generated using the trained super-learner models.

Next Steps

The next steps for this supervised approach are to incorporate conformal prediction into the super-learner model pipeline. By partitioning a third hold out dataset, we can simultaneously estimate the concentrations using our trained super learner *and* obtain reasonable uncertainty bounds for these predictions. With the ability to generate uncertainty estimates together with our model predictions, we can develop *smart* data collection strategies to enable the robotic boat to collect additional samples optimized to decrease model uncertainty and not waste time re-treading portions of the pond where the model is already highly confident.

4.5 Methods for Unsupervised Classification of Hyperspectral Scenes in Novel Environments

A second modality for the robotic team is to utilize unsupervised machine learning strategies to identify key constituents for which we *do not* have reference sensors. Many common approaches like endmember mixing models treat measured reflectance spectra as a linear

combination of representative source spectra whose coefficients indicate a relative abundance within a pixel. By utilizing the Self Organizing Map and Generative Topographic Map codes I've developed, we can identify key endmembers present in the scene for which the autonomous boat is not equipped to directly measure. Further, by examining the learned weights of each class and their distribution throughout the scene, we can attempt to assign individual classes to known chemical signatures.

Unsupervised Classification

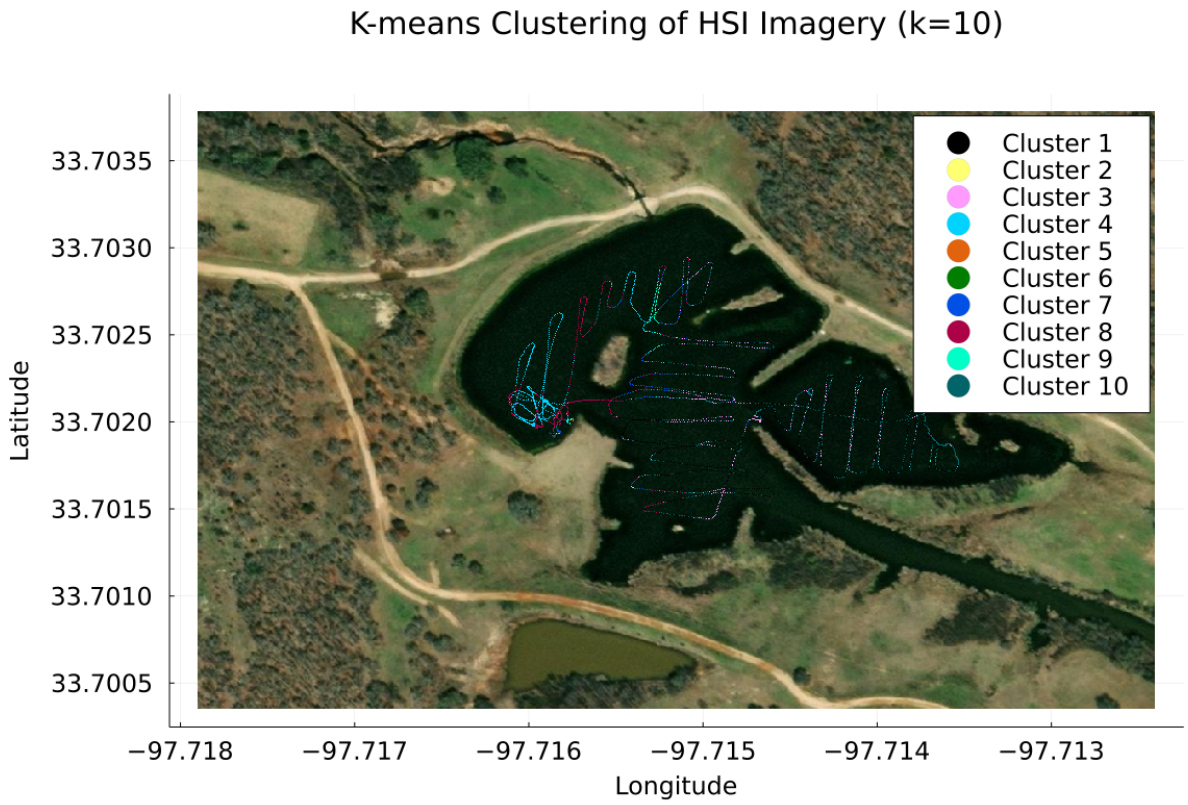


Figure 4.11: Unsupervised clustering of hyper-spectral image data using the K-means algorithm.

CHAPTER 5

TIME SERIES METHODS FOR AIR QUALITY

With the proliferation of low-cost sensors for a wide range of IoT and wearable biometric applications, a systematic approach to both quality control and performing comprehensive time series analysis has significant value. It is highly desirable to be able to answer some basic questions using *just* the time series alone. For example: What is the likely sensor uncertainty given a time series of observations? How frequently should observations be made to adequately resolve typical temporal variability? How representative is a single observation of what one expects to see over a temporal (and spatial) window? Can we construct predictive models for the time series of a single sensor given a sufficient volume of sample data? In this chapter we seek to address these questions with direct application to the data collected by our low cost air quality sensor network. First, we will demonstrate how the *temporal variogram* provides a way to assess the intrinsic sensor uncertainty from a time series, and in the process, develop an open-source Julia implementation of a variety of variogram methods that can be used for any generic time series. Next, we present two techniques for physics-based time series modeling: the Hankel Alternative View Of Koopman (HAVOK) method, and the Hamiltonian Neural Network. It should be noted that despite the rapid pace of development in the fields of data-driven and scientific machine learning, many recently developed techniques like the Universal Differential Equations (UDEs), Hamiltonian Neural Networks, and others have yet to see wide spread application on noisy, real-world datasets. Our secondary goal for this chapter is therefore to demonstrate how, with some slight modifications, these techniques can be applied to real-world problems.

5.1 Time-Series Methods for Uncertainty Quantification

As we've already demonstrated, the shrinking cost of sensing technologies has improved our ability to create dense sensing networks. In order to make effective use of these sensors, and to

provide high quality data that can be used for critical decision making, it is vital we establish both the relevant sampling time scale and reasonable uncertainty estimates for measurements obtained by these sensors. For low cost sensors in particular, the manufacture supplied uncertainty estimates tend to be highly conservative so as to minimize their responsibility for variation in device performance, and in general, to prevent unnecessary returns. Consider for example, the sample time series for particulate matter concentrations at a variety of size fractions collected by one of our low-cost sensing nodes for a single 24-hour period (figure 5.1: These sample data illustrate many typical features of air quality time series: there is

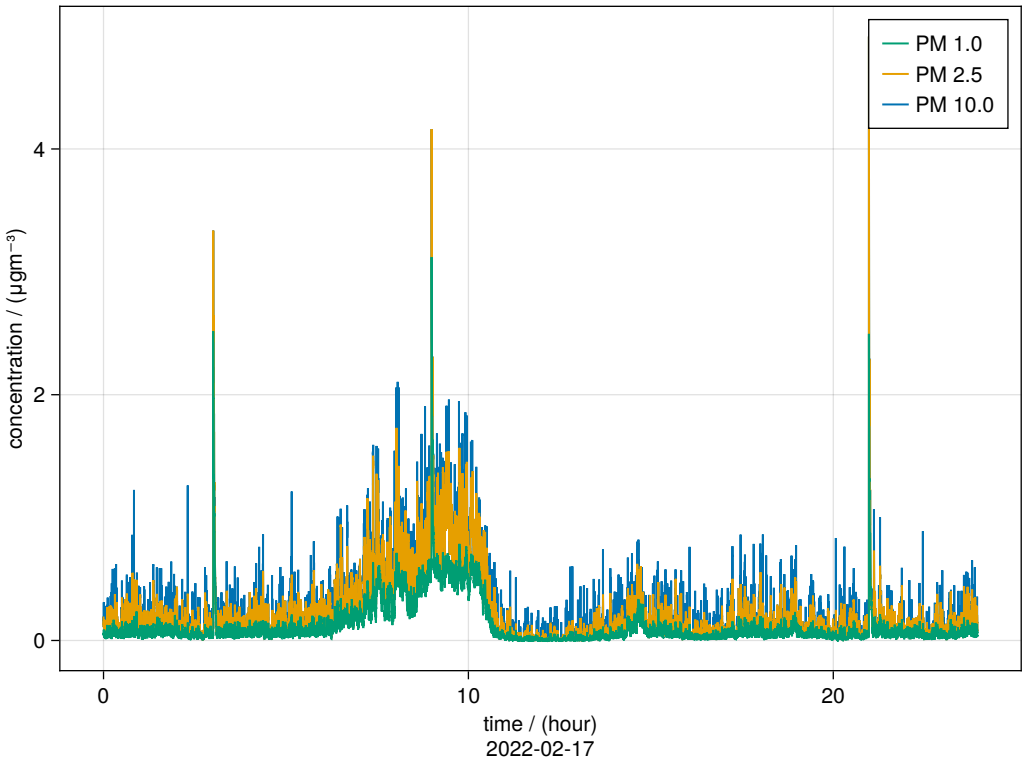


Figure 5.1: Time series of particulate matter at size fractions 1.0, 2.5, and 10.0 μm captured at a single location over one 24-hour day.

a baseline level of noise, there are general periodic trends (an increase in concentration at all size fractions near morning traffic around hour 10), and there are intermittent transient events. If we suppose that we *already* know what the shortest time resolution needed to

resolve the important transients is, say Δt, then a straight forward approach to develop a baseline notion of uncertainty for a time series, $z(t)$, is to convolve a rolling window of width Δt across the signal so that at any time t we may define the associated *pseudo-observation* to be

$$\bar{z}(t) = \text{mean}\left(\{z_j\}\right) \pm \text{std}\left(\{z_j\}\right); \quad \text{where } j \in \left\{ j \middle| \, t - \frac{\Delta t}{2} \leq t_j \leq t + \frac{\Delta t}{2} \right\} \tag{5.1}$$

where $\text{std}\left(\{z_j\}\right)$ is called the *representativeness uncertainty* as it measures how representative a single measurement is for the entire sampling window.

Unfortunately it may be difficult to know the relevant times scales of key transient events in advance. Given the fact that it is increasingly common for modern sensing systems (like our low-cost PM monitors) to provide sampling rates north of 1 Hz, we do not want to *shoot ourselves in the foot* by arbitrarily choosing an averaging window that will smooth away all the relevant features. Therefore, we seek to develop an alternative technique that can simultaneously establish the relevant temporal resolution for sampling whilst also enabling us to estimate the aleatoric, that is, intrinsic uncertainty of the sensing system.

5.1.1 Uncertainty Quantification with Temporal Variograms

The key feature that differentiates time series from other data sources is that at short time scales, neighboring samples are not independent; measurements taken close in time tend to be more correlated than measurements taken far apart. One way to measure this idea of self-similarity is by the auto-correlation function, which for a continuous signal $z(t)$ is

$$R_{zz}(\Delta t) = \int_{\infty}^{\infty} z(t + \Delta t) z^*(\Delta t) dt. \tag{5.2}$$

This can be efficiently computed for real-valued time series in $\mathcal{O}(n \log n)$ using the *Fast Fourier Transform* via

$$R_{zz}(\Delta t) = \mathcal{F}^{-1}\left[\mathcal{F}(z) \cdot (\mathcal{F}(z))^*\right] \tag{5.3}$$

The first peak of this autocorrelation function is an excellent method to identify the relevant time-scale: high autocorrelation means we gain little extra information between points offset by a time lag of Δt. This puts us on the right track but does not provide a straight forward interpretation for an intrinsic uncertainty. Taking motivation from the representativeness uncertainty, let us instead consider the expected variance between neighboring samples as a function of the time lag Δt. That is,

$$2\gamma(\Delta t) = \mathrm{Var}\left(z(t+\Delta t) - z(t)\right) \tag{5.4}$$

where γ is called the *semi-variogram* for $z(t)$. Expanding this definition yields

$$2\gamma(\Delta t) = \mathbb{E}\left[\left(z(t+\delta t) - z(t) - \mathbb{E}\left(z(t+\Delta t) - z(t)\right)\right)^2\right] \tag{5.5}$$

which for timescales where $z(t)$ is sufficiently stationary simplifies to yield

$$\gamma(\Delta t) = \frac{1}{2}\mathbb{E}\left[\left(z(t+\Delta t) - z(t)\right)^2\right]. \tag{5.6}$$

For a perfect measurement system we would expect that taking $\lim_{\Delta t \to 0} \gamma(\Delta t) = 0$, since measurements taken at the same time by the same instrument *should* be identical. Intrinsic uncertainty will therefore manifest as a non-zero limit and taking the square-root of the variogram will yield units that match our original time series and provide a reasonable uncertainty estimate. For a given set of samples, we construct the *empircal variogram* as

$$\hat{\gamma}(\Delta t) = \frac{1}{2N(\Delta t)} \sum_{i}^{N(\Delta t)} \left(z(t_i + \Delta t) - z(t_i)\right)^2 \tag{5.7}$$

where $N(\Delta t)$ are the number of available observation pairs $(z_i, z(t_i + \Delta t))$ for a lag of Δt.

There are three key parameters of the variogram $\gamma(\Delta t)$ which we want to estimate from $\hat{\gamma}(\Delta t)$:

- **Nugget**: The y-intercept of $\gamma(\Delta t)$ representing the limit as $\Delta t \to 0$.

- **Sill**: The value of $\gamma(\Delta t)$ as $\Delta t \to \infty$. In other words, the expected variance for uncorrelated samples of our time series. *Note* the **partial sill** is defined to be the sill minus the nugget.

- **Range**: The Δt at which γ realizes it asymptote. Effectively this is the time lag beyond which samples are uncorrelated and can be used as a proxy for the relevant sampling time scale.

To do this, we first compute $\hat{\gamma}(\Delta t)$ as above and then fit a model to $\gamma(\Delta t)$. There are a number of popular choices depending on the structure of a particular time series. In my open source Julia implementation, `TimeSeriesTools.jl`, the following models are made available:

- **Spherical**:

$$\gamma(\Delta t) = b + C_0 \left(1.5\frac{\Delta t}{r} - 0.5\frac{(\Delta t)^2}{r^2} \right) \tag{5.8}$$

- **Exponential**:

$$\gamma(\Delta t) = b + C_0 \left(1 - \exp\left(-\frac{\Delta t}{r} \right) \right) \tag{5.9}$$

- **Gaussian**:

$$\gamma(\Delta t) = b + C_0 \left(1 - \exp\left(-\frac{(\Delta t)^2}{r^2} \right) \right) \tag{5.10}$$

- **Circular**:

$$\gamma(\Delta t) = b + C_0 \left(1 - (2\pi)\cos^{-1}\left(\frac{\Delta t}{r} \right) + \frac{2\Delta t}{\pi r}\sqrt{1 - \frac{(\Delta t)^2}{r^2}} \right) \tag{5.11}$$

- **Cubic**:

$$\gamma(\Delta t) = b + C_0 \left(7\left(\frac{\Delta t}{r} \right)^2 - 8.75\left(\frac{\Delta t}{r} \right)^3 + 3.5\left(\frac{\Delta t}{r} \right)^5 - 0.75\left(\frac{\Delta t}{r} \right)^7 \right) \tag{5.12}$$

- **Pentaspherical**:

$$\gamma(\Delta t) = b + C_0 \left(\frac{15}{8} \left(\frac{\Delta t}{r} \right) - \frac{5}{4} \left(\frac{\Delta t}{r} \right)^3 + \frac{3}{8} \left(\frac{\Delta t}{r} \right)^5 \right) \tag{5.13}$$

- **Sine Hole**

$$\gamma(\Delta t) = b + C_0 \left(1 - \frac{\sin(\pi \Delta t / r)}{\pi \Delta t / r} \right) \tag{5.14}$$

where C_0 is the partial sill, b is the nugget, and r is the range. Figure 5.2 demonstrates these fits for the PM_{10} time series from above.

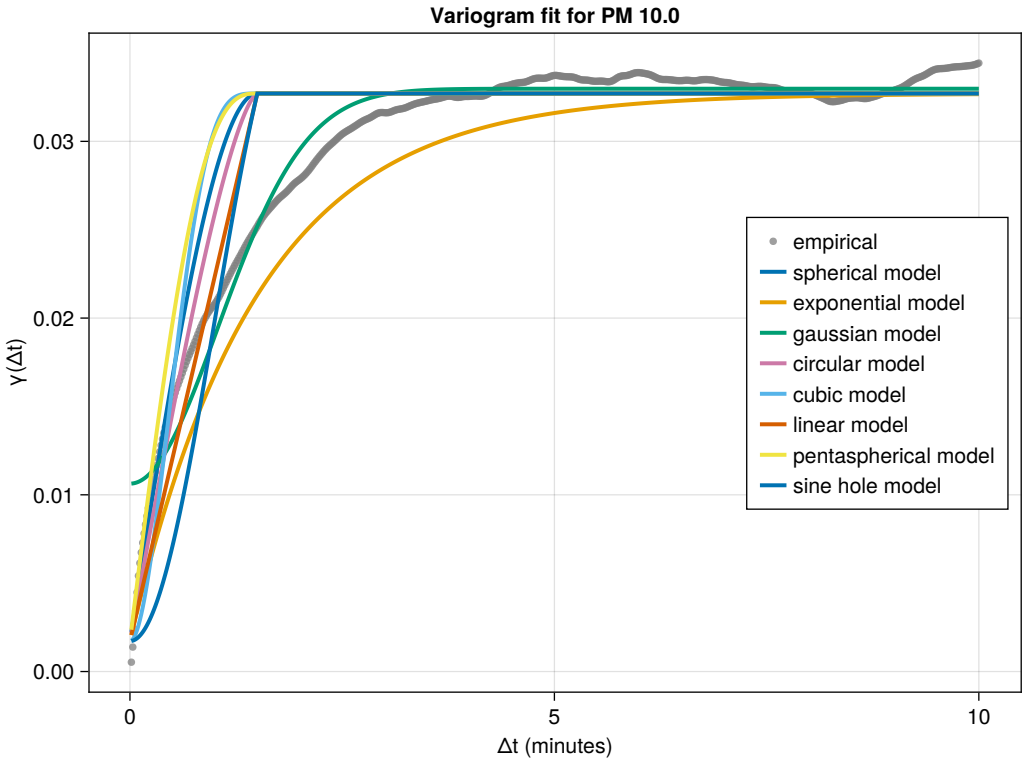


Figure 5.2: The empirical variogram and a variety of model fits obtained for the a PM 10.0 single-day time series

5.1.2 Next Steps

As of right now, I have developed an open source library to compute the temporal variogram and extract the estimated uncertainty for generic time series data. I have also developed a

pipeline for the acquisition and storage of time series data in publicly available S3 buckets on the Open Storage Network. My plan now is to evaluate the uncertainty for a variety of sensors from our network for data collected across the previous 1-3 years. We will then be able to investigate how stable these uncertainty estimates are as a function of time. For example, do we see any seasonal trends during the coldest winter months and warmest summer months? Does the uncertainty appear to increase over time so that we might be able to automatically identify when a sensor should be serviced/replaced?

5.2 Physics Informed modeling techniques for Air Quality Data

In order to provide actionable insights we must be able to effectively model the dynamics of our collected time series. In a perfect world, we would measure all relevant physical quantities such that the time evolution of local air quality at each sensor could be obtained by simulating the relevant micro-physics. However, low cost sensor networks are not equipped with all the necessary reference grade instruments needed to perform such simulations; accurate winds speed and direction sensors alone can cost hundreds to thousands of dollars and remote sensing data products are often unreliable at the ground level (i.e. in the human head space). We therefore are motivated to develop techniques to model our collected time series using only the data provided at a single node. There are many approaches to this task in the statistics and machine learning literature including statistical models like ARIMA and deep learning methods like Recurrent Neural Networks (Adhikari and Agrawal, 2013; Petneházi, 2019). While these methods can often lead to robust short term predictions, they do not incorporate prior physics knowledge and therefore are not primed to take advantage of underlying dynamical laws. Recently two interesting physics-informed, data driven techniques have been developed for just this type of scenario. The first we shall examine is the so-called *Hankel Alternative View Of Koopman* (HAVOK) framework which extends the principle of dynamic mode decomposition to nonlinear systems (Brunton et al., 2017). The second is

an technique dubbed the *Hamiltonian Neural Network* which extends the notion of a Neural Ordinary Differential Equation to allow a neural network to learn coordinate transformations or the original time series data which satisfy *Hamiltons equations* (Greydanus et al., 2019).

5.2.1 A Hybrid HAVOK UDE Approach

If we are to take a physics informed approach to modeling our time series, a good place to start is with the question: *How much information about the physical state of a system can be extracted from a single time series.* The answer, surprisingly, is *more than you might expect.* For simplicity let us consider a discrete, deterministic dynamical system described by a smooth function $f : M \to M$ that maps points in the state-space manifold M to new points in the same manifold. We can define an *observable* $y : M \to \mathbb{R}$ as a smooth function which maps a state $x_t \in M$ to a number. The Takens Embedding Theorem then suggests that if the dynamical system described by f has a strange attractor of box counting dimension d, then there exists a $k \geq 2d$ such that the embedding $\mathbf{y} : M \to \mathbb{R}^k$ given by

$$\mathbf{y}(x) = \big(y(x), y(f(x)), y(f^2(x)), ..., y(f^{k-1}(x))\big) \tag{5.15}$$

is a diffeomorphism. In other words, the time series $(y(x_t), y(x_{t+1}), y(x_{t+2}), ...)$ formed by tracking the evolution of a single point x_t in the state manifold M under the dynamical law $x_k \mapsto x_{k+1} = f(x_k)$ captures the entire structure of the original dynamics in the state manifold (Takens, 2006). This remarkable result provides justification for our modeling and suggests that for practical purposes, we will want to construct our models by considering dynamics on vectors formed by so called *time delay embeddings* like that of the previous equation.

With this justification in hand, the next logical step is to propose a model for the dynamics which describe the evolution of such delay-embeddings. To do this, we will utilize technqieus from Koopman Operator theory. Koopman operator theory provides a framework for understanding the time evolution of a dynamical system at the level of observable

functions rather than directly from the state itself. This is highly analogous to the equivalence between the Schrodinger and Heisenberg pictures in Quantum Mechanics where the former provides time evolution via dynamics of the state vector, $|\alpha(t)\rangle$, and the later provides dynamics on the observable operators, $X(t)$, of a system (Sakurai and Commins, 1995, p 80-84). In this same spirit, we seek to identify an operator $\mathcal{K}$, called the *Koopman Operator* for our dynamical system which satisfies

$$\mathcal{K}\left[y(x_k)\right] = y(f(x_k)) = y(x_{k+1}), \tag{5.16}$$

that is, $\mathcal{K}$ takes an observable y and maps it to a new value which is just the observable evaluated at the *next* state after an update from f. This is highly appealing because such an operator (if it exists) is *linear* and results in a *linear* dynamical system for the time series of an observable, namely

$$y_t = \mathcal{K}^t(y_0) = y(f^t(x_0)). \tag{5.17}$$

The goal should now be clear: if we are able to utilize the data from our time series (interpreted as an *observable* of the underlying system's state space) to estimate the operator $\mathcal{K}$, then we will be able to model the dynamics of our time series for which a time delay embedding of sufficient length should produce an attractor that is diffeomorphic to the underlying dynamics of the system's true state. Unfortunately, the price paid for the linearity of $\mathcal{K}$ is that its representation may demand an infinite dimensional basis (Brunton et al., 2021). Therefore, in practice, we hope that sufficient data will enable us to identify a suitable subspace of M for which a finite approximation of $\mathcal{K}$, say K, keeps our observables bounded. One approach (add citation for Mezic et al here) is to attempt an eigen decomposition of K using our time series data which relies on carefully crafting the observable functions y. In our case, we do not have direct control over the available observables; we are limited to the sensors provided by each sensing node. As an alternative approach, we take inspiration from

the popular method of Dynamic Mode Decomposition (DMD) to enable approximations of the Koopman operator from time series observations.

Originally, DMD was introduced to allow for the decomposition of expensive fluid dynamics simulations into a basis of *modes* (in the spirit of normal modes) such that the time evolution across the entire domain can be obtained by a linear combination of such modes with some simple time-dependent coefficients. With this as our inspiration, we can take a similar approach by forming a so called *Hankel matrix*, H, from our time series as

$$H = \begin{bmatrix} y_1 & y_2 & \cdots & y_p \\ y_2 & y_3 & \cdots & y_{p+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_q & y_{q+1} & \cdots & y_{p+q} \end{bmatrix} \tag{5.18}$$

(NOTE: we should update this to match our definition i.e. starting at y_1 or y_q as the first element) where each column is a shifted time delay embedding of length q and each row corresponds to a subsample of our observable time series of length p. We can decompose this matrix using the SVD so that

$$H = U\Sigma V^T, \tag{5.19}$$

where the singular values $\sigma_k = \Sigma_{kk}$ are ranked hierarchically by the relative contribution to H. If the singular values drop off fast enough, a reasonable approximation to H can be obtained by keeping only r-many singular values and singular vectors from the decomposition. This provides us a data-driven approach to approximating the Koopman operator for a subspace (that is a subset of singular vectors) that is approximately invariant. To see this, we can rewrite our Hankel matrix in terms of $\mathcal{K}$ so that

$$H = \begin{bmatrix} y_1 & \mathcal{K}(y_1) & \cdots & \mathcal{K}^{p-1}(y_1) \\ \mathcal{K}(y_1) & \mathcal{K}^2(y_1) & \cdots & \mathcal{K}^{p}(y_1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{K}^{q-1}(x_1) & \mathcal{K}^{q}(x_1) & \cdots & \mathcal{K}^{p+q-1}(x_1) \end{bmatrix} \tag{5.20}$$

this direct connection to the propagation by $\mathcal{K}$ suggests we *should* be able to successfully obtain a linear model describing the dynamics of the singular vectors $v_i(t)$. The potential for multiple attractors as well as the challenges of noise in real datasets suggests that a linear model alone may not be good enough to recover the full dynamics and therefore the HAVOK model proposed by Brunton seeks to fit a model on the first $r-1$ singular vectors with *external forcing* provided by the r$^{\text{th}}$ component:

$$\frac{d}{dt}\mathbf{v}(t) = A\mathbf{v}(t) + Bv_r(t) \tag{5.21}$$

where $\mathbf{v} = (v_1, v_2, ..., v_{r-1})^T$. To demonstrate this approach, consider the famous Lorenz dynamical system given by

$$\dot{x} = \sigma(y - x) \tag{5.22}$$

$$\dot{y} = x(\rho - z) - y \tag{5.23}$$

$$\dot{z} = xy - \beta z \tag{5.24}$$

with parameters chosen so that $\sigma = 10$, $\rho = 20$, and $\beta = 8/3$. Because the SVD provides a hierarchical ranking of the singular vectors by the relative scale of the singular values, we imagine this $v_r(t)$ to be an intermitent external forcing do the the fact that it represents a low energy mode. A visualizualition of a sample trajectory for this system as well as the *embedded* attractor formed from the first three principal vectors v_1, v_2, v_3 of the singular value decomposition of the associated Hankel matrix for the "observable" $x(t)$ are illustrated in Figure 5.3.

This is a clear demonstration of Takens embedding theorem in action: the attractor formed from the first three singular vectors of the Hankel matrix for the single observable $x(t)$ appear to capture key features of the original attractor formed from the state vector $(x(t), y(t), z(t))^T$. The rows of the matrix U correspond to the *modes* of the singular value decomposition and allow us to decompose the original delay embedding of the observable

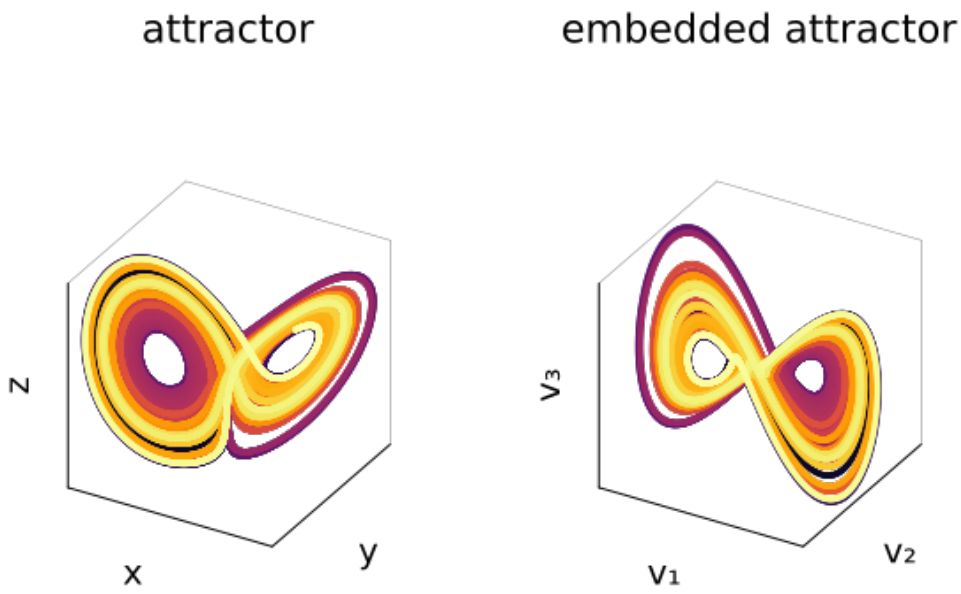


Figure 5.3: Comparing the original Lorenz attractor (left) to the embedded attractor learned after performing the SVD (right). Color indicates the time along the trajectory.

time series into this new representation. In this example, we have chosen $r = 15$ as is done in (Brunton et al., 2017). A more programmatic approach is to evaluate r based on some information criterion. Alternatively, this can be left as a hyperparameter and adjusted to achieve a better fit. These modes are visualized in Figure 5.4 below.

The matrices A and B for our HAVOK model are obtained by standard linear regression. (We should add further description of how to form the linear system here, .e.g. we do linear regression on a non-square matrix of weights to let us fit A and B simultaneously.) The coefficients of these matrices for the Lorenz system described above are illustrated in Figure 5.5. We note that the form of the A matrix appears to closely follow the Toeplitz structure one would expect from the matrix of a finite difference stencil. We can then examine the time series obtained by integrating the HAVOK model forward to see how well we model the original dynamics of the singular vectors v_i. The first of these is shown in Figure 5.6 To further evaluate the quality of the resulting model, we can visualize the scatter diagram

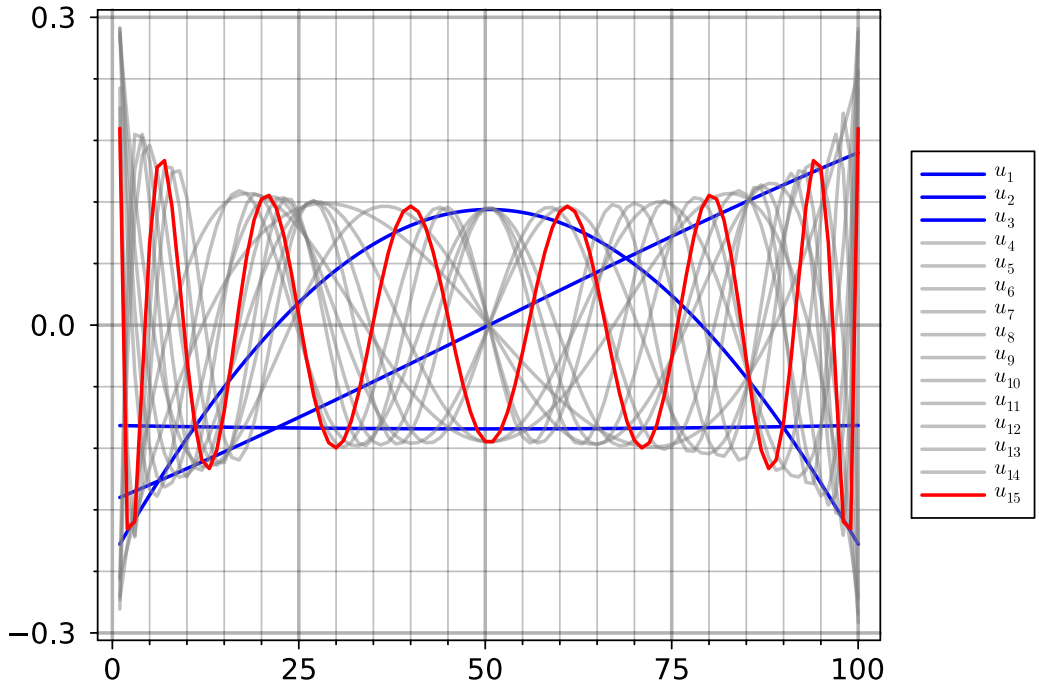


Figure 5.4: Eigenmodes of the embedded attractor extracted from the SVD.

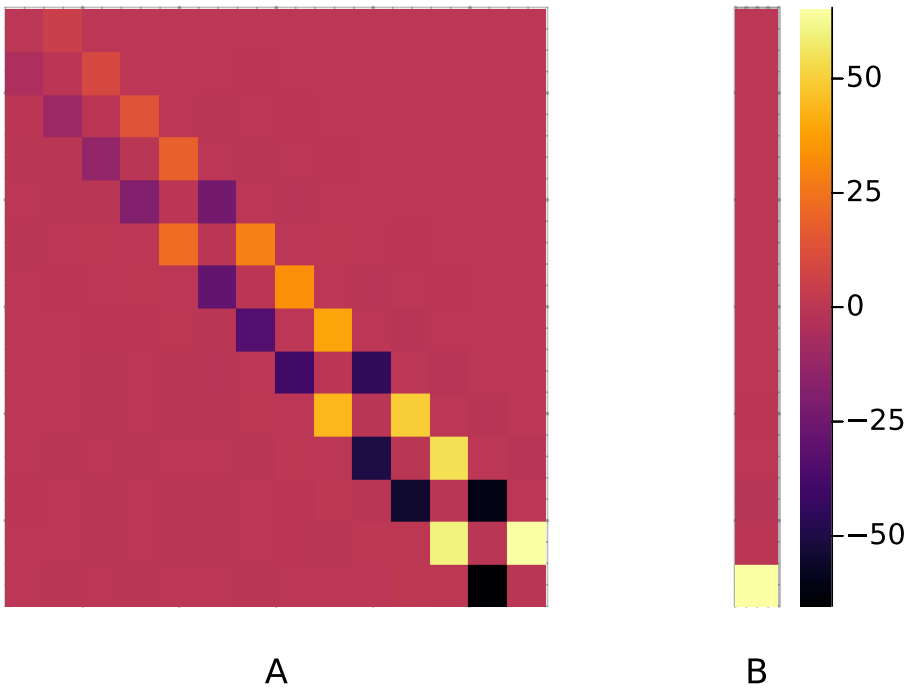


Figure 5.5: Heatmap of the linear Koopman operator together with the forcing activation.

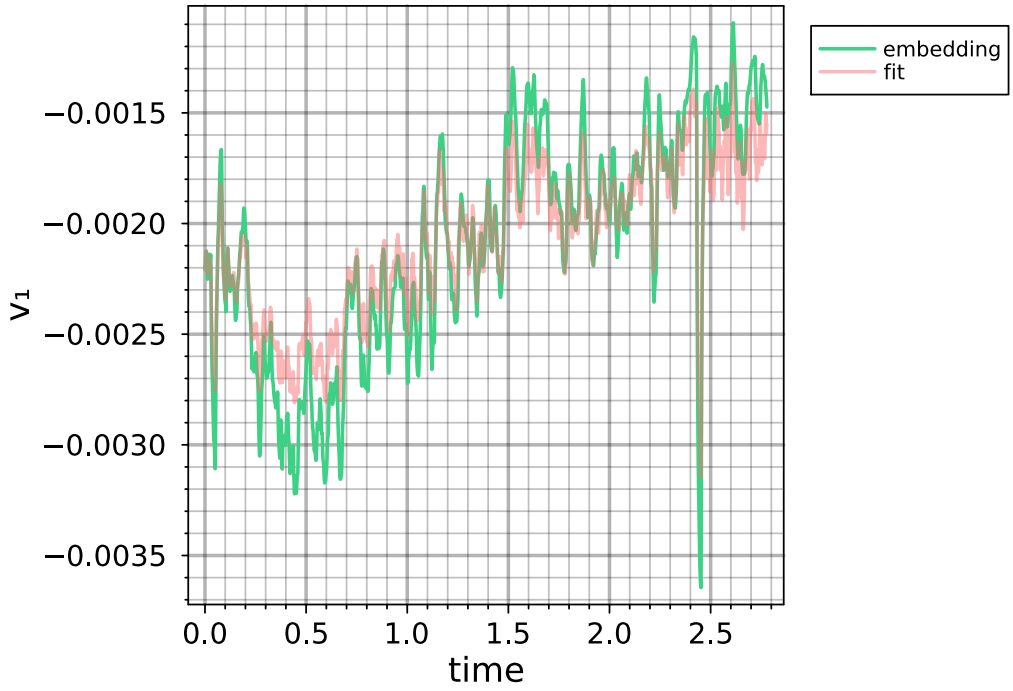


Figure 5.6: The reconstructed timeseries for v_1 via the learned HAVOK model.

of the predicted values for $v_1(t)$ on training data (the points that were used in the Hankel Matrix) as well as some holdout data. The results are shown in Figure 5.7.

Finally, we can visualize the original time series series and the embedded attractor by differentiating regions with high externel forcing (salmon colored) from the regions where the observable is approximately linear (green colored), that is where the dynamics of the observable are invariant to our learned Koopman operator. This is illustrated in Figures 5.9 and 5.10.

Results so far

What happens if we try and apply this HAVOK approach to a sample of our PM time series? This section details preliminary results with the method. Figure 5.11 illustrated the embedded attractor obtained by visualizing the first 3 singular vectors of the SVD for a $PM_{2.5}$ time series. After attempting a variety of cut of values r as well as a variable number

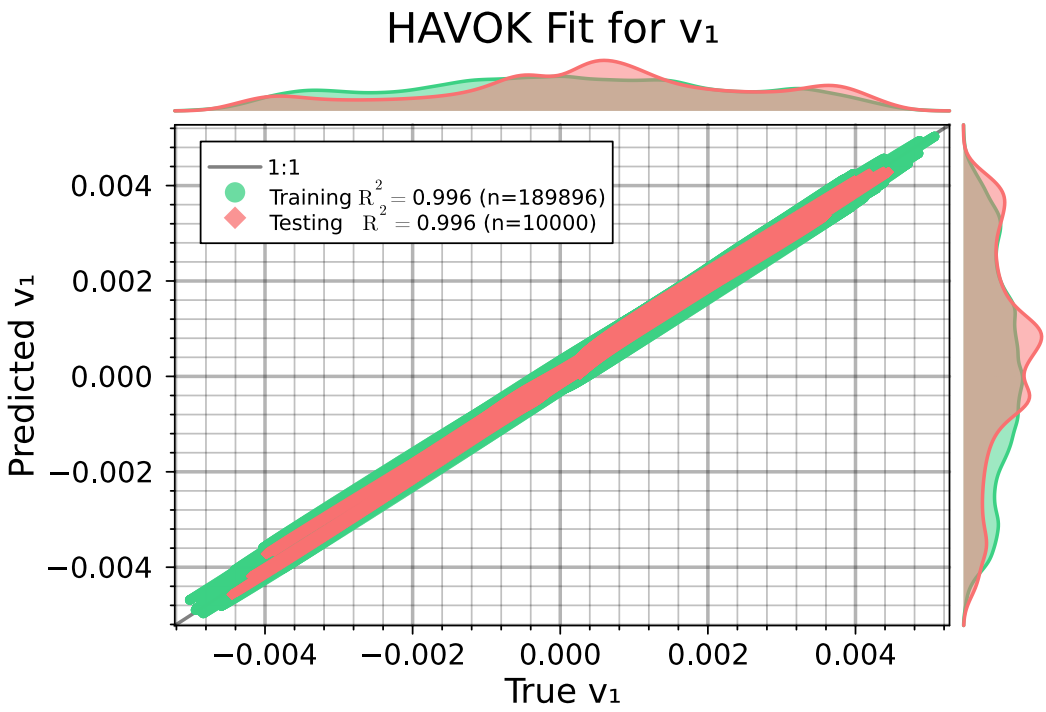


Figure 5.7: A scatterplot of the resulting HAVOK fit

of forcing terms, the *best fit* values for the A and B matrices are visualized in 5.12. As in the case of the Lorenz system, the A matrix has an interesting structure of alternating positive and negative coefficients in an anti-symmetric pattern along the diagonals. The learned coefficients of the external forcing activation matrix B appear to be biased towards negative values. Time series obtained for the first three components are visualized in Figure **??**. In figure 5.14 we show the same data limited to the first 2.5 hours. This appears to suggest that the HAVOK models has learned a reasonable representation of the dynamics, however errors appear to accumulate over past 2.5 hours such that the model fails for long time integration.

Finally, if we examine the original time series and embedded attractor colored by the magnitude of the first forcing vector, we obtain Figures 5.15, and 5.16.

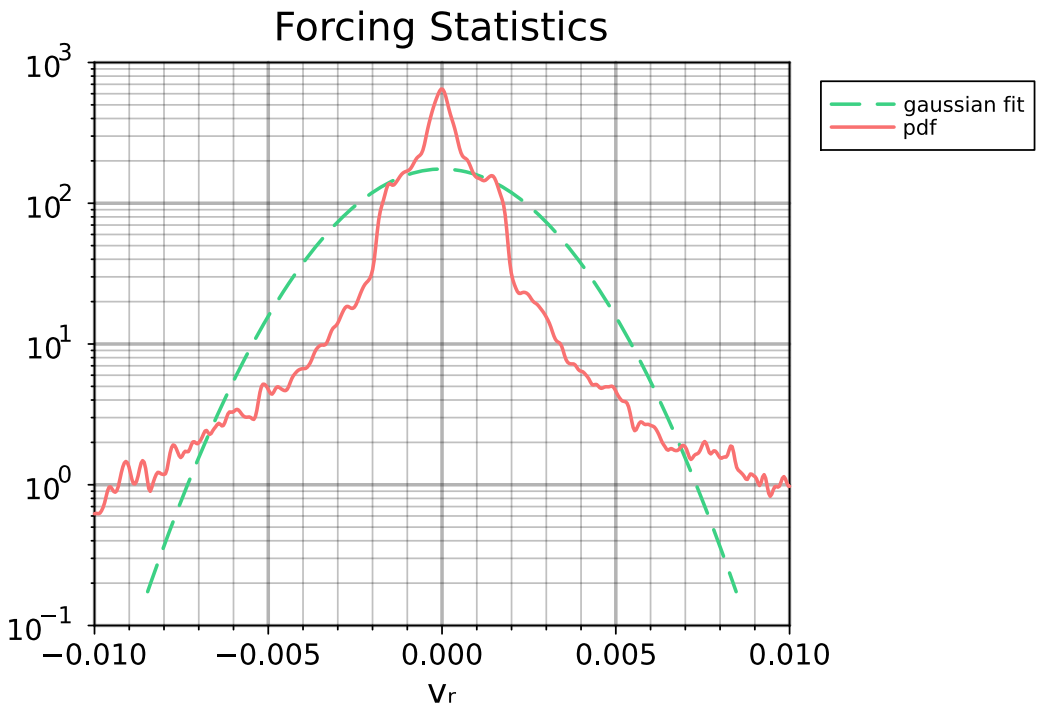


Figure 5.8: The statistics of the learned forcing function. The sharpness of the distribution (in comparison to a Normal distribution) indicates that the forcing is *intermittent*

Next Steps

The preliminary results of our application of HAVOK to model PM data are very encouraging. The time series obtained in Figure 5.14 suggest we have achieved a reasonable first order model, however as the integration period extends, the agreement is quickly lost. This is likely due to our observations not being entirely invariant under the approximated Koopman operator and therefore it is unreasonable to expect that our model will be successful with linear and external forcing terms alone. To address this, I plan to combine the HAVOK approach with the Universal Differential Equations (UDEs) framework established by Rackauckas et al in (Rackauckas et al., 2020). That is, we can augment our learned HAVOK model to become

$$\frac{d\mathbf{v}}{dt} = A\mathbf{v}(t) + Bv_r(t) + \mathrm{NN}(\mathbf{v}, t) \tag{5.25}$$

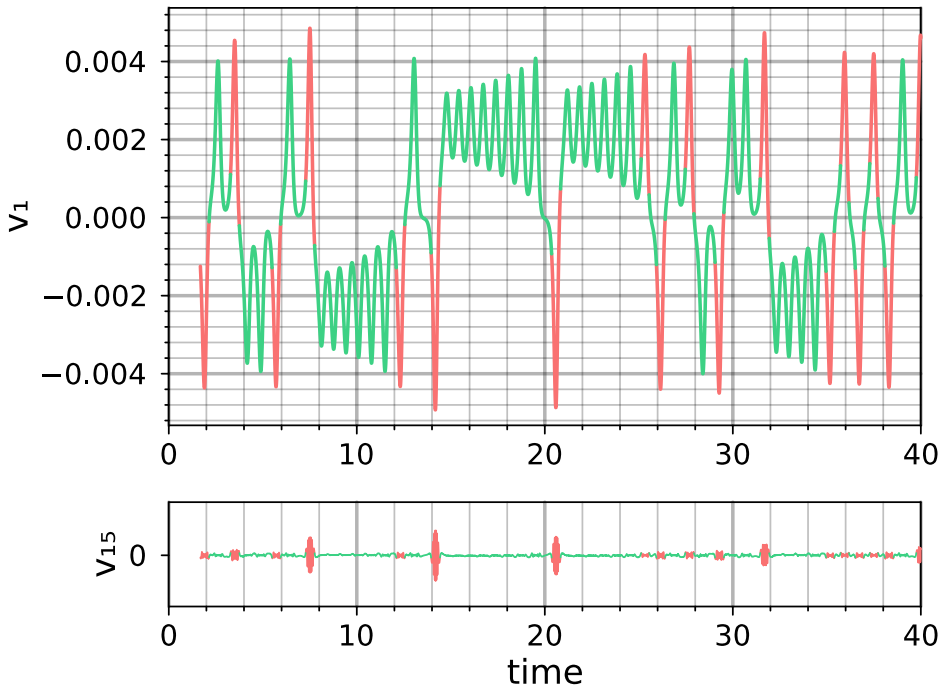


Figure 5.9: The time series for v_1 marked where the forcing function is above a specified threshold.

where $\mathrm{NN}(\mathbf{v}, t)$ is a neural network. Since neural networks are universal function approximators, we can train the NN to learn the missing dynamics not present in our simple HAVOK model. Once we have accomplished this, we can then use sparse regression techniquese such as SINDy as in (Brunton et al., 2016) to obtain a functional form for the missing nonlinear terms which complete the model.

5.2.2 Hamiltonian Neural Networks

The HAVOK modeling approach outlined in the previous section can largely be understood as a *data-driven* method. By carefully curating the Hankel matrix from our time series observations, we open the door to techniques from linear algebra that enable the generation of a straight-froward dynamical model. As an alternative method, let us now consider how we can use neural networks to directly model our time series data by forcing a neural network to learn useful transformations that simplify the modeling task.

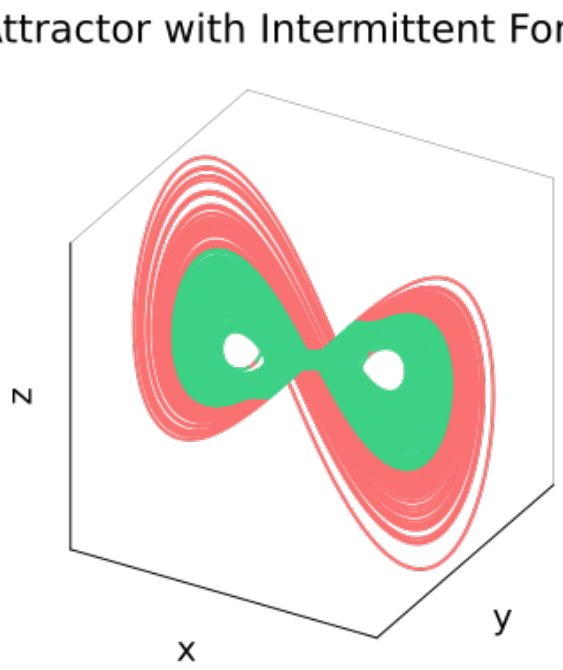


Figure 5.10: The embedded attractor colored by the presence of external forcing.

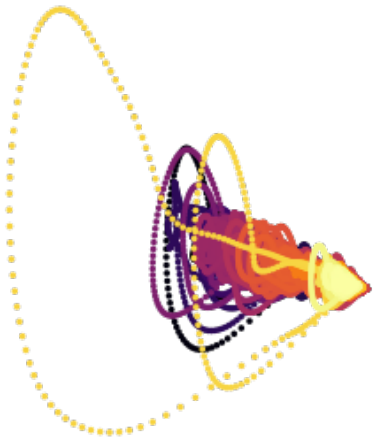


Figure 5.11: The embedded attractor for PM 2.5 time-series data.

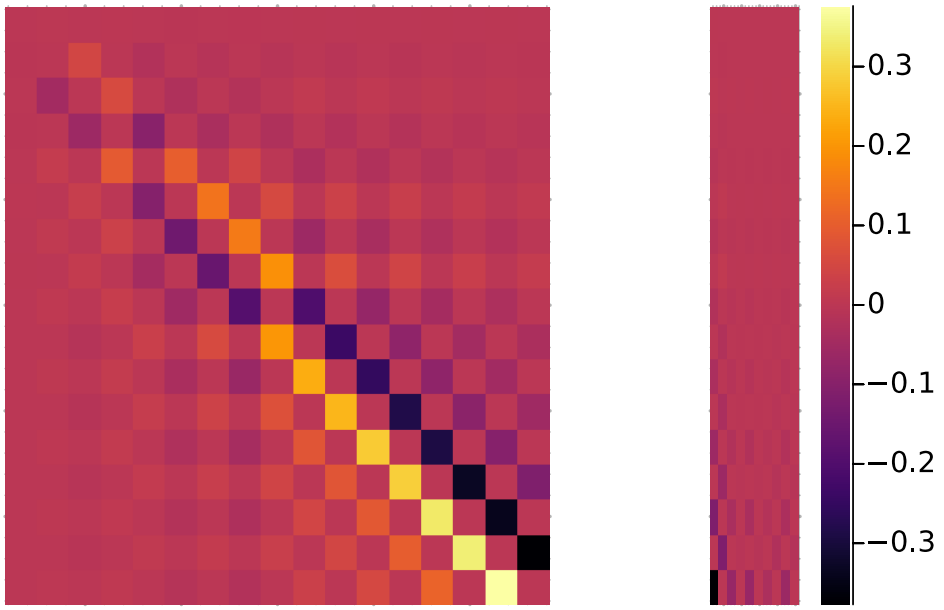


Figure 5.12: Heatmap of the linear Koopman operator together with the forcing activation.

For a number of years one of the most popular deep learning approaches for time series forecasting was the family of so-called *recurrent neural networks* (Petneházi, 2019). These models take the form

$$u_k = u_{k-1} + \mathrm{NN}(u_{k-1}; \theta) \tag{5.26}$$

so that the output of the k^{th} layer is formed by starting with the previous output and correcting by the output of a neural network $\mathrm{NN}(u_{k-1}, \theta)$. Recently Chen et al observed that this equation can be understood to be the discrete Euler scheme implementation for a continuous ODE of the form

$$\dot{u} = \mathrm{NN}(u, \theta). \tag{5.27}$$

Therefore, by utilizing our adjoint method for determining the sensitivity of an ODE model to the parameters θ, once can construct ODE layers whose output is the solution to a differential equations (Chen et al., 2018). By extension, a reasonable time series forecasting

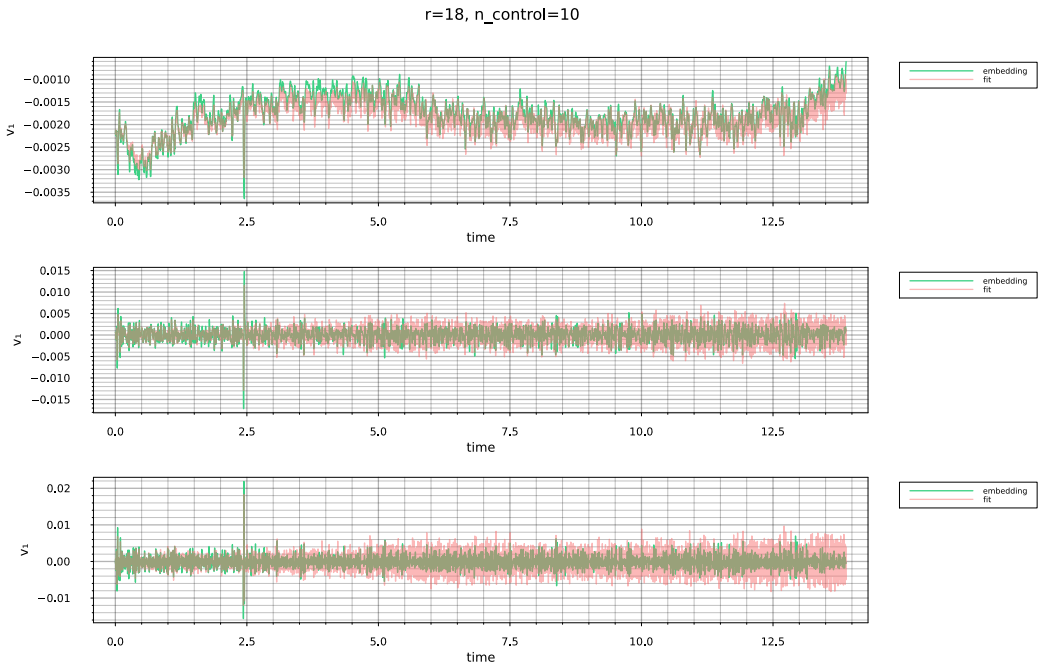


Figure 5.13: The reconstructed time-series for the first three embedding coordinates using the HAVOK fit.

approach would be to learn a model such that our time series $z(t)$ satisfies

$$\dot{z}(t) = \mathrm{NN}(z, t; \theta). \tag{5.28}$$

This technique is very general however it does not take advantage of any physical constraints imposed for time series sampling real, physical processes. Much effort is now being spent on how to regularize this approach to produce physics-informed Neural ODEs such that obey physical constraints like energy conservation (Sholokhov et al., 2023). Still, these physics-informed constraints are not themselves built into the model (i.e. the neural network) but rather the weights are encouraged during training to conform to physics priors. An interesting alternative development came from Greydanus et al who took a different approach by incorporating physical laws directly into the neural network structure (Greydanus et al., 2019). Their method, the Hamiltonian Neural Network, seeks to train a neural

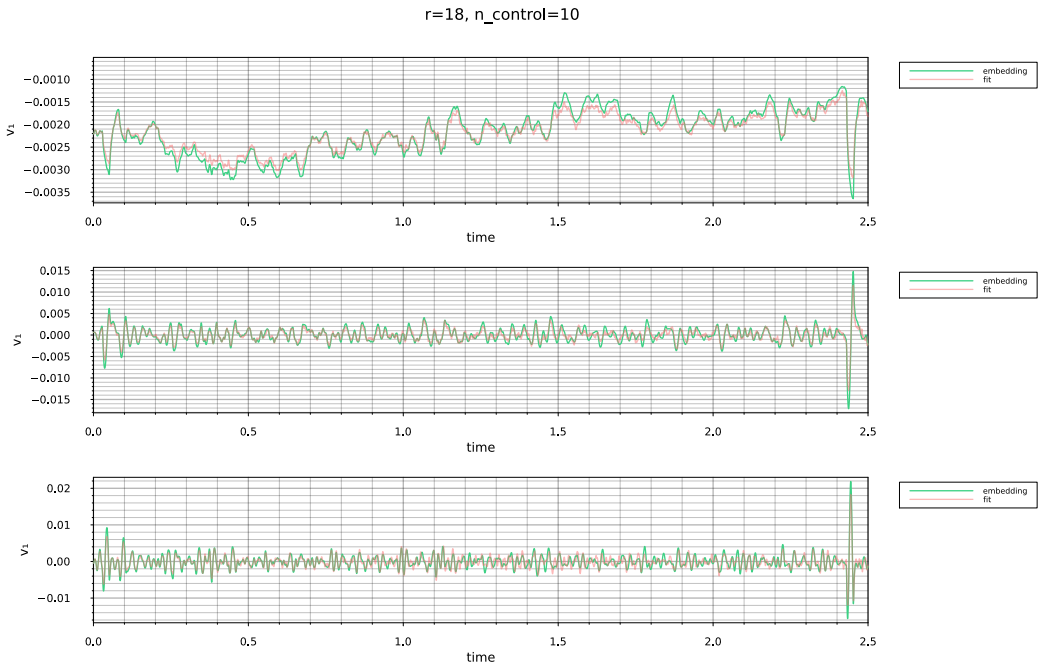


Figure 5.14: A zoomed in view of the same time-series reconstruction for the first 2.5 hours showing a very decent fit. Some kind of error appears to be accumulating over time.

network so that it satisfies Hamilton's equations, namely

$$\begin{aligned} H &= \mathrm{NN}(q, p, \theta) \\ \dot{q} &= \frac{\partial H}{\partial p} \\ \dot{p} &= -\frac{\partial H}{\partial q} \end{aligned} \tag{5.29}$$

where q and p are the generalized coordinates and momenta for a system. By constructing a loss function of the form

$$\mathcal{L}_{\mathrm{HNN}} = \left| \frac{\partial H}{\partial p} - \dot{q}|_2 + |\frac{\partial H}{\partial q} + \dot{p} \right|_2 \tag{5.30}$$

they found that the HNN is able to learn a valid representation of the Hamiltonian for a variety of physical systems (spring, double pendulum, etc.). Further, since Hamilton's equations must be integrated to obtain trajectories for a system, the learned Hamiltonian can easily explore counterfactual configurations not present in the supplied training data by

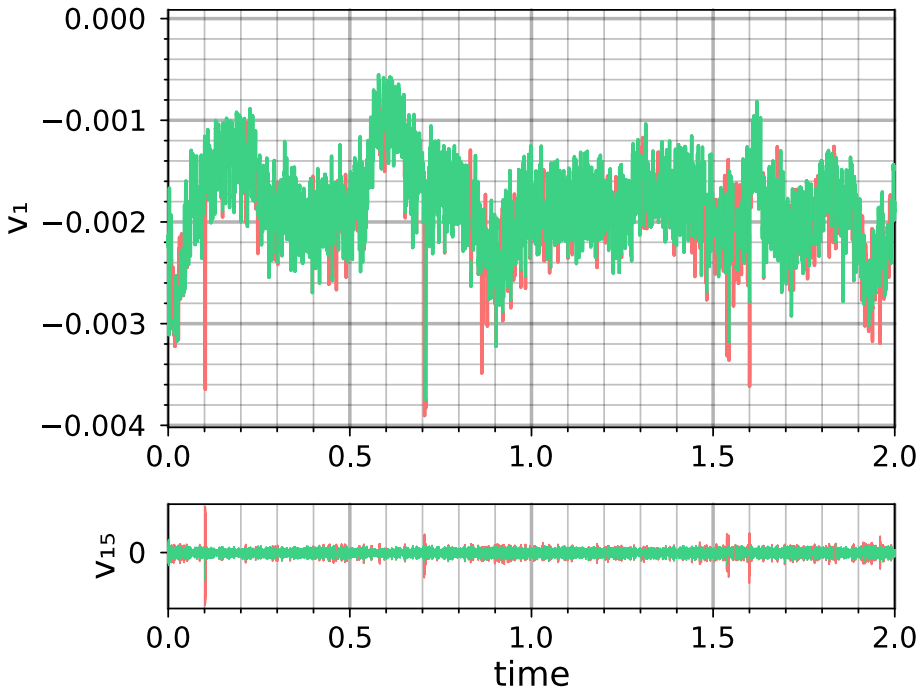


Figure 5.15: The first embedding coordinate with forcing above a critical threshold identified.

simply adjusting the Hamiltonian by addition of a constant (i.e. shifting the total energy). Because the total energy is a constant of motion, learning a representation of the Hamiltonian also improved the long term energy conservation of integrated trajectories as compared to directly attempting to predict $\dot{q}, \dot{p}$ with a neural network alone.

To make approach feasible for real data, one must supply both position and momenta time series which may not be readily available. In their final example, the authors demonstrate how to accomplish this with an autoencoder network. By using additional encoder/decoder models, they were able to take the raw images of a *video* of a swinging pendulum and *learn* an effective transformation into a q, p pair that they then use to learn the Haniltonian. Specifically, the HNN loss function is augmented to include two additional terms. The first is the standard autoencoder loss which demands an input that is encoded and then decoded be similar to the original input. The second is an additional loss term used to encourage the moment portion of the encoder to be some function of the velocity of the position component,

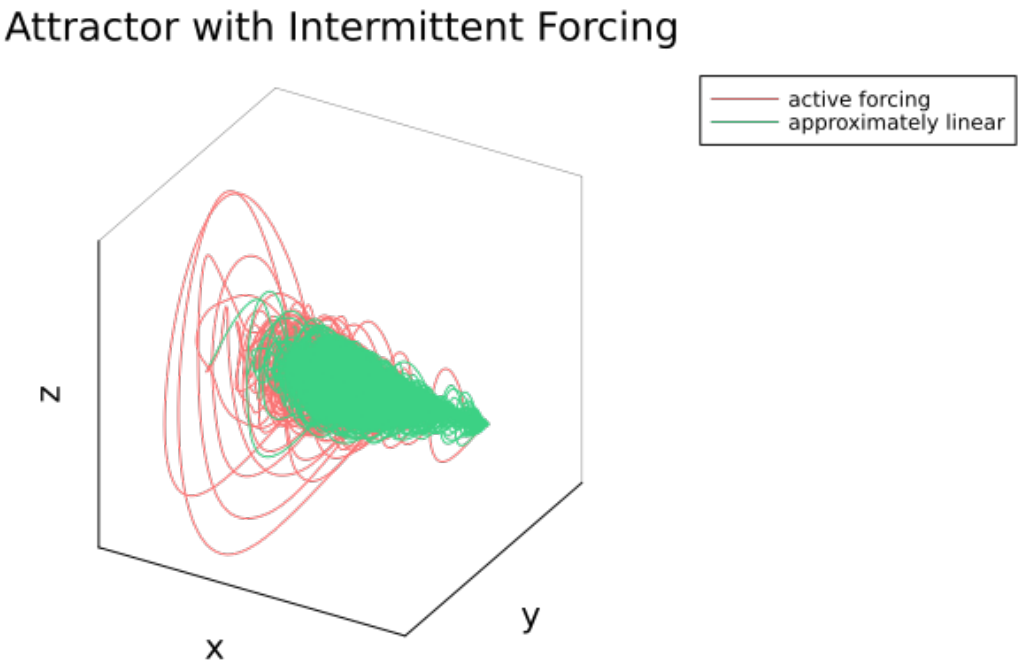


Figure 5.16: The original attractor now colored by external forcing above a threshold.

that is,

$$\mathcal{L}_{\text{momentum}} = \left| p^t - (q^{t+1} - q^t) \right|_2 \tag{5.31}$$

Results

I have implemented the HNN model for time series data collected by one of our central nodes. For this implementation I have joined time smoothed time series for $PM_{2.5}$ together with temperature, pressure, and humidity time series from the same node. Using this dataset with the HNN training procedure outlined above, I have been able to generate preliminary mappings of the learned Hamiltonian such as those in Figure 5.17

Next Steps

Now that I've developed the code to train an HNN, my plan is to attempt to train one for a data set consisting of roughly 1 week of observations from a Central Node to make sure

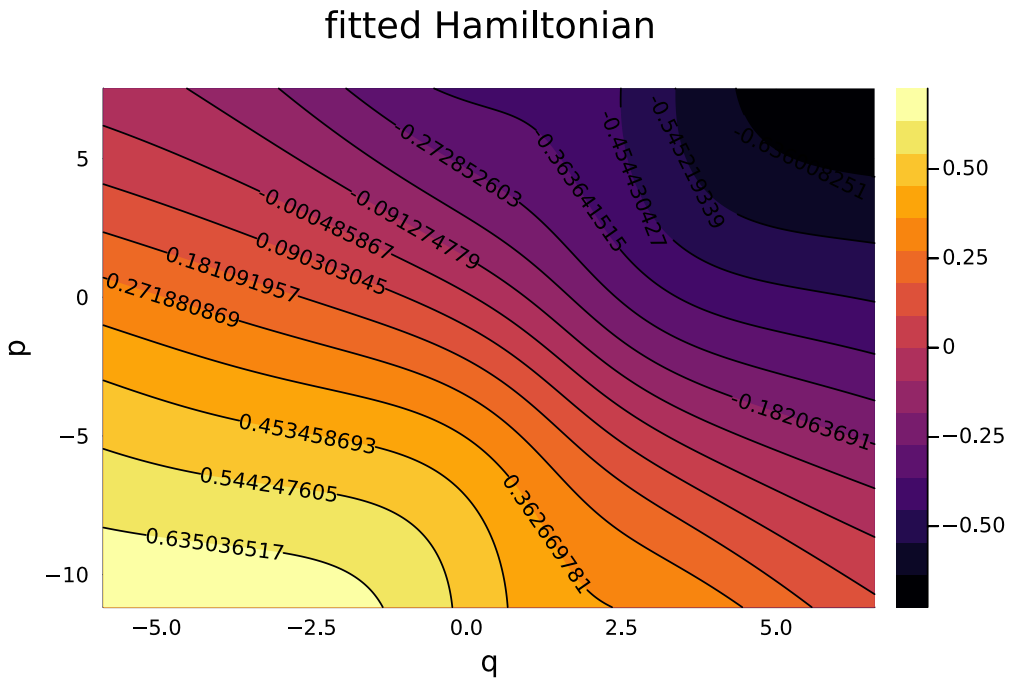


Figure 5.17: The landscape of the Hamiltonian learned using the HNN procedure as a function of the learned phase-space coordinates (q, p).

we include a few iterations of the diurnal cycle. With the trained HNN we can visualize the trajectory of the system through phase space, in particular, as it relates to the level surfaces of the learned Hamiltonian. I expect that for relatively calm conditions and short time scales, the trajectories will conform to the level sets of H so that the total energy is conserved. I am particularly interested in whether or not the magnitude of the gradient of H can be used as an indicator of potential externel pollution sources. Similarly, I am curious about how long a trained HNN will be valid for a single sensor, as well as whether or not an HNN trained for a single node will be transferable to multiple nodes.

CHAPTER 6

A CHEMICAL DATA ASSIMILATION FRAMEWORK FOR INDOOR AIR QUALITY ASSESSMENT

As described in the introduction, the COVID-19 pandemic has catalyzed global concern for indoor air quality. In order to provide actionable insights, a comprehensive analysis of the relevant chemistry is needed to establish how chemical constituents interact in indoor spaces. Much of outdoor air chemistry is driven by light; short wavelength photons can break bonds through photolysis and changing light conditions due to the rotation of the Earth introduce diurnal variations which drive relevant atmospheric reactions. Very little UV light makes it through building windows and into the indoor spaces. Additionally there is a large variability in indoor light sources including LED, fluorescent, incandescent and others which each have very different spectral properties. Therefore to establish a baseline for the relevant chemical processes indoors pertaining to air quality, we have developed an advanced sensing suite called the HEART chamber as described in chapter 2. By combining high quality measurements for a variety of source gases and aerosols together with detailed characterization of indoor photolysis we are able to produce rich data sets constraining the breadth of possible reactions in indoor spaces. Using this high quality data, we can then use techniques of data assimilation to constrain complicated chemical kinetics mechanism in order to infer the abundance of species that are too reactive to directly measure.

In this chapter we first introduce the relevant physics for reaction kinetics and photolysis which we will use in the construction and evaluation of our reaction mechanism models. Then we present preliminary results on a detailed evaluation of photolysis rates as well as initial measurements from our HEART chamber chemical kinetics evaluations using our data assimilation framework.

6.1 Physics of Chemical Reactions: Chemical Reaction Kinetics

6.1.1 Overview

Since the early successes of Newton's descriptions of mechanics by means of simple forces acting on masses, scientists have sought to understand the dynamics of chemical reactions in terms of the detailed microphysics of molecular collisions. As we shall see, this approach can be utilized productively to justify the complicated temperature and pressure dependence of the reaction rate coefficients of many elementary reaction. However, even when considering the asymmetric structure of many molecules, and therefore, the dependence on orientation at the collision site, kinetic theory alone is unable to model reaction rates in all relevant temperature and pressure regimes. To do this, one can utilize the modern treatment of *Potential Energy Surface* (PES) theory together with the notion of short-lived, unstable intermediate reaction states to calculate reaction rate coefficient functions for specific reactants. *Ab initio* solution of the Schrodinger equation for the relevant nuclear geometries ($3N$ reaction coordinates for N atoms) together with scattering and spectroscopic methods as suggested by (Neumark, 1992) have led to significant improvements in our understanding of reaction dynamics.

However the computational complexity of this task makes it prohibitively expensive to perform at the scale required for our desired chemical mechanism which consists of many hundreds to thousands of reactants together with as many as 16,000 unique reactions. Therefore in the following discussion, we shall primarily utilize the kinetic theory to justify the functional form for *most* rate coefficients with some reference to the extensions made by PES theory. We note that in practice, kinetic evaluations such as the periodic reports from the NASA Jet Propulsion Laboratory (Burkholder et al., 2020) utilize (justified) empirical fits to provide suggested functional forms for reaction rate coefficients.

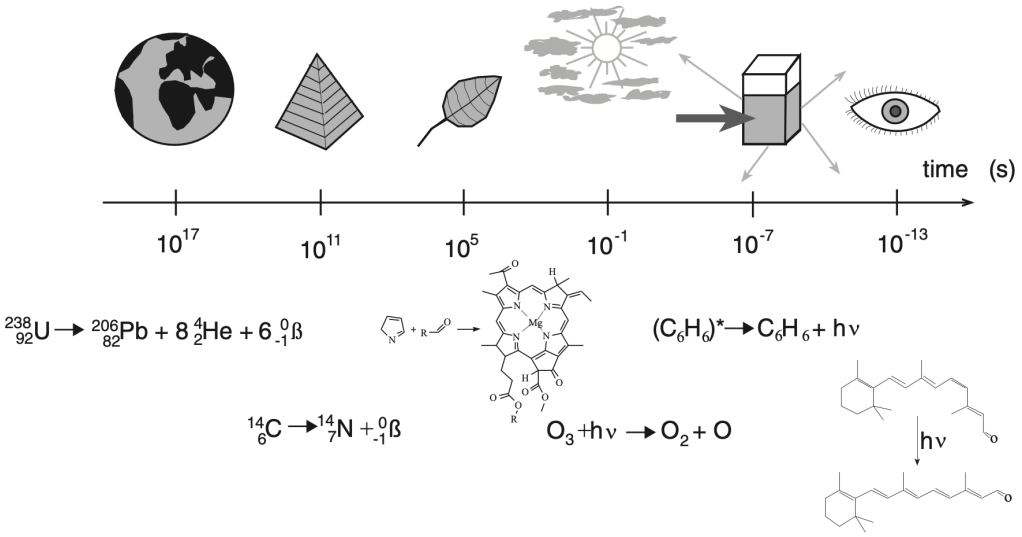


Figure 6.1: An illustration of the broad range of reaction time scales from the long-lived nuclear decay to rapid degradation of molecules by photolysis. Figure taken from (Arnaut and Burrows, 2006)

6.1.2 Chemical Equilibrium and the Law of Mass Action

Before outlining the dynamics involved during complex chains of chemical reactions, it is worth spending some time to consider how we should treat chemical equilibrium. Usually in these scenarios we can not hold the internal energy fixed due to interactions with the environment, but rather, the temperature and pressure can often be treated as so. For example, we may be interested in gas-phase reactions occurring in ambient indoor air at or near standard temperature and pressure. In such scenarios, one finds the relevant potential energy to be the Gibbs, given by

$$G = U - TS + PV, \tag{6.1}$$

which is minimized in equilibrium under constant temperature and pressure.

This equation leads to the convenient thermodynamic identity

$$dG = -SdT + VdP + \sum_i \mu_i dN_i \tag{6.2}$$

from which we may identify the *chemical potential* of the i^{th} species as

$$\mu_i = \left(\frac{\partial G}{\partial N_i}\right)_{T,P,N_{j\neq i}}. \tag{6.3}$$

The fact that each μ_i depends only on intensive state variables allows us to further simplify the relationship by considering what would happen were we to gradually increase the size of the system while maintaining the values of intensive parameters T, P, μ_i. The result is G must increase in direct proportion to the increase in each N_i, that is:

$$G = \sum_i \mu_i N_i. \tag{6.4}$$

From equation 6.4 it is clear that the μ_i can be understood as molecular *potentials* (i.e. chemical energy per molecule) in analogy to the notion of electric potential as a energy per unit charge.

For an ideal gas consisting of a single component we can combine equation 6.4 together with the identity

$$V = \left(\frac{\partial G}{\partial P}\right)_{T,N} \tag{6.5}$$

to obtain

$$\left(\frac{\partial \mu}{\partial P}\right)_{T,N} = \left(\frac{\partial}{\partial P}\frac{G}{N}\right)_{T,N} = \frac{1}{N}\left(\frac{\partial G}{\partial P}\right)_{T,N} = \frac{V}{N} = \frac{kT}{P} \tag{6.6}$$

so that by integration from a reference pressure, say $P_0 = 1$ atm, we obtain the handy expression for the chemical potential

$$\mu(T,P) = \mu(T,P_0) + kT\ln(P/P_0) \tag{6.7}$$

which we shall use again momentarily.

Returning now to the notion of chemical equilibrium, recall that we must have $dG = 0$ so that G is minimized. At constant temperature and pressure, this means

$$0 = dG = \cancel{-SdT} + \cancel{VdP} + \sum_i \mu_i dN_i = \sum_i \mu_i dN_i \tag{6.8}$$

and therefore, a generic reaction of the form

$$\nu_1 X_1 + \nu_2 X_2 + \cdots \leftrightarrow \nu_3 X_3 + \nu_4 X_4 \cdots \tag{6.9}$$

with chemical species X_i and stoichiometric coefficients ν_i satisfies the condition that

$$\nu_1\mu_1 + \nu_2\mu_2 + \cdots = \nu_3\mu_3 + \nu_4\mu_4 + \cdots . \tag{6.10}$$

If we now make use of equation 6.7 with the identification of $\mu^0 = \mu(T, P_0)$, then we obtain

$$\sum_i^{\text{products}} \nu_i\mu_i^0 + \nu_i kT\ln(P_i/P_0) = \sum_j^{\text{reactants}} \nu_j\mu_j^0 + \nu_j kT\ln(P_j/P_0). \tag{6.11}$$

collecting terms involving the μ_k^0 to one side and multiplying through by Avogadro's constant, we obtain

$$RT\ln\left(\frac{\prod_j^{\text{reactants}}\left(\frac{P_i}{P_0}\right)^{\nu_i}}{\prod_j^{\text{products}}\left(\frac{P_j}{P_0}\right)^{\nu_j}}\right) = R\left(\sum_j^{\text{reactants}} \nu_j\mu_j^0 - \sum_i^{\text{products}} \nu_i\mu_i^0\right) = \Delta G^0 \tag{6.12}$$

so that by exponentiation, we arrive at the simple expression:

$$\frac{\prod_j^{\text{products}}\left(\frac{P_j}{P_0}\right)^{\nu_j}}{\prod_i^{\text{reactants}}\left(\frac{P_i}{P_0}\right)^{\nu_i}} = \exp(-\Delta G^0/RT) \tag{6.13}$$

which through further application of the ideal gas law yields

$$\boxed{\frac{\prod_j^{\text{products}}(X_j)^{\nu_j}}{\prod_i^{\text{reactants}}(X_i)^{\nu_i}} = K_{\text{eq}}} \tag{6.14}$$

Here K_{eq} is a temperature dependent constant called the *equilibrium constant* for the reaction, and equation 6.14 is called the *law of mass action*. This expression indicates what we can expect to find if we allow our reactive system to proceed far enough to reach equilibrium. We shall later utilize this expression to perform a thermodynamic *sanity check* as is described in (Boldi, 1994).

6.1.3 Reaction Rate Laws

Having established the expected behavior at equilibrium, our task now is to establish the correct dynamical laws describing the variety of reactions which take place. To begin, let us consider again a generic chemical reaction of the form

$$\nu_1 X_1 + \nu_2 X_2 + \cdots \longrightarrow \nu_3 X_3 + \nu_4 X_4 + \cdots \tag{6.15}$$

To describe the dynamical process of a reaction, we can introduce a parameter ξ called the *reaction extent* such that at any time we have

$$\xi(t) = \frac{|N_i(t) - N_i(0)|}{\nu_i} \tag{6.16}$$

where $N_i(t)$ is the number of the i^{th} species and ν_i is the stoichiometric coefficient. The reaction rate is then easily understood as the rate of change of the reaction extent,

$$r := \frac{d\xi}{dt} = \frac{1}{\nu_i}\left|\frac{dN_i(t)}{dt}\right|. \tag{6.17}$$

Manipulating this expression to introduce the volume then leads us to

$$r(t) = \frac{1}{\nu_i}\left|\frac{dN_i}{dt}\frac{V}{V}\right| = \frac{V}{\nu_i}\left|\frac{d[X_i]}{dt}\right| \tag{6.18}$$

where $[X_i]$ denotes the concentration (number density) of the i^{th} species participating in the reaction. All this is to say that upon rearranging the expression, we find

$$\left|\frac{d[X_i]}{dt}\right| = \nu_i \frac{r(t)}{V} = \nu_i v, \tag{6.19}$$

or in words, the absolute change in concentration of the i^{th} species as a function of time is proportional the quantity $v = r(t)/V$ (called the reaction *velocity*) scaled by the stoichiometric coefficient ν_i of the i^{th} species. For the purposes of our modeling tasks, we must now establish appropriate forms for the reaction velocity in terms of the relevant thermodynamic variables (i.e. temperature, and pressure) and constituent concentrations.

To begin, let us consider a bimolecular reaction

$$A + B \longrightarrow \text{Products.} \tag{6.20}$$

The simplest approach to modeling the reaction velocity for reactions of this type is to make the assumption that *each molecular collision leads to a reaction.* From this perspective, should then be able to derive the reaction velocity using the established statistics of molecular velocities together with appropriate data for the size of each reactant.

Let us treat reacting species as *hard spheres* of radii r_A and r_B respectively, or in other words, the molecules are spheres which only interact by contact (no long range interactions). Then a collision will occurs if the separation d_{AB} satisfies

$$d_{AB} = r_A + r_B \tag{6.21}$$

NOTE: Add image of hard spheres here

To start, let's consider collisions where B are stationary and A is seen to move with velocity $\mathbf{v}_A$.

NOTE: Add image of hard spheres forming cylinder here

Then during a time dt, the molecule A sweeps out a cylindrical volume

$$dV = \pi d_{AB}^2 v_A dt \tag{6.22}$$

in which collisions may occur. Supposing we have a density N_B/V of species B, then the collision rate (collisions per second) for a single particle of A will be

$$\frac{dV}{dt}\frac{N_B}{V} = \frac{\pi d_{AB}^2 v_A N_B}{V} \tag{6.23}$$

and therefore, the reaction velocity for N_A particles with average velocity $\overline{\mathbf{v}}_A$ is

$$v = \frac{\pi d_{AB}^2 \overline{v}_A N_A N_B}{V^2} \tag{6.24}$$

if instead particles of both A and B move relative to each other, then their *relative* velocity is given by the law of cosines:

$$v_{AB}^2 = v_A^2 + v_B^2 - 2v_A v_B \cos(\theta) \tag{6.25}$$

which, since all directions are equally probable, yields an average relative velocity of

$$\overline{v_{AB}}^2 = \sqrt{\overline{v_A}^2 + \overline{v_B}^2}. \tag{6.26}$$

The relative velocity describes the motion of the reduced mass $\mu = m_A m_B/(m_A + m_B)$ and therefore under the standard Maxwellian velocity distribution, leads to

$$\overline{v_{AB}} = \sqrt{\frac{8kT}{\pi\mu}} \tag{6.27}$$

Combining everything together yields the reaction velocity

$$v = \pi d_{AB}^2 \frac{N_A}{V}\frac{N_B}{V}\sqrt{\frac{8kT}{\pi\mu}} = \pi d_{AB}^2 [A][B]\sqrt{\frac{8kT}{\pi\mu}} \tag{6.28}$$

which we might further simplify as

$$v = k[A][B] \tag{6.29}$$

$$k = \pi d_{AB}^2 \sqrt{\frac{8kT}{\pi\mu}} = \sigma\sqrt{\frac{8k_B T}{\pi\mu}} \tag{6.30}$$

where k is called the *reaction rate coefficient* and σ is the *reaction cross section.*

Excellent! We have discovered a couple of key features for the reaction velocity, namely, that it depends on a polynomial combination of reactant concentrations, *and* that there is clear temperature dependence due to the relationship between molecular velocities and temperature.

There are, however, some obvious limitations.

1. Not all collisions occur with orientations favorable for reaction (i.e. the hard sphere model isn't realistic for species).

2. Not all collisions will have enough enough energy for the reaction to proceed.

These limitations were well known, and in particular, Arrhenius suggested a competing function based on empirical studies of with

$$k = A\exp(-\alpha/T) \tag{6.31}$$

where α is some constant that depends on the reaction taking place. We can address the first point in a *hand-wavy* manner by simply including a geometric correction factor, $g \leq 1$, to account for the distribution of favorable orientations. To address the second point, it is worth establishing some minimal energy required for the reaction, E_a, by examining our Maxwellian distribution in closer detail. As we shall see, this will allow us to recover the exponential dependence suggested by Arrhenius.

The velocities $\mathbf{v}_A$ and $\mathbf{v}_B$ of each colliding pair of reactants define a plane. Therefore, for ease of calculation, we approximate the distribution of velocities near the collision site by the 2-dimensional Maxwellian speed distribution:

$$f(v) = \frac{\mu}{k_B T} v \exp\left(-\frac{\mu v^2}{k_B T}\right) \tag{6.32}$$

so that the fraction of particles with speed in the range $[v, v+dv]$ is

$$\frac{dN(v)}{N_{tot}} = f(v)dv = \frac{\mu}{k_B T} v \exp\left(-\frac{\mu v^2}{k_B T}\right) dv. \tag{6.33}$$

For an ideal gas under no external forces, we may identify the energy $\epsilon = \mu v^2/2$ so that $d\epsilon = \mu v dv$. Therefore, in energy space, this ratio becomes

$$\frac{dN(\epsilon)}{N_{tot}} = \frac{1}{k_B T}\exp(-\epsilon/k_B T)d\epsilon \tag{6.34}$$

If E_a is the minimum (*activation*) Energy required to engage the reactants, then the ratio of reactants with sufficient energy for the reaction to proceed is determined by integration to be

$$\left.\frac{N(\epsilon)}{N_{tot}}\right|_{\epsilon > E_a} = \int_{\mathbb{E}_a}^{\infty} \frac{1}{k_B T}\exp(-\epsilon/k_B T)d\epsilon = \exp(-E_a/k_b T) \tag{6.35}$$

so that we may justify an additional correction to our reaction rate coefficient

$$k = g\pi d_{AB}^2 \sqrt{\frac{8k_B T}{\pi \mu}} \exp(-E_a/k_B T) \tag{6.36}$$

The final augmentation we can perform without before leaving collision theory behind is to observe that the cross-section σ was treated independently from the requirement of a minimum activation energy. With that in mind, suppose instead that a reaction cross section *only makes sense* if the reactants have the required minimum energy, that is

$$\sigma(\epsilon) = \begin{cases} \pi d_{AB}^2 & \epsilon > E_A \\ 0 & \text{otherwise} \end{cases} \tag{6.37}$$

If this is true, we can not keep the cross section outside of the ensemble average.

Allowing ourselves to utilize the full, three-dimensional Maxwellian speed distribution, we have

$$\begin{aligned} k &= \int_0^\infty \sigma(v) v f(v) dv \\ &= 4 \left(\frac{\mu}{2\pi k_B T} \right)^{3/2} \int_0^\infty \sigma(v) v \cdot v^2 \exp\left(-\frac{\mu v^2}{2k_B T} \right) dv \end{aligned} \tag{6.38}$$

which in terms of energy yields

$$\begin{aligned} k(T) &= \left(\frac{1}{\pi \mu} \right)^{1/2} \left(\frac{2}{k_B T} \right)^{3/2} \int_0^\infty \epsilon \sigma(\epsilon) \exp(-\epsilon/k_B T) d\epsilon \\ &= \left(\frac{1}{\pi \mu} \right)^{1/2} \left(\frac{2}{k_B T} \right)^{3/2} \int_{E_a}^\infty \epsilon \pi d_{AB}^2 \exp(-\epsilon/k_B T) d\epsilon \\ &= \pi d_{AB}^2 \sqrt{\frac{8k_B T}{\pi \mu}} \left(1 + \frac{E_a}{k_B T} \right) \exp(-E_a/k_B T) \end{aligned} \tag{6.39}$$

In summary, we have established that for bimolecular reactions, the (simple) theory of molecular collisions allows us to model the reaction velocity by

$$v = k(T)[A][B] \tag{6.40}$$

$$k(T) \approx \pi d_{AB}^2 \sqrt{\frac{8k_BT}{\pi\mu}} \left(1 + \frac{E_a}{k_BT}\right) \exp(-E_a/k_BT) \tag{6.41}$$

To derive a more accurate form for the rate coefficient k, we can result to PES, simulation, and statistical sampling techniques (Partridge et al., 1993; Wu et al., 2006, for example). For our purposes, it is enough to have justified the kinds of functional dependence on temperature we can expect to encounter when when collating available data on rate coefficients in the present literature.

6.2 Summary of Chemical Mechanism Kinetics

With this justification in hand, we are now ready to describe the chemical kinetics for each of the reaction types we will utilize in our chemical mechanism. We have already introduced bimolelcular reactions which involve the collision of two chemical species. These are reactions of the form

$$\mathrm{A} + \mathrm{B} \longrightarrow \text{Products} \tag{6.42}$$

The next relevant reaction type we will utilize are so called *trimolecular* (also, termolecular) reactions. These require the collision of three species such that

$$\mathrm{A} + \mathrm{B} + \mathrm{M} \longrightarrow \text{Products} \tag{6.43}$$

where M is a placeholder for any *third* molecule.

The final type of reaction we will consider is photolysis for which we may write

$$\mathrm{A} + h\nu \longrightarrow \text{Products.} \tag{6.44}$$

Bimolecular and trimolecular reactions have rate constants that can, in general, be written as

$$k(T,P) = f(T,P)\exp(-E_a/k_BT) \tag{6.45}$$

where E_a is the activation energy, and the function $f(T,P)$ encodes extra temperature and pressure dependence for each particular reaction, as in the example provided by collision theory in equation 6.41.

The reaction rate for photolysis reactions is (unsurprisingly) called the *photolysis rate* and is a function of three key parameters: The incident spectral irradiance, I measured in $\text{Photonts} \cdot \text{s}^{-1} \cdot \text{cm}^{-2} \cdot \text{nm}^{-1}$ and measures the effective photon flux through a cross section of area per wavelength. The second key parameter is the absorption cross section σ measured in cm^2. And the third is the quantum yield, Φ which is unitless (King, 2013). Together these three quantities combine to give a photolysis rate of

$$k(T) = \int_0^\infty I(\lambda)\sigma(T,\lambda)\Phi(T,\lambda)d\lambda \tag{6.46}$$

By way of analogy, we can think of the spectral irradiance as darts being thrown at a target. The absorption cross section, σ, is therefore the effective size of the dartboard, and the quantum yield Φ is the chance that a dart thrown at the target actually sticks into the board. We should note that the photolysis rate is often denoted by j, however we have chosen to use k to simplify writing the kinetic model for the *entire* reactive system.

Suppose we have a collection of M reactants in a mechanism consisting of N-many reactions of each of the above-mentioned types. The time dependent concentrations of each species, $u_i(t)$ can therefore be written as

$$\begin{cases} \dfrac{du_i}{dt}(t) = -\sum_j r_j S_{ij} \\ r_j = k_j \prod\limits_{\ell}^{S_{\ell j}>0} u_\ell^{S_{ij}} \end{cases} \tag{6.47}$$

where S_{ij} is the so-called *Stoichiometry matrix* whose entries are the signed stoichiometric coefficients such that S_{ij} is positive if species i appears as a reactant in reaction j, and negative if it appears as a product.

In order to simulate such a mechanism, we will need to provide functional forms for each reaction rate coefficient function, k_j. In the next section, we describe our methodology for doing this using absorption cross section and quantum yield data scraped from publicly available scientific databases together with real-time irradiance spectra captured using a high resolution UV-NIR spectrometer.

6.3 Characterization of Photolysis

To estimate the photolysis rates for photochemical reactions we must first collect data for the absorption cross section and quantum yields of all relevant chemical species. The MPI-Mainz UV/VIS Spectral Atlas of Gaseous Molecules of Atmospheric Interest contains experimental data meticulously gathered from thousands of research publications for a plethora of chemical species (Keller-Rudek et al., 2013). We scraped this database for all absorption cross section and quantum yield data provided at a variety of temperatures which we then use to generate training data in order to fit a Gaussian Process Regression model to estimate σ and Φ as a function of temperature. For the remainder of this section, we will describe the machine learning methodology for the particular case of ozone, O_3, and its associated photocatalytic dissociation $O_3 \longrightarrow O(1D) + O_2$.

6.3.1 Absorption Cross Sections

The collected absorption cross section data ozone are illustrated in a log-plot in Figure 6.2. As we can see, the bulk of the available data are for temperatures near the range $292 - 295$ Kelvin with a few sparse records at higher and lower temperatures. Because the Gaussian Process Regression model requires the Cholseky decomposition of a kernel matrix of size

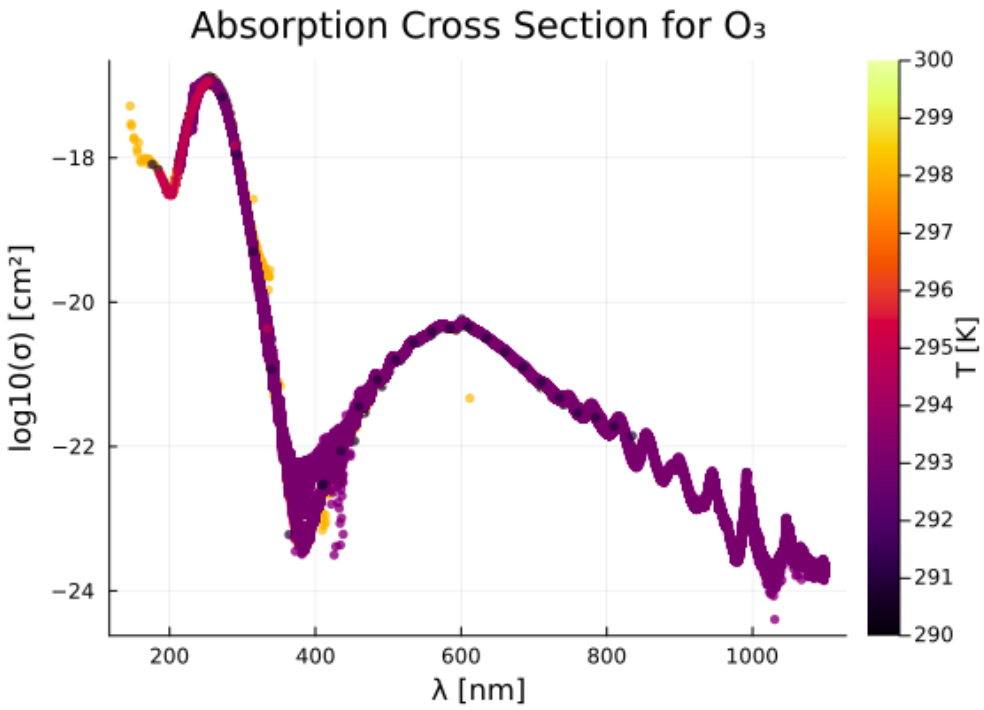


Figure 6.2: The collected absorption cross section data for Ozone, O_3, across a variety of temperatures.

$N \times N$ (N being the number of data records), we first perform a representative sub-sampling to generate a suitable train/test partition. The distribution of these records is illustrated in Figure 6.3.

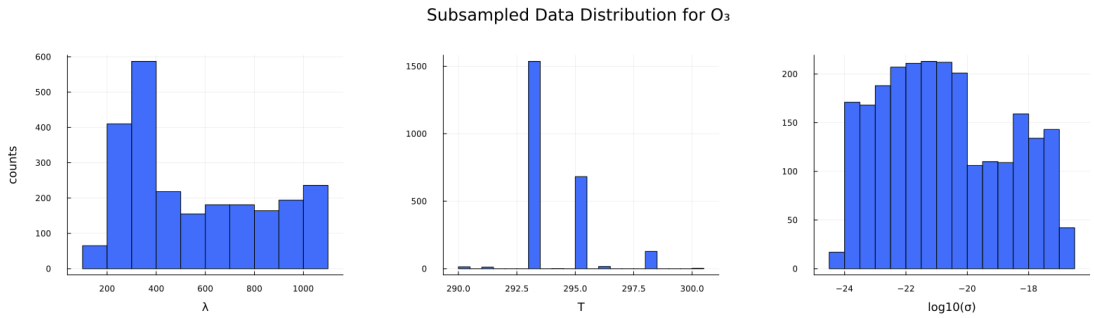


Figure 6.3: The distribution of subsampled O_3 cross section data

Using this dataset, we train a Gaussian Process Regression model to estimate $\log(\sigma)$. We fit the logarithm in order to make sure that the model learns the key absorption peaks since

the values of σ are observed to vary by multiple orders of magnitude. We chose to use a squared exponential kernel with hyperparameters for the variances as well as lengths scales for both λ and T. Additionally, the mean value for the GPR is set to a default value of -30 to prevent spurious values from being estimated in wavelength regions of few records. A scatterplot evaluating the resulting fit is given in Figure 6.4 From this, we can clearly

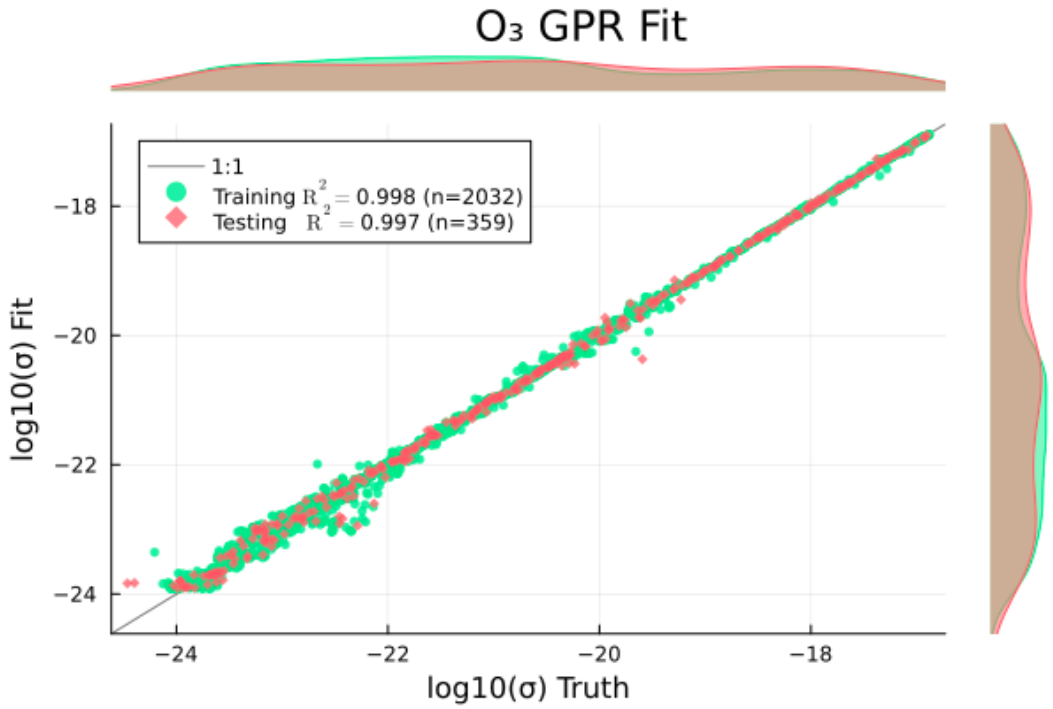


Figure 6.4: A scatter plot of the resulting GPR fit evaluated on the training set (green) and a holdout testing dataset (red).

see that the model has done a good job fitting the collected data. Next, we use the model to predict $\sigma(T, \lambda)$ for a variety of temperatures as illustrated in Figure 6.5 These curves illustrate the temperature dependence of our GPR fit. For temperatures above 298 K for which we had few records near 400 nm, the model has biased towards the mean value of -30 (i.e. an effective cross section of 0 cm^2. Because the data is sparse in this temperature range we should not implicitly trust the inferred values at high temperatures. To further illustrate this effect, we can plot the estimated cross section for two selected temperatures

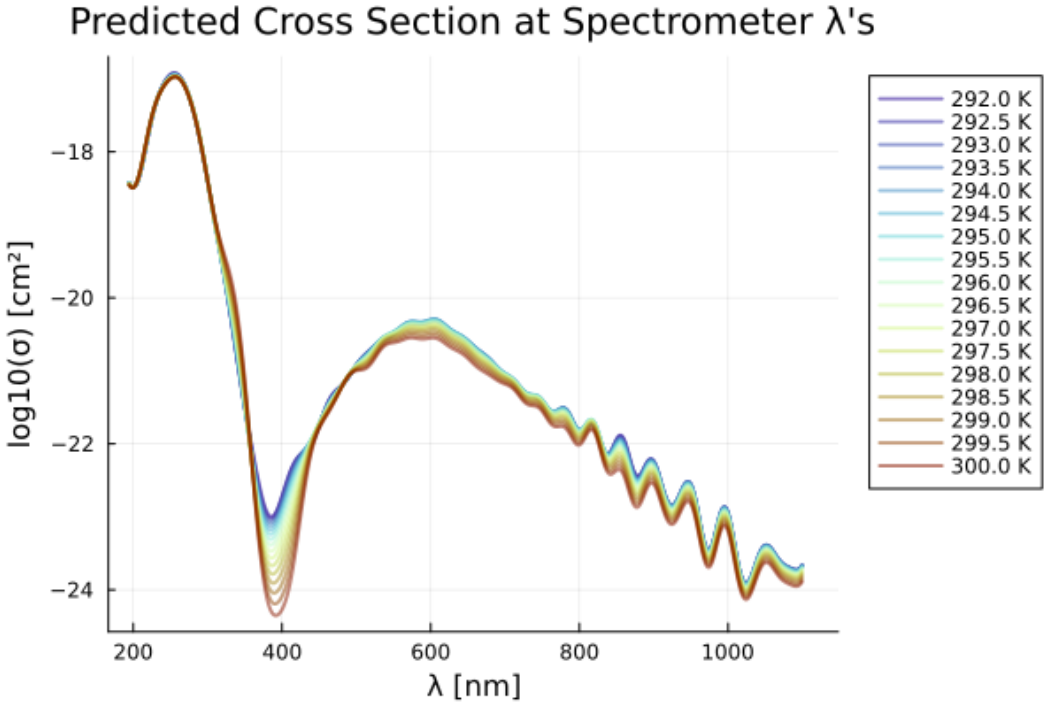


Figure 6.5: Predicted O_3 cross section as a function of temperature.

with uncertainty estimates obtained from the variance of the posterior distribution. An example of this is shown in Figure 6.6 below. The fit for 293 Kelvin is well represented in

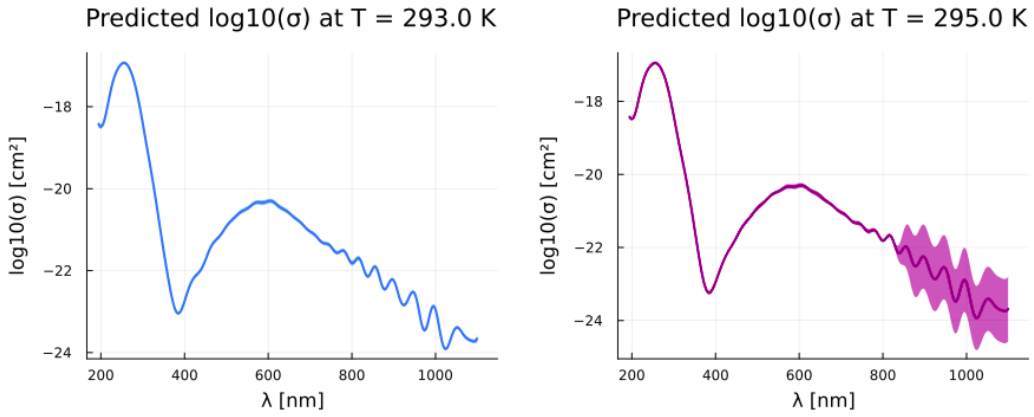


Figure 6.6: Predicted O_3 cross section for two temperatures with the associated uncertainty visualized. At $T = 293$ K, we have many data points and therefore the associated fit uncertainty is small. For $T = 295$ K, there are many fewer data points available above 800 nm leading to a larger fit uncertainty.

the training data and therefore the associated uncertainty is low. For a temperature of 295

K, there are few available measurements above 800 nm, and consequently, the uncertainty estimated by our GPR is increased in this region (as we should hope).

6.3.2 Quantum Yields

To fit the quantum yield data, the procedure is nearly identical to that of the absorption cross section, however there is often far less data available for the quantum yield. As we shall demonstrate for the case of $O_3 + h\nu \longrightarrow O(1D) + O_2$, for regions where we do not have enough data to establish a suitable fit, we default to either a value of 1 or 0 depending on the region of wavelength space. Figure 6.7 illustrates the available quantum yield data for this reaction. In the panel on the left, we see that the quantum yield appears to have little

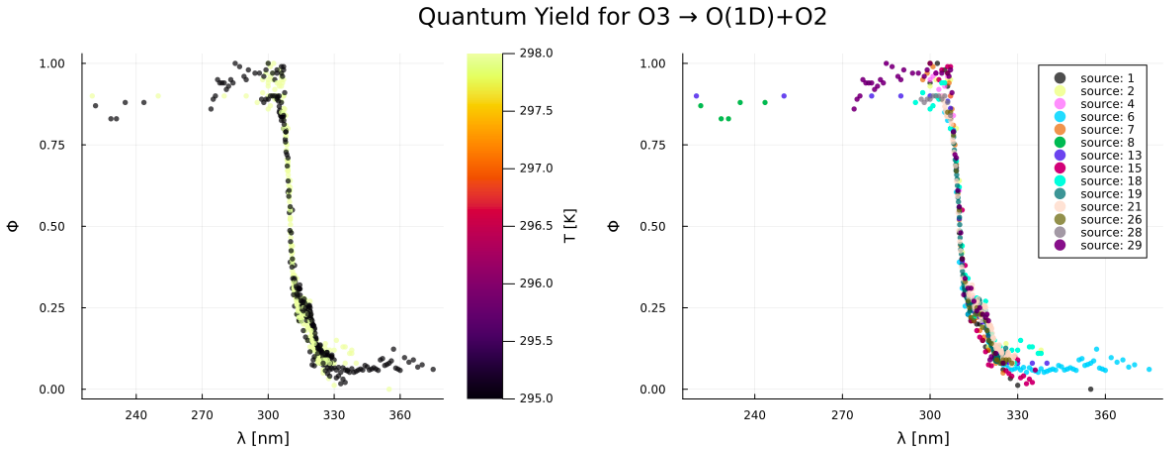


Figure 6.7: Data for the quantum yield of O_3. Sources of data are far more sparse than for the absorption cross section.

temperature dependence. On the right we color the data points according to the reference they were scraped from. The distribution of the the data is illustrated in Figure 6.8. Here we see that the available data come from two temperatures and most of the wavelength dependence is estimated near steep drop in Φ between 300 and 330 nm.

The resulting GPR fit is evaluated in Figure 6.9. It is obvious that we have far less data to fit, and as a consequence, we see in Figure 6.10 that the model estimates a quantum yield

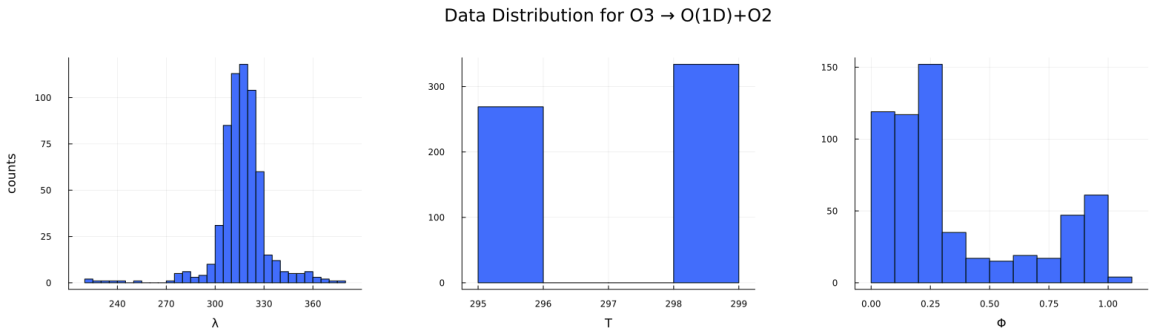


Figure 6.8: Distribution for the O_3 quantum yield data.

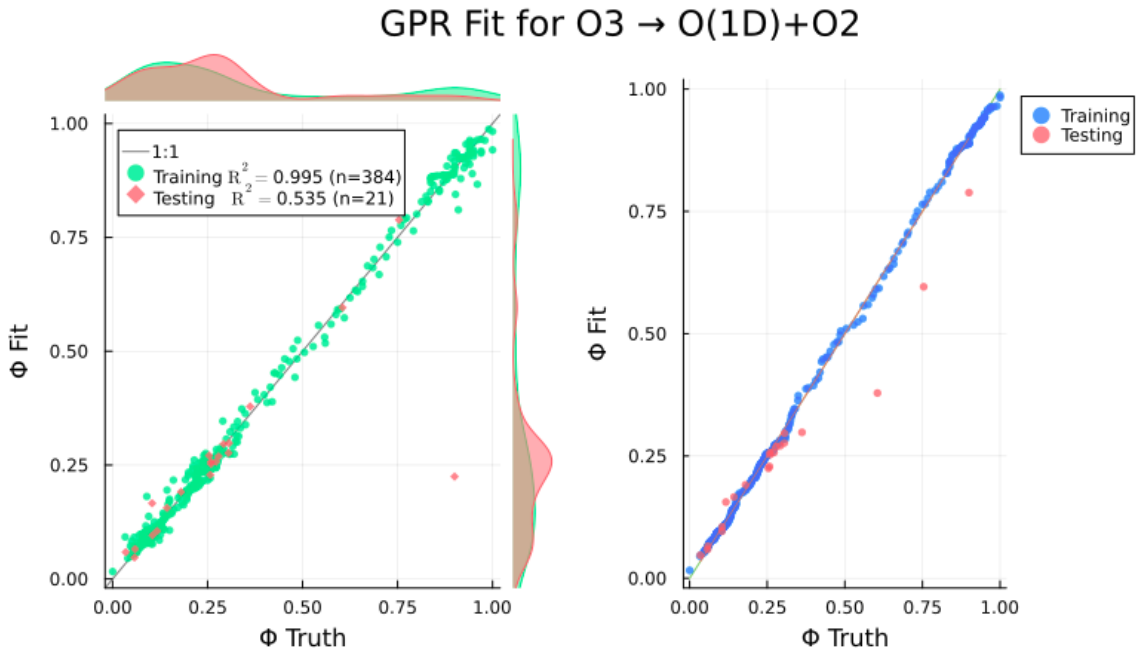
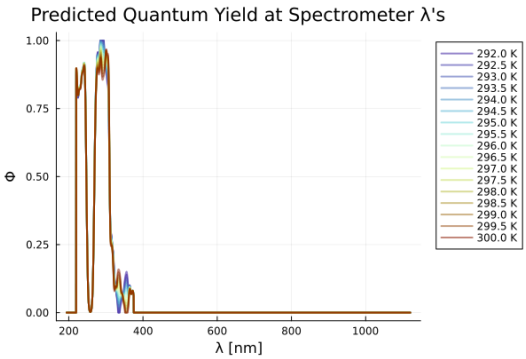
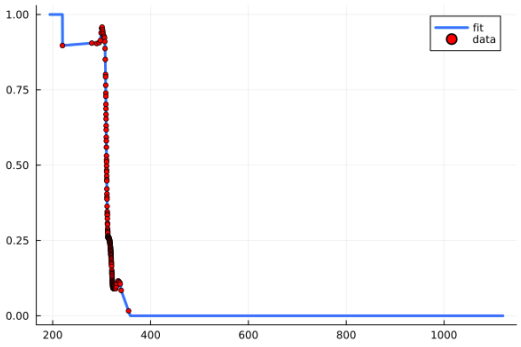


Figure 6.9: Resulting GPR fit for O_3 quantum yield data. Left: A scatter plot. Right: a quantile-quantile plot.

of zero between 250 and 270 nanometers where there is a massive data gap. As we can not justify this drop on physical grounds, we construct a hybrid fit by interpolating between the estimates in these regions. For wavelengths below the first available datapoint, we default to a value of 1. Similarly, for wavelengths beyond our last available record, we default to a value of 0.



(a) Predicted quantum yield for O_3



(b) Final quantum yield estimates

Figure 6.10: Estimated quantum yields as a function of temperature and the resulting final fit.

6.3.3 Irradiance Spectra and Photolysis Rate Determination

The last piece needed to estimate photolysis rates is the spectral irradiance. Using a calibrated Ocean Optics HR4000 UV-VIS spectrometer, we are able to measure the absolute irradiance spectrum every 15 seconds in units of $\mu\text{W} \cdot \text{cm}^{-2} \cdot \text{nm}^{-1}$. To obtain the desired units in terms of Photons, we utilize the plank relation $E(\lambda) = hc/\lambda$ to convert to $\text{Photons} \cdot s^{-1} \cdot \text{cm}^{-2} \cdot \text{nm}^{-1}$. A sample spectrum captured within an office building is shown in Figure 6.11.

6.3.4 Photolysis Rate Determination

Finally, using numerical quadrature to integrate the measured irradiance spectrum together with the estimated absorption cross section and quantum yield, we are able to obtain the photolysis rate for each desired reaction. A comparison of all three quantities is provided in figure 6.12 from which we estimate a photolysis rate of 10^{-4} s.

6.4 Chemical Data Assimilation

In this section we present preliminary results from our chemical data assimilation model. The underlying chemical mechanism is an extension of *AutoChem*, a NASA release software

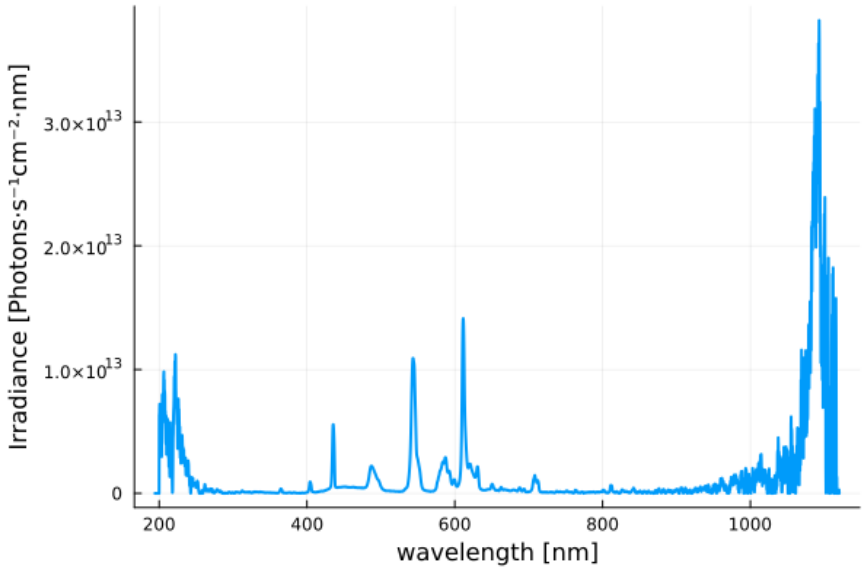


Figure 6.11: A plot of the spectral irradiance captured within a typical office building and averaged over 8 hours.

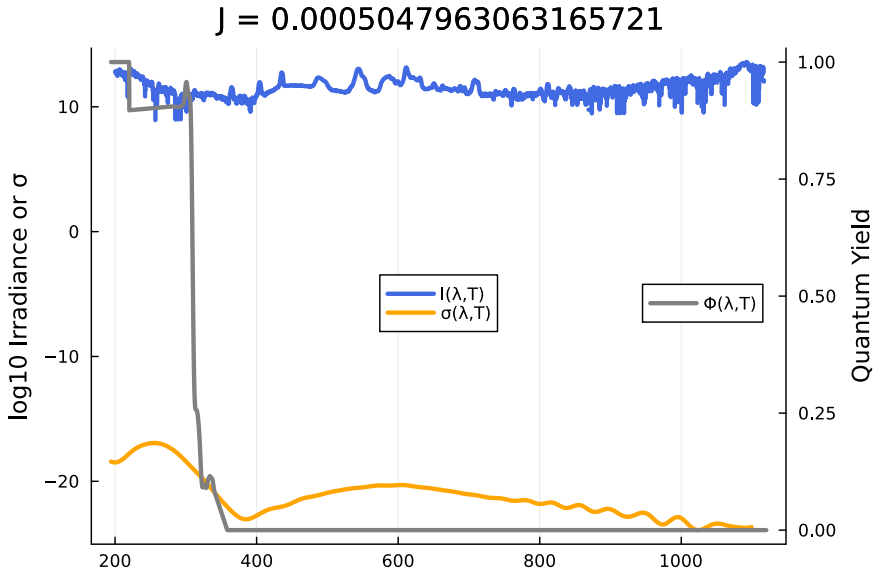


Figure 6.12: A comparison of the absorption cross section, quantum yield, and captured irradiance spectrum used to determine the photolysis rate for an O_3 photolysis channel.

from David Lary (Lary, 1999; Lary et al., 2003) augmented with reactions from the *Master Chemical Mechanism* (Saunders et al., 2003). Together this yields a reactive system of over 5000 chemical species and 16,000 chemical reactions. The preliminary results discussed in

this section have been carried out for a subset of the full mechanism consisting of 1000 critical reactions including relevant ion chemistry. For computational efficiency, we have developed the code using the Julia programming language to take advantage of the cutting-edge differential equations suite, `DifferentialEquations.jl` which provides optimized solvers for the large, stiff systems encountered in our chemical mechanism (Rackauckas and Nie, 2017).

To develop our data processing pipeline and the associated assimilation code for use with the HEART chamber described in chapter 2, we have collected an initial data set using a subset of our sensors to sample the ambient air within the room. These include a variety of gas sensors for CO_2, NO_x, O_3, H_2O_2, H_2O, and others in addition to ambient temperature and air pressure measurements. Data were collected over an 8 hour period providing over 1 million individual records.

The first step of the assimilation pipeline is to apply the 4d-var algorithm outlined in chapter 3 in order to estimate reasonable initial conditions for all species integrated in the mechanism. These values are initialized using typical atmospheric values provided by the Master Chemical Mechanism together with the initial samples from our instruments. Figure 6.13 A two stage training procedure was utilized starting with the ADAM optimizer which is known to converge rapidly. A second, final round of optimization is then performed using the BFGS optimizer to make sure we are not stuck in a local minimum.

Next, the Ensemble Kalman Filter is used to integrate the starting concentrations estimated by the 4d-var step forward in time. At each instance where we have a measurement, the EKF algorithm adjusts the current concentrations to minimize the difference between simulation and data with consideration for the uncertainties of each. This allows for the simulation to remain valid over long integration periods *and* makes it possible to infer the time-dependent concentrations of chemical species which we do not measure directly. Figure 6.14 illustrates the time series of H_2O2 as a volumetric mixing ratio in parts-per-trillion

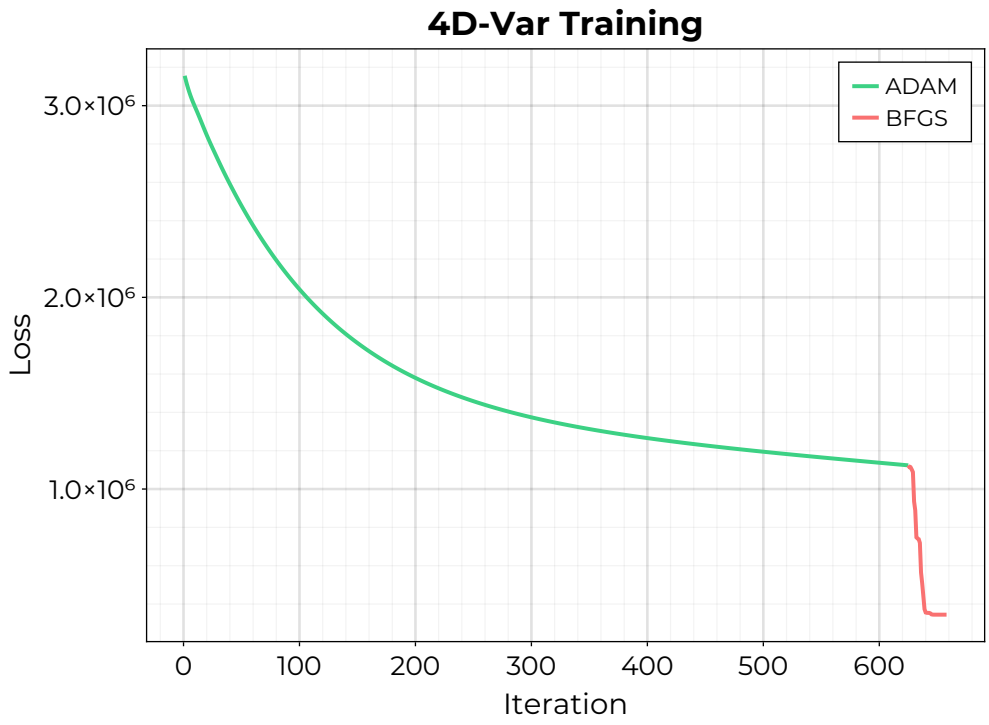


Figure 6.13: Training losses for 4d-var applied to our initial data collection.

(ppt). The gas analyzer producing the H_2O_2 concentrations are close to the detection limits as evidenced by the large error bars. However, due to the propagation of information between chemical species by the EKF algorithm, the estimated analysis uncertainty is actually reduced as visualized by the colored ribbon around each estimate.

Figure 6.15 illustrates similar time series obtained for the concentration of formaldehyde (HCHO). For this species, the analysis is able to follow the measurements however the assimilation has resulted in increased uncertainty estimates.

Finally, in Figure 6.16 we show the time series of concentrations for the highly reactive hydroxyl radical, OH. The hydroxyl radical is a highly reactive oxygen species which is difficult to measure directly due to its short lifetime (it tends to react with 50 molecular diameters). Using this assimilation framework we are able to infer the abundance of OH in the ambient room to be at the scale of a few parts-per-quadrillion.

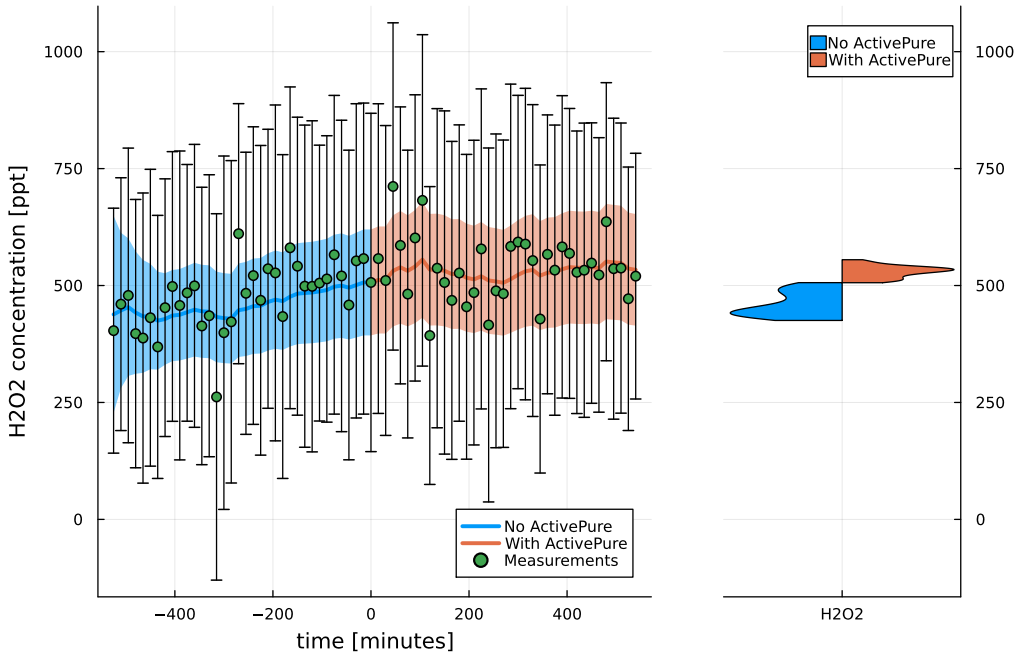


Figure 6.14: Assimilation results for H_2O_2.

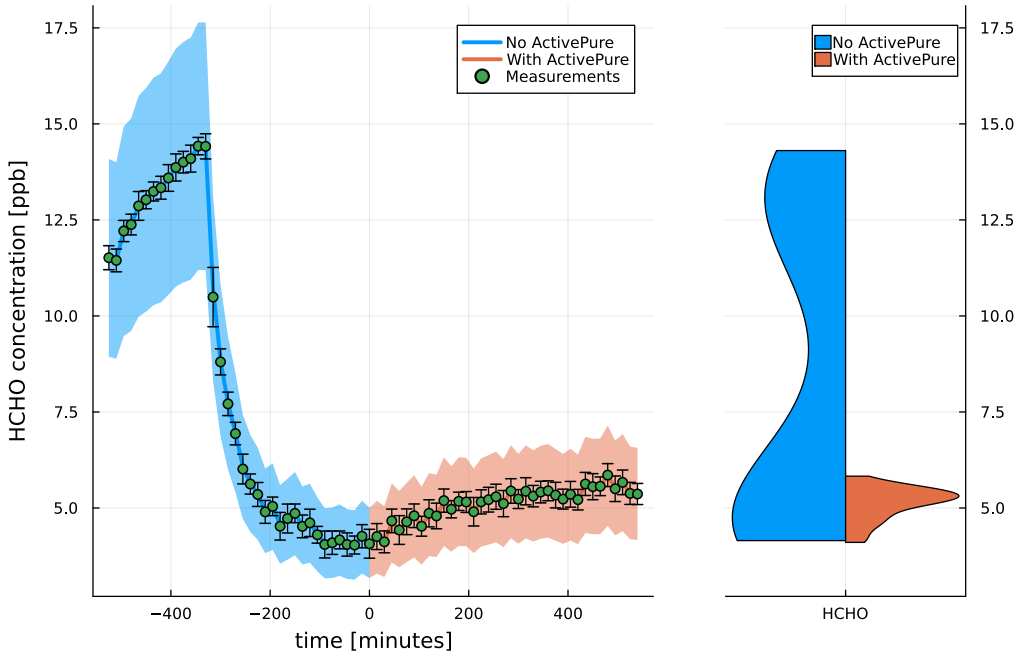


Figure 6.15: Assimilation results for formaldehyde.

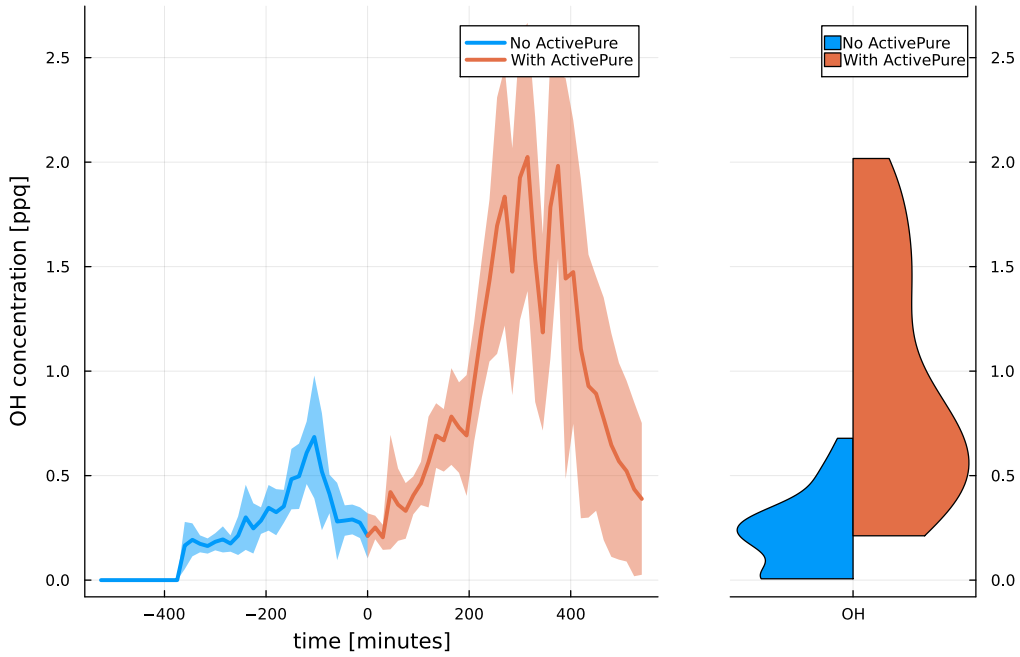


Figure 6.16: Assimilation results for the hydroxyl radical OH

The code for this project is made freely available at `https://github.com/john-waczak/AutoChem.jl`. A list of the reactions used in this preliminary investigation and their associated reaction rate coefficients are listed in Appendix A.

6.5 Next Steps

The next steps for this project include the following:

1. Extending the mechanism to include heterogeneous (mixed phase) reaction types.
2. Extending mechanism to include fixed rate surface deposition.
3. Development of a cd-EKF code for better uncertainty estimation.
4. Perform fixed temperature/pressure long time integration tests per the Boldi thesis to evaluate agreement with predictions from equilibrium thermodynamics.

5. Update reaction rates with recent JPL Kinetic evaluations.

CHAPTER 7

LIMITATIONS AND FUTURE WORK

CHAPTER 8

CONCLUSIONS

UPDATE REQUIRED!!!

APPENDIX A

CHEMICAL REACTION MECHANISMS

UPDATE REQUIRED!

A.1 Simple Ion Mechanism

Here we can include the automatically generated documentation for the Ion Mechanism and any others.

A.1.1 Bimolecular Reactions

#	Bimolecular Reaction	Reaction Rate Coeff
1	$Cl + HNO_3 \longrightarrow HCl + NO_3$	$k = (2.0\text{e-}16)$
2	$Br + H_2 \longrightarrow H + HBr$	$k = (8.0\text{e-}11)\exp(\text{-}8970.0/T)(T/298.0)^{0.43}$
3	$Br + O_3 \longrightarrow O_2 + BrO$	$k = (1.7\text{e-}11)\exp(\text{-}800.0/T)$
4	$Br + OH \longrightarrow O(^3P) + HBr$	$k = (4.56\text{e-}12)\exp(\text{-}8715.0/T)$

5	$\mathrm{Br} + \mathrm{CH_4} \longrightarrow \mathrm{HBr} + \mathrm{CH_3}$	$k = (7.8\text{e-}11)\exp(-76330.0/T)$
6	$\mathrm{Br} + \mathrm{H_2O} \longrightarrow \mathrm{OH} + \mathrm{HBr}$	$k = 0$
7	$\mathrm{Br} + \mathrm{HCO} \longrightarrow \mathrm{CO} + \mathrm{HBr}$	$k = (2.8\text{e-}10)\exp(-800.0/T)$
8	$\mathrm{Br} + \mathrm{HO_2} \longrightarrow \mathrm{O_2} + \mathrm{HBr}$	$k = (1.5\text{e-}11)\exp(-600.0/T)$
9	$\mathrm{Br} + \mathrm{N_2O} \longrightarrow \mathrm{N_2} + \mathrm{BrO}$	$k = (3.32\text{e-}10)\exp(-154800.0/T)$
10	$\mathrm{Br} + \mathrm{NO_3} \longrightarrow \mathrm{BrO} + \mathrm{NO_2}$	$k = (1.6\text{e-}11)$
11	$\mathrm{Br} + \mathrm{H_2O_2} \longrightarrow \mathrm{HBr} + \mathrm{HO_2}$	$k = (1.0\text{e-}11)\exp(-3000.0/T)$

12	$\mathrm{Br + HCHO \longrightarrow HBr + HCO}$	$k = (1.7\text{e-}11)\exp(-800.0/T)$
13	$\mathrm{Br + CH_3OH \longrightarrow HBr + HO_2 + HCHO}$	$k = (7.59\text{e-}13)\exp(-3119.0/T)$
14	$\mathrm{Br + OClO \longrightarrow BrO + ClO}$	$k = (2.6\text{e-}11)\exp(-1300.0/T)$
15	$\mathrm{Br + Cl_2O_2 \longrightarrow BrCl + ClOO}$	$k = (3.0\text{e-}12)$
16	$\mathrm{Br + CH_3OOH \longrightarrow HBr + CH_3O_2}$	$k = (2.63\text{e-}12)\exp(-1610.0/T)$
17	$\mathrm{Br + BrONO_2 \longrightarrow Br_2 + NO_3}$	$k = (6.3\text{e-}11)\exp(213.0/T)$
18	$\mathrm{BrO + BrO \longrightarrow Br + O_2 + Br}$	$k = (9.23\text{e-}13)\exp(250.0/T)$

19	$\mathrm{BrO} + \mathrm{BrO} \longrightarrow \mathrm{O_2} + \mathrm{Br_2}$	$k = (1.8\text{e-}13)\exp(250.0/T)$
20	$\mathrm{BrO} + \mathrm{ClO} \longrightarrow \mathrm{Br} + \mathrm{OClO}$	$k = (1.6\text{e-}12)\exp(430.0/T)$
21	$\mathrm{BrO} + \mathrm{ClO} \longrightarrow \mathrm{Br} + \mathrm{ClOO}$	$k = (2.9\text{e-}12)\exp(220.0/T)$
22	$\mathrm{BrO} + \mathrm{ClO} \longrightarrow \mathrm{O_2} + \mathrm{BrCl}$	$k = (5.8\text{e-}13)\exp(170.0/T)$
23	$\mathrm{BrO} + \mathrm{HO_2} \longrightarrow \mathrm{O_3} + \mathrm{HBr}$	$k = \exp(500.0/T)$
24	$\mathrm{BrO} + \mathrm{HO_2} \longrightarrow \mathrm{O_2} + \mathrm{HOBr}$	$k = (3.4\text{e-}12)\exp(540.0/T)$
25	$\mathrm{BrO} + \mathrm{NO_3} \longrightarrow$ $\mathrm{Br} + \mathrm{O_2} + \mathrm{NO_2}$	$k = (1.0\text{e-}12)$

26	$\mathrm{BrO} + \mathrm{CH_3O_2} \longrightarrow \mathrm{O_2} + \mathrm{Br} + \mathrm{CH_3O}$	$k = (3.23\text{e-}11)\exp(-332.0/T)$
27	$\mathrm{CH_3} + \mathrm{H_2O} \longrightarrow \mathrm{OH} + \mathrm{CH_4}$	$k = (1.2\text{e-}14)\exp(-62190.0/T)(T/298.0)^{2.9}$
28	$\mathrm{CH_3} + \mathrm{HCl} \longrightarrow \mathrm{Cl} + \mathrm{CH_4}$	$k = (3.88\text{e-}13)\exp(-9640.0/T)$
29	$\mathrm{CH_4} + \mathrm{H_2O} \longrightarrow \mathrm{CH_3} + \mathrm{H_2O_2}$	$k = (3.0\text{e-}13)\exp(-77740.0/T)$
30	$\mathrm{CH_4} + \mathrm{HCO} \longrightarrow \mathrm{CH_3} + \mathrm{HCHO}$	$k = (1.36\text{e-}13)\exp(-94200.0/T)(T/298.0)^{2.85}$
31	$\mathrm{CH_4} + \mathrm{CH_3O} \longrightarrow \mathrm{CH_3} + \mathrm{CH_3OH}$	$k = (2.6\text{e-}13)\exp(-37000.0/T)$

32	$CH_4 + CH_3O_2 \longrightarrow CH_3 + CH_3OOH$	$k = (3.0\text{e-}13)\exp(-77320.0/T)$
33	$CO + HO_2 \longrightarrow OH + CO_2$	$k = (2.5\text{e-}10)\exp(-98940.0/T)$
34	$CO + CH_3O \longrightarrow CH_3 + CO_2$	$k = (2.6\text{e-}11)\exp(-49390.0/T)$
35	$CO + NO_2 \longrightarrow NO + CO_2$	$k = (1.48\text{e-}10)\exp(-141420.0/T)$
36	$CO + NO_3 \longrightarrow NO_2 + CO_2$	$k = (4.0\text{e-}19)$
37	$Cl + H_2 \longrightarrow H + HCl$	$k = (3.7\text{e-}11)\exp(-2300.0/T)$
38	$Cl + O_3 \longrightarrow O_2 + ClO$	$k = (2.9\text{e-}11)\exp(-260.0/T)$

39	$\mathrm{Cl} + \mathrm{CH_4} \longrightarrow \mathrm{HCl} + \mathrm{CH_3}$	$k = (1.1\text{e-}11)\exp(\text{-}1400.0/T)$
40	$\mathrm{Cl} + \mathrm{H_2O} \longrightarrow \mathrm{OH} + \mathrm{HCl}$	$k = (2.79\text{e-}11)\exp(\text{-}72080.0/T)$
41	$\mathrm{Cl} + \mathrm{HCl} \longrightarrow \mathrm{H} + \mathrm{Cl_2}$	$k = (1.66\text{e-}7)\exp(\text{-}198590.0/T)$
42	$\mathrm{Cl} + \mathrm{HO_2} \longrightarrow \mathrm{O_2} + \mathrm{HCl}$	$k = (1.8\text{e-}11)\exp(170.0/T)$
43	$\mathrm{Cl} + \mathrm{HO_2} \longrightarrow \mathrm{OH} + \mathrm{ClO}$	$k = (4.1\text{e-}11)\exp(\text{-}450.0/T)$
44	$\mathrm{Cl} + \mathrm{N_2O} \longrightarrow \mathrm{N_2} + \mathrm{ClO}$	$k = (2.16\text{e-}10)\exp(\text{-}140150.0/T)$
45	$\mathrm{Cl} + \mathrm{NO_3} \longrightarrow \mathrm{ClO} + \mathrm{NO_2}$	$k = (2.4\text{e-}11)$

46	$\mathrm{Cl} + \mathrm{ClOO} \longrightarrow \mathrm{ClO} + \mathrm{ClO}$	$k = (1.2\text{e-}11)$
47	$\mathrm{Cl} + \mathrm{ClOO} \longrightarrow \mathrm{O_2} + \mathrm{Cl_2}$	$k = (2.3\text{e-}10)$
48	$\mathrm{Cl} + \mathrm{H_2O_2} \longrightarrow \mathrm{HCl} + \mathrm{HO_2}$	$k = (1.1\text{e-}11)\exp(\text{-}980.0/T)$
49	$\mathrm{Cl} + \mathrm{HCHO} \longrightarrow \mathrm{HCl} + \mathrm{HCO}$	$k = (8.2\text{e-}11)\exp(\text{-}34.0/T)$
50	$\mathrm{Cl} + \mathrm{HOCl} \longrightarrow \mathrm{OH} + \mathrm{Cl_2}$	$k = (2.5\text{e-}12)\exp(\text{-}130.0/T)$
51	$\mathrm{Cl} + \mathrm{CH_3O_2} \longrightarrow \mathrm{CH_3O} + \mathrm{ClO}$	$k = (7.7\text{e-}11)$
52	$\mathrm{Cl} + \mathrm{OClO} \longrightarrow \mathrm{ClO} + \mathrm{ClO}$	$k = (3.4\text{e-}11)\exp(160.0/T)$

53	$\mathrm{Cl} + \mathrm{Cl_2O_2} \longrightarrow \mathrm{Cl_2} + \mathrm{ClOO}$	$k = (1.0\text{e-}10)$
54	$\mathrm{Cl} + \mathrm{CH_3OCl} \longrightarrow \mathrm{Cl_2} + \mathrm{HO_2} + \mathrm{HCHO}$	$k = (4.9\text{e-}11)$
55	$\mathrm{Cl} + \mathrm{CH_3OOH} \longrightarrow \mathrm{HCl} + \mathrm{CH_3O_2}$	$k = (5.9\text{e-}11)$
56	$\mathrm{Cl} + \mathrm{BrONO_2} \longrightarrow \mathrm{NO_3} + \mathrm{BrCl}$	$k = (2.0\text{e-}11)\exp(329.0/T)$
57	$\mathrm{Cl} + \mathrm{ClONO_2} \longrightarrow \mathrm{Cl_2} + \mathrm{NO_3}$	$k = (6.5\text{e-}12)\exp(135.0/T)$
58	$\mathrm{Cl} + \mathrm{CH_3ONO_2} \longrightarrow \mathrm{NO_2} + \mathrm{HCl} + \mathrm{HCHO}$	$k = (1.3\text{e-}11)\exp(\text{-}1200.0/T)$
59	$\mathrm{Cl} + \mathrm{CH_3O_2NO_2} \longrightarrow \mathrm{NO_3} + \mathrm{HCl} + \mathrm{HCHO}$	$k = (4.0\text{e-}13)\exp(\text{-}640.0/T)$

60	$\mathrm{ClO} + \mathrm{ClO} \longrightarrow \mathrm{Cl} + \mathrm{ClOO}$	$k = (3.0\text{e-}11)\exp(\text{-}2450.0/T)$
61	$\mathrm{ClO} + \mathrm{ClO} \longrightarrow \mathrm{O_2} + \mathrm{Cl_2}$	$k = (1.0\text{e-}12)\exp(\text{-}1590.0/T)$
62	$\mathrm{ClO} + \mathrm{ClO} \longrightarrow \mathrm{Cl} + \mathrm{OClO}$	$k = (3.5\text{e-}13)\exp(\text{-}1370.0/T)$
63	$\mathrm{ClO} + \mathrm{HO_2} \longrightarrow \mathrm{O_2} + \mathrm{HOCl}$	$k = (4.8\text{e-}13)\exp(700.0/T)$
64	$\mathrm{ClO} + \mathrm{HO_2} \longrightarrow \mathrm{O_3} + \mathrm{HCl}$	$k = \exp(710.0/T)$
65	$\mathrm{ClO} + \mathrm{NO_3} \longrightarrow \mathrm{NO_2} + \mathrm{ClOO}$	$k = (4.7\text{e-}13)$
66	$\mathrm{ClO} + \mathrm{NO_3} \longrightarrow \mathrm{NO_2} + \mathrm{OClO}$	$k = 0$

67	$\mathrm{ClO} + \mathrm{HCHO} \longrightarrow \mathrm{HCO} + \mathrm{HOCl}$	$k = (1.0\text{e-}15)$
68	$\mathrm{ClO} + \mathrm{CH_3O_2} \longrightarrow \mathrm{O_2} + \mathrm{CH_3OCl}$	$k = (2.6\text{e-}13)\exp(263.0/T)$
69	$\mathrm{ClO} + \mathrm{CH_3O_2} \longrightarrow \mathrm{CH_3O} + \mathrm{ClOO}$	$k = (4.9\text{e-}12)\exp(\text{-}330.0/T)$
70	$\mathrm{ClO} + \mathrm{CH_3O_2} \longrightarrow$ $\mathrm{O_2} + \mathrm{Cl} + \mathrm{CH_3O}$	$k = (4.91\text{e-}12)\exp(\text{-}332.0/T)$
71	$\mathrm{H} + \mathrm{O_3} \longrightarrow \mathrm{OH} + \mathrm{O_2}$	$k = (1.4\text{e-}10)\exp(\text{-}480.0/T)$
72	$\mathrm{H} + \mathrm{Br_2} \longrightarrow \mathrm{Br} + \mathrm{HBr}$	$k = (1.84\text{e-}10)\exp(\text{-}558.0/T)(T/298.0)^{0.5}$

73	$H + CH_4 \longrightarrow H_2 + CH_3$	$k = (5.82\text{e-}13)\exp(\text{-}33630.0/T)(T/298.0)^{3.0}$
74	$H + H_2O \longrightarrow OH + H_2$	$k = (6.83\text{e-}12)\exp(\text{-}80810.0/T)(T/298.0)^{1.6}$
75	$H + HCl \longrightarrow H_2 + Cl$	$k = (1.32\text{e-}11)\exp(\text{-}14220.0/T)$
76	$H + HO_2 \longrightarrow O(^3P) + H_2O$	$k = (2.4\text{e-}12)$
77	$H + HO_2 \longrightarrow OH + OH$	$k = (7.2\text{e-}11)$
78	$H + HO_2 \longrightarrow H_2 + O_2$	$k = (5.6\text{e-}12)$

79	$H + N_2O \longrightarrow N_2 + OH$	$k = (9.793\text{e-}11)\exp(\text{-}56640.0/T)$
80	$H + NO_2 \longrightarrow OH + NO$	$k = (4.0\text{e-}10)\exp(\text{-}340.0/T)$
81	$H + HONO \longrightarrow H_2 + NO_2$	$k = (2.0\text{e-}11)\exp(\text{-}3700.0/T)$
82	$H + CH_3Br \longrightarrow CH_3 + HBr$	$k = (4.34\text{e-}11)\exp(\text{-}2646.0/T)$
83	$H_2 + Br_2 \longrightarrow HBr + HBr$	$k = (6.81\text{e-}9)\exp(\text{-}20430.0/T)$
84	$H_2O + N_2O_5 \longrightarrow HNO_3 + HNO_3$	$k = (2.0\text{e-}21)$

85	$H_2O + ClONO_2 \longrightarrow HOCl + HNO_3$	$k = (2.0\text{e-}21)$
86	$HCO + H_2O \longrightarrow OH + HCHO$	$k = (8.54\text{e-}13)\exp(\text{-}109300.0/T)(T/298.0)^{1.35}$
87	$HCO + HONO \longrightarrow NO_2 + HCHO$	$k = (2.0\text{e-}21)T^{2.37}\exp(\text{-}1940.0/T)$
88	$HCl + NO_3 \longrightarrow Cl + HNO_3$	$k = (5.0\text{e-}17)$
89	$HCl + N_2O_5 \longrightarrow HNO_3 + ClNO_2$	$k = (6.97\text{e-}21)$
90	$HCl + ClONO_2 \longrightarrow Cl_2 + HNO_3$	$k = (1.0\text{e-}20)$

91	$HCl + HO_2NO_2 \longrightarrow HOCl + HNO_3$	$k = (1.0e\text{-}21)$
92	$HO_2 + HO_2 \longrightarrow O_2 + H_2O_2$	$k = (2.2e\text{-}13)\exp(600.0/T)(1.0+(0.6*P/1013.25))$
93	$HO_2 + NO_3 \longrightarrow$ $O_2 + OH + NO_2$	$k = (2.15e\text{-}12)$
94	$HO_2 + NO_3 \longrightarrow O_2 + HNO_3$	$k = (2.15e\text{-}12)$
95	$HO_2 + HCHO \longrightarrow HCO + H_2O_2$	$k = (3.3e\text{-}12)\exp(-48800.0/T)$
96	$HO_2 + CH_3O_2 \longrightarrow O_2 + CH_3OOH$	$k = (3.42e\text{-}13)\exp(780.0/T)$

97	$HO_2 + CH_3O_2 \longrightarrow O_2 + H_2O + HCHO$	$k = (3.42\text{e-}14)\exp(780.0/T)$
98	$CH_3O_2 + CH_3O_2 \longrightarrow O_2 + CH_3O + CH_3O$	$k = (3.67\text{e-}14)\exp(365.0/T)$
99	$CH_3O_2 + CH_3O_2 \longrightarrow O_2 + CH_3OH + HCHO$	$k = (6.6\text{e-}14)\exp(365.0/T)$
100	$N + NO \longrightarrow O(^3P) + N_2$	$k = (3.1\text{e-}11)$
101	$N + O_2 \longrightarrow O(^3P) + NO$	$k = (4.4\text{e-}12)\exp(\text{-}3220.0/T)$
102	$N + O_3 \longrightarrow NO + O_2$	$k = (1.0\text{e-}16)$
103	$N + OH \longrightarrow H + NO$	$k = (3.8\text{e-}11)\exp(85.0/T)$

104	$N + NO_2 \longrightarrow O(^3P) + N_2O$	$k = (3.0e\text{-}12)$
105	$NO + O_3 \longrightarrow O_2 + NO_2$	$k = (1.8e\text{-}12)\exp(-1370.0/T)$
106	$NO + BrO \longrightarrow Br + NO_2$	$k = (8.8e\text{-}12)\exp(260.0/T)$
107	$NO + ClO \longrightarrow Cl + NO_2$	$k = (6.4e\text{-}12)\exp(290.0/T)$
108	$NO + HO_2 \longrightarrow OH + NO_2$	$k = (3.5e\text{-}12)\exp(250.0/T)$
109	$NO + N_2O \longrightarrow N_2 + NO_2$	$k = (2.51e\text{-}13)\exp(-206270.0/T)$
110	$NO + NO_3 \longrightarrow NO_2 + NO_2$	$k = (1.5e\text{-}11)\exp(170.0/T)$

111	$NO + CH_3O_2 \longrightarrow CH_3O + NO_2$	$k = (4.1e\text{-}12)\exp(180.0/T)$
112	$NO + CH_3O_2 \longrightarrow CH_3ONO_2$	$k = (4.1e\text{-}15)\exp(180.0/T)$
113	$NO + OClO \longrightarrow NO_2 + ClO$	$k = (3.4e\text{-}13)$
114	$NO + HNO_3 \longrightarrow NO_2 + HONO$	$k = (7.43e\text{-}21)$
115	$NO_2 + NO_3 \longrightarrow$ $O_2 + NO + NO_2$	$k = (4.5e\text{-}14)\exp(\text{-}1260.0/T)$
116	$NO_3 + HBr \longrightarrow Br + HNO_3$	$k = (1.0e\text{-}16)$
117	$NO_3 + HCHO \longrightarrow HCO + HNO_3$	$k = (5.8e\text{-}16)$

118	$NO_3 + CH_3OH \longrightarrow CH_3O + HNO_3$	$k = (1.3\text{e-}12)\exp(\text{-}2560.0/T)$
119	$NO_3 + CH_3O_2 \longrightarrow$ $O_2 + CH_3O + NO_2$	$k = (1.0\text{e-}12)$
120	$NO_3 + OClO \longrightarrow$ $O_2 + ClO + NO_2$	$k = (2.0\text{e-}15)$
121	$O(^1D) + CO \longrightarrow CO_2$	$k = (7.3\text{e-}11)$
122	$O(^1D) + H_2 \longrightarrow H + OH$	$k = (1.1\text{e-}10)$
123	$O(^1D) + N_2 \longrightarrow O(^3P) + N_2$	$k = (1.8\text{e-}11)\exp(107.0/T)$
124	$O(^1D) + NO \longrightarrow N + O_2$	$k = (5.0\text{e-}11)$

125	$O(^1D) + O_2 \longrightarrow O(^3P) + O_2$	$k = (3.2\text{e-}11)\exp(67.0/T)$
126	$O(^1D) + O_3 \longrightarrow$ $O(^3P) + O(^3P) + O_2$	$k = (1.2\text{e-}10)$
127	$O(^1D) + O_3 \longrightarrow O_2 + O_2$	$k = (1.2\text{e-}10)$
128	$O(^1D) + CH_4 \longrightarrow OH + CH_3$	$k = (1.13\text{e-}10)$
129	$O(^1D) + CH_4 \longrightarrow H_2 + HCHO$	$k = (7.5\text{e-}12)$
130	$O(^1D) + CH_4 \longrightarrow$ $H + H + HCHO$	$k = (3.0\text{e-}11)$
131	$O(^1D) + CO_2 \longrightarrow O(^3P) + CO_2$	$k = (7.4\text{e-}11)\exp(120.0/T)$

132	$O(^1D) + Cl_2 \longrightarrow O(^3P) + Cl_2$	$k = (7.0e\text{-}11)$
133	$O(^1D) + Cl_2 \longrightarrow Cl + ClO$	$k = (2.1e\text{-}10)$
134	$O(^1D) + H_2O \longrightarrow OH + OH$	$k = (2.2e\text{-}10)$
135	$O(^1D) + HBr \longrightarrow O(^3P) + HBr$	$k = (3.0e\text{-}11)$
136	$O(^1D) + HBr \longrightarrow OH + Br$	$k = (1.5e\text{-}10)$
137	$O(^1D) + HCl \longrightarrow O(^3P) + HCl$	$k = (1.35e\text{-}11)$
138	$O(^1D) + HCl \longrightarrow H + ClO$	$k = (3.6e\text{-}11)$

139	$O(^1D) + HCl \longrightarrow OH + Cl$	$k = (1.0e\text{-}10)$
140	$O(^1D) + N_2O \longrightarrow N + NO_2$	$k = (2.45e\text{-}13)$
141	$O(^1D) + N_2O \longrightarrow NO + NO$	$k = (7.2e\text{-}11)$
142	$O(^1D) + N_2O \longrightarrow N_2 + O_2$	$k = (4.4e\text{-}11)$
143	$O(^1D) + CF_2Cl_2 \longrightarrow Cl + Cl$	$k = (1.4e\text{-}10)$
144	$O(^3P) + H_2 \longrightarrow H + OH$	$k = (9.0e\text{-}18)$
145	$O(^3P) + O_3 \longrightarrow O_2 + O_2$	$k = (8.0e\text{-}12)\exp(\text{-}2060.0/T)$

146	$O(^3P) + OH \longrightarrow H + O_2$	$k = (2.3e\text{-}11)\exp(110.0/T)$
147	$O(^3P) + Br_2 \longrightarrow Br + BrO$	$k = (1.4e\text{-}11)$
148	$O(^3P) + BrO \longrightarrow O_2 + Br$	$k = (1.9e\text{-}11)\exp(230.0/T)$
149	$O(^3P) + CH_3 \longrightarrow H + HCHO$	$k = (1.4e\text{-}10)$
150	$O(^3P) + CH_4 \longrightarrow OH + CH_3$	$k = (8.32e\text{-}12)\exp(\text{-}35500.0/T)(T/298.0)^{1.56}$
151	$O(^3P) + ClO \longrightarrow Cl + O_2$	$k = (3.0e\text{-}11)\exp(70.0/T)$
152	$O(^3P) + H_2O \longrightarrow OH + OH$	$k = (1.85e\text{-}11)\exp(\text{-}71260.0/T)(T/298.0)^{0.946}$

153	$O(^3P) + HBr \longrightarrow OH + Br$	$k = (6.6e\text{-}12)\exp(-1540.0/T)$
154	$O(^3P) + HCl \longrightarrow OH + Cl$	$k = (1.0e\text{-}11)\exp(-3340.0/T)$
155	$O(^3P) + HO_2 \longrightarrow OH + O_2$	$k = (2.7e\text{-}11)\exp(224.0/T)$
156	$O(^3P) + NO_2 \longrightarrow O_2 + NO$	$k = (6.5e\text{-}12)\exp(120.0/T)$
157	$O(^3P) + NO_3 \longrightarrow O_2 + NO_2$	$k = (1.7e\text{-}11)$
158	$O(^3P) + BrCl \longrightarrow Cl + BrO$	$k = (2.2e\text{-}11)$
159	$O(^3P) + H_2O_2 \longrightarrow OH + HO_2$	$k = (1.4e\text{-}12)\exp(-2000.0/T)$

160	$O(^3P) + HCHO \longrightarrow OH + HCO$	$k = (3.4e\text{-}11)\exp(-1600.0/T)$
161	$O(^3P) + HOBr \longrightarrow OH + BrO$	$k = (1.4e\text{-}10)\exp(-430.0/T)$
162	$O(^3P) + HOCl \longrightarrow OH + ClO$	$k = (1.7e\text{-}13)$
163	$O(^3P) + HONO \longrightarrow OH + NO_2$	$k = (2.0e\text{-}11)\exp(-3000.0/T)$
164	$O(^3P) + OClO \longrightarrow O_2 + ClO$	$k = (2.5e\text{-}12)\exp(-950.0/T)$
165	$O(^3P) + CH_3Br \longrightarrow OH + Br$	$k = (7.6e\text{-}13)\exp(-890.0/T)$
166	$O(^3P) + HNO_3 \longrightarrow OH + NO_3$	$k = (3.0e\text{-}17)$

167	$O(^3P) + ClONO_2 \longrightarrow NO_2 + OClO$	$k = (1.5e\text{-}12)\exp(-800.0/T)$
168	$O(^3P) + ClONO_2 \longrightarrow NO_3 + ClO$	$k = (1.5e\text{-}12)\exp(-800.0/T)$
169	$O(^3P) + HO_2NO_2 \longrightarrow OH + O_2 + NO_2$	$k = (7.8e\text{-}11)\exp(-3400.0/T)$
170	$O_2 + CH_3 \longrightarrow OH + HCHO$	$k = (3.0e\text{-}16)$
171	$O_2 + CH_4 \longrightarrow CH_3 + HO_2$	$k = (6.7e\text{-}11)\exp(-238120.0/T)(T/298.0)^{1.56}$
172	$O_2 + HCO \longrightarrow CO + HO_2$	$k = (5.5e\text{-}12)$

173	$O_2 + CH_3O \longrightarrow HO_2 + HCHO$	$k = (7.2e\text{-}14)\exp(-1080.0/T)$
174	$O_3 + BrO \longrightarrow$ $Br + O_2 + O_2$	$k = (5.0e\text{-}15)$
175	$O_3 + CH_3 \longrightarrow O_2 + CH_3O$	$k = (5.1e\text{-}12)\exp(-210.0/T)$
176	$O_3 + CH_3 \longrightarrow$ $H + O_2 + HCHO$	$k = (5.1e\text{-}12)\exp(-210.0/T)$
177	$O_3 + ClO \longrightarrow O_2 + ClOO$	$k = (1.5e\text{-}17)$
178	$O_3 + ClO \longrightarrow O_2 + OClO$	$k = (1.0e\text{-}18)\exp(-4000.0/T)$
179	$O_3 + HO_2 \longrightarrow$ $OH + O_2 + O_2$	$k = (1.4e\text{-}14)\exp(-600.0/T)$

180	$O_3 + NO_2 \longrightarrow O_2 + NO_3$	$k = (1.2\text{e-}13)\exp(-2450.0/T)$
181	$O_3 + HONO \longrightarrow O_2 + HNO_3$	$k = (5.0\text{e-}19)$
182	$O_3 + Cl_2O_2 \longrightarrow$ $O_2 + ClO + ClOO$	$k = (1.0\text{e-}19)$
183	$OH + CO \longrightarrow H + CO_2$	$k = (1.5\text{e-}13)(1.0 + (0.6 * P/1013.25))$
184	$OH + H_2 \longrightarrow H + H_2O$	$k = (5.0\text{e-}12)\exp(-2000.0/T)$
185	$OH + O_3 \longrightarrow O_2 + HO_2$	$k = (1.9\text{e-}12)\exp(-1000.0/T)$
186	$OH + OH \longrightarrow O(^3P) + H_2O$	$k = (4.2\text{e-}12)\exp(-240.0/T)$

187	$OH + Br_2 \longrightarrow Br + HOBr$	$k = (1.2\text{e-}11)\exp(400.0/T)$
188	$OH + BrO \longrightarrow Br + HO_2$	$k = (7.425\text{e-}11)$
189	$OH + BrO \longrightarrow O_2 + HBr$	$k = (7.5\text{e-}13)$
190	$OH + CH_4 \longrightarrow H_2O + CH_3$	$k = (2.8\text{e-}14)T^{0.667}\exp(\text{-}1575.0/T)$
191	$OH + Cl_2 \longrightarrow Cl + HOCl$	$k = (1.4\text{e-}12)\exp(\text{-}900.0/T)$
192	$OH + ClO \longrightarrow Cl + HO_2$	$k = (9.35\text{e-}12)\exp(120.0/T)$
193	$OH + ClO \longrightarrow O_2 + HCl$	$k = (1.65\text{e-}12)\exp(120.0/T)$

194	$OH + HBr \longrightarrow Br + H_2O$	$k = (1.1e\text{-}11)$
195	$OH + HCl \longrightarrow Cl + H_2O$	$k = (2.6e\text{-}12)\exp(-350.0/T)$
196	$OH + HO_2 \longrightarrow O_2 + H_2O$	$k = (4.8e\text{-}11)\exp(250.0/T)$
197	$OH + N_2O \longrightarrow N_2 + HO_2$	$k = (3.8e\text{-}17)$
198	$OH + NO_3 \longrightarrow HO_2 + NO_2$	$k = (2.3e\text{-}11)$
199	$OH + H_2O_2 \longrightarrow H_2O + HO_2$	$k = (2.9e\text{-}12)\exp(-160.0/T)$
200	$OH + HCHO \longrightarrow H_2O + HCO$	$k = (8.8e\text{-}12)\exp(25.0/T)$

201	$OH + HOCl \longrightarrow ClO + H_2O$	$k = (3.0e\text{-}12)\exp(-500.0/T)$
202	$OH + HONO \longrightarrow H_2O + NO_2$	$k = (1.8e\text{-}11)\exp(-390.0/T)$
203	$OH + CH_3OH \longrightarrow H_2O + CH_3O$	$k = (5.0e\text{-}13)\exp(-380.0/T)$
204	$OH + OClO \longrightarrow O_2 + HOCl$	$k = (4.5e\text{-}13)\exp(800.0/T)$
205	$OH + ClNO_2 \longrightarrow NO_2 + HOCl$	$k = (3.5e\text{-}14)$
206	$OH + HNO_3 \longrightarrow H_2O + NO_3$	$k = 2.4e\text{-}14 \cdot \exp(460.0/T) + \frac{6.50e\text{-}34 \cdot M \cdot \exp(1335}{1 + \frac{6.50e\text{-}34 \cdot M \cdot \exp(1335.0}{2.7e\text{-}17 \cdot exp(2199.0/T}}$
207	$OH + CH_3OCl \longrightarrow Cl + H_2O + HCHO$	$k = (2.4e\text{-}12)\exp(-360.0/T)$

208	$OH + CH_3OOH \longrightarrow H_2O + CH_3O_2$	$k = (1.9e\text{-}12)\exp(190.0/T)$
209	$OH + CH_3OOH \longrightarrow OH + H_2O + HCHO$	$k = (1.0e\text{-}12)\exp(190.0/T)$
210	$OH + ClONO_2 \longrightarrow NO_3 + HOCl$	$k = (1.2e\text{-}12)\exp(-330.0/T)$
211	$OH + HO_2NO_2 \longrightarrow O_2 + H_2O + NO_2$	$k = (1.5e\text{-}12)\exp(360.0/T)$
212	$OH + CH_3ONO_2 \longrightarrow NO_2 + H_2O + HCHO$	$k = (4.0e\text{-}13)\exp(-845.0/T)$
213	$OH + CH_3O_2NO_2 \longrightarrow NO_3 + H_2O + HCHO$	$k = (1.0e\text{-}13)\exp(-640.0/T)$

214	$N_2^+ + O(^3P) \longrightarrow NO^+ + N$	$k = (1.4e\text{-}10)$
215	$N_2^+ + O_2 \longrightarrow O_2^+ + N_2$	$k = (6.0e\text{-}11)$
216	$N_2^+ + NO \longrightarrow NO^+ + N_2$	$k = (5.0e\text{-}10)$
217	$N_2^+ + CO_2 \longrightarrow CO_2^+ + N_2$	$k = (7.7e\text{-}10)$
218	$N_2^+ + CO \longrightarrow CO^+ + N_2$	$k = (7.4e\text{-}11)$
219	$N^+ + O_2 \longrightarrow O_2^+ + N$	$k = (2.8e\text{-}10)$
220	$N^+ + NO \longrightarrow NO^+ + N$	$k = (8.0e\text{-}10)$

221	$N^+ + CO_2 \longrightarrow CO_2^+ + N$	$k = (7.5e\text{-}10)$
222	$N^+ + CO_2 \longrightarrow CO^+ + NO$	$k = (2.5e\text{-}10)$
223	$O^+ + N_2 \longrightarrow NO^+ + N$	$k = (1.2e\text{-}12)$
224	$O^+ + O_2 \longrightarrow O_2^+ + O(^3P)$	$k = (1.2e\text{-}12)$
225	$O^+ + NO \longrightarrow NO^+ + O(^3P)$	$k = (1.1e\text{-}11)$
226	$O^+ + CO_2 \longrightarrow CO_2^+ + O(^3P)$	$k = (1.1e\text{-}9)$
227	$O^+ + CO_2 \longrightarrow O_2^+ + CO$	$k = (1.1e\text{-}9)$

228	$O_2^+ + NO \longrightarrow NO^+ + O_2$	$k = (4.4e\text{-}10)$
229	$Ar^+ + N_2 \longrightarrow N_2^+ + Ar$	$k = (2.5e\text{-}12)$
230	$Ar^+ + CO_2 \longrightarrow CO_2^+ + Ar$	$k = (4.2e\text{-}10)$
231	$Ar^+ + O_2 \longrightarrow O_2^+ + Ar$	$k = (5.5e\text{-}11)$
232	$Ar^+ + NO \longrightarrow NO^+ + Ar$	$k = (2.0e\text{-}10)$
233	$CO_2^+ + O_2 \longrightarrow O_2^+(CO_2)$	$k = (1.0e\text{-}10)$
234	$CO_2^+ + NO \longrightarrow NO^+ + CO_2$	$k = (2.0e\text{-}11)$

235	$CO_2^+ + CO \longrightarrow CO^+ + CO_2$	$k = (1.9e\text{-}12)$
236	$CO^+ + CO_2 \longrightarrow CO_2^+ + CO$	$k = (1.42e\text{-}9)$
237	$CO_2^+CO_2 + O_2 \longrightarrow O_2^+ + 2CO_2$	$k = (1.0e\text{-}10)$
238	$CO_2^+CO_2 + O_2 \longrightarrow$ $O_2^+(CO_2) + CO_2$	$k = (2.7e\text{-}11)$
239	$O_2^+(CO_2) + CO_2 \longrightarrow O_2^+ + 2CO_2$	$k = (2.4e\text{-}13)$
240	$O_2^+(CO_2) + H_2O \longrightarrow$ $O_2^+(H_2O) + CO_2$	$k = (1.1e\text{-}9)$
241	$O_2^+(CO_2)_2 + H_2O \longrightarrow$ $O_2^+(H_2O) + 2CO_2$	$k = (2.3e\text{-}9)$

242	$O_2^+(H_2O) + H_2O \longrightarrow$ $H_3O^+ + OH + O_2$	$k = (2.04e\text{-}10)$
243	$O_2^+(H_2O) + H_2O \longrightarrow H_3O^+OH + O_2$	$k = (9.96e\text{-}10)$
244	$O_2^+(H_2O)_2 + H_2O \longrightarrow$ $O_2^+(H_2O) + 2H_2O$	$k = (3.24e\text{-}10)$
245	$O_2^+O_2 + CO_2 \longrightarrow O_2^+(CO_2) + O_2$	$k = (4.0e\text{-}13)$
246	$O_2^+O_2 + H_2O \longrightarrow O_2^+(H_2O) + O_2$	$k = (1.7e\text{-}9)$
247	$H_3O^+OH + H_2O \longrightarrow H_3O^+(H_2O) + OH$	$k = (1.4e\text{-}9)$

248	$O(^3P) + e^- \longrightarrow O^-$	$k = (1.3e\text{-}15)$
249	$O_3 + e^- \longrightarrow O^- + O_2$	$k = (9.1e\text{-}12)\exp(-46.0/T)(T/300.0)^{-1.0}$
250	$O_3^- + CO_2 \longrightarrow CO_3^- + O_2$	$k = (5.5e\text{-}10)$
251	$CO_3^- + NO \longrightarrow NO_2^- + CO_2$	$k = (1.0e\text{-}11)$
252	$CO_3^- + NO_2 \longrightarrow NO_3^- + CO_2$	$k = (2.0e\text{-}10)$
253	$CO_4^- + NO \longrightarrow NO_3^- + CO_2$	$k = (4.8e\text{-}11)$
254	$CO_4^- + O(^3P) \longrightarrow CO_3^- + O_2$	$k = (1.4e\text{-}10)$

255	$CO_4^- + O_3 \longrightarrow O_3^- + CO_2 + O_2$	$k = (1.3\text{e-}10)$
256	$NO_2^- + O_3 \longrightarrow NO_3^- + O_2$	$k = (1.2\text{e-}10)$
257	$NO_3^- + CO_2 \longrightarrow CO_3^- + NO_2$	$k = (1.0\text{e-}11)$
258	$O_2^+O_2 + h\nu \longrightarrow O_2^+ + O_2$	$k = (0.13)$
259	$O_2^+(H_2O) + h\nu \longrightarrow O_2^+ + H_2O$	$k = (0.18)$
260	$O_2^+(H_2O)_2 + h\nu \longrightarrow O_2^+(H_2O) + H_2O$	$k = (0.27)$
261	$CO_3^- + h\nu \longrightarrow e^- + CO_2 + O(^3P)$	$k = (0.01)$

262	$CO_3^- + h\nu \longrightarrow e^- + CO_2$	$k = (0.076)$
263	$CO_4^- + h\nu \longrightarrow O_2^- + CO_2$	$k = (0.0028)$
264	$NO_2^- + h\nu \longrightarrow e^- + NO_2$	$k = (0.018)$
265	$NO_3^- + h\nu \longrightarrow O(^3P) + NO_2^-$	$k = (0.026)$
266	$NO_3^- + h\nu \longrightarrow$ $e^- + NO + O_2$	$k = (0.002)$
267	$O^- + h\nu \longrightarrow e^- + O(^3P)$	$k = (0.62)$
268	$O_2^- + h\nu \longrightarrow e^- + O_2$	$k = (0.17)$

269	$O_3^- + h\nu \longrightarrow e^- + O_3$	$k = (0.021)$
270	$O_3^- + h\nu \longrightarrow O^- + O_2$	$k = (0.21)$
271	$N_2^+ + e^- \longrightarrow 2N$	$k = (2.7e\text{-}7)$
272	$N_2^+ + O^- \longrightarrow N_2 + O(^3P)$	$k = (1.5e\text{-}7)$
273	$N_2^+ + O_2^- \longrightarrow N_2 + O_2$	$k = (1.5e\text{-}7)$
274	$N_2^+ + O_3^- \longrightarrow N_2 + O_3$	$k = (1.5e\text{-}7)$
275	$N_2^+ + NO_2^- \longrightarrow N_2 + NO_2$	$k = (1.5e\text{-}7)$

276	$N_2^+ + NO_3^- \longrightarrow N_2 + NO_3$	$k = (1.7e\text{-}7)$
277	$N_2^+ + CO_3^- \longrightarrow$ $N_2 + CO_2 + O(^3P)$	$k = (1.5e\text{-}7)$
278	$N_2^+ + NO_2^-(H2O)_2 \longrightarrow$ $N_2 + NO_2 + 2H_2O$	$k = (1.5e\text{-}7)$
279	$N_2^+ + CO_4^- \longrightarrow$ $N_2 + CO_2 + O_2$	$k = (1.5e\text{-}7)$
280	$O_2^+ + e^- \longrightarrow 2O(^3P)$	$k = (2.8e\text{-}7)(T/300.0)^{-0.63}$
281	$O_2^+ + O^- \longrightarrow O_2 + O(^3P)$	$k = (1.0e\text{-}7)(T/300.0)^{-0.5}$
282	$O_2^+ + O_2^- \longrightarrow 2O_2$	$k = (4.2e\text{-}7)(T/300.0)^{-0.5}$

283	$O_2^+ + O_3^- \longrightarrow O_2 + O_3$	$k = (4.0\text{e-}7)(T/300.0)^{-0.5}$
284	$O_2^+ + NO_2^- \longrightarrow O_2 + NO_2$	$k = (4.1\text{e-}7)(T/300.0)^{-0.5}$
285	$O_2^+ + NO_3^- \longrightarrow O_2 + NO_3$	$k = (1.3\text{e-}7)(T/300.0)^{-0.5}$
286	$O_2^+ + CO_3^- \longrightarrow$ $O_2 + CO_2 + O(^3P)$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
287	$O_2^+ + NO_2^-(H2O)_2 \longrightarrow$ $O_2 + NO_2 + 2H_2O$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
288	$O_2^+ + CO_4^- \longrightarrow 2O_2 + 2CO_2$	$k = (2.0\text{e-}7)$
289	$NO^+ + e^- \longrightarrow N + O(^3P)$	$k = (4.0\text{e-}7)(T/300.0)^{-1.0}$

290	$NO^+ + O^- \longrightarrow NO + O(^3P)$	$k = (4.9e\text{-}7)(T/300.0)^{-0.5}$
291	$NO^+ + O_2^- \longrightarrow NO + O_2$	$k = (6.0e\text{-}7)(T/300.0)^{-0.5}$
292	$NO^+ + O_3^- \longrightarrow NO + O_3$	$k = (1.0e\text{-}7)(T/300.0)^{-0.5}$
293	$NO^+ + NO_2^- \longrightarrow NO + NO_2$	$k = (1.0e\text{-}7)(T/300.0)^{0.5}$
294	$NO^+ + NO_3^- \longrightarrow NO_2 + NO_3$	$k = (1.3e\text{-}7)(T/300.0)^{-0.5}$
295	$NO^+ + CO_3^- \longrightarrow$ $NO + CO_2 + O(^3P)$	$k = (1.2e\text{-}7)(T/300.0)^{-0.5}$
296	$NO^+ + NO_2^-(H2O)_2 \longrightarrow$ $NO + NO_2 + 2H_2O$	$k = (1.0e\text{-}7)(T/300.0)^{-0.5}$

297	$NO^+ + CO_4^- \longrightarrow NO + CO_2 + O_2$	$k = (1.0\text{e-}7)(T/300.0)^{-0.5}$
298	$CO_2^+ + e^- \longrightarrow CO + O(^3P)$	$k = (2.7\text{e-}7)$
299	$CO_2^+ + O^- \longrightarrow CO_2 + O(^3P)$	$k = (1.0\text{e-}7)(T/300.0)^{-0.5}$
300	$CO_2^+ + O_2^- \longrightarrow CO_2 + O_2$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
301	$CO_2^+ + O_3^- \longrightarrow CO_2 + O_3$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
302	$CO_2^+ + NO_2^- \longrightarrow CO_2 + NO_2$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
303	$CO_2^+ + NO_3^- \longrightarrow CO_2 + NO_3$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$

304	$CO_2^+ + CO_3^- \longrightarrow 2CO_2 + O(^3P)$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
305	$CO_2^+ + NO_2^-(H2O)_2 \longrightarrow$ $CO_2 + NO_2 + 2H_2O$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
306	$CO_2^+ + CO_4^- \longrightarrow 2CO_2 + O_2$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
307	$H_3O^+ + e^- \longrightarrow H_2 + OH$	$k = (1.3e\text{-}6)(T/300.0)^{-0.7}$
308	$H_3O^+ + O^- \longrightarrow$ $H_2 + O(^3P) + OH$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
309	$H_3O^+ + O_2^- \longrightarrow$ $H_2 + O_2 + OH$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
310	$H_3O^+ + O_3^- \longrightarrow$ $H_2 + O_3 + OH$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$

311	$H_3O^+ + NO_2^- \longrightarrow$ $H_2 + NO_2 + OH$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
312	$H_3O^+ + NO_3^- \longrightarrow$ $H_2 + NO_3 + OH$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
313	$H_3O^+ + CO_3^- \longrightarrow$ $H_2 + CO_2 + O(^3P) + OH$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
314	$H_3O^+ + NO_2^-(H2O)_2 \longrightarrow$ $H_2 + NO_2 + 2H_2O + OH$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
315	$H_3O^+ + CO_4^- \longrightarrow$ $H_2 + CO_2 + O_2 + OH$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
316	$H_3O^+(H_2O) + e^- \longrightarrow$ $H_2 + OH + H_2O$	$k = (2.8e\text{-}6)(T/300.0)^{-0.15}$
317	$H_3O^+(H_2O) + O_2^- \longrightarrow$ $H_2 + O_2 + OH + H_2O$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$

318	$H_3O^+(H_2O) + O_3^- \longrightarrow$ $H_2 + O_3 + OH + H_2O$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
319	$H_3O^+(H_2O) + NO_2^- \longrightarrow$ $H_2 + NO_2 + OH + H_2O$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
320	$H_3O^+(H_2O) + NO_3^- \longrightarrow$ $H_2 + NO_3 + OH + H_2O$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
321	$H_3O^+(H_2O) + CO_3^- \longrightarrow$ $H_2 + CO_2 + O(^3P) + OH + H_2O$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
322	$H_3O^+(H_2O) + NO_2^-(H2O)_2 \longrightarrow$ $H_2 + NO_2 + OH + 3H_2O$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
323	$CO_2^+CO_2 + e^- \longrightarrow 2CO_2$	$k = (2.0\text{e-}6)(T/300.0)^{-1.0}$
324	$O_2^+O_2 + e^- \longrightarrow 2O_2$	$k = (2.0\text{e-}6)(T/300.0)^{-1.0}$

325	$O_2^+O_2 + O^- \longrightarrow 2O_2 + O(^3P)$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
326	$O_2^+O_2 + O_2^- \longrightarrow 3O_2$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
327	$O_2^+O_2 + O_3^- \longrightarrow 2O_2 + O_3$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
328	$O_2^+O_2 + NO_3^- \longrightarrow 2O_2 + NO_3$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
329	$O_2^+O_2 + CO_3^- \longrightarrow$ $2O_2 + CO_2 + O(^3P)$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
330	$O_2^+O_2 + NO_2^-(H2O)_2 \longrightarrow$ $2O_2 + NO_2 + 2H_2O$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$
331	$O_2^+O_2 + CO_4^- \longrightarrow 3O_2 + CO_2$	$k = (2.0e\text{-}7)(T/300.0)^{-0.5}$

332	$O_2^- + HNO_3 \longrightarrow NO_3^- + HO_2$	$k = (2.8e\text{-}9)$
333	$O_2^- + HCl \longrightarrow Cl^- + HO_2$	$k = (1.6e\text{-}9)$
334	$O^- + O_3 \longrightarrow O_3^- + O(^3P)$	$k = (8.0e\text{-}10)$
335	$O^- + NO_2 \longrightarrow NO_2^- + O(^3P)$	$k = (1.0e\text{-}9)$
336	$O^- + H_2 \longrightarrow OH^- + H$	$k = (3.2e\text{-}11)$
337	$O^- + CH_4 \longrightarrow OH^- + CH_3$	$k = (1.0e\text{-}10)$
338	$O^- + H_2 \longrightarrow H_2O + e^-$	$k = (6.0e\text{-}10)$

339	$O^- + O(^3P) \longrightarrow O_2 + e^-$	$k = (1.9e\text{-}10)$
340	$O^- + NO \longrightarrow NO_2 + e^-$	$k = (2.8e\text{-}10)$
341	$O^- + HCl \longrightarrow Cl^- + OH$	$k = (2.0e\text{-}9)$
342	$O^- + HNO_3 \longrightarrow NO_3^- + OH$	$k = (3.0e\text{-}9)$
343	$O_2^- + O(^3P) \longrightarrow O_2 + O^-$	$k = (1.5e\text{-}10)$
344	$O_2^- + O_3 \longrightarrow O_2 + O_3^-$	$k = (7.8e\text{-}10)$
345	$O_2^- + NO_2 \longrightarrow NO_2^- + O_2$	$k = (7.8e\text{-}10)$

346	$O_2^- + O(^3P) \longrightarrow e^- + O_3$	$k = (1.5e\text{-}10)$
347	$O_2^- + H \longrightarrow e^- + HO_2$	$k = (1.5e\text{-}10)$
348	$O_2^-(H2O) + CO_2 \longrightarrow CO_4^- + HO_2$	$k = (5.8e\text{-}10)$
349	$O_3^- + O(^3P) \longrightarrow O_2^- + O_2$	$k = (2.5e\text{-}10)$
350	$O_3^- + H \longrightarrow OH^- + O_2$	$k = (8.4e\text{-}10)$
351	$O_3^- + NO_2 \longrightarrow NO_3^- + O_2$	$k = (2.8e\text{-}10)$
352	$O_3^- + NO \longrightarrow NO_3^- + O(^3P)$	$k = (4.5e\text{-}12)$

353	$CO_3^- + N_2O_5 \longrightarrow NO_3^- + CO_2 + NO_3$	$k = (2.8e\text{-}10)$
354	$CO_3^- + O(^3P) \longrightarrow O_2^- + CO_2$	$k = (1.1e\text{-}10)$
355	$CO_3^- + H \longrightarrow OH^- + CO_2$	$k = (1.7e\text{-}10)$
356	$NO_2^- + NO_2 \longrightarrow NO_3^- + NO$	$k = (2.0e\text{-}13)$
357	$NO_2^- + H \longrightarrow OH^- + NO$	$k = (3.0e\text{-}10)$
358	$NO_2^- + HCl \longrightarrow Cl^- + HONO$	$k = (1.4e\text{-}9)$
359	$NO_2^- + HNO_3 \longrightarrow NO_3^- + HONO$	$k = (1.6e\text{-}9)$

360	$NO_2^- + N_2O_5 \longrightarrow NO_3^- + 2NO_2$	$k = (7.0e\text{-}10)$
361	$OH^- + NO_2 \longrightarrow NO_2^- + 2OH$	$k = (1.1e\text{-}9)$
362	$OH^- + O_3 \longrightarrow O_3^- + 2OH$	$k = (9.0e\text{-}10)$
363	$OH^- + H \longrightarrow e^- + 2H_2O$	$k = (1.4e\text{-}9)$
364	$OH^- + O(^3P) \longrightarrow e^- + 2HO_2$	$k = (2.0e\text{-}10)$
365	$O_4^- + O(^3P) \longrightarrow O_3^- + 2O_2$	$k = (4.0e\text{-}10)$
366	$O_4^- + CO_2 \longrightarrow CO_4^- + 2O_2$	$k = (4.3e\text{-}10)$

367	$O_4^- + NO \longrightarrow NO_3^- + 2O_2$	$k = (2.5e\text{-}10)$
368	$O_4^- + H_2O \longrightarrow O_2^-(H2O) + 2O_2$	$k = (1.0e\text{-}10)$
369	$CO_4^- + H \longrightarrow CO_3^- + 2OH$	$k = (2.2e\text{-}10)$
370	$CO_4^- + H_2O \longrightarrow O_2^-(H2O) + 2CO_2$	$k = (2.5e\text{-}10)$
371	$Cl^- + H \longrightarrow HCl + 2e^-$	$k = (9.3e\text{-}10)$
372	$Cl^- + HNO_3 \longrightarrow HCl + 2NO_3^-$	$k = (1.6e\text{-}9)$
373	$Cl^- + N_2O_5 \longrightarrow NO_3^- + 2ClNO_2$	$k = (9.4e\text{-}10)$

A.1.2 Trimolecular Reactions

#	Trimolecular Reaction	Reaction Rate Coeff
1	$Br + NO_2 \longrightarrow BrONO + M$	$k_0 = M(4.2e\text{-}31)(T/300.0)^{-2.4}$ $k_\infty = (2.7e\text{-}11)(T/300.0)^{0.0}$ $f_c = (0.55)$
2	$BrO + NO_2 \longrightarrow BrONO_2 + M$	$k_0 = M(5.2e\text{-}31)(T/300.0)^{-3.2}$ $k_\infty = (6.9e\text{-}12)(T/300.0)^{-2.9}$ $f_c = (0.6)$
3	$CH_3 + O_2 \longrightarrow CH_3O_2 + M$	$k_0 = M(1.0e\text{-}30)(T/300.0)^{-3.3}$ $k_\infty = (2.2e\text{-}12)(T/300.0)^{1.0}$ $f_c = (0.27)$
4	$CO + H \longrightarrow HCO + M$	$k_0 = M(1.9e\text{-}33)\exp(-7000.0/T)(T/300.0)^{0.0}$ $k_\infty = (T/298.0)^{0.0}$ $f_c = (0.0)$

5	$CO + O(^3P) \longrightarrow CO_2 + M$	$k_0 = M(1.7e\text{-}33)\exp(-12550.0/T)(T/300.0)^{0.0}$ $k_\infty = (4.2e\text{-}12)\exp(-199540.0/T)(T/298.0)^{0.0}$ $f_c = (0.6)$
6	$Cl_2O_2 + M \longrightarrow ClO + ClO$	$k_0 = M(1.35e\text{-}5)\exp(-8720.0/T)(T/300.0)^{1.0}$ $k_\infty = (1.8e15)\exp(-8450.0/T)(T/300.0)^{0.0}$ $f_c = (0.6)$
7	$Cl + NO_2 \longrightarrow ClNO_2 + M$	$k_0 = M(1.8e\text{-}31)(T/300.0)^{-2.0}$ $k_\infty = (1.0e\text{-}10)(T/300.0)^{-1.0}$ $f_c = (0.6)$
8	$Cl + O_2 \longrightarrow ClOO + M$	$k_0 = M(2.7e\text{-}33)(T/300.0)^{-1.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
9	$ClO + ClO \longrightarrow Cl_2O_2 + M$	$k_0 = M(2.2e\text{-}32)(T/300.0)^{-3.1}$ $k_\infty = (3.5e\text{-}12)(T/300.0)^{-1.0}$ $f_c = (0.6)$

10	$\mathrm{ClO} + \mathrm{NO_2} \longrightarrow \mathrm{ClONO_2} + \mathrm{M}$	$k_0 = M(1.6\text{e-}31)(T/300.0)^{\text{-}3.4}$ $k_\infty = (2.0\text{e-}11)(T/300.0)^{0.0}$ $f_c = (0.0) + (0.0) \cdot \exp(\text{-T}/430.0)$
11	$\mathrm{ClONO_2} + \mathrm{M} \longrightarrow \mathrm{ClO} + \mathrm{NO_2}$	$k_0 = M(T/300.0)^{0.0}$ $k_\infty = (2.75\text{e-}6)\exp(\text{-}95100.0/T)(T/300.0)^{0.0}$ $f_c = (0.0)$
12	$\mathrm{ClOO} + \mathrm{M} \longrightarrow \mathrm{Cl} + \mathrm{O_2}$	$k_0 = M(2.8\text{e-}10)\exp(\text{-}1820.0/T)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
13	$\mathrm{HO_2} + \mathrm{HO_2} \longrightarrow \mathrm{H_2O_2} + \mathrm{O_2}$	$k_0 = M(1.9\text{e-}33)\exp(980.0/T)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

14	$H + O_2 \longrightarrow HO_2 + M$	$k_0 = M(6.2e\text{-}32)(T/300.0)^{-1.6}$ $k_\infty = (7.5e\text{-}11)(T/300.0)^{0.0}$ $f_c = (0.0) + (0.0) \cdot \exp(-T/498.0)$
15	$HO_2 + NO_2 \longrightarrow HO_2NO_2 + M$	$k_0 = M(1.8e\text{-}31)(T/300.0)^{-3.2}$ $k_\infty = (4.7e\text{-}12)(T/300.0)^{-1.4}$ $f_c = (0.6)$
16	$HO_2NO_2 + M \longrightarrow HO_2 + NO_2$	$k_0 = M(5.0e\text{-}6)\exp(-10000.0/T)(T/300.0)^{0.0}$ $k_\infty = (2.6e15)\exp(-10900.0/T)(T/300.0)^{0.0}$ $f_c = (0.6)$
17	$HONO + M \longrightarrow NO + OH$	$k_0 = M(6.4e6)T^{-3.8}\exp(-25340.0/T)$ $k_\infty = (1.2e19)T^{-1.23}\exp(-25010.0/T)$ $f_c = (0.62)$

18	$CH_3O_2NO_2 + M \longrightarrow CH_3O_2 + NO_2$	$k_0 = M(9.0e\text{-}5)\exp(-9690.0/T)(T/300.0)^{0.0}$ $k_\infty = (1.1e16)\exp(-10560.0/T)(T/300.0)^{0.0}$ $f_c = (0.4)$
19	$CH_3O + NO_2 \longrightarrow CH_3ONO_2 + M$	$k_0 = M(2.8e\text{-}29)(T/300.0)^{-4.5}$ $k_\infty = (2.0e\text{-}11)(T/300.0)^{0.0}$ $f_c = (0.44)$
20	$CH_3O_2 + NO_2 \longrightarrow CH_3O_2NO_2 + M$	$k_0 = M(2.5e\text{-}30)(T/300.0)^{-5.5}$ $k_\infty = (7.5e\text{-}12)(T/300.0)^{0.0}$ $f_c = (0.4)$
21	$N_2 + O(^1D) \longrightarrow N_2O + M$	$k_0 = M(3.5e\text{-}37)(T/300.0)^{-0.6}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
22	$N_2O_5 + M \longrightarrow NO_2 + NO_3$	$k_0 = M(0.0022)\exp(-11080.0/T)(T/300.0)^{-4.4}$ $k_\infty = (9.7e14)\exp(-11080.0/T)(T/300.0)^{0.1}$ $f_c = (0.0) + (0.0) \cdot \exp(-T/250.0) + \exp(extrm-1$

23	$NO_2 + NO_3 \longrightarrow N_2O_5 + M$	$k_0 = M(2.7e\text{-}30)(T/300.0)^{-3.4}$ $k_\infty = (2.0e\text{-}12)(T/300.0)^{0.2}$ $f_c = (0.0) + (0.0) \cdot \exp(-T/250.0) + \exp(extrm-1$
24	$NO_2 + O(^3P) \longrightarrow NO_3 + M$	$k_0 = M(9.0e\text{-}32)(T/300.0)^{-2.0}$ $k_\infty = (2.2e\text{-}11)(T/300.0)^{0.0}$ $f_c = (0.0) + (0.0) \cdot \exp(-T/1300.0)$
25	$NO_2 + OH \longrightarrow HNO_3 + M$	$k_0 = M(2.47e\text{-}30)(T/300.0)^{-2.97}$ $k_\infty = (1.45e\text{-}11)(T/300.0)^{-2.77}$ $f_c = (0.6)$
26	$NO + NO \longrightarrow NO_2 + NO_2$	$k_0 = M \exp(530.0/T)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

27	$NO + O(^3P) \longrightarrow NO_2 + M$	$k_0 = M(1.0e\text{-}31)(T/300.0)^{-1.6}$ $k_\infty = (3.0e\text{-}11)(T/300.0)^{0.3}$ $f_c = (0.0) + (0.0) \cdot \exp(-T/1850.0)$
28	$NO + OH \longrightarrow HONO + M$	$k_0 = M(7.4e\text{-}31)(T/300.0)^{-2.4}$ $k_\infty = (3.2e\text{-}11)(T/300.0)^{0.0}$ $f_c = (0.0) + (0.0) \cdot \exp(-T/1300.0)$
29	$O(^3P) + O_2 \longrightarrow O_3 + M$	$k_0 = M(5.64e\text{-}34)(T/300.0)^{-2.8}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
30	$OH + OH \longrightarrow H_2O_2 + M$	$k_0 = M(8.0e\text{-}31)(T/300.0)^{-0.8}$ $k_\infty = (3.0e\text{-}11)(T/300.0)^{0.0}$ $f_c = (0.5)$

31	$CH_3ONO_2 + M \longrightarrow CH_3O + NO_2$	$k_0 = M(T/300.0)^{0.0}$ $k_\infty = (1.0e13)\exp(-140.0/T)(T/300.0)^{0.0}$ $f_c = (0.0)$
32	$O_2^+ + O_2 \longrightarrow O_2^+O_2 + M$	$k_0 = M(1.0e-30)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
33	$O_2^+ + H_2O \longrightarrow O_2^+(H_2O) + M$	$k_0 = M(2.8e-28)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
34	$O_2^+ + CO_2 \longrightarrow O_2^+(CO_2) + M$	$k_0 = M(1.7e-29)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
35	$CO_2^+ + CO_2 \longrightarrow CO_2^+CO_2 + M$	$k_0 = M(2.5e-28)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

36	$O_2^+(CO_2) + CO_2 \longrightarrow O_2^+(CO_2)_2 + M$	$k_0 = M(5.0\text{e-}30)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
37	$O_2^+(H_2O) + H_2O \longrightarrow O_2^+(H_2O)_2 + M$	$k_0 = M(1.3\text{e-}27)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
38	$H_3O^+ + H_2O \longrightarrow H_3O^+(H_2O) + M$	$k_0 = M(3.4\text{e-}27)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
39	$H_3O^+(H_2O) + H_2O \longrightarrow H_3O^+(H_2O)_2 + M$	$k_0 = M(2.3\text{e-}27)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
40	$H_3O^+(H_2O) + M \longrightarrow H_3O^+ + H_2O + M$	$k_0 = M(7.0\text{e-}26)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

41	$\mathrm{H_3O^+(H_2O)_2 + H_2O \longrightarrow H_3O^+(H_2O)_3 + M}$	$k_0 = M(2.4\text{e-}27)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
42	$\mathrm{H_3O^+(H_2O)_2 + M \longrightarrow H_3O^+(H_2O) + H_2O + M}$	$k_0 = M(7.0\text{e-}18)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
43	$\mathrm{H_3O^+(H_2O)_3 + H_2O \longrightarrow H_3O^+(H_2O)_4 + M}$	$k_0 = M(9.0\text{e-}28)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
44	$\mathrm{H_3O^+(H_2O)_3 + M \longrightarrow H_3O^+(H_2O)_2 + H_2O + M}$	$k_0 = M(4.0\text{e-}14)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
45	$\mathrm{H_3O^+(H_2O)_4 + M \longrightarrow H_3O^+(H_2O)_3 + H_2O + M}$	$k_0 = M(6.0\text{e-}12)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

46	$O_2 + e^- \longrightarrow O_2^- + M$	$k_0 = M(2.0\text{e-}31)\exp(\text{-}600.0/T)(T/300.0)^{1.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
47	$O^- + CO_2 \longrightarrow$ $CO_3^- + CO_2 + M$	$k_0 = M(1.1\text{e-}27)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
48	$O_2^- + CO_2 \longrightarrow CO_4^- + M$	$k_0 = M(1.3\text{e-}29)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
49	$O_2^+ + O^- \longrightarrow$ $O_2 + O(^3P) + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
50	$O_2^+ + O_2^- \longrightarrow 2O_2 + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

51	$O_2^+ + O_3^- \longrightarrow$ $O_2 + O_3 + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
52	$O_2^+ + NO_2^- \longrightarrow$ $O_2 + NO_2 + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
53	$O_2^+ + NO_3^- \longrightarrow$ $O_2 + NO_3 + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
54	$O_2^+ + CO_3^- \longrightarrow$ $O_2 + CO_2 + O(^3P) + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
55	$O_2^+ + NO_2^-(H2O)_2 \longrightarrow$ $O_2 + NO_2 + 2H_2O + M$	$k_0 = M(1.0\text{e-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

56	$NO^+ + O^- \longrightarrow$ $NO + O(^3P) + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
57	$NO^+ + O_2^- \longrightarrow$ $NO + O_2 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
58	$NO^+ + O_3^- \longrightarrow$ $NO + O_3 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
59	$NO^+ + NO_2^- \longrightarrow$ $NO + NO_2 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
60	$NO^+ + NO_3^- \longrightarrow$ $NO + NO_3 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

61	$NO^+ + CO_3^- \longrightarrow$ $NO + CO_2 + O(^3P) + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
62	$NO^+ + NO_2^-(H2O)_2 \longrightarrow$ $NO + NO_2 + 2H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
63	$NO^+ + CO_4^- \longrightarrow$ $NO + CO_2 + O_2 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
64	$CO_2^+ + O_2^- \longrightarrow$ $CO_2 + O_2 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
65	$CO_2^+ + O_3^- \longrightarrow$ $CO_2 + O_3 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

66	$CO_2^+ + NO_2^- \longrightarrow$ $CO_2 + NO_2 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
67	$CO_2^+ + NO_3^- \longrightarrow$ $CO_2 + NO_3 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
68	$CO_2^+ + CO_3^- \longrightarrow$ $CO_2 + CO_2 + O(^3P) + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
69	$CO_2^+ + NO_2^-(H2O)_2 \longrightarrow$ $CO_2 + NO_2 + 2H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
70	$CO_2^+ + CO_4^- \longrightarrow$ $2CO_2 + O_2 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

71	$H_3O^+ + O^- \longrightarrow$ $H_2 + O(^3P) + OH + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
72	$H_3O^+ + O_2^- \longrightarrow$ $H_2 + O_2 + OH + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
73	$H_3O^+ + O_3^- \longrightarrow$ $H_2 + O_3 + OH + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
74	$H_3O^+ + NO_2^- \longrightarrow$ $H_2 + NO_2 + OH + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
75	$H_3O^+ + NO_3^- \longrightarrow$ $H_2 + NO_3 + OH + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

76	$H_3O^+ + CO_3^- \longrightarrow$ $H_2 + CO_2 + O(^3P) + OH + M$	$k_0 = M(3.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
77	$H_3O^+ + NO_2^-(H2O)_2 \longrightarrow$ $H_2 + NO_2 + 2H_2O + OH + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
78	$H_3O^+ + CO_4^- \longrightarrow$ $H_2 + CO_2 + O_2 + OH + M$	$k_0 = M(3.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
79	$H_3O^+(H_2O) + O^- \longrightarrow$ $H_2 + O(^3P) + OH + H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
80	$H_3O^+(H_2O) + O_2^- \longrightarrow$ $H_2 + O_2 + OH + H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

81	$H_3O^+(H_2O) + O_3^- \longrightarrow$ $H_2 + O_3 + OH + H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
82	$H_3O^+(H_2O) + NO_2^- \longrightarrow$ $H_2 + NO_2 + OH + H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
83	$H_3O^+(H_2O) + NO_3^- \longrightarrow$ $H_2 + NO_3 + OH + H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
84	$H_3O^+(H_2O) + CO_3^- \longrightarrow$ $H_2 + CO_2 + O(^3P) + OH + H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
85	$H_3O^+(H_2O) + NO_2^-(H2O)_2 \longrightarrow$ $H_2 + NO_2 + OH + 3H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

86	$H_3O^+(H_2O) + CO_4^- \longrightarrow$ $H_2 + CO_2 + O_2 + OH + H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
87	$O_2^+O_2 + O^- \longrightarrow$ $O_2 + O(^3P) + O_2 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
88	$O_2^+O_2 + O_2^- \longrightarrow 3O_2 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
89	$O_2^+O_2 + NO_3^- \longrightarrow$ $2O_2 + NO_3 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
90	$O_2^+O_2 + CO_3^- \longrightarrow$ $2O_2 + CO_2 + O(^3P) + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

91	$O_2^+O_2 + NO_2^-(H2O)_2 \longrightarrow$ $2O_2 + NO_2 + 2H_2O + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
92	$O_2^+O_2 + CO_4^- \longrightarrow$ $3O_2 + CO_2 + M$	$k_0 = M(1.0e\text{-}25)(T/300.0)^{-2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
93	$O_2^- + O_2 \longrightarrow O_4^- + M$	$k_0 = M(3.4e\text{-}31)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
94	$O_2^- + H_2O \longrightarrow O_2^-(H2O) + M$	$k_0 = M(2.2e\text{-}28)(T/300.0)^{0.0}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
95	$O_2^-(H2O) + NO \longrightarrow$ $NO_3^- + H_2O + M$	$k_0 = M(T/300.0)^{0.0}$ $k_\infty = (2.0e\text{-}10)(T/300.0)^{0.0}$ $f_c = (0.0)$

A.1.3 Photolysis Reactions

#	Photolysis Reaction
1	$Br_2 + h\nu \longrightarrow Br + Br$
2	$BrCl + h\nu \longrightarrow Cl + Br$
3	$BrO + h\nu \longrightarrow O(^3P) + Br$
4	$BrO + h\nu \longrightarrow Br + O(^1D)$
5	$BrONO + h\nu \longrightarrow Br + NO_2$
6	$BrONO_2 + h\nu \longrightarrow NO_2 + BrO$
7	$BrONO_2 + h\nu \longrightarrow Br + NO_3$

8	$CF_2Cl_2 + h\nu \longrightarrow Cl + Cl$
9	$CH_3Br + h\nu \longrightarrow Br + CH_3$
10	$CH_4 + h\nu \longrightarrow H + CH_3$
11	$CO_2 + h\nu \longrightarrow O(^3P) + CO$
12	$CO_2 + h\nu \longrightarrow CO + O(^1D)$
13	$Cl_2 + h\nu \longrightarrow Cl + Cl$
14	$Cl_2O_2 + h\nu \longrightarrow Cl + ClOO$

15	$\mathrm{ClNO_2} + h\nu \longrightarrow \mathrm{Cl} + \mathrm{NO_2}$
16	$\mathrm{ClO} + h\nu \longrightarrow \mathrm{O}(^3\mathrm{P}) + \mathrm{Cl}$
17	$\mathrm{ClONO_2} + h\nu \longrightarrow \mathrm{Cl} + \mathrm{NO_3}$
18	$\mathrm{ClOO} + h\nu \longrightarrow \mathrm{O}(^3\mathrm{P}) + \mathrm{ClO}$
19	$\mathrm{H_2O} + h\nu \longrightarrow$ $\mathrm{H} + \mathrm{H} + \mathrm{O}(^3\mathrm{P})$
20	$\mathrm{H_2O} + h\nu \longrightarrow \mathrm{H_2} + \mathrm{O}(^1\mathrm{D})$
21	$\mathrm{H_2O} + h\nu \longrightarrow \mathrm{H} + \mathrm{OH}$

22	$H_2O + h\nu \longrightarrow H_2 + O^+$
23	$H_2O_2 + h\nu \longrightarrow OH + OH$
24	$HCHO + h\nu \longrightarrow H_2 + CO$
25	$HCHO + h\nu \longrightarrow H + HCO$
26	$HCHO + h\nu \longrightarrow$ $H + H + CO$
27	$HCl + h\nu \longrightarrow H + Cl$
28	$HO_2 + h\nu \longrightarrow O(^3P) + OH$

29	$HO_2NO_2 + h\nu \longrightarrow NO_2 + HO_2$
30	$HO_2NO_2 + h\nu \longrightarrow OH + NO_3$
31	$HOBr + h\nu \longrightarrow OH + Br$
32	$HOCl + h\nu \longrightarrow Cl + OH$
33	$HONO + h\nu \longrightarrow NO + OH$
34	$HNO_3 + h\nu \longrightarrow OH + NO_2$
35	$HNO_3 + h\nu \longrightarrow O(^3P) + HONO$

36	$CH_3O_2NO_2 + h\nu \longrightarrow NO_2 + CH_3O_2$
37	$CH_3O_2NO_2 + h\nu \longrightarrow NO_3 + CH_3O$
38	$CH_3OCl + h\nu \longrightarrow Cl + CH_3O$
39	$CH_3OH + h\nu \longrightarrow H_2 + HCHO$
40	$CH_3OH + h\nu \longrightarrow OH + CH_3$
41	$CH_3ONO_2 + h\nu \longrightarrow NO_2 + CH_3O$
42	$CH_3OOH + h\nu \longrightarrow OH + CH_3O$

43	$\mathrm{CH_3OOH} + h\nu \longrightarrow$ $\mathrm{O}(^3\mathrm{P}) + \mathrm{H} + \mathrm{CH_3O}$
44	$\mathrm{N_2O} + h\nu \longrightarrow \mathrm{N_2} + \mathrm{O}(^1\mathrm{D})$
45	$\mathrm{N_2O_5} + h\nu \longrightarrow$ $\mathrm{O}(^3\mathrm{P}) + \mathrm{NO} + \mathrm{NO_3}$
46	$\mathrm{N_2O_5} + h\nu \longrightarrow \mathrm{NO_2} + \mathrm{NO_3}$
47	$\mathrm{NO} + h\nu \longrightarrow \mathrm{O}(^3\mathrm{P}) + \mathrm{N}$
48	$\mathrm{NO_2} + h\nu \longrightarrow \mathrm{NO} + \mathrm{O}(^1\mathrm{D})$
49	$\mathrm{NO_2} + h\nu \longrightarrow \mathrm{O}(^3\mathrm{P}) + \mathrm{NO}$

50	$NO_3 + h\nu \longrightarrow O(^3P) + NO_2$
51	$NO_3 + h\nu \longrightarrow O_2 + NO$
52	$O_2 + h\nu \longrightarrow O(^3P) + O(^1D)$
53	$O_2 + h\nu \longrightarrow O(^3P) + O(^3P)$
54	$O_3 + h\nu \longrightarrow O(^3P) + O_2$
55	$O_3 + h\nu \longrightarrow O_2 + O(^1D)$
56	$OClO + h\nu \longrightarrow O(^3P) + ClO$

57	$\mathrm{OClO} + h\nu \longrightarrow \mathrm{ClO} + \mathrm{O}(^1\mathrm{D})$

A.2 Master Chemical Mechanism

REFERENCES

Adhikari, R. and R. K. Agrawal (2013). An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*.

Ahmed, S. E., S. Pawar, and O. San (2020). Pyda: A hands-on introduction to dynamical data assimilation with python. *Fluids 5*(4), 225.

Altmeyer, P. (2023). Conformal prediction.

Arnaut, L. and H. Burrows (2006). *Chemical kinetics: from molecular structure to chemical reactivity*. Elsevier.

Baillarin, S. J., A. Meygret, C. Dechoz, B. Petrucci, S. Lacherade, T. Tremas, C. Isola, P. Martimort, and F. Spoto (2012). Sentinel-2 level 1 products and image processing performances. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pp. 7003–7006.

Bäumker, M. and F. Heimes (2001). New calibration and computing method for direct georeferencing of image and scanner data using the position and angular data of an hybrid inertial navigation system. In *OEEPE Workshop, Integrated Sensor Orientation*, pp. 1–17.

Bishop, C. M., M. Svensén, and C. K. Williams (1998). Gtm: The generative topographic mapping. *Neural computation 10*(1), 215–234.

Boiarskii, B. and H. Hasegawa (2019). Comparison of ndvi and ndre indices to detect differences in vegetation and chlorophyll content. *J. Mech. Contin. Math. Sci 4*, 20–29.

Boldi, R. A. (1994). A model of the ion chemistry of electrified convection.

Brunton, S. L., B. W. Brunton, J. L. Proctor, E. Kaiser, and J. N. Kutz (2017). Chaos as an intermittently forced linear system. *Nature communications 8*(1), 19.

Brunton, S. L., M. Budišić, E. Kaiser, and J. N. Kutz (2021). Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*.

Brunton, S. L., J. L. Proctor, and J. N. Kutz (2016). Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine 49*(18), 710–715.

Burkholder, J., S. Sander, J. Abbatt, J. Barker, C. Cappa, J. Crounse, T. Dibble, R. Huie, C. Kolb, M. Kurylo, et al. (2020). Chemical kinetics and photochemical data for use in atmospheric studies; evaluation number 19. Technical report, Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space ….

Burns, R. (1993). Origin of electronic spectra of minerals in the visible and near-infrared region. *Remote Geochemical Analysis*, 3–29.

Burroughs, H. E., C. Muller, W. Yao, and Q. Yu (2014). An Evaluation of Filtration and Air Cleaning Equipment Performance in Existing Installations with Regard to Acceptable IAQ Attainment. In A. Li, Y. Zhu, and Y. Li (Eds.), *Proceedings of the 8th International Symposium on Heating, Ventilation and Air Conditioning*, Volume 262, pp. 267–275. Berlin, Heidelberg: Springer Berlin Heidelberg. Series Title: Lecture Notes in Electrical Engineering.

Carneiro, J., M. A. Cole, and E. Strobl (2021). The effects of air pollution on students' cognitive performance: Evidence from brazilian university entrance tests. *Journal of the Association of Environmental and Resource Economists 8*, 1051 – 1077.

Chen, R. T., Y. Rubanova, J. Bettencourt, and D. K. Duvenaud (2018). Neural ordinary differential equations. *Advances in neural information processing systems 31*.

Costello, A., M. Abbas, A. Allen, S. Ball, S. Bell, R. Bellamy, S. Friel, N. Groce, A. Johnson, M. Kett, M. Lee, C. Levy, M. Maslin, D. McCoy, B. McGuire, H. Montgomery, D. Napier, C. Pagel, J. Patel, J. A. P. de Oliveira, N. Redclift, H. Rees, D. Rogger, J. Scott, J. Stephenson, J. Twigg, J. Wolff, and C. Patterson (2009). Managing the health effects of climate change: Lancet and university college london institute for global health commission. *The Lancet 373*(9676), 1693–1733.

Curran, P. J. (1989). Remote sensing of foliar chemistry. *Remote sensing of environment 30*(3), 271–278.

Edenhofer, O., R. Pichs-Madruga, Y. Sokona, E. Farahani, S. Kadner, K. Seyboth, A. Adler, I. Baum, S. Brunner, P. Eickemeier, et al. (2014). *Climate change 2014: Mitigation of climate change*. Cambridge University Press.

Emmerich, S. J., T. McDowell, and W. Anis (2005). Investigation of the impact of commercial building envelope airtightness on HVAC energy use. Technical Report NIST IR 7238, National Institute of Standards and Technology, Gaithersburg, MD.

Finewax, Z., D. Pagonis, M. S. Claflin, A. V. Handschy, W. L. Brown, O. Jenks, B. A. Nault, D. A. Day, B. M. Lerner, J. L. Jimenez, P. J. Ziemann, and J. A. Gouw (2021, September). Quantification and source characterization of volatile organic compounds from exercising and application of chlorine-based cleaning products in a university athletic center. *Indoor Air 31*(5), 1323–1339.

Fisher, M. and D. J. Lary (1995, oct). Lagrangian 4-dimensional variational data assimilation of chemical species. *Quarterly Journal of the Royal Meteorological Society 121*(527, Part A), 1681–1704.

Gao, X., B. Coull, X. Lin, P. Vokonas, A. Spiro, L. Hou, J. Schwartz, and A. A. Baccarelli (2021). Short-term air pollution, cognitive performance, and nonsteroidal anti-inflammatory drug use in the veterans affairs normative aging study. *Nature aging 1 5*, 430.

Greydanus, S., M. Dzamba, and J. Yosinski (2019). Hamiltonian neural networks. *Advances in neural information processing systems 32*.

Haines, A., R. S. Kovats, D. Campbell-Lendrum, and C. Corvalán (2006). Climate change and human health: Impacts, vulnerability and public health. *Public Health 120*(7), 585–596.

Hunt, G. R. (1977). Spectral signatures of particulate minerals in the visible and near infrared. *Geophysics 42*(3), 501–513.

Innes, M., A. Edelman, K. Fischer, C. Rackauckas, E. Saba, V. B. Shah, and W. Tebbutt (2019). A differentiable programming system to bridge machine learning and scientific computing. *arXiv preprint arXiv:1907.07587*.

Keller-Rudek, H., G. Moortgat, R. Sander, and R. Sörensen (2013). The mpi-mainz uv/vis spectral atlas of gaseous molecules of atmospheric interest. *Earth System Science Data 5*(2), 365–373.

King, M. (2013). Calculating photolysis rates and estimating photolysis lifetimes. *ECG Environmental Briefs 1*, 1–2.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetics 43*(1), 59–69.

Krebs, B. and S. Luechinger (2021). Air pollution, cognitive performance, and the role of task proficiency. *Planetary Health eJournal*.

Lamping, G. A. and C. O. Muller. Air Cleaning in Practice – School Sustainability and Commercial Building Field Study Results. pp. 13.

Lary, D. J. (1999). Data assimilation: a powerful tool for atmospheric chemistry. *Philosophical Transactions of the Royal Society of London Series a-Mathematical Physical and Engineering Sciences 357*(1763), 3445–3457.

Lary, D. J., B. V. Khattatov, and H. Y. Mussa (2003). Chemical data assimilation: A case study of solar occultation data from the ATLAS 1 mission of the Atmospheric Trace Molecule Spectroscopy Experiment (ATMOS). *Journal of Geophysical Research-Atmospheres 108*(D15).

Lary, D. J., T. Lary, and B. Sattler (2015). Using machine learning to estimate global pm2.5 for environmental health studies. *Environmental Health Insights 9s1*, EHI.S15664. PMID: 26005352.

Main-Knorn, M., B. Pflug, J. Louis, V. Debaecker, U. Müller-Wilm, and F. Gascon (2017). Sen2cor for sentinel-2. In *Image and Signal Processing for Remote Sensing XXIII*, Volume 10427, pp. 37–48. SPIE.

Masson-Delmotte, V., P. Zhai, H.-O. Pörtner, D. Roberts, J. Skea, P. Shukla, A. Pirani, W. Moufouma-Okia, C. Péan, R. Pidcock, et al. (2018). *Global warming of 1.5°C. An IPCC special report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty.* Intergovernmental Panel on Climate Change.

Memarzadeh, F. and W. Xu (2012, March). Role of air changes per hour (ACH) in possible transmission of airborne infections. *Building Simulation 5*(1), 15–28.

Mostafa, M. M. R. and K. P. Schwarz (2000). A multi-sensor system for airborne image capture and georeferencing. *Photogrammetric Engineering and Remote Sensing 66*, 1417–1424.

Muller, R., M. Lehner, R. Muller, P. Reinartz, M. Schroeder, and B. Vollmer (2002). A program for direct georeferencing of airborne and spaceborne line scanner images. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences 34*(1), 148–153.

Navalgund, R. R., V. Jayaraman, and P. Roy (2007). Remote sensing applications: An overview. *current science*, 1747–1766.

Neumark, D. M. (1992). Transition state spectroscopy of bimolecular chemical reactions. *Annual Review of Physical Chemistry 43*(1), 153–176.

Ng, L. C., A. K. Persily, and S. J. Emmerich (2015, October). IAQ and energy impacts of ventilation strategies and building envelope airtightness in a big box retail building. *Building and Environment 92*, 627–634.

Ni, Y., C. T. Loftus, A. Szpiro, M. T. Young, M. F. Hazlehurst, L. E. Murphy, F. A. Tylavsky, A. Mason, K. Z. Lewinn, S. Sathyanarayana, E. S. Barrett, N. R. Bush, and C. J. Karr (2021). Associations of pre- and postnatal air pollution exposures with child cognitive performance and behavior: A multi-cohort study. *ISEE Conference Abstracts*.

Odabasi, M. (2008, March). Halogenated Volatile Organic Compounds from the Use of Chlorine-Bleach-Containing Household Products. *Environmental Science & Technology 42*(5), 1445–1451. Publisher: American Chemical Society.

Organization, W. H. et al. (2021). *WHO global air quality guidelines: particulate matter (PM2. 5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide*. World Health Organization.

Partridge, H., C. W. Bauschlicher Jr, J. R. Stallcop, and E. Levin (1993). Ab initio potential energy surface for h–h2. *The Journal of chemical physics 99*(8), 5951–5960.

Pease, L. F., N. Wang, T. I. Salsbury, R. M. Underhill, J. E. Flaherty, A. Vlachokostas, G. Kulkarni, and D. P. James (2021, June). Investigation of potential aerosol transmission and infectivity of SARS-CoV-2 through central ventilation systems. *Building and Environment 197*, 107633.

Petneházi, G. (2019). Recurrent neural networks for time series forecasting. *arXiv preprint arXiv:1901.00069*.

Rackauckas, C., D. Kelly, Q. Nie, J. Li, C. Warner, M. Dhairya, J. Fang, E. Zhou, R. Supekar, S. Sandhu, et al. (2020). Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2012.09345*.

Rackauckas, C. and Q. Nie (2017). Differentialequations. jl–a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of open research software 5*(1).

Sakurai, J. J. and E. D. Commins (1995). Modern quantum mechanics, revised edition.

Saunders, S. M., M. E. Jenkin, R. Derwent, and M. Pilling (2003). Protocol for the development of the master chemical mechanism, mcm v3 (part a): tropospheric degradation of non-aromatic volatile organic compounds. *Atmospheric Chemistry and Physics 3*(1), 161–180.

Shehab, M. and F. D. Pope (2019). Effects of short-term exposure to particulate matter air pollution on cognitive performance. *Scientific Reports 9*.

Sholokhov, A., Y. Liu, H. Mansour, and S. Nabi (2023). Physics-informed neural ode (pinode): embedding physics into models using collocation points. *Scientific Reports 13*(1), 10166.

Srikrishna, D. (2022, November). Long-term experience with rapid air filtration (6 to 15 air changes per hour) in a K-5 elementary school using HEPA and Do-It-Yourself (DIY) air purifiers during the COVID-19 pandemic. preprint, Infectious Diseases (except HIV/AIDS).

Takens, F. (2006). Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*, pp. 366–381. Springer.

Thenkabail, P. S., R. B. Smith, and E. De Pauw (2000). Hyperspectral vegetation indices and their relationships with agricultural crop characteristics. *Remote sensing of Environment 71*(2), 158–182.

United Nations (2015). Transforming our world: the 2030 agenda for sustainable development. *UN General Assembly*.

Vacca, G. (2019). Overview of open source software for close range photogrammetry. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 42*, 239–245.

Van Der Meer, F. (2004). Analysis of spectral absorption features in hyperspectral imagery. *International journal of applied earth observation and geoinformation 5*(1), 55–68.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks 5*, 241–259.

World Health Organization (2018). Climate change and health.

Wu, T., H.-J. Werner, and U. Manthe (2006). Accurate potential energy surface and quantum reaction rate calculations for the h+ ch4→ h2+ ch3 reaction. *The Journal of chemical physics 124*(16).

Yoon, H. H., H. A. Fernandez, F. Nigmatulin, W. Cai, Z. Yang, H. Cui, F. Ahmed, X. Cui, M. G. Uddin, E. D. Minot, et al. (2022). Miniaturized spectrometers with a tunable van der waals junction. *Science 378*(6617), 296–299.

Zaatari, M., A. Novoselac, and J. Siegel (2016, May). Impact of ventilation and filtration strategies on energy consumption and exposures in retail stores. *Building and Environment 100*, 186–196.

Zheng, W., J. Hu, Z. Wang, J. Li, Z. Fu, H. Li, J. Jurasz, S. Chou, and J. Yan (2021, August). COVID-19 Impact on Operation and Energy Consumption of Heating, Ventilation and Air-Conditioning (HVAC) Systems. *Advances in Applied Energy 3*, 100040.

BIOGRAPHICAL SKETCH

UPDATE REQUIRED!!!

CURRICULUM VITAE

UPDATE REQUIRED!