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

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! Make sure to include lab colleagues *and* ActivePure colleagues and a nod to Dr. Cooper, OSU teachers, and friends...

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

UPDATE REQUIRED!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The rapid pace of global change poses a significant and ever-present threat to human prosperity. To facilitate the development of remediation technologies and 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. The goal of this applied physics dissertation is to expand the boundary of what is possible in realm of sensing by leveraging machine learning in a *principled way* to produce actionable insights. To that end, this work invokes three key case studies to demonstrate how machine learning can be used effectively to fill to enable

To that end, the guiding question of this work can be summarized succinctly as: *How can we best utilize the measurements we have together with the physics we know to estimate the quantities we care about?*

1.0.1 Global Change

1.0.2

- first we should discuss global change and the necessity for improved sensing and simulation - justify the use of machine learning in order to fill the gaps where theory provides the motivation between what we measure and what we want to predict but no easily-represented physical relationship exists. - this is a good place to bring in the - we can then discuss the advent of physics-informed machine learning, and better, scientific machine learning to fill in the gaps and not be Snake Oil Salesmen (TM). In particular the role of numerical methods has not been subsumed by machine learning. At best these techniques are complementary

and any performance claims need to be carefully benchmarked so as to not oversell the abilities of machine learning. - Many of these emerging methods are promising but have seen little application to noisy, real-world data. One can only examine the Lorentz equations so much before transitioning to actual atmospheric dynamics - We can do a paragraph on the specific areas we are trying to address: (1) using machine learning to establish the nonlinear relationship between measured spectra and constituent concentrations in water, (2) data-based techniques for evaluating the uncertainty of single sensor time series with application to low cost sensing, (3) physics based models for time series dynamics of Air Quality data (i.e. there will never be enough sensors on a low cost station to explain variability – we would need a car presence sensor, etc... but we can model most of the dynamics and treat the rest as intermittent forcing), (3) how can we fuse sensing and simulation in order to infer the abundance of chemicals below detectable limits (in particular with application to indoor air chemistry).

Much of the current machine learning landscape is dominated by highly abstract problems like image recognition, language prediction

In particular, this work focuses on three key areas

we should not throw the baby out with the bath water. Nor should we attempt to reinvent the wheel... we can use machine learning to *fill in the gaps* where our current theory can not easily provide a direct link between the data we have and the target we wish to model but a relationship is easily justified on physical grounds. Further, we can use our knowledge of physical laws to impose strong constraints on the space of available models that may significantly improve the capability of machine learning models to extrapolate beyond the boundaries of their training data sets.

make the point that for physical theories are typically only amenable to analytic techniques when sufficiently constrained to domains with high symmetry or where nonlinear interactions can be controlled so that linear effects dominate over bounded of higher order effects (re: perturbation theory). For common real world scenarios where edges are

sharp, functions aren't smooth, and approximating assumptions can not be readily applied, we typically must resort to numerical methods which seek to solve the dynamical laws on computational domains with well posed boundary and initial conditions.

In the first proposed project, we will apply machine learning techniques to provide a sensor calibration where theory suggests

To that end, this work focuses on improving sensing capabilities in three key domains: water quality, outdoor air quality, and indoor air quality.

physical sensing and physics-informed machine learning. The world we inhabit is undergoing rapid and transformative changes, with pressing environmental challenges demanding innovative solutions. At the heart of this quest lies the critical need for a deeper understanding of our complex environmental systems, coupled with the ability to derive actionable insights from the wealth of data at our disposal. This dissertation represents my contribution to addressing these challenges by bridging the gap between the physical world and advanced data-driven methodologies. Through the fusion of physical sensing technologies and machine learning, my research endeavors to unlock profound insights into environmental phenomena, ultimately facilitating informed decisions and sustainable practices in an ever-changing world.

This goal is pursued by applying physics informed approaches together with a suite of sensing and computational technologies, implementing the reusable paradigm of software defined sensors, i.e. physical sensing elements wrapped in a software layer. This software layer can serve a variety of purposes such as calibration and the provision of enhanced or derived data products. It is part of a broader effort in the MINTS-AI laboratory at the University of Texas at Dallas. Where MINTS-AI is an acronym, Multi-Scale Multi-Use Integrated Intelligent Interactive Sensing in Service of Society for Actionable Insights.

Comprehensive environmental sensing is a timely and beneficial endeavor for a variety of reasons. The growing awareness of major environmental issues such as climate change,

pollution, and habitat loss necessitates effective environmental monitoring and management. Comprehensive environmental sensing can provide real-time data on air and water quality, weather patterns, and other environmental factors, assisting in the identification and resolution of environmental issues. This assists in the development and implementation of policies and strategies aimed at reducing environmental impact and increasing sustainability. Given that, for instance, air quality can have significant effects on human health, this has particular societal value.

Comprehensive sensing of the environment can improve decision-making. The real-time and accurate data provided by environmental sensors can aid in informed decision-making regarding various aspects such as traffic management, industrial regulation, and crop planning. For instance, data on air quality can be used to inform decisions about reducing pollution levels, while data on weather patterns can help farmers to plan their crops and reduce water usage. Comprehensive sensing of the environment can be instrumental in emergency response. Real-time data on weather patterns, air quality, water levels and resources, and seismic activity can help emergency responders to prepare for and respond to natural disasters such as hurricanes, floods, and earthquakes. The quick and accurate information can enable effective and timely response, potentially saving lives and reducing the impact of the disaster.

Many advances in technology have enabled the creation of comprehensive sensing systems that can monitor and analyze data from various sensors and devices in real-time. In this thesis we use a range of technologies including autonomous robotic teams [@Dunbabin2012; @Rubenstein2014; @Chen2017], hyperspectral imaging [@Plaza2009; @Li2018; @Zhu2017], mesh networks utilizing the Internet of Things (IoT) [@Gubbi2013; @Atzori2010; @Al-Fuqaha2015], machine learning (ML) [@Goodfellow2016; @LeCun2015; @Jordan2015], edge computing, high-performance computing, wearable sensors and modern high-performance dynamic programming languages such as Julia [@Bezanson2017] designed for numerical and

scientific computing. These technologies have facilitated the collection and processing of large amounts of data from multiple sources, resulting in more accurate and comprehensive environmental monitoring.

Today, the most well known application of machine learning techniques

CHAPTER 2

PHYSICAL SENSING

2.1 Hyperspectral Imaging

The following are notes from the Manolakis textbook that I originally kept here. NOTE: we will need to either make new figures or correctly cite these for attribution.

2.1.1 Hyperspectal Imaging Sensors

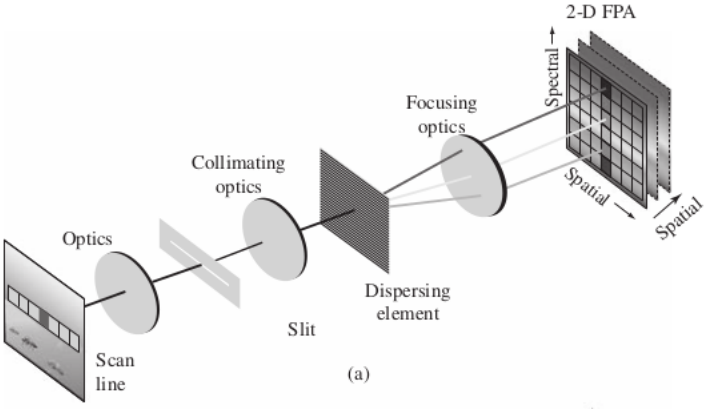
- *Hyperspectral Sensors* aka imaging spectrometers
  - scanning mechanism
  - imaging system
  - spectrometer
- 3 types of resolution
  1. spatial
  2. spectral
  3. radiant
  4. (temporal?)

2.1.2 Spectral-Spatial Data Collection and Organization

Data collected into *datacube* with 2 spatial dimensions, 1 spectral dimension

Different types of rigs:

- Pushbroom scanner (ours)
- Staring System
- Fourier Transform Imaging Spectrometer (FTIS)

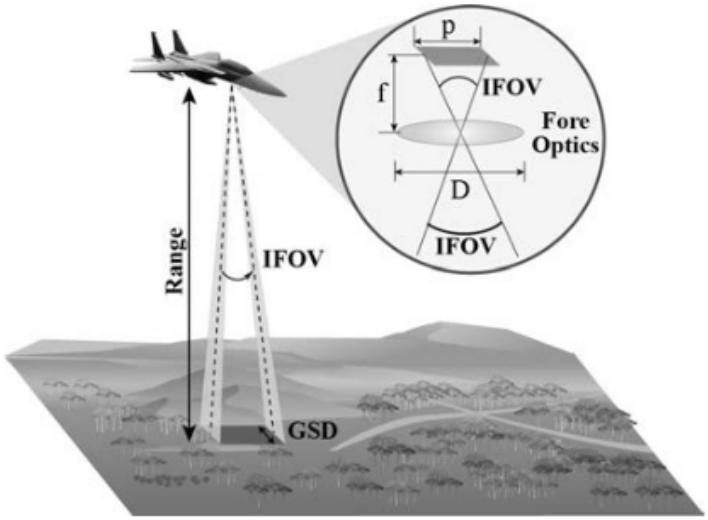


2.1.3 Spatial Sampling

- ground resolution elements are mapped to picture elements (pixels)

- IFOV: Instantaneous Field of View

- Cross track dimension: the projection of the long axis of the slit (i.e. the axis of the pushbroom sensors)
- Along track dimension: the direction accumulated by traveling
- Ground Sample Distance: physical size of projected pixel element



2.1.4 Spectral Sampling

- Recovery of spectral info is imperfect due to finite sampling
- Spectral Response Function: is the weighing function that describes the wavelengths that are transmitted to a particular spectral sample

2.1.5 Radiometric Sampling

- detector transforms radiant power to electrical signal
- electrical signal converted to number via analog-to-digital converter
- photon detectors

2.1.6 Signal Considerations

Strength of signal is determined by:

- Terrain composition affects amount of radiant energy reflected/emitted from ground resolution element
- Range: Intensity drops off by inverse square law. Further you are away, the worse the signal
- Spectral Bandwidth: output signal of detector element is proportional to spectral bandwidth of the detector
- Instantaneous Field of View: Decreasing IFOV increases spatial resolution but weakens the signal
- Dwell Time: the time required to sweep the IFOV across the ground resolution element, i.e. the time-on-pixel. Longer dwell time $\rightarrow$ more accumulated photons $\rightarrow$ more signal.

2.2 Remote Sensing

mention different types of satellite data platforms (mostly optical), differences in orbits, coverage, etc... Also good to discuss the increasing use of drones for a variety of applications including intelligent agriculture, geophysics, mapping, etc...

2.2.1 Infrared Sensing Phenomenology

Main passive sources of EM radiation for remote sensing are light emitted by the sun and the self-emission via black-body radiation of objects due to their temperature.

2.2.2 Sources of Infrared Radiation

- **spectral radiant exitance** power per unit area emitted by the sun. We can treat this as a black body with temperature $5800K$, maximum emittance at $\lambda = 0.50\ \mu m$.
- The Earth is $300K$ with maximum spectral radiant emittance at $\lambda = 9.7\ \mu m$. This is known as the **thermal infrared**

2.2.3 Atmospheric Propagation

- Key parameter is the **path length** of atmosphered traveled through before it arrives at the remote sensing system. Main effects are:
  - **Atmospheric Scattering**: diffusion of radiation by particles in the atmosphere
  - **Absorption**
- Useful remote sensing spectral regions are obtained via the **Transmission Spectrum**.
  - **Reflective Range**: $0.35 - 2.5\ \mu m$. Dominated by solar illumination
  - **Water Absorption**: $0.2 - 2.5\ \mu m$.
- **Atmospheric Windows**: Regions of low atmospheric absorption

2.2.4 Reflectance and Emissivity Spectra

There are three processes that occur when EM radiation meets and interface:

1. **Reflection**: Solar illumination dominates here. Consequently, this part of the spectrum is used to characterize the surface
   - **Specular Reflectors**: Flat surfaces that act like mirrors, i.e. $\theta_i = \theta_r$.
   - **Diffuse (Lambertian) Reflectors**: Rough surfaces that reflect uniformly in all directions.

- **Real Reflectors**: Somewhere between the specular and diffuse.

2. **Absorption**

3. **Transmission**

- Fractions vary as a function of λ
- Remote sensing usually cares about *diffuse* reflectors because this is the dominant type of most materials (water being an exception).
- Reflectance of a material is characterized by its **Reflectance Spectrum**, that is, the percent of incident light reflected as a function of wavelength.
  - Dips in reflectance spectrum are called **absorption features**
  - Peaks are called **Reflectance Peaks**
- **Emissivity Spectrum**: The ratio of radiant emittance at a given temperature to the radiant emittance of a black body at the same temperature.

2.3 Coordinated Robotic Teams

- Remote Sensing: data acquisition, processing, and interpretation of images, and related data, obtained from aircraft and satellites that record the interaction between matter and electromagnetic radiation
- Source: the source of electromagnetic radiation, e.g. the sun, black-body radiation, microwave radar, etc...
- Atmospheric Radiation: The EM radiation propagating through the atmosphere. Moderated by various processes including absorption and scattering

- Earth's Surface Interactions: Amount and spectral distribution of radiation emitted/reflected by the earth's surface. This depends on
  - physical properties of the matter
  - wavelength of EM radiation that is sensed

2.4 In-situ Chemical Sensing in Water

Here we can give an overview of the sensors utilized on the boat for the robot team studies... We should include any information about the uncertainties. We can also comment on their use in fresh v.s. salt water environments, e.g. we should discuss fluorometers, sonar, and any other relevant sensors from the boat

2.5 Low Cost Sensors for (Outdoor) Air Quality

The outdoor part here is optional. Here I seek to provide specific details of the classes of low cost sensors used for PM (optical particle counters based on Mie scattering), thermistors (for tempature), humidity sensors, pressure sensors, gas sensors, etc... This can be a rough outline for those that specifically are used in my research projects.

2.6 The HEART Chamber

CHAPTER 3

MACHINE LEARNING METHODS

NOTE: Good way to start is with the question: What is a model? We can discuss the differences between mechanistic models (i.e. physics equations), their free parameters (constants of nature) and other types of models such as the non-parametric, nonlinear models used in machine learning. Further, we can discuss how machine learning models often display have the desirable trait of being a **universal approximator**. This will naturally lead to a discussion of function expansions (Taylor, Fourier, other polynomial expansions, etc...) and how they scale poorly (exponential) with increasing dimension. Machine learning models essentially allow us to do the same thing: perform a function expansion given data but in a way that can scale well to arbitrary dimension (features) of data. This allows us to build predictive models without necessarily needing to prescribe *all* of the physics. From another perspective, this framework allows us to incorporate our physics knowledge from well-behaved or linearized domains in order to *fit the residual* behavior with a sophisticated data-driven approach.

discuss role in science in particular (i.e. calibration, modeling, etc...)

discuss pushback against use in science and need for methods which simultaneously provide uncertainty bounds. Also describe types of ML e.g. supervised, unsupervised, generative, etc...

this is a good place for the Chihuahua vs Muffin meme... and use this to motivate incorportating physical knowledge into the machine learning process as a key feature for scientific applications... we have more constraints!

also discuss statistical v.s. deep learning

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:

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

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 Exploratory Data Analysis

3.2.1 Correlation

3.2.2 Mutual Information

3.3 Model Training Methodology

3.3.1 Data Sampling

Dr. Lary's method (from Gaussian Process Code) for representative sampling to reduce data size

- Testing holdout set

- k-fold Cross Validation

We should use this subsection to discuss why k-fold cross validation is ideal where data size/time permits so that you can get a sense of each model's dependence on the input data. Similarly, it is important we end the discussion by pointing out that the final **production** model is to be trained on the *entire* dataset after we have evaluated which model is best to use and the ideal hyperparameters, etc...

3.3.2 Model Evaluation

Here we should discuss how we evaluate models, e.g. scatter plots, quantile-quantile plots, histograms, confusion matrices, various correctness metrics, etc...

3.3.3 Hyperaprameter Selection

Discuss grid search vs random search vs Bayesian Optimization, etc...

3.3.4 Occam's Razor and Feature Importance Ranking

I should point out my contributions to various julia packages in MLJ for doing feature importance determination here

Linear Regression

Tree Based Methods

Neural Networks

Perhaps explore ideas from Chris Olah's blog??? From a perspective of basic function composition... as in Rackauckas's blog

Shapley Values

Here we can discuss Shapley Values as a model agnostic method for feature importance rankings with the caveat that their computation can take quite a while...

3.4 Supervised Regression Techniques

3.4.1 Neural Networks

3.4.2 Gaussian Process Regression

Based on my notes from this repo

The following is based on the book **Gaussian Processes for Machine Learning** by Carl Edward Rasmussen and Christopher K. I. Williams (Rasmussen et al., 2006). You can find the free online book here.

To explain/derive the Gaussian Process model for regression, let's first consider a motivating example: **Linear Regression**. We will use this guiding example to derive GRP from a *weight space view*. After this derivation, we will suggest a simpler, but more abstract derivation using a *function space view*. First let's set up some data we can use for training:

The Weight Space View

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

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

- $\mathbf{x}_i$ 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**. 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).

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

and our targets into a target vector

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

so that the full training set becomes

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

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

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

Note, 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

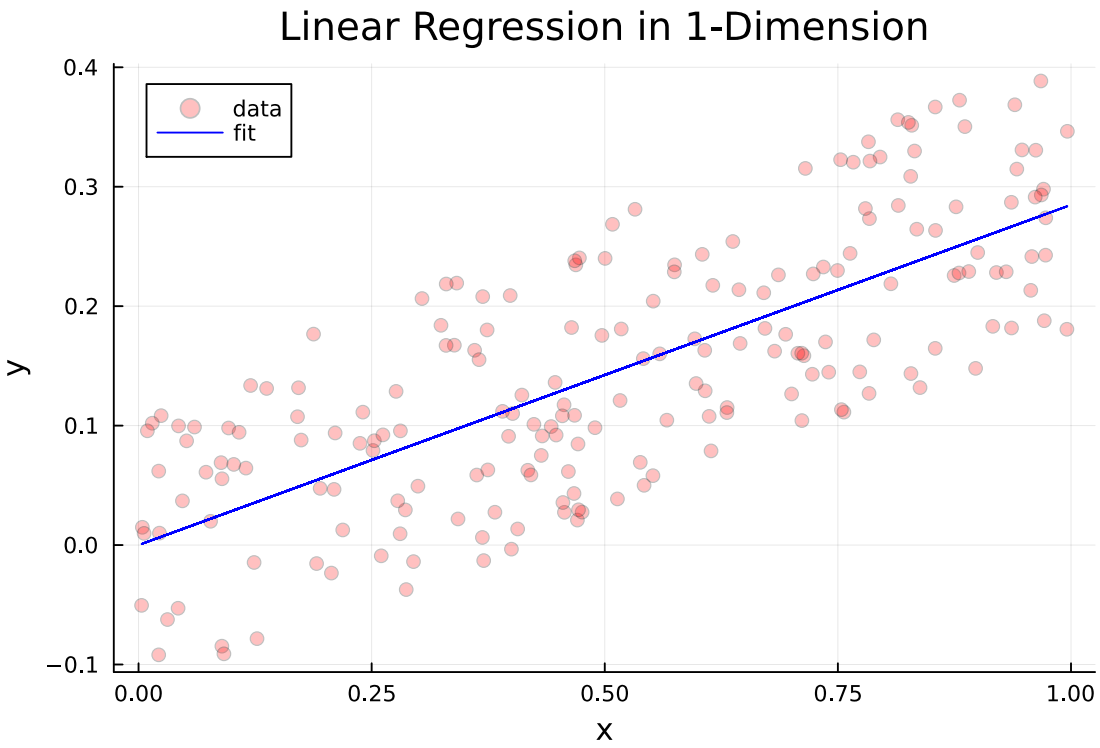


Figure 3.1: A standard linear regression fit on some noisy data.

We can also fit a y-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 are 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

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

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

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

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

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

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

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

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.

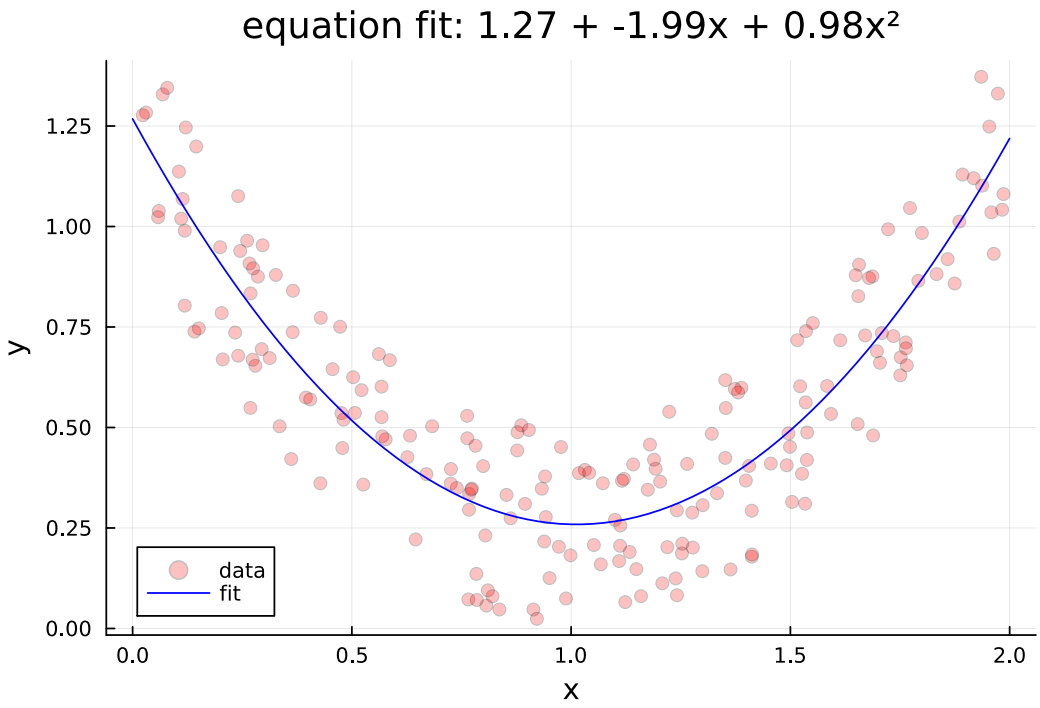


Figure 3.2: An example of polynomial regression for which we fit a quadratic polynomial to some noisy data.

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

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:

$$\begin{aligned}
\frac{1}{\sigma_n^2}\Phi(K + \sigma_n^2 I) &= \frac{1}{\sigma_n^2}\Phi\left(\Phi^T\Sigma_p\Phi + \sigma_n^2 I\right) && (3.71)\\
&= \frac{1}{\sigma_n^2}\Phi\Phi^T\Sigma_p\Phi + \Phi I && (3.72)\\
&= \left(\frac{1}{\sigma_n^2}\Phi\Phi^T\right)\Sigma_p\Phi + \left(\Phi I \Phi^{-1}\Sigma_p^{-1}\right)\Sigma_p\Phi && (3.73)\\
&= \left(\frac{1}{\sigma_n^2}\Phi\Phi^T + \Sigma_p^{-1}\right)\Sigma_p\Phi && (3.74)\\
&= A\Sigma_p\Phi && (3.75)
\end{aligned}$$

From there we see that

$$\begin{aligned}
A^{-1}\frac{1}{\sigma_n^2}\Phi\left(K + \sigma_n^2 I\right) &= \Sigma_p\Phi && (3.76)\\
\Rightarrow \frac{1}{\sigma_n^2}A^{-1}\Phi &= \Sigma_p\Phi\left(K + \sigma_n^2 I\right)^{-1} && (3.77)\\
\Rightarrow \frac{1}{\sigma_n^2}\phi_*^T A^{-1}\Phi &= \phi_*^T\Sigma_p\Phi\left(K + \sigma_n^2 I\right)^{-1} && (3.78)
\end{aligned}$$

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:

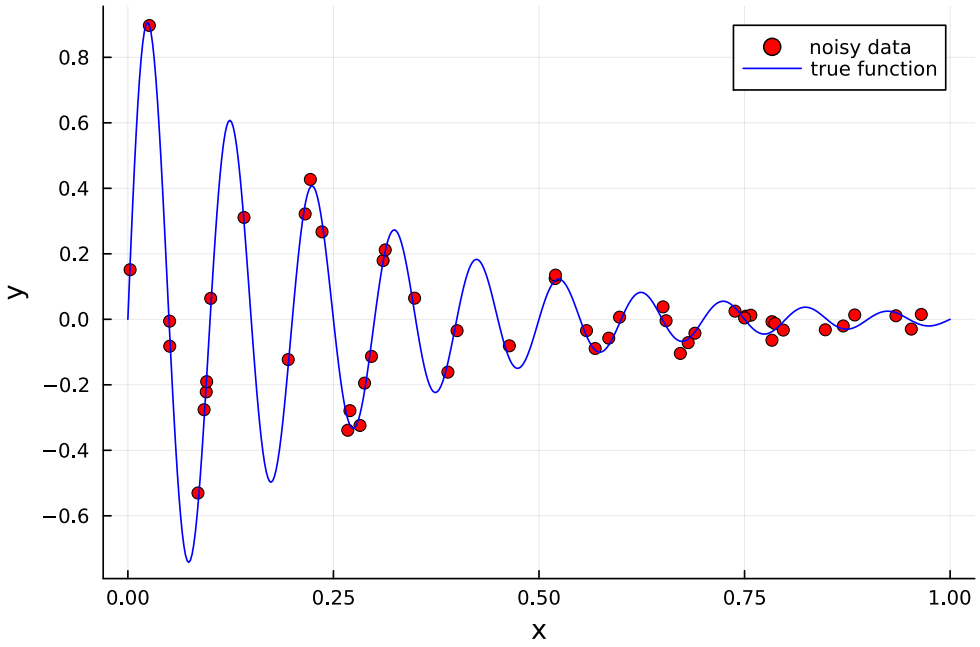
$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) - \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{3.135}$$

which leads to the predictive distribution

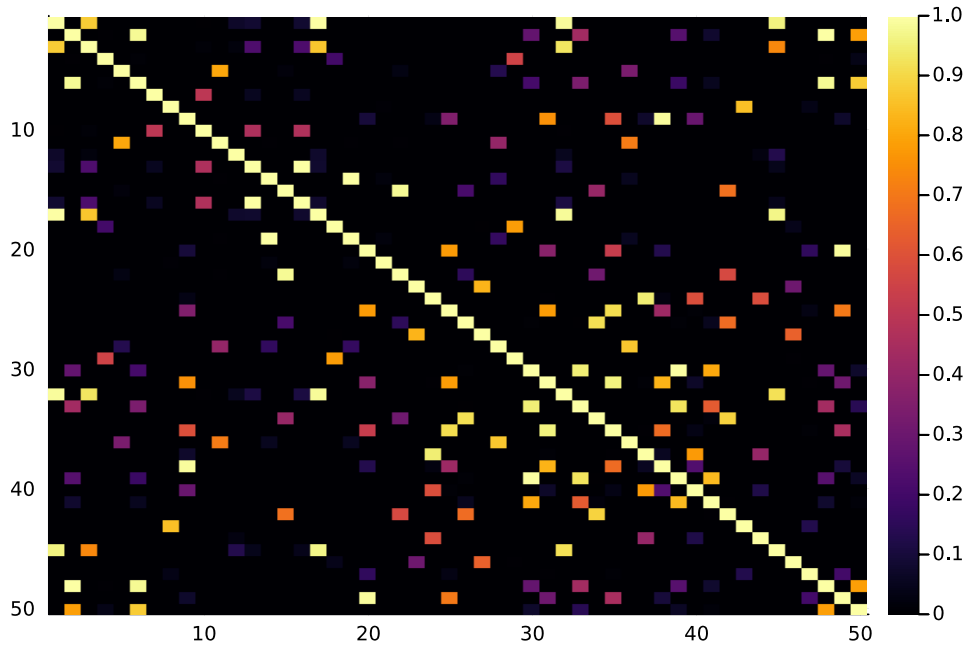
$$\boxed{p(\mathbf{f}_*|X_*,X,\mathbf{y}) = \mathcal{N}\left(K_*^T\left[K+\sigma_n^2 I\right]^{-1}\mathbf{f},\ K_{**} - K_*^T\left[K+\sigma_n^2 I\right]^{-1}K_*\right)} \tag{3.136}$$

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()`.

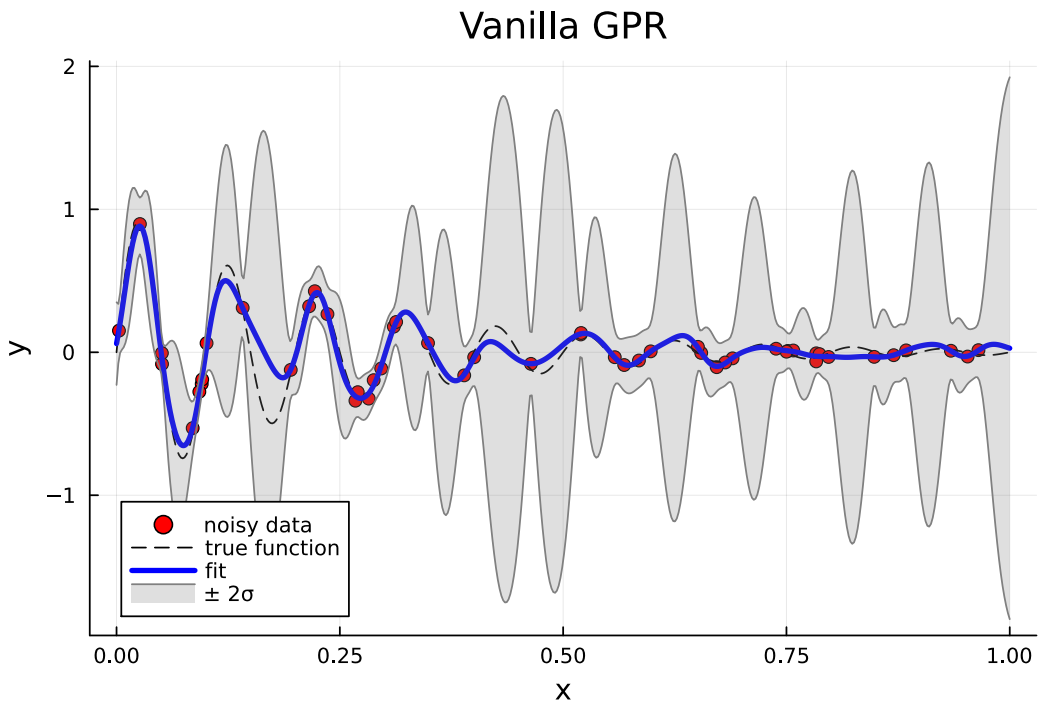


Figure 3.3: A Gaussian Process fit to the training data illustrating the predication (means) and a $\pm 2\sigma$ uncertainty interval. We can clearly see how the uncertainty is larger for inputs far away from supplied training data.

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. See stack exchange link.

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. (Refer to prior section on this in the Theoretical Techniques chapter). 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.

3.4.3 Decision Trees

3.5 Unsupervised Classification

3.5.1 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*

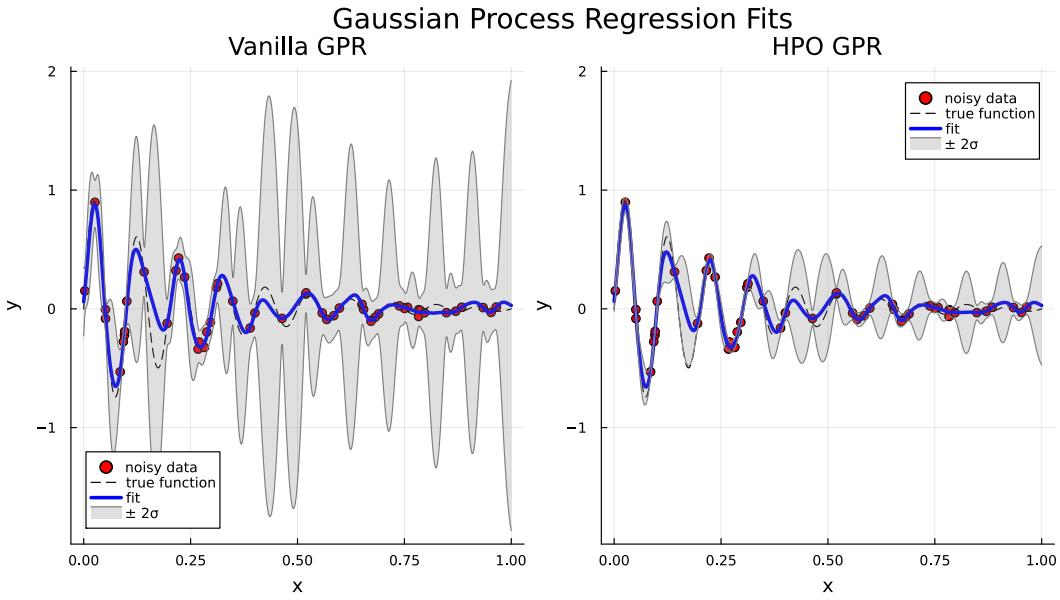


Figure 3.4: Left: the original Gaussian Process obtained with our default choice of kernel parameters. Right: A much better Gaussian Process obtained after performing hyperparameter optimization on the kernel parameters. Note how the mean function still does an excellent job fitting the data points while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

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 unsupervised 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

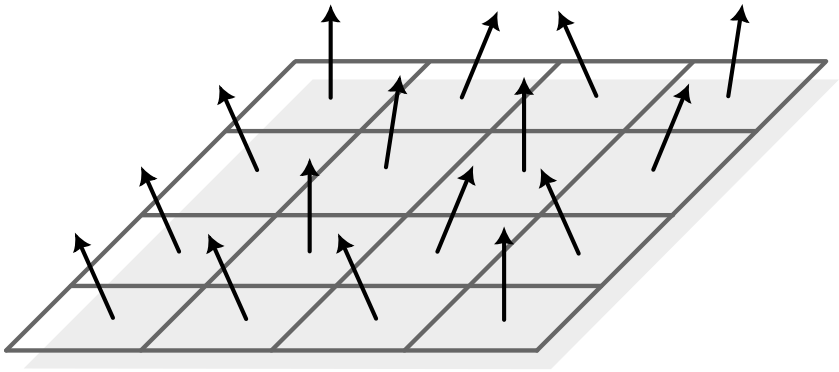


Figure 3.5: A Self Organizing Map with nodes configured in a rectangular grid topology with weight vectors randomly initialized.

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:

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.

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.

3.5.2 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

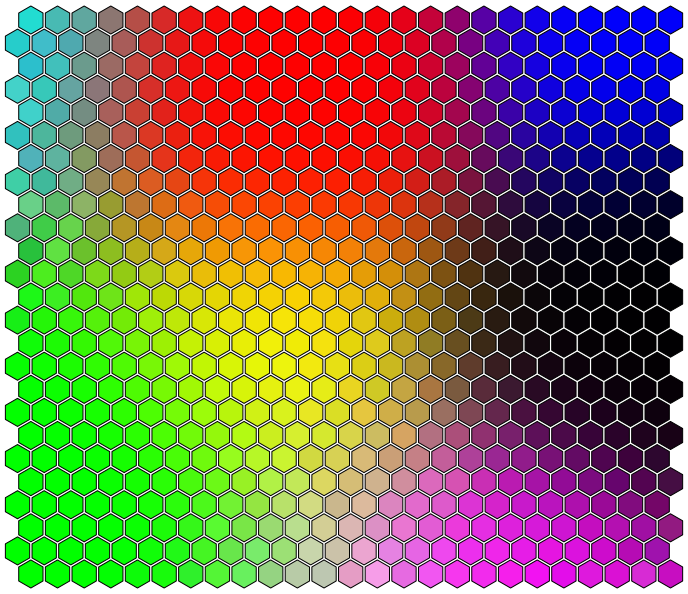


Figure 3.6: A trained Self Organizing Map of 25×25 cells in a hexagonal topology trained on a dataset consisting of the three colors red, green, and blue.

- 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

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

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.

3.6 Model Ensembling

MLJ and SciKitLearn documentation sites will have good references for this, I think.

3.6.1 Bagging

e.g. Random Forests

3.6.2 Boosting

e.g. XGBoost

3.6.3 Stacking

e.g. Super Learners. Use the example of model stacking from MLJ documentation to describe our approach.

3.7 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.

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.

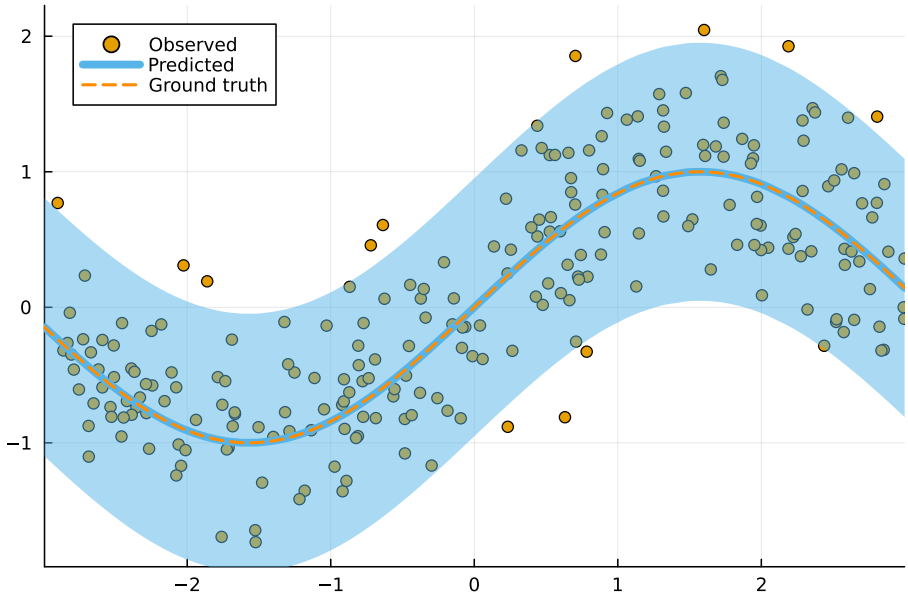


Figure 3.7: An example of conformal prediction for a model estimating a noisy one-dimensional target. The blue shaded region illustrates the learned confidence interval. Image taken from (Altmeyer, 2023)

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.

3.7.1 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

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

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

3.7.2 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.

UPDATE REQUIRED: we should add a couple more paragraphs referencing the key papers that develop this technique as well as a few example use cases it has been applied to.

3.8 Scientific Machine Learning

3.8.1 Physics-Informed Neural Networks

3.8.2 Universal Differential Equations

3.9 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˘ 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):

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

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

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

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

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

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

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

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

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

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

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

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.

3.9.1 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,

$$\begin{aligned}
0 &= \mathbb{E}[u_k^{(t)} - u_k^{(a)}] && (3.236)\\
&= \mathbb{E}[(u_k^{(b)} + \xi_k^{(b)}) - (\nu + K_k w_k)] && (3.237)\\
&= \mathbb{E}[(u_k^{(b)} + \xi_k^{(b)}) - (\nu + K_k H_k u_k^{(t)} + K_k \xi_k^{(m)})] && (3.238)\\
&= \mathbb{E}[u_k^{(b)}] + \mathbb{E}[\xi_k^{(b)}] - \mathbb{E}[\nu] - K_k H_k \mathbb{E}[u_k^{(t)}] - K_k \mathbb{E}[\xi_k^{(m)}] && (3.239)\\
&= u_k^{(b)} + 0 - \nu - K_k H_k u_k^{(b)} - 0 && (3.240)\\
&= u_k^{(b)} - \nu - K_k H_k u_k^{(b)} && (3.241)\\
\Rightarrow \nu &= u_k^{(b)} - K_k H_k u_k^{(b)} && (3.242)
\end{aligned}$$

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.

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

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

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

3.9.2 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.

3.9.3 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

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

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

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

3.9.4 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.*

3.9.5 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

4.1 Rapid Georectification and Processing of Pushbroom Hyperspectral Imagery

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. In this paper, we demonstrate a procedure for the rapid georeferencing of pushbroom imagery and demonstrate its application for real-time remote sensing... Results indicate that a typical image can be processed in 10 seconds while it takes x seconds to capture.

For decades, multi-spectral imagers* have seen wide spread adoption 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 (add references here).

(**add a figure showing the different wavelength bands of publically available satellites**)

These uses are justified by the reflectance features of materials across the electromagnetic spectrum; water has vibrational modes in the IR, pigments have absorption peaks in the

visible, etc...(discuss the particular features present in different regions of the reflectance spectrum). 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 the constant signals infrared to infer the abundance of chlorophyll, and consequently, the health of plants, to identify forest canopy, etc..(**add citations and expand**)

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 chemicals constituents in solution (**add more detail and references**) by deconvolution of a sample spetrum against libraries carefully collected of reference spectra: individual chemicals be uniquely identified by the characterstic location and shape of their absorption features. We should mention something about algal blooms– harmful species display shifted reflectance peaks that can be used to identify toxic species. This information is effectively filtered away by the broad passbands of existing (find the spectra used in that book that show the different species of algae). To that end, *hyperspectral imagers* (HSI), which sample hundreds of wavelength binds at each pixel, have become the natural next step for remote sensing platforms with many planned to deployed in the coming years (**add references about soon-to-be-deployed HSI satellites**).

At the terrestrial level, drones (commonly, quadcoptrs, 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... These capabilities provide significant aid for structural analysis, smart agriculture, etc... Today, kilogram-scale HSI can be comfortably mounted to the payload of film-scale drones such as the AltaX and advancements in spectral sensing such as (**reference Ethan Minot's recent paper**) suggest that sizes of HSI will continue to shrink further expanding their application in this domain.

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 ((**add citation here**)). 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.) (**add citation**). For an HSI platform to function in real time, three key computational tasks are critical: 1. **FileIO**: captured 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 must be avaiable to transmit the final

The first can be readily accomplished by means of light weight, high volume solid state drives. 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 hyper-spectral datacube right away, once can generate the desired data products on board (NDVI for example) and transfer only the relevant information to a ground station

We should now make the argument about the need for improved methods for processing of HSI due the their dramatically larger size, i.e. the standard workflow of collecting imagery and then post processing (re: Open Drone Map) is a fantastic and well tested solution

but prevents the use in time-critical applications. We can also reference the continued improvements to single-board computers such as the raspberry pi 4b (8 Gb of Ram), Jetson family (GPU equipped), and intel nuc (powerful work horses with miniscule form factors). To make real-time possible we need two things: 1. Conversion from raw data (digital counts) to physical units like Reflectance 2. Georeferencing of captured imagery so features can be quickly geolocated

We can always re-process the imagery later for an in-depth reanalysis, but for time-critical applications, we need all of the processing to happen on board the drone.

Add a paragraph about georeferencing in particular: - Georectification = georeference + orthorectification - outline other papers that have discussed georectification techniques: - ground control points (not reasonable for dangerous or water-based environments) - IMU / GPS - different type of sensors: square, pushbroom, whiskbroom

Discuss configurations of imagers and the development of georeferencing strategies, e.g. starting with the Muller paper and going to today. Mention potential for the incorporation of digital elevation maps together enabled by GPUs and video game engines (**reference that one paper that suggests using Unity or something similar for the projective geometry optimized for Cuda on NVIDIA GPUs**).

In our previous work, we demonstrated a prototype autonomous robot team employing a drone based hyper-spectral imager which can learn the mapping from reflectance spectra to concentrations of variety of chemicals-of-concern by utilizing the information contained in the reflectance spectra captured with a HSI together with a (Lary et al., 2021). In this paper, we present a procedure based on the method of (Muller et al., 2002) for the rapid processing and georeferencing of imagery captured by a pushbroom HSI mounted on an autonomous. Associated code can be freely accessed and downloaded in (**add reference to our repository**)

(**This point can be saved for the supervised learning paper**) what we really want (usually) isn't the full spectra at each pixel

IMU - inertial measurement unit

4.1.1 Autonomous Aerial Vehicle

subsection detailing the robot team, and specifically, the drone setup (HSI, NUC, Flir, etc...) - HSI takes images - IMU/GPS onboard position/orientation determination - Intel NUC 1: manages HSI - SSD - Intel NUC 2: Processing - Ubiquity long-range wifi antenna (for streaming data products)

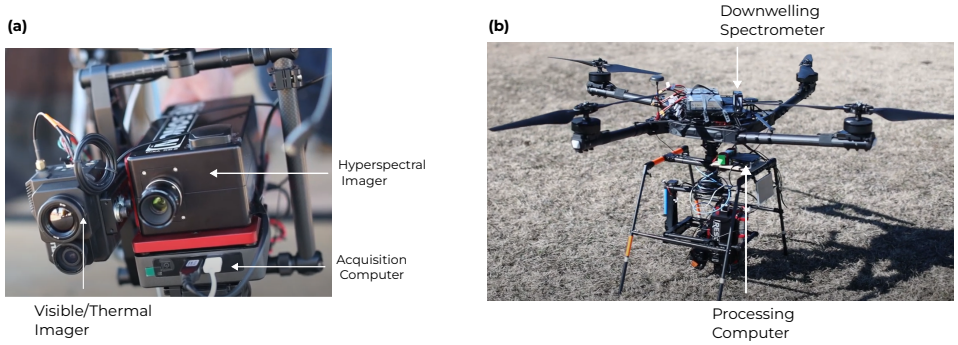


Figure 4.1: Components of the Autonomous Drone HSI platform.

4.1.2 Real time processing of HSI imagery

- overview of processing pipeline

- Collection of raw HSI by imager - Reading stored binary (ENVI) radiance data into tensor - Reading Flight Data - Interpolate flight data from IMU to match HSI times - Reading stored downwelling irradiance spectrum - Interpolate irradiance spectrum to match HSI λs - Conversion to Reflectance under the assumption of a Lambertian surface (perfectly diffuse) - Georectify to obtain new coordinates - resample to regular grid - compute derived λ-metrics (NDVI, etc...) - serve result (save to HDF5 or serve via web map service over wifi)

Georectification Procedure.

- collect datacube

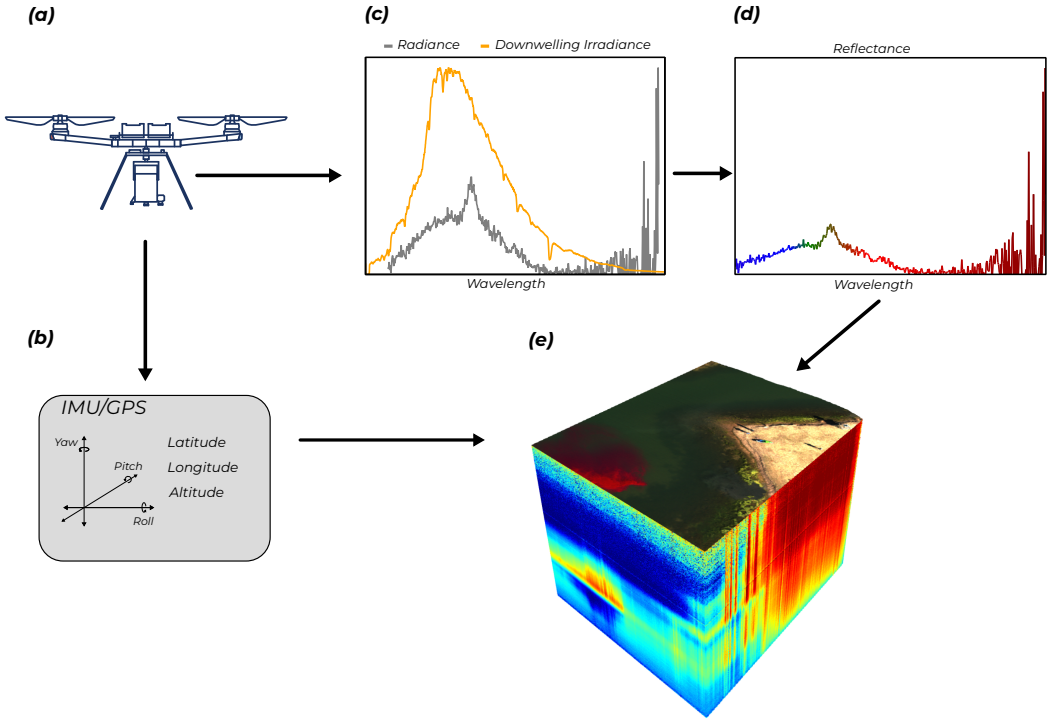


Figure 4.2: Solar Irradiance spectrum captured by downwelling spectrometer.

- read HSI in ENVI format
- read Flight Data
- interpolate flight data to HSI times
- read Downwelling irradiance spectrum
- convert radiance to reflectance
- georectify to obtain new coordinates
- resample to regular grid
- save results to HDF5

Code and data availability via OSN and Github

4.1.3 Georectification Procedure

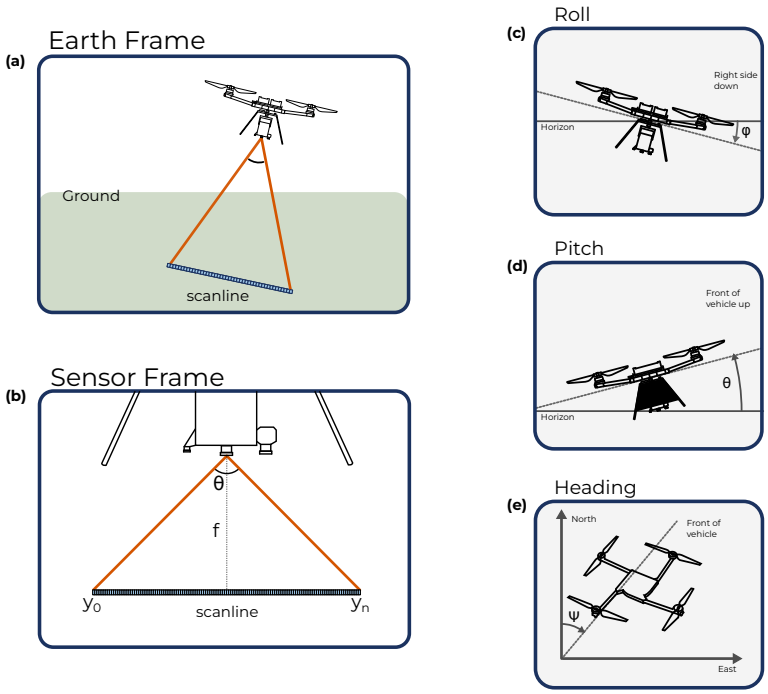


Figure 4.3: Visual representation of scan-line geometry for the drone based hyperspectral imaging platform.

4.2 Results

NOTE: we need to update these timings by first copying the HSI to the local drive (off of the external hard drive). This will probably have some effect on the read/write speeds for the test.

NOTE Stuff to do in this section:

- Create table with processing timings for each step on a typical HSI
- Create second table comparing time for full processing including resampling to a variety of scales (e.g. 5 cm, 10 cm, 25 cm) We will want a version of the processing function that does not include the uncertainties and the spectral indices

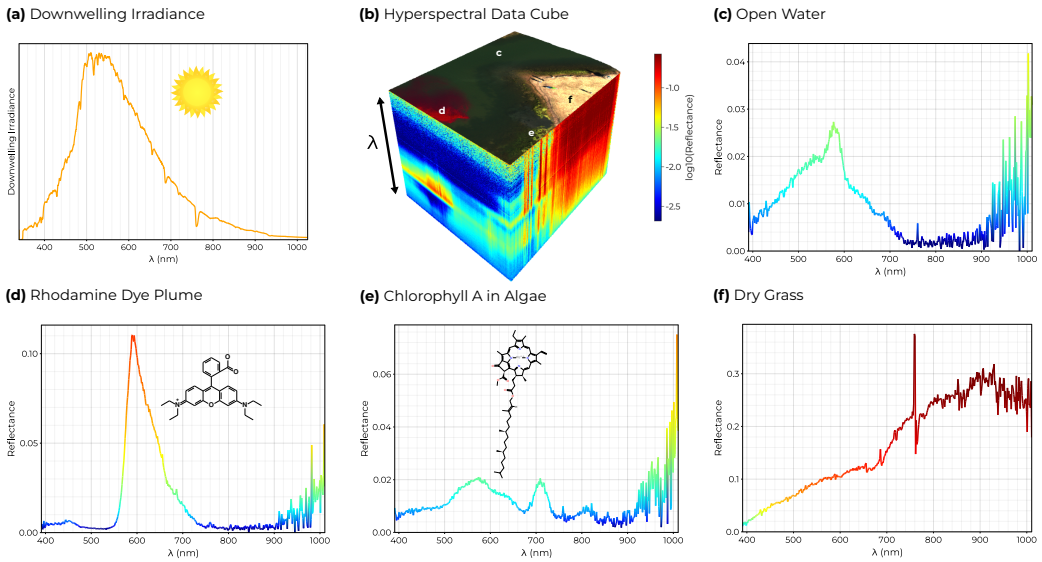


Figure 4.4: Annotated view of a hyperspectral data cube showcasing sampled spectra for a variety of constituents.

- create pipeline visualization showing size of each of the data products and their step in the pipeline (ending with a dataproduct like the NDVI)
- Create a figure showing the georeferenced image on top of the satellite background
- Create another figure of the entire map with a spectral index applied to one of the dye-release datasets (to demonstrate use for real-time application like oil-spill)
- create a figure of a sample HSI datacube in 3-d like from the previous paper with the rgb image on the top.

4.3 Discussion

- Incorporation of drone into autonomous robotic team - Long range wireless network (ubiquiti) for data downlink - Incorporation of edge-ML for direct determination of chemical concentration - Utility for navigation, waste removal, etc...

Discuss drawbacks (striping in images)

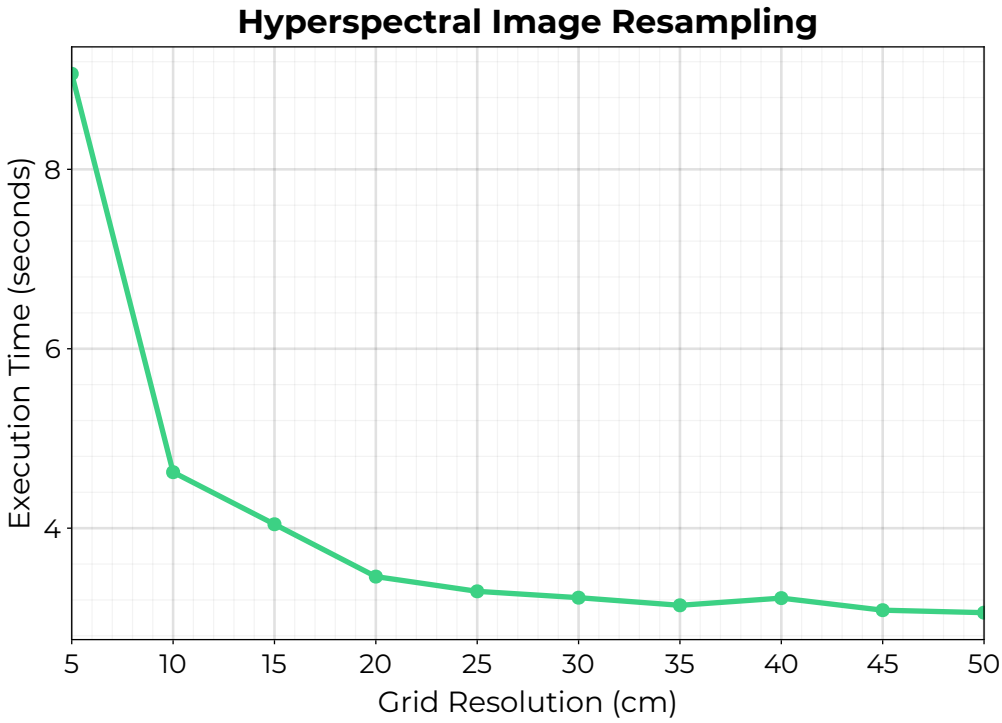


Figure 4.5: This is a figure. Schemes follow the same formatting. If there are multiple panels, they should be listed as: (**a**) Description of what is contained in the first panel. (**b**) Description of what is contained in the second panel. Figures should be placed in the main text near to the first time they are cited. A caption on a single line should be centered.

Authors should discuss the results and how they can be interpreted from the perspective of previous studies and of the working hypotheses. The findings and their implications should be discussed in the broadest context possible. Future research directions may also be highlighted.

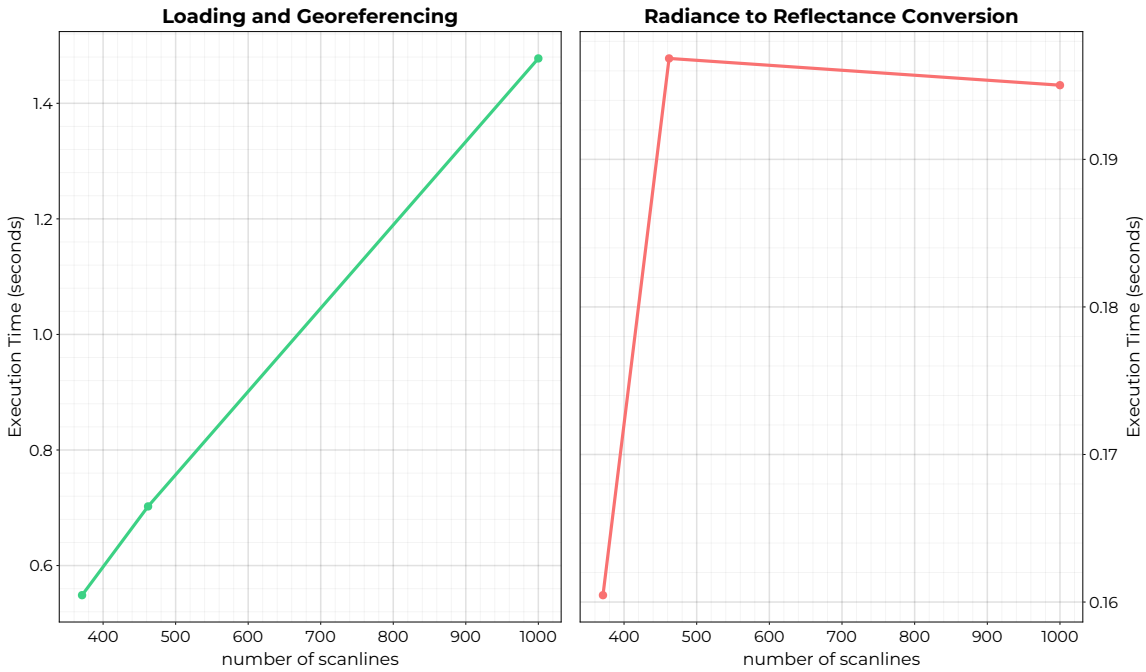


Figure 4.6: Description

4.4 Supervised Regression with Uncertainty Quantification

4.4.1 Hyperspectral Imaging and Remote Sensing

paper creating new spectral index for wheat rust (Zheng et al., 2018)

overview of NDVI (Huang et al., 2020)

paper using spectral indices for desertification (Lamqadem et al., 2018)

HSI sattelite systems

HSI drone systems

Machine Learning using HSI

add Xiao He reference here (Yu et al., 2021)

4.4.2 In-situ Measurements / Robot Teams

Meteorology

Atmospheric Sensing

add in reference to our AQ sensor network

4.4.3 Julia Programming Language

scientific computing

differentiable programming (∂P)

MLJ

discuss general *learning network* framework and utilization of Directed Acyclic Graphs for pipeline construction

basic overview of MLJ as a Julia package (Blaom et al., 2020)

detailed description of *learning networks* (Blaom and Vollmer, 2020)

4.5 Materials and Methods

4.5.1 Autonomous Collection and Processing of Hyperspectral Images

website for alta x (fre, 2021)

$$R = \pi \frac{L}{E} \tag{4.1}$$

4.5.2 Georectification

paper by Muller (Muller et al., 2002)

paper by Baumker (Bäumker and Heimes, 2001)

paper by Mostafa (Mostafa and Schwarz, 2000)

NOTE: need to double check (Muller et al., 2002) for convention used (Φ vs ϕ)

Heading correction due to longitude deviation in UTM system:

$$\kappa_{\text{geodetic}} = \kappa_{\text{geographic}} - \tan^{-1}\left(\sin\Phi \tan(\lambda - \lambda_{cm})\right) \tag{4.2}$$

scale factor:

$$s = \frac{z_{\text{sensor}} - z_{\text{ground}}}{f} \tag{4.3}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{\text{Object}}^{\text{UTM}} = f_{\text{geo}}^{\text{UTM}} \begin{pmatrix} \lambda \\ \Phi \\ z \end{pmatrix}_{\text{GPS}}^{\text{geo}} + sT_{\text{IMU}}^{\text{UTM}} R_{\text{sensor}}^{\text{IMU}}(\kappa, \phi, \omega) \begin{pmatrix} 0 \\ y_i \\ f \end{pmatrix}_{\text{object}}^{\text{sensor}} \tag{4.4}$$

4.5.3 In-situ Data Collection via Robotic Sentinels

autonomous boat website (??, Ott)

data collection process

Data Collocation

CO

CDOM

HDO

Na^{+}

Ca^{2+}

Other Targets

Feature Engineering with Remote Sensing Indices

include discussion of PCA as well as Standardization

4.5.4 Exploratory Data Analysis

Variable Correlations

Machine Learning Models

include Linear Models, Neural Networks, Decision Trees, Bagging, Boosting, and Stacking

The field of Machine Learning has seen the invention of a plethora of models able to perform inference for regression tasks. Naturally, this leads to the challenge of evaluating which model is best for a given problem. In 1992 David Wolpert developed the notion of Stacked Generalization to divorce arbitrary human choice from the model selection process. [1] The key idea is as follows: fit an instance of each model compatible with your dataset. Each model learns different features of the data so that an adjudicating model (called the

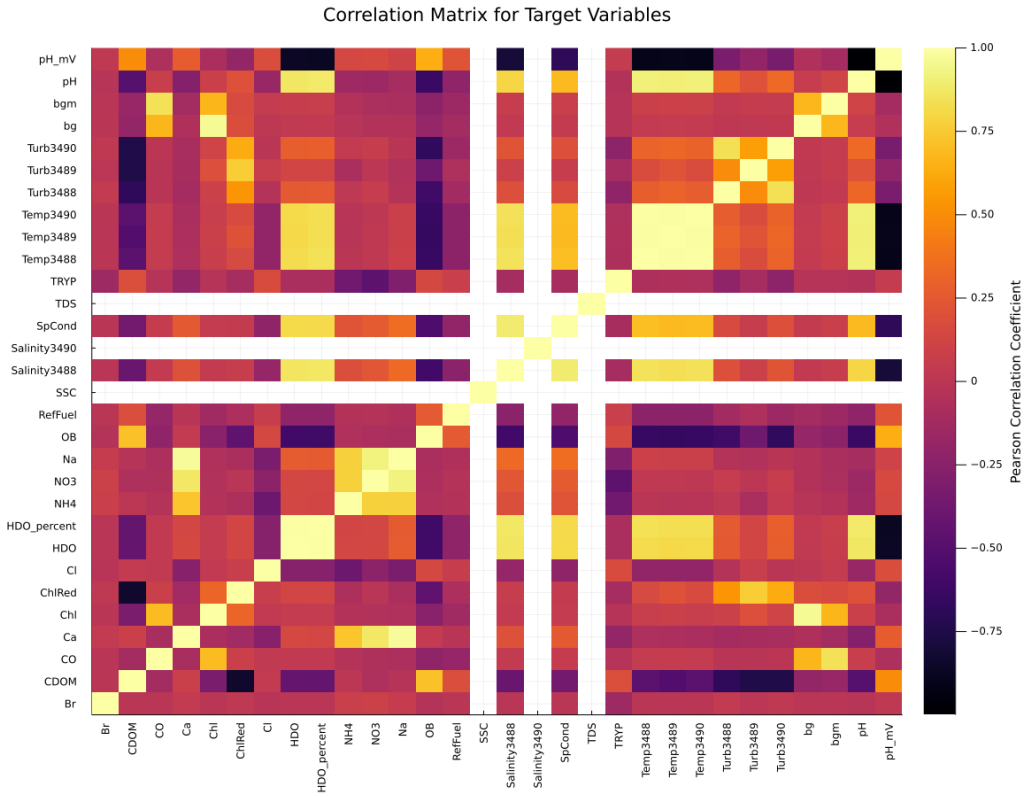


Figure 4.7: Correlation Matrix for Target variables measured by the boat.

super-learner) can then be trained to map the predictions from each individual learner to a final output prediction. This is incredibly easy to accomplish in the Julia programming language via the MLJ machine learning framework [2]. We train a variety of learners including artificial neural networks, bagged decision trees, boosted decision trees, polynomial regressors, quantile regressors, LASSO regressors, and K-nearest-neighbors regressors. To ensure proper data hygiene, individual learners are first trained on separate cross validation folds so as not to bias the super-learner. After training the super-learner, each individual model is then trained on the full dataset to produce the best possible model.

paper on stacking (Wolpert, 1992)

4.5.5 Model Analysis

aleatoric and epistemic uncertainty

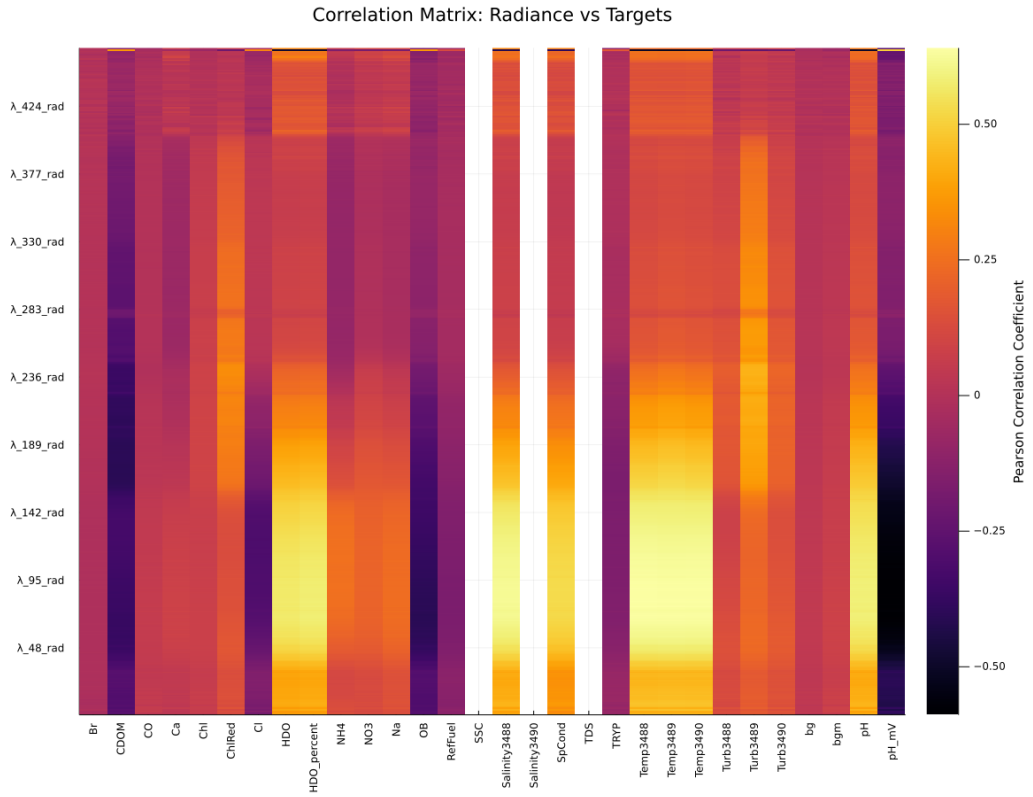


Figure 4.8: Correlation Matrix for Radiance measurements versus Target Variables.

The black-box nature most machine learning methods makes it difficult to consider how errors propagate through a trained model during the inference process. For many problems such as image classification this is not an issue, however, in physical sensing error considerations are vital to establish detection limits and prediction confidence. Typically, there are three considerations we must make: measurement uncertainty– how much we can trust our model's inputs, measurement representativeness– how well each data point represents data within a local neighborhood, and model error– how far the prediction is from the true value. Machine learning models are typically trained to minimize the model error. The Julia programming language makes it possible to handle all three.

To address measurement uncertainty, we take advantage of Julia's advanced automatic differentiation capabilities to enable differentiable programming (P). This means that we can take derivatives of our trained ML model with respect to each input feature to perform

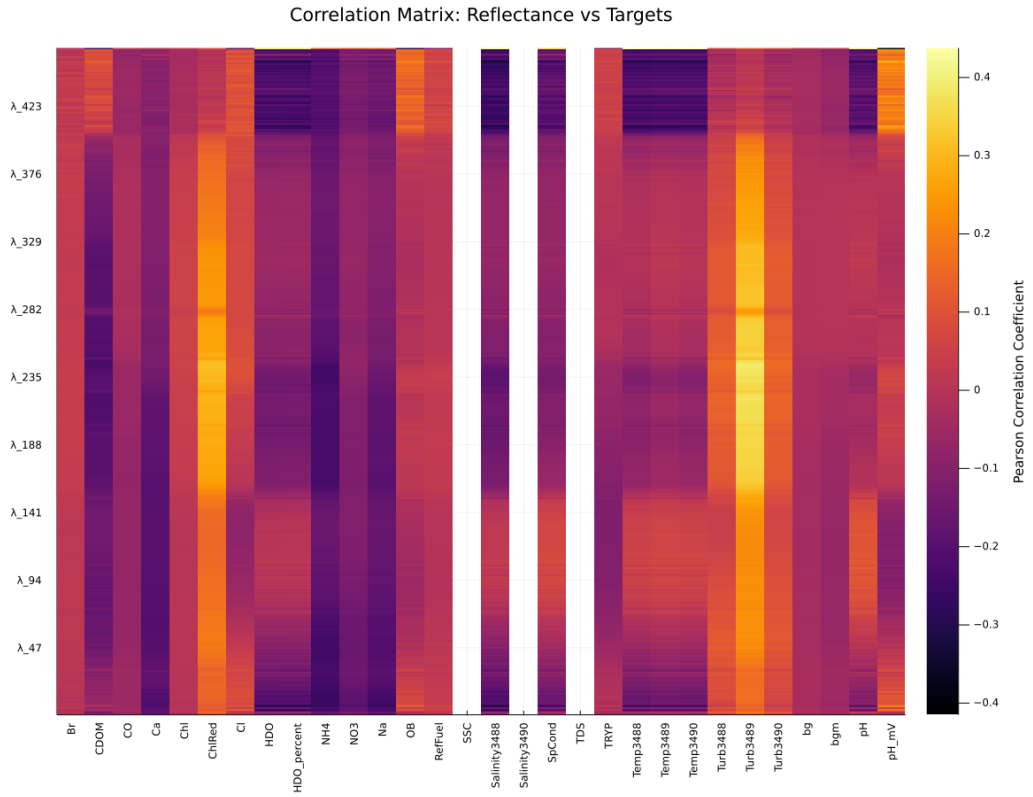


Figure 4.9: Correlation Matrix for Reflectance measurements versus Target Variables.

a sensitivity analysis. The final output uncertainty is then given via linear error propagation theory. [5]

Measurement representativeness can be trickier to handle. For example, in fluids it is common for two species to form a mixing layer that presents as a sharp discontinuity in measurement values. To quantify the degree to which neighboring points diverge we compute the average deviation across input measurements in a neighborhood around each data sample. This information can then be presented with model outputs to identify the regions where our model is valid. Similarly for ensemble ML models we compute the prediction deviation across all base learners to identify the representativeness of a data sample. Inputs that cause a large spread in base learner predictions indicate low representativeness in the training set. This analysis can then be used to plan further data collection missions that minimize time spent collecting data we have already present in our models.

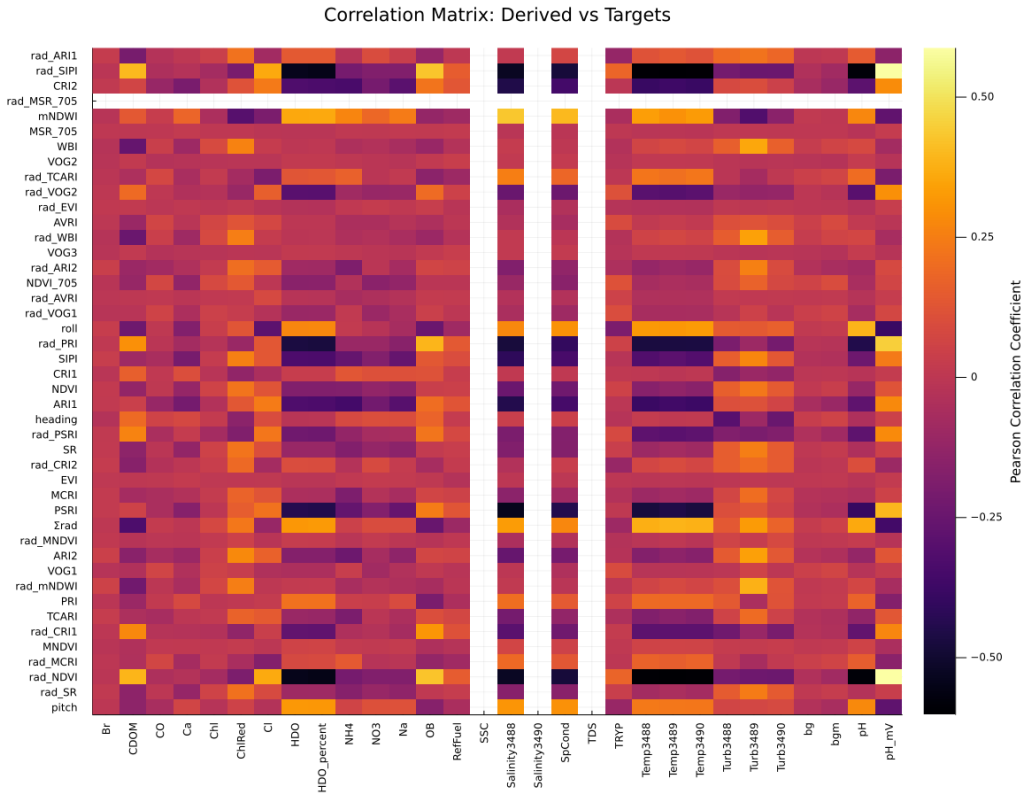


Figure 4.10: Correlation Matrix for derived measurements versus Target Variables.

Uncertainty Propagation via (∂P)

Feature Ranking via SHAP values

shape value paper used in ShapML.jl (Strumbelj and Kononenko, 2013)

Data Representativeness

i.e. the average deviation across pixels used average during collocation process. Also, we can look at relative disagreement between model predictions in bagged ensemble.

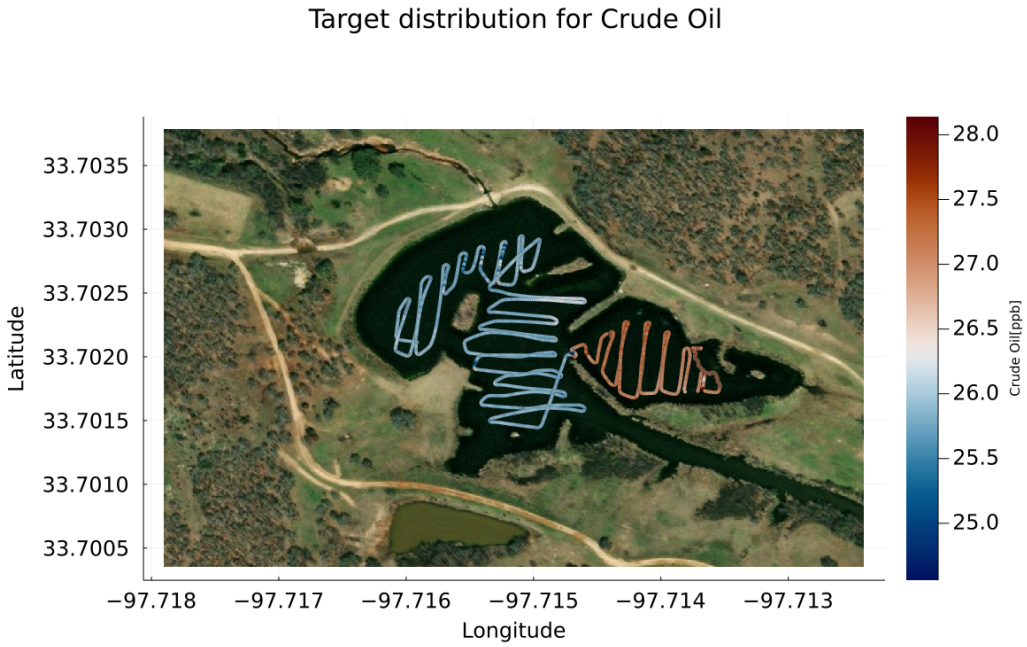


Figure 4.11: Target distribution of CO at Scotty's Ranch.

4.6 Results

4.6.1 CO

4.7 Methods for Unsupervised Classification of Hyperspectral Scenes in Novel Environments

Unsupervised Classification

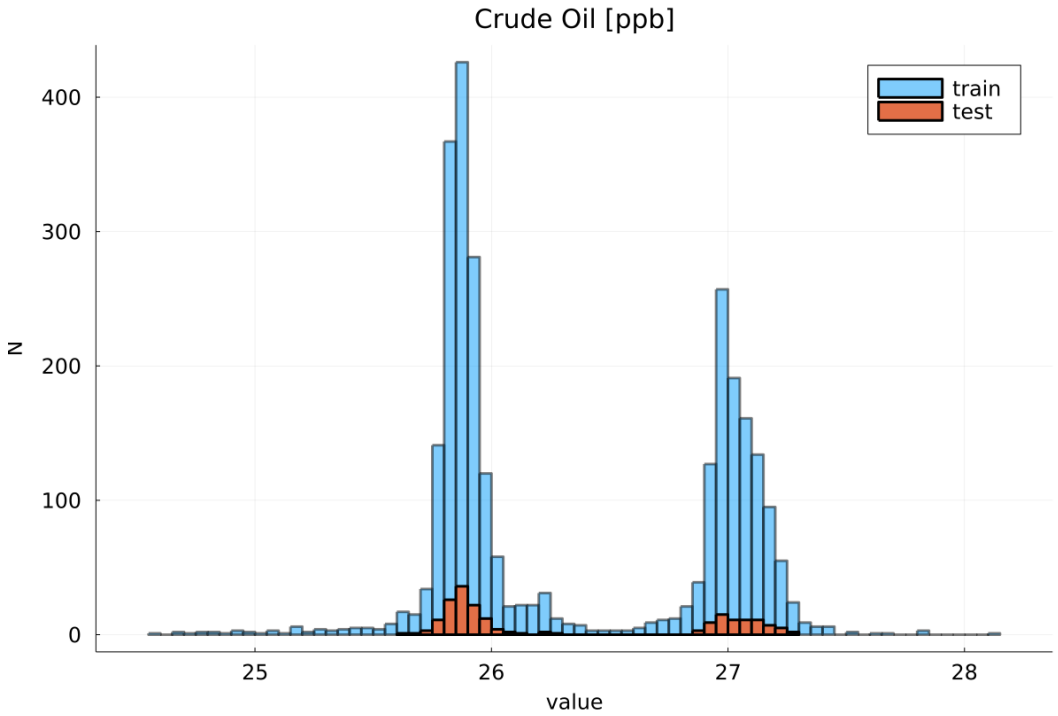


Figure 4.12: Stratified train-test split for CO.

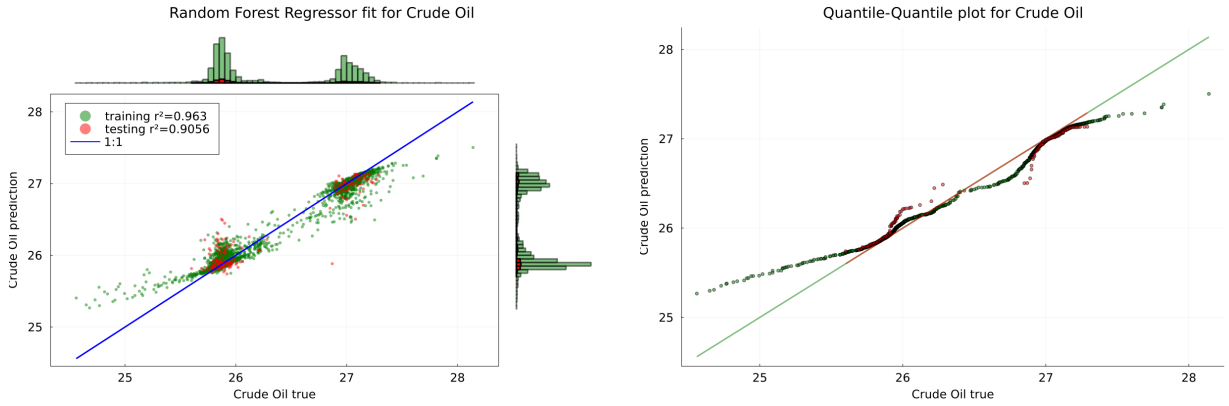


Figure 4.13: (**a**) Scatterplot for the trained random forest Crudo Oil model. (**b**) Quantile-Quantile plot for the same fit.

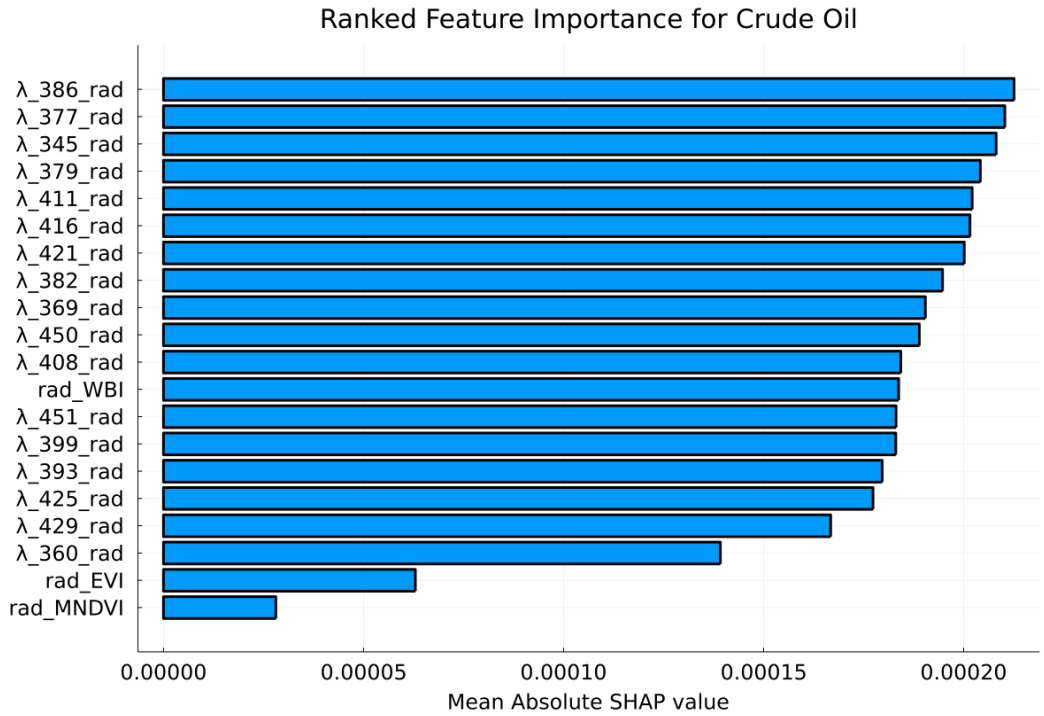


Figure 4.14: Ranked feature importance for Crude Oil. Importance is determined via mean absolute SHAP value.

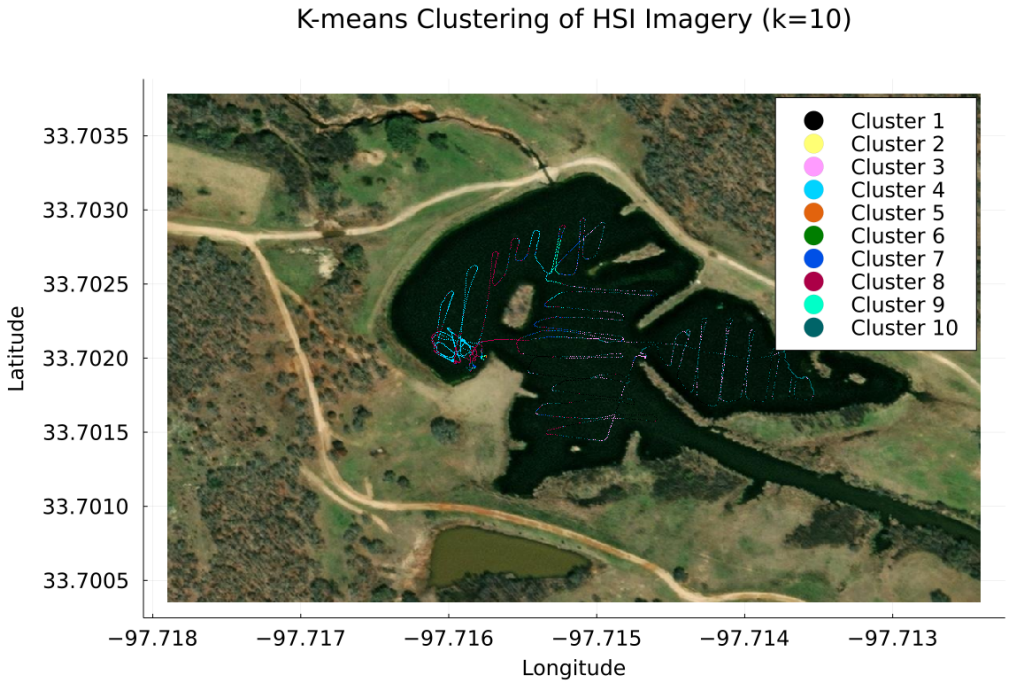


Figure 4.15: Unsupervised clustering of hyper-spectral image data using the K-means algorithm.

CHAPTER 5

A CHEMICAL DATA ASSIMILATION FRAMEWORK FOR INDOOR AIR QUALITY

5.1 Physics of Chemical Reactions: Chemical Reaction Kinetics

5.1.1 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

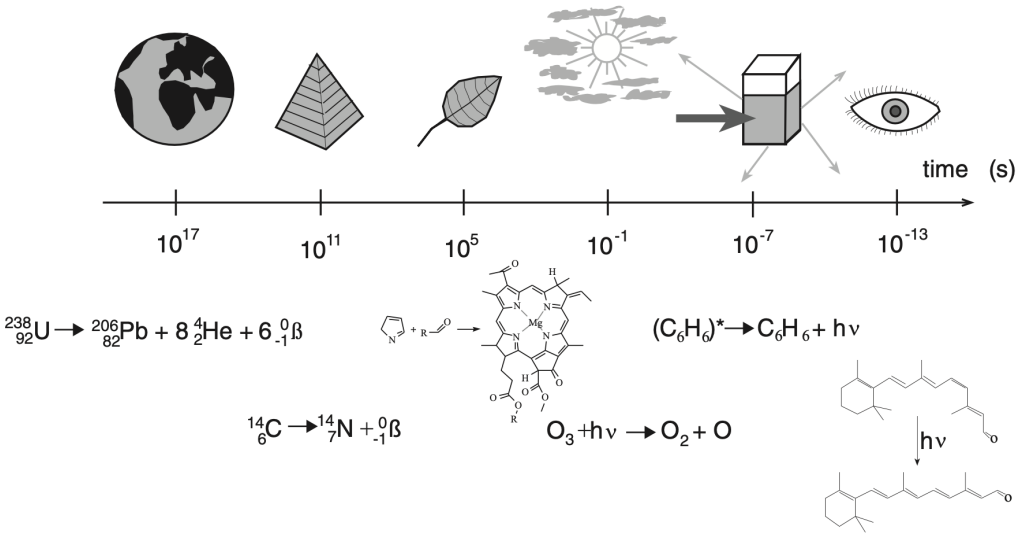


Figure 5.1: An illustration of the broad range of reaction time scales from the long-lived nuclear decay to rapid degradation of molecules by photolysis. Figure taken from (Arnaut and Burrows, 2006)

NASA Jet Propulsion Laboratory (Burkholder et al., 2020) utilize (justified) empirical fits to provide suggested functional forms for reaction rate coefficients.

5.1.2 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

$$G = U - TS + PV, \tag{5.1}$$

which is minimized in equilibrium under constant temperature and pressure.

This equation leads to the convenient thermodynamic identity

$$dG = -SdT + VdP + \sum_i \mu_i dN_i \tag{5.2}$$

from which we may identify the *chemical potential* of the i^{th} species as

$$\mu_i = \left(\frac{\partial G}{\partial N_i}\right)_{T,P,N_{j\neq i}}. \tag{5.3}$$

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:

$$G = \sum_i \mu_i N_i. \tag{5.4}$$

From equation 5.4 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 5.4 together with the identity

$$V = \left(\frac{\partial G}{\partial P}\right)_{T,N} \tag{5.5}$$

to obtain

$$\left(\frac{\partial \mu}{\partial P}\right)_{T,N} = \left(\frac{\partial}{\partial P}\frac{G}{N}\right)_{T,N} = \frac{1}{N}\left(\frac{\partial G}{\partial P}\right)_{T,N} = \frac{V}{N} = \frac{kT}{P} \tag{5.6}$$

so that by integration from a reference pressure, say $P_0 = 1$ atm, we obtain the handy expression for the chemical potential

$$\mu(T,P) = \mu(T,P_0) + kT\ln(P/P_0) \tag{5.7}$$

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

$$0 = dG = \cancel{-SdT} + \cancel{VdP} + \sum_i \mu_i dN_i = \sum_i \mu_i dN_i \tag{5.8}$$

and therefore, a generic reaction of the form

$$\nu_1 X_1 + \nu_2 X_2 + \cdots \leftrightarrow \nu_3 X_3 + \nu_4 X_4 \cdots \tag{5.9}$$

with chemical species X_i and stoichiometric coefficients ν_i satisfies the condition that

$$\nu_1\mu_1 + \nu_2\mu_2 + \cdots = \nu_3\mu_3 + \nu_4\mu_4 + \cdots . \tag{5.10}$$

If we now make use of equation 5.7 with the identification of $\mu^0 = \mu(T, P_0)$, then we obtain

$$\sum_i^{\text{products}} \nu_i\mu_i^0 + \nu_i kT\ln(P_i/P_0) = \sum_j^{\text{reactants}} \nu_j\mu_j^0 + \nu_j kT\ln(P_j/P_0). \tag{5.11}$$

collecting terms involving the μ_k^0 to one side and multiplying through by Avogadro's constant, we obtain

$$RT\ln\left(\frac{\prod_j^{\text{reactants}}\left(\frac{P_i}{P_0}\right)^{\nu_i}}{\prod_j^{\text{products}}\left(\frac{P_j}{P_0}\right)^{\nu_j}}\right) = R\left(\sum_j^{\text{reactants}} \nu_j\mu_j^0 - \sum_i^{\text{products}} \nu_i\mu_i^0\right) = \Delta G^0 \tag{5.12}$$

so that by exponentiation, we arrive at the simple expression:

$$\frac{\prod_j^{\text{products}}\left(\frac{P_j}{P_0}\right)^{\nu_j}}{\prod_i^{\text{reactants}}\left(\frac{P_i}{P_0}\right)^{\nu_i}} = \exp(-\Delta G^0/RT) \tag{5.13}$$

which through further application of the ideal gas law yields

$$\boxed{\frac{\prod_j^{\text{products}}(X_j)^{\nu_j}}{\prod_i^{\text{reactants}}(X_i)^{\nu_i}} = K_{\text{eq}}} \tag{5.14}$$

Here K_{eq} is a temperature dependent constant called the *equilibrium constant* for the reaction, and equation 5.14 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).

5.1.3 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

$$\nu_1 X_1 + \nu_2 X_2 + \cdots \longrightarrow \nu_3 X_3 + \nu_4 X_4 + \cdots \tag{5.15}$$

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

$$\xi(t) = \frac{|N_i(t) - N_i(0)|}{\nu_i} \tag{5.16}$$

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,

$$r := \frac{d\xi}{dt} = \frac{1}{\nu_i}\left|\frac{dN_i(t)}{dt}\right|. \tag{5.17}$$

Manipulating this expression to introduce the volume then leads us to

$$r(t) = \frac{1}{\nu_i}\left|\frac{dN_i}{dt}\frac{V}{V}\right| = \frac{V}{\nu_i}\left|\frac{d[X_i]}{dt}\right| \tag{5.18}$$

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

$$\left|\frac{d[X_i]}{dt}\right| = \nu_i \frac{r(t)}{V} = \nu_i v, \tag{5.19}$$

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

$$A + B \longrightarrow \text{Products.} \tag{5.20}$$

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

$$d_{AB} = r_A + r_B \tag{5.21}$$

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

$$dV = \pi d_{AB}^2 v_A dt \tag{5.22}$$

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

$$\frac{dV}{dt}\frac{N_B}{V} = \frac{\pi d_{AB}^2 v_A N_B}{V} \tag{5.23}$$

and therefore, the reaction velocity for N_A particles with average velocity $\overline{\mathbf{v}}_A$ is

$$v = \frac{\pi d_{AB}^2 \overline{v}_A N_A N_B}{V^2} \tag{5.24}$$

if instead particles of both A and B move relative to each other, then their *relative* velocity is given by the law of cosines:

$$v_{AB}^2 = v_A^2 + v_B^2 - 2v_A v_B \cos(\theta) \tag{5.25}$$

which, since all directions are equally probable, yields an average relative velocity of

$$\overline{v_{AB}}^2 = \sqrt{\overline{v_A}^2 + \overline{v_B}^2}. \tag{5.26}$$

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

$$\overline{v_{AB}} = \sqrt{\frac{8kT}{\pi\mu}} \tag{5.27}$$

Combining everything together yields the reaction velocity

$$v = \pi d_{AB}^2 \frac{N_A}{V}\frac{N_B}{V}\sqrt{\frac{8kT}{\pi\mu}} = \pi d_{AB}^2 [A][B]\sqrt{\frac{8kT}{\pi\mu}} \tag{5.28}$$

which we might further simplify as

$$v = k[A][B] \tag{5.29}$$

$$k = \pi d_{AB}^2 \sqrt{\frac{8kT}{\pi\mu}} = \sigma\sqrt{\frac{8k_B T}{\pi\mu}} \tag{5.30}$$

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

$$k = A \exp(-\alpha/T) \tag{5.31}$$

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:

$$f(v) = \frac{\mu}{k_B T} v \exp\left(-\frac{\mu v^2}{k_B T}\right) \tag{5.32}$$

so that the fraction of particles with speed in the range $[v, v + dv]$ is

$$\frac{dN(v)}{N_{tot}} = f(v)dv = \frac{\mu}{k_B T} v \exp\left(-\frac{\mu v^2}{k_B T}\right) dv. \tag{5.33}$$

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

$$\frac{dN(\epsilon)}{N_{tot}} = \frac{1}{k_B T} \exp(-\epsilon/k_B T) d\epsilon \tag{5.34}$$

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

$$\left.\frac{N(\epsilon)}{N_{tot}}\right|_{\epsilon>E_a} = \int_{\mathbb{E}_a}^{\infty} \frac{1}{k_B T} \exp(-\epsilon/k_B T) d\epsilon = \exp(-E_a/k_b T) \tag{5.35}$$

so that we may justify an additional correction to our reaction rate coefficient

$$k = g\pi d_{AB}^2 \sqrt{\frac{8k_B T}{\pi\mu}} \exp(-E_a/k_B T) \tag{5.36}$$

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

$$\sigma(\epsilon) = \begin{cases} \pi d_{AB}^2 & \epsilon > E_A \\ 0 & \text{otherwise} \end{cases} \tag{5.37}$$

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

$$\begin{aligned} k &= \int_0^{\infty} \sigma(v) v f(v) dv \\ &= 4 \left(\frac{\mu}{2\pi k_B T}\right)^{3/2} \int_0^{\infty} \sigma(v) v \cdot v^2 \exp\left(-\frac{\mu v^2}{2k_B T}\right) dv \end{aligned} \tag{5.38}$$

which in terms of energy yields

$$\begin{aligned} k(T) &= \left(\frac{1}{\pi\mu}\right)^{1/2} \left(\frac{2}{k_B T}\right)^{3/2} \int_0^{\infty} \epsilon\sigma(\epsilon) \exp(-\epsilon/k_B T) d\epsilon \\ &= \left(\frac{1}{\pi\mu}\right)^{1/2} \left(\frac{2}{k_B T}\right)^{3/2} \int_{E_a}^{\infty} \epsilon\pi d_{AB}^2 \exp(-\epsilon/k_B T) d\epsilon \\ &= \pi d_{AB}^2 \sqrt{\frac{8k_B T}{\pi\mu}} \left(1 + \frac{E_a}{k_B T}\right) \exp(-E_a/k_B T) \end{aligned} \tag{5.39}$$

In summary, we have established that for bimolecular reactions, the (simple) theory of molecular collisions allows us to model the reaction velocity by

$$v = k(T)[A][B] \tag{5.40}$$

$$k(T) \approx \pi d_{AB}^2 \sqrt{\frac{8k_BT}{\pi\mu}} \left(1 + \frac{E_a}{k_BT}\right) \exp(-E_a/k_BT) \tag{5.41}$$

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.

5.2 Photolysis

Give an overview for photolysis (perhaps borrow from Dr. Lary's Textbook)

For experimental reasons, the wavelengths (λ) most commonly used for initiating photochemical processes vary between the ultraviolet ($200 - 250$ nm) and the near infrared ($750 - 800$ nm). Light at these wavelengths has an energy which corresponds roughly to $600 - 50$ kJ mol^{-1}, which is very close to the energies of many chemical bonds.

5.3 Summary of Chemical Mechanism Kinetics

Give a summary of kinetics for each of the 3 reaction types. Then finish by writing the form of the combined differential equations for the entire system (and it's Jacobian)

5.4 Characterization of Photolysis

5.4.1 Absorption Cross Sections

5.4.2 Quantum Yields

5.4.3 Photolysis Rate Determination

5.5 HEART Chamber Sensing System

5.6 Chemical Data Assimilation

5.7 An Evaluation of Photocatalytic Ionization

CHAPTER 6

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? I 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. Next, we present two techniques for physics inspired time series modeling: the Hankel Alternative View Of Koopman (HAVOK) method, and the Hamiltonian Neural Network.

6.1 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 manufacturer responsibility for variation in device performance. To address this, we have developed a technique using a data-driven temporal variogram to estimate reasonable values for sensor uncertainties from their time series data.

6.1.1 Uncertainty Quantification with Temporal Variograms

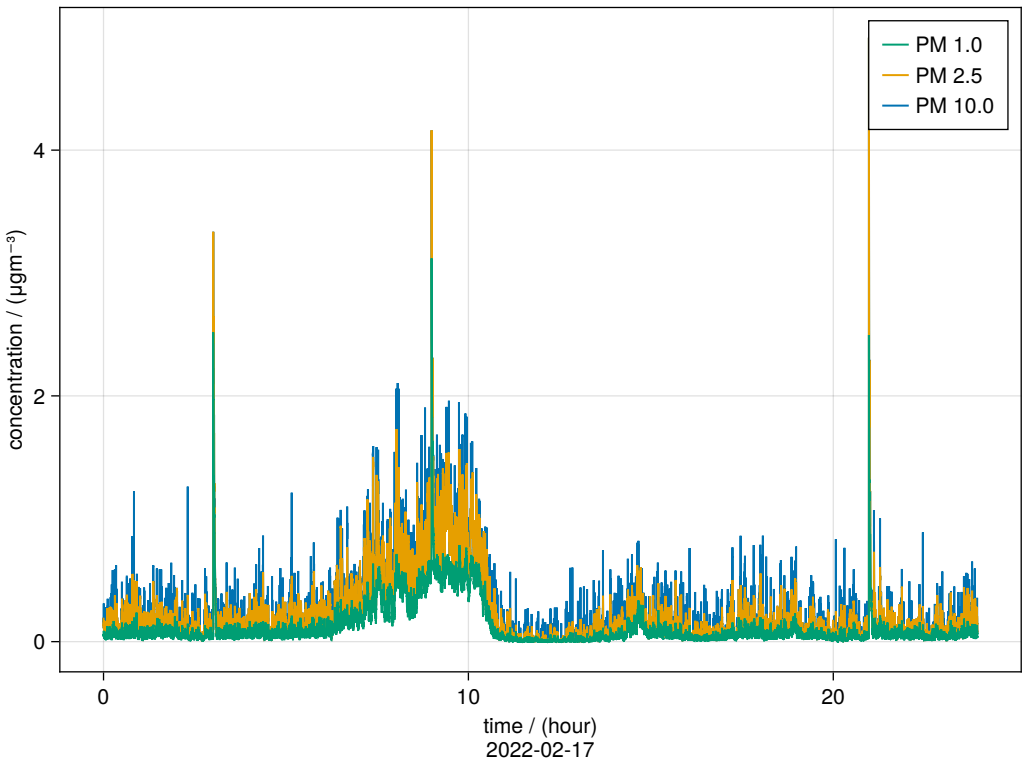


Figure 6.1: Time series of particulate matter at size fractions 1.0, 2.5, and 10.0 μm for a single day

6.1.2 Next Steps

The next step is to apply the methods to examine the trends in uncertainty estimates for each of the models for a longer period of time (like a year) for a handful of sensors. Do we see any trends that suggest seasonal variability in the uncertainty or a general trend that might suggest end of life for the sensors?

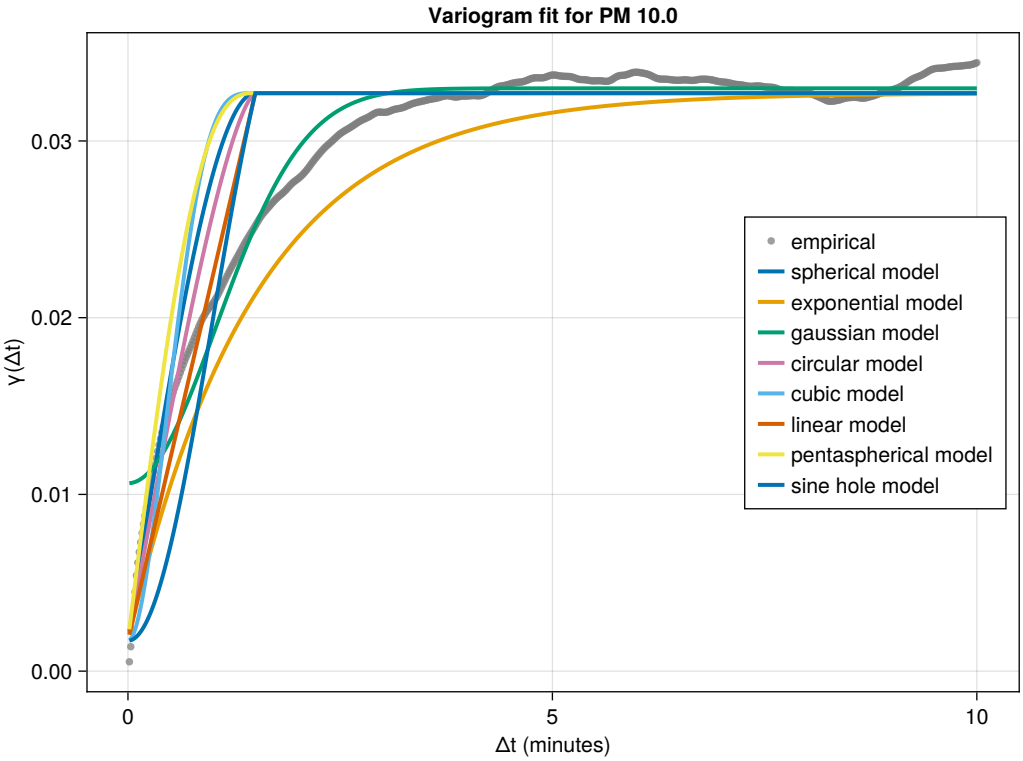


Figure 6.2: The empirical variogram and a variety of model fits obtained for the a PM 10.0 single-day time series

6.2 Physics Informed modeling techniques for Air Quality Data

6.2.1 A Hybrid HAVOK UDE Approach

Motivating example: the Lorenz attractor.

Now what happens if we attempt this procedure on a real, noisy dataset:

Now we can justify the use of a UDE approach to fit missing non-linear terms once we have a trained HAVOK model.

6.2.2 Hamiltonian Neural Networks

An alternative approach to modeling the time series is to consider a system under no external forcing. We can describe the dynamics for this system using *some* set of generalized

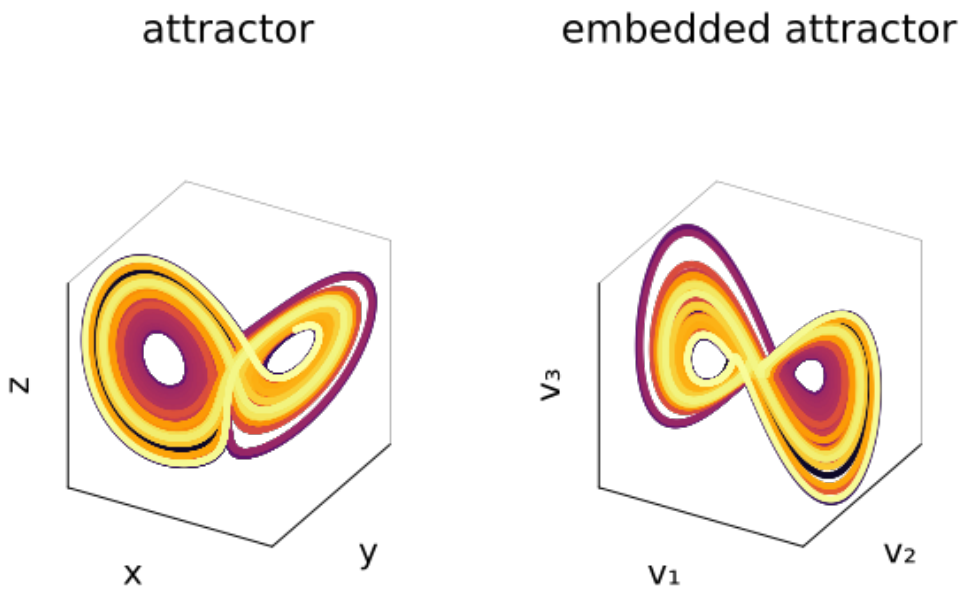


Figure 6.3: Comparing the original Lorenz attractor (left) to the embedded attractor learned after performing the SVD (right). Color indicates the time along the trajectory.

coordinates via Hamiltons equations. Now we suggest to learn an appropriate set of coordinates (q, p) via an auto-encoder structure so that we can then *learn* a Hamiltonian function for which these coordinates satisfy Hamilton's equations for short times. Once we have this model, we can then analyze the motion of our systems state on the Hamiltonian surface. When there is some kind of external forcing driving us away from the level sets of H can we expect to see large jumps in PM concentration? Further, does the Hamiltonian function learned for a single system generalize well to multiple distributed sensors?

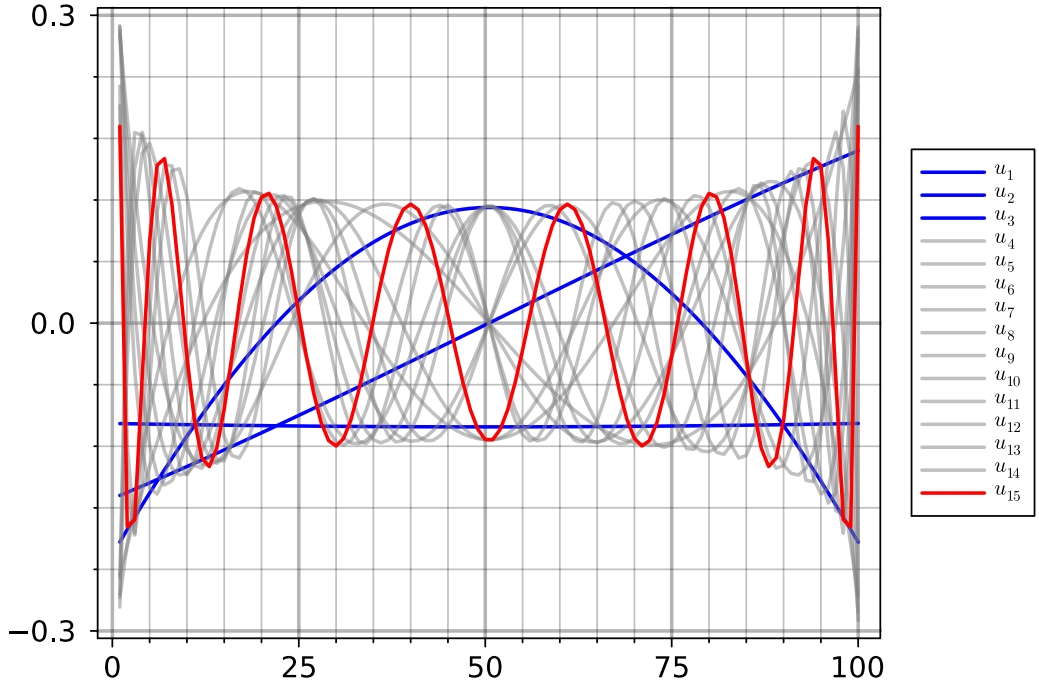


Figure 6.4: Eigenmodes of the embedded attractor extracted from the SVD.

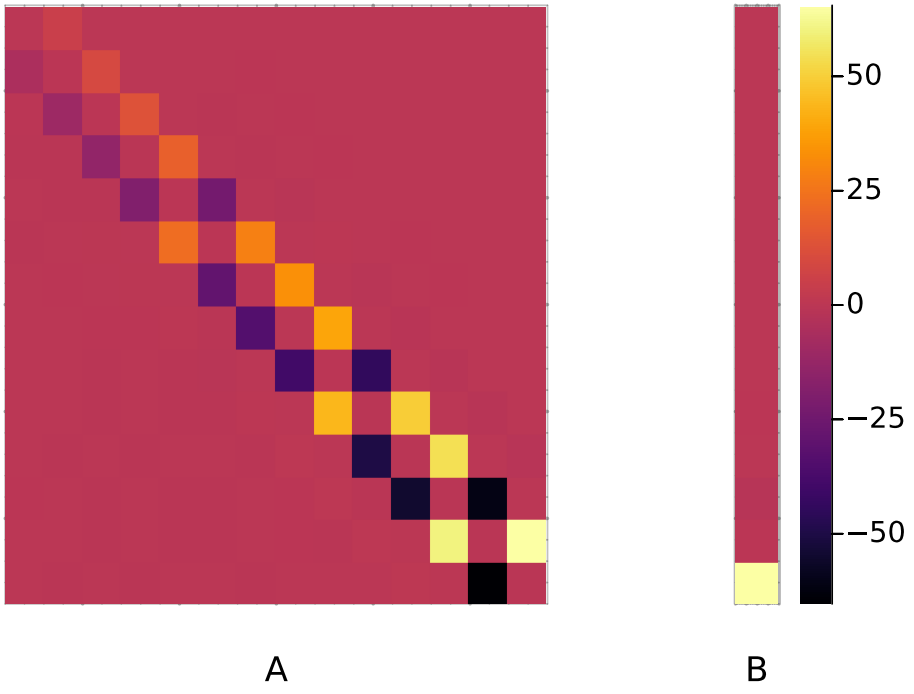


Figure 6.5: Heatmap of the linear Koopman operator together with the forcing activation.

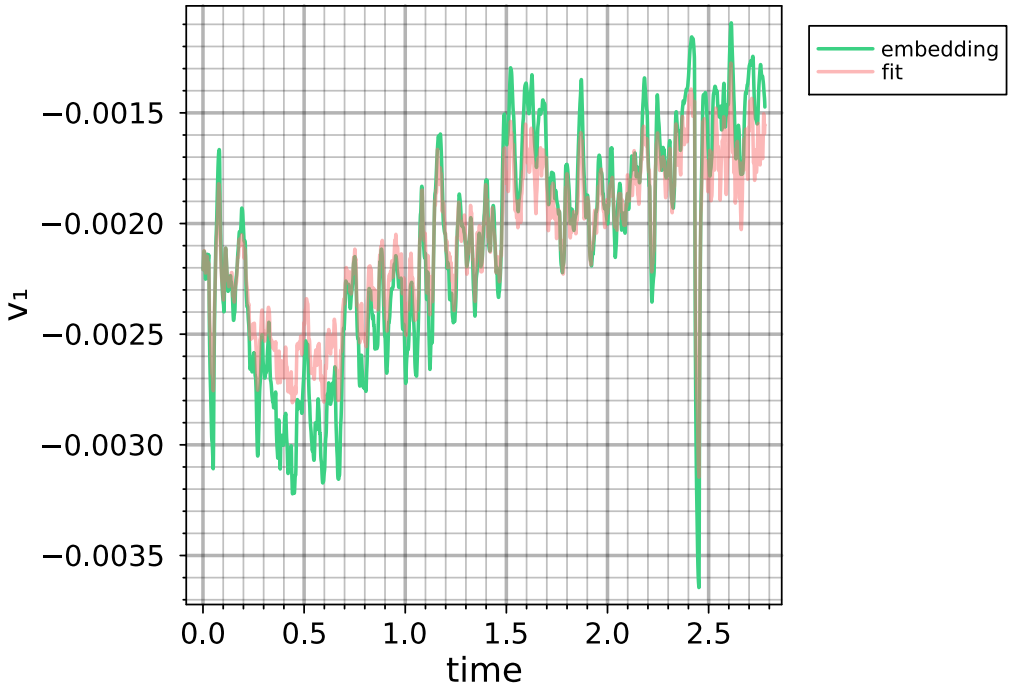


Figure 6.6: The reconstructed timeseries for v_1 via the learned HAVOK model.

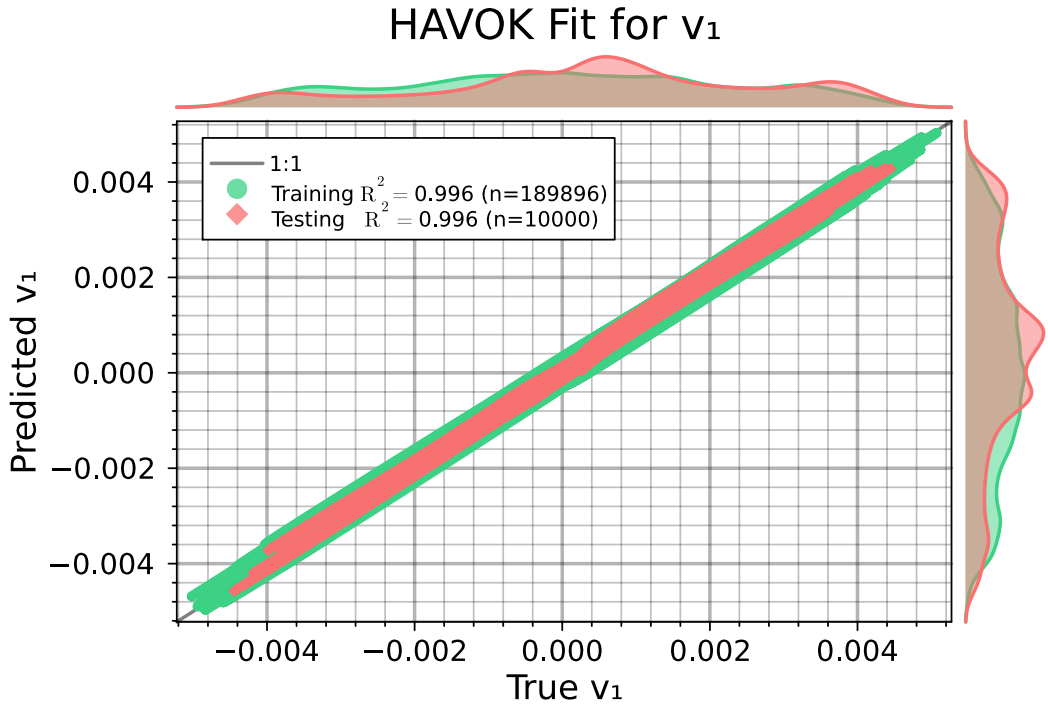


Figure 6.7: A scatterplot of the resulting HAVOK fit

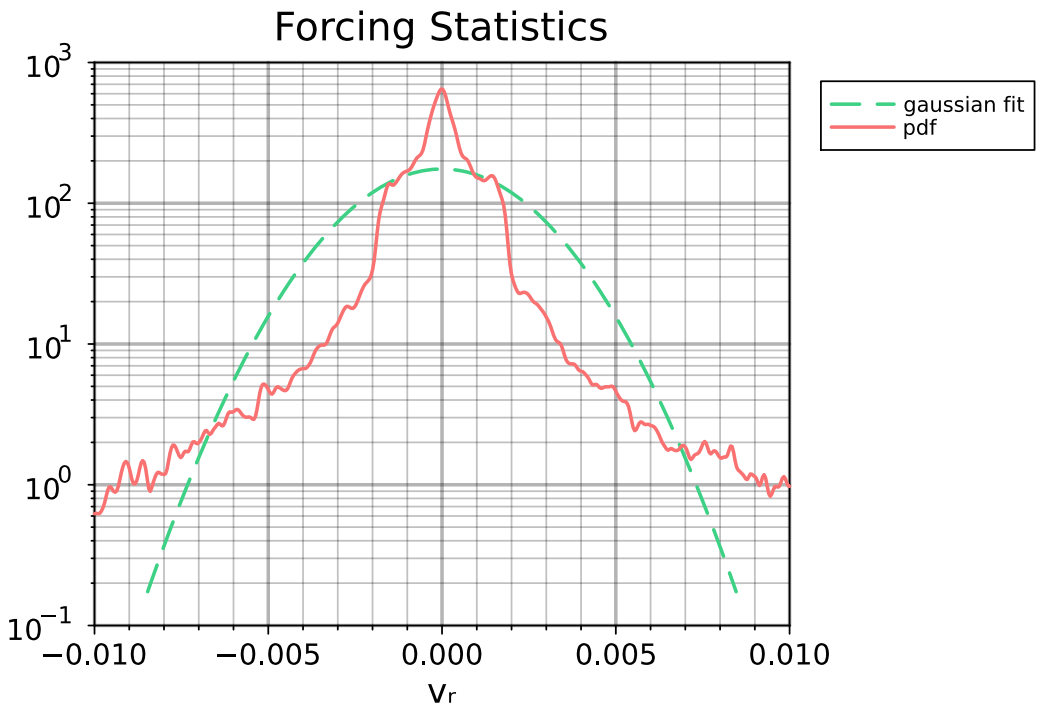


Figure 6.8: The statistics of the learned forcing function. The sharpness of the distribution (in comparison to a Normal distribution) indicates that the forcing is *intermittent*

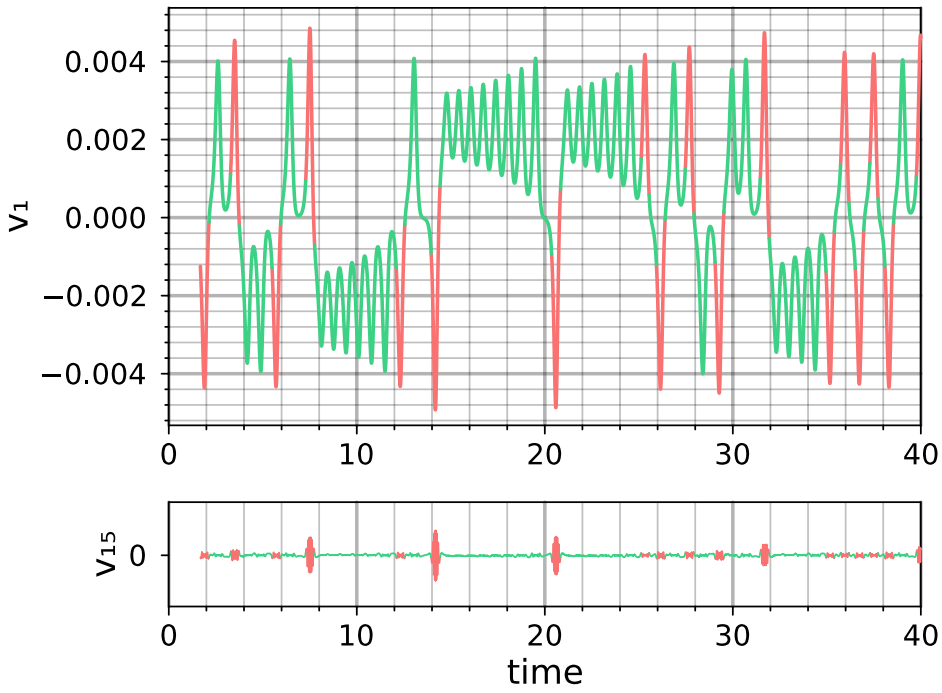


Figure 6.9: The time series for v_1 marked where the forcing function is above a specified threshold.

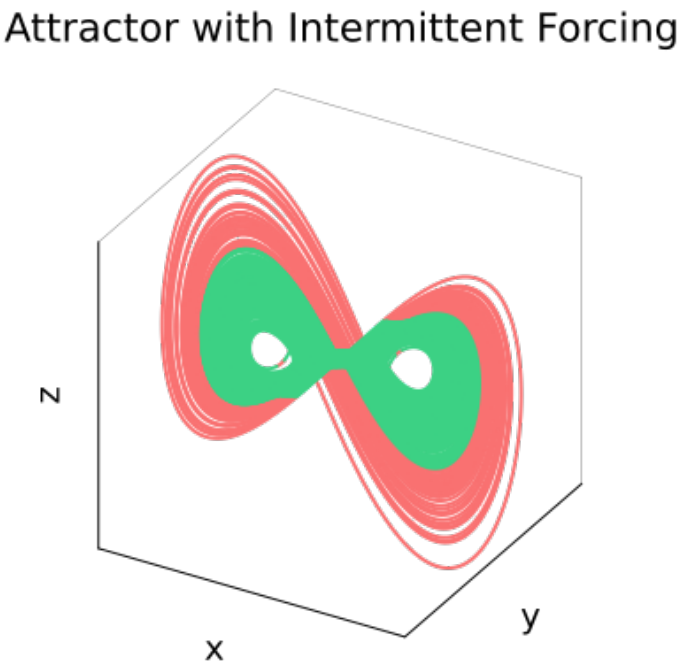


Figure 6.10: The embedded attractor colored by the presence of external forcing.

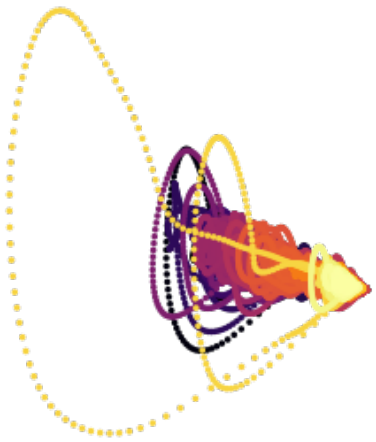


Figure 6.11: The embedded attractor for PM 2.5 time-series data.

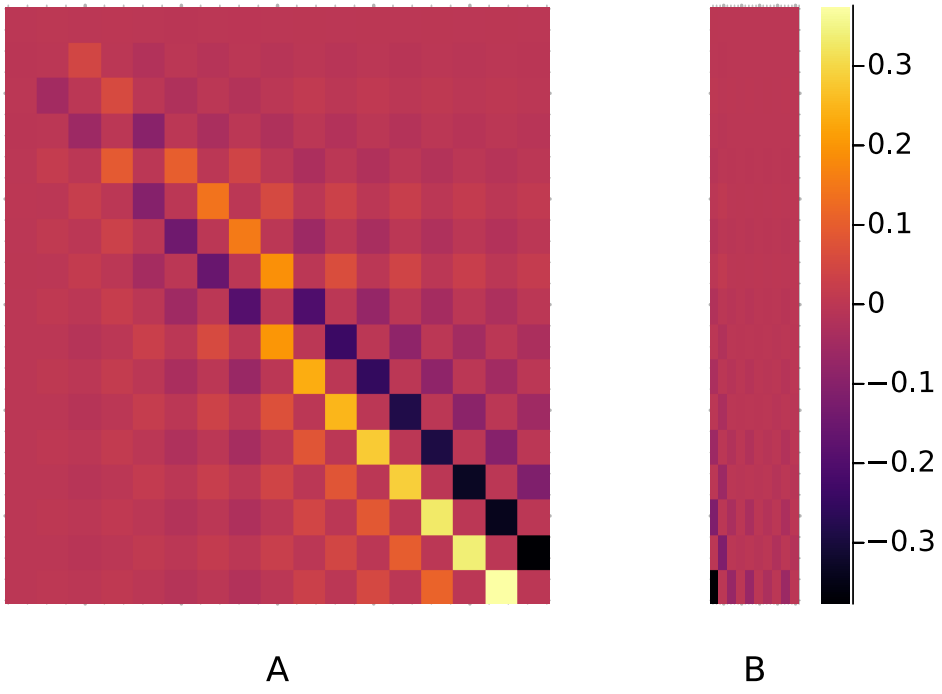


Figure 6.12: Heatmap of the linear Koopman operator together with the forcing activation.

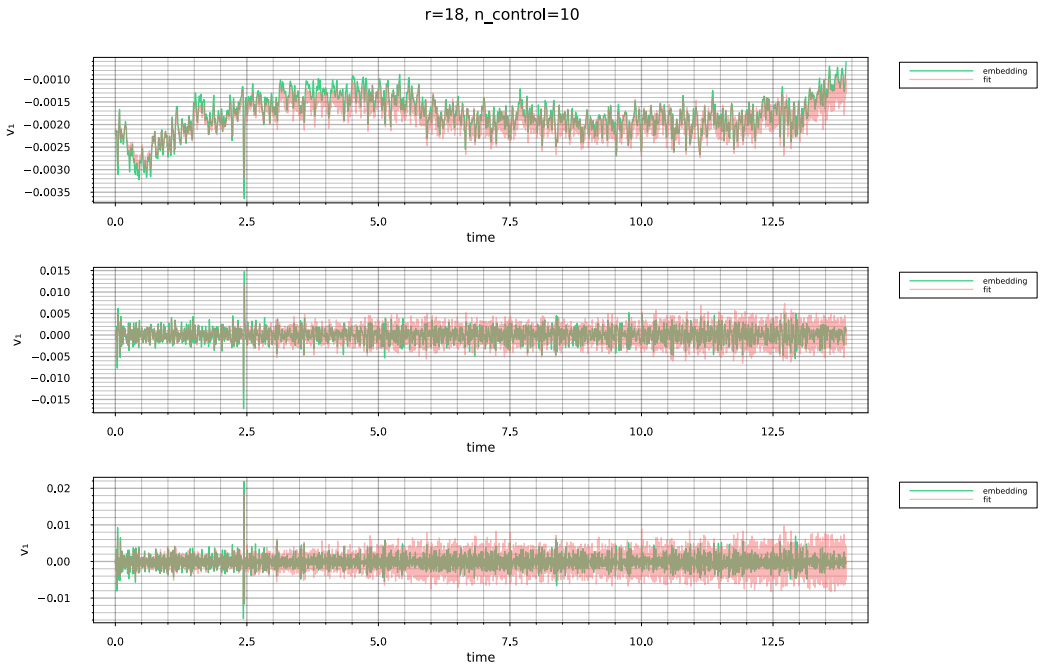


Figure 6.13: The reconstructed time-series for the first three embedding coordinates using the HAVOK fit.

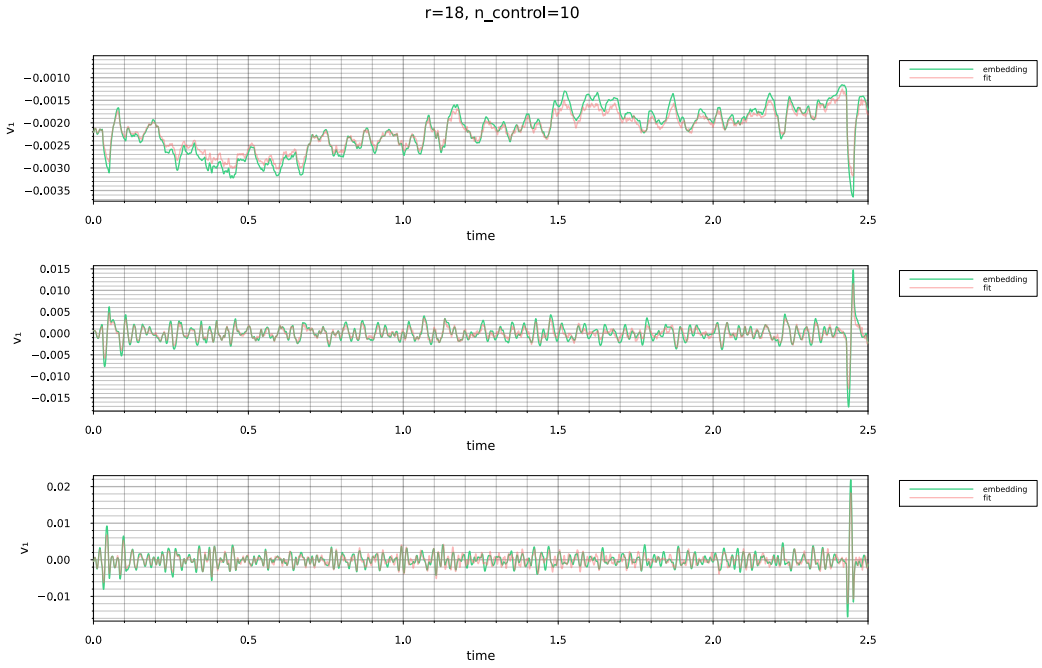


Figure 6.14: A zoomed in view of the same time-series reconstruction for the first 2.5 hours showing a very decent fit. Some kind of error appears to be accumulating over time.

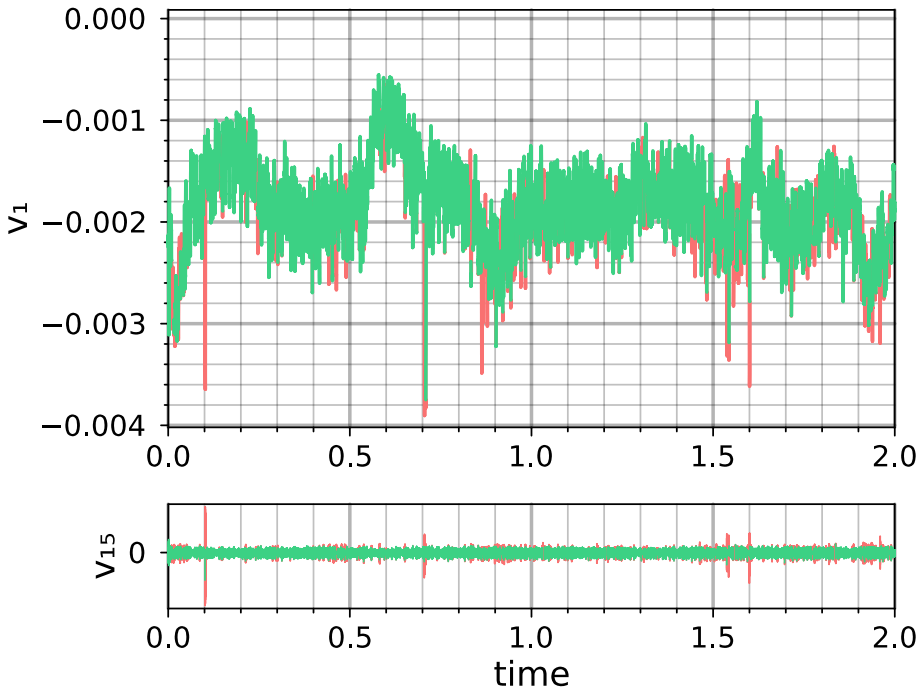


Figure 6.15: The first embedding coordinate with forcing above a critical threshold identified.

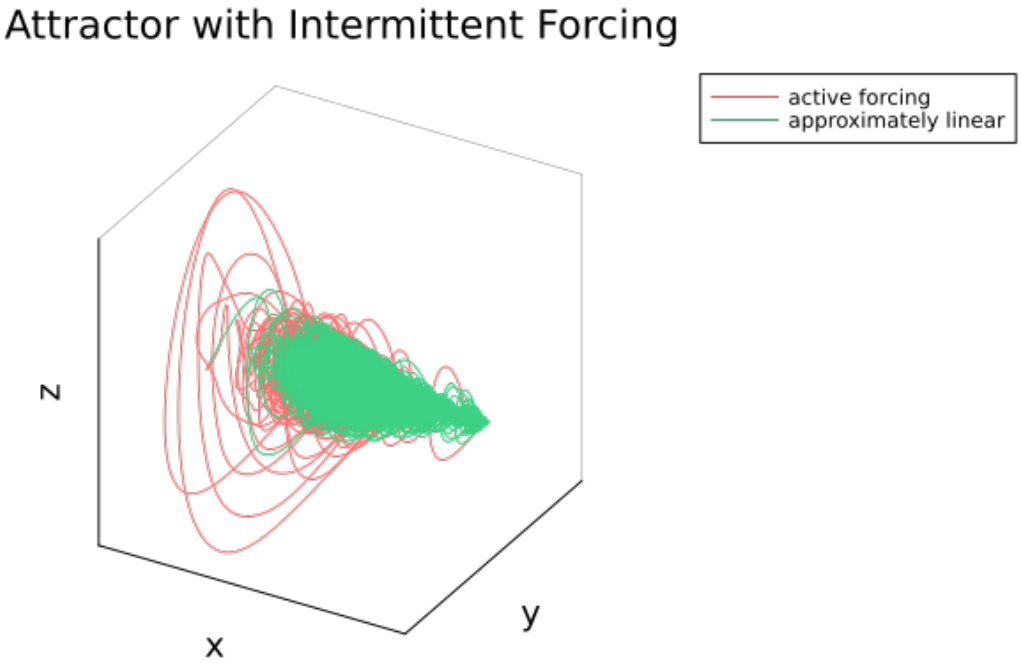
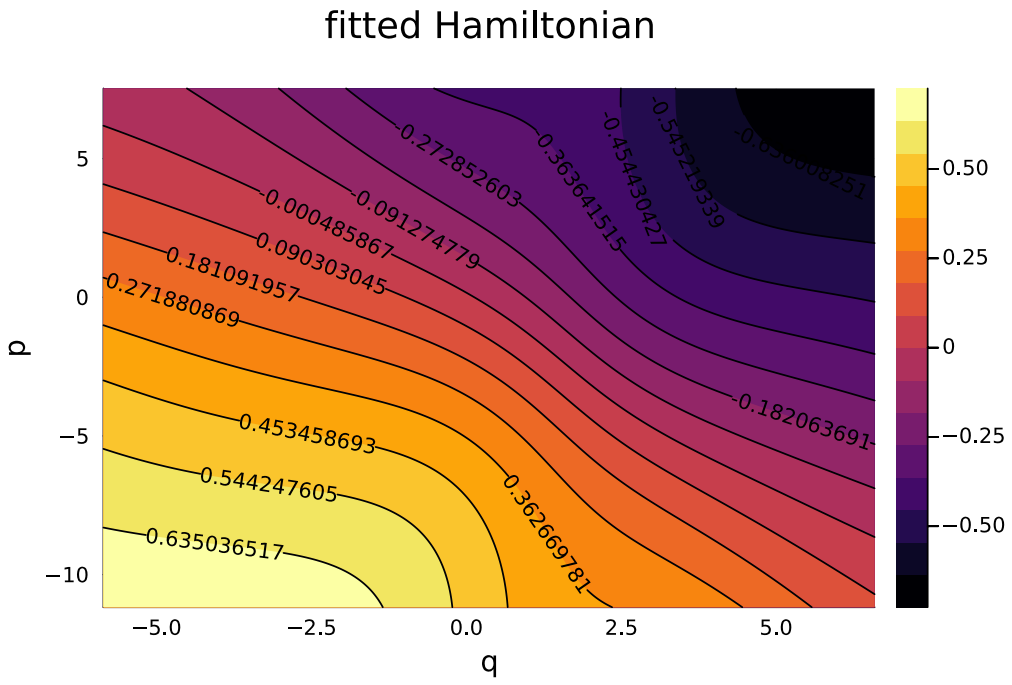


Figure 6.16: The original attractor now colored by external forcing above a threshold.


fitted Hamiltonian
p
q
5
0
−5
−10
−5.0
−2.5
0.0
2.5
5.0
0.50
0.25
0
−0.25
−0.50
-0.272852603
-0.363641515
-0.454430427
-0.545219339
-0.000485867
-0.091274779
0.090303045
0.181091957
-0.271880869
-0.182063691
0.453458693
0.544247605
0.635036517
0.362669781

CHAPTER 7

LIMITATIONS AND FUTURE WORK

7.1 Robot Team

7.1.1 Super Resolution

Propose methods for spatial/temporal/spectral super-resolution. In particular, comment on upcoming satellite deployments like EnMAP and others which will provide hyperspectral imagery.

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	$\mathrm{Br} + \mathrm{CH_4} \longrightarrow \mathrm{HBr} + \mathrm{CH_3}$	$k = (7.8\text{e-}11)\exp(\text{-}76330.0/T)$
6	$\mathrm{Br} + \mathrm{H_2O} \longrightarrow \mathrm{OH} + \mathrm{HBr}$	$k = 0$
7	$\mathrm{Br} + \mathrm{HCO} \longrightarrow \mathrm{CO} + \mathrm{HBr}$	$k = (2.8\text{e-}10)\exp(\text{-}800.0/T)$
8	$\mathrm{Br} + \mathrm{HO_2} \longrightarrow \mathrm{O_2} + \mathrm{HBr}$	$k = (1.5\text{e-}11)\exp(\text{-}600.0/T)$
9	$\mathrm{Br} + \mathrm{N_2O} \longrightarrow \mathrm{N_2} + \mathrm{BrO}$	$k = (3.32\text{e-}10)\exp(\text{-}154800.0/T)$
10	$\mathrm{Br} + \mathrm{NO_3} \longrightarrow \mathrm{BrO} + \mathrm{NO_2}$	$k = (1.6\text{e-}11)$
11	$\mathrm{Br} + \mathrm{H_2O_2} \longrightarrow \mathrm{HBr} + \mathrm{HO_2}$	$k = (1.0\text{e-}11)\exp(\text{-}3000.0/T)$

12	$\mathrm{Br} + \mathrm{HCHO} \longrightarrow \mathrm{HBr} + \mathrm{HCO}$	$k = (1.7\text{e-}11)\exp(-800.0/T)$
13	$\mathrm{Br} + \mathrm{CH_3OH} \longrightarrow$ $\mathrm{HBr} + \mathrm{HO_2} + \mathrm{HCHO}$	$k = (7.59\text{e-}13)\exp(-3119.0/T)$
14	$\mathrm{Br} + \mathrm{OClO} \longrightarrow \mathrm{BrO} + \mathrm{ClO}$	$k = (2.6\text{e-}11)\exp(-1300.0/T)$
15	$\mathrm{Br} + \mathrm{Cl_2O_2} \longrightarrow \mathrm{BrCl} + \mathrm{ClOO}$	$k = (3.0\text{e-}12)$
16	$\mathrm{Br} + \mathrm{CH_3OOH} \longrightarrow \mathrm{HBr} + \mathrm{CH_3O_2}$	$k = (2.63\text{e-}12)\exp(-1610.0/T)$
17	$\mathrm{Br} + \mathrm{BrONO_2} \longrightarrow \mathrm{Br_2} + \mathrm{NO_3}$	$k = (6.3\text{e-}11)\exp(213.0/T)$
18	$\mathrm{BrO} + \mathrm{BrO} \longrightarrow$ $\mathrm{Br} + \mathrm{O_2} + \mathrm{Br}$	$k = (9.23\text{e-}13)\exp(250.0/T)$

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	$H + CH_4 \longrightarrow H_2 + CH_3$	$k = (5.82\text{e-}13)\exp(-33630.0/T)(T/298.0)^{3.0}$
74	$H + H_2O \longrightarrow OH + H_2$	$k = (6.83\text{e-}12)\exp(-80810.0/T)(T/298.0)^{1.6}$
75	$H + HCl \longrightarrow H_2 + Cl$	$k = (1.32\text{e-}11)\exp(-14220.0/T)$
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	$\mathrm{HCl} + \mathrm{HO_2NO_2} \longrightarrow \mathrm{HOCl} + \mathrm{HNO_3}$	$k = (1.0\text{e-}21)$
92	$\mathrm{HO_2} + \mathrm{HO_2} \longrightarrow \mathrm{O_2} + \mathrm{H_2O_2}$	$k = (2.2\text{e-}13)\exp(600.0/T)(1.0+(0.6*P/1013.25))$
93	$\mathrm{HO_2} + \mathrm{NO_3} \longrightarrow$ $\mathrm{O_2} + \mathrm{OH} + \mathrm{NO_2}$	$k = (2.15\text{e-}12)$
94	$\mathrm{HO_2} + \mathrm{NO_3} \longrightarrow \mathrm{O_2} + \mathrm{HNO_3}$	$k = (2.15\text{e-}12)$
95	$\mathrm{HO_2} + \mathrm{HCHO} \longrightarrow \mathrm{HCO} + \mathrm{H_2O_2}$	$k = (3.3\text{e-}12)\exp(\text{-}48800.0/T)$
96	$\mathrm{HO_2} + \mathrm{CH_3O_2} \longrightarrow \mathrm{O_2} + \mathrm{CH_3OOH}$	$k = (3.42\text{e-}13)\exp(780.0/T)$

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	$O_3 + NO_2 \longrightarrow O_2 + NO_3$	$k = (1.2\text{e-}13)\exp(\text{-}2450.0/T)$
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	$OH + H_2 \longrightarrow H + H_2O$	$k = (5.0\text{e-}12)\exp(\text{-}2000.0/T)$
185	$OH + O_3 \longrightarrow O_2 + HO_2$	$k = (1.9\text{e-}12)\exp(\text{-}1000.0/T)$
186	$OH + OH \longrightarrow O(^3P) + H_2O$	$k = (4.2\text{e-}12)\exp(\text{-}240.0/T)$

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	$H_3O^+ + NO_2^- \longrightarrow$ $H_2 + NO_2 + OH$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
312	$H_3O^+ + NO_3^- \longrightarrow$ $H_2 + NO_3 + OH$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
313	$H_3O^+ + CO_3^- \longrightarrow$ $H_2 + CO_2 + O(^3P) + OH$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
314	$H_3O^+ + NO_2^-(H2O)_2 \longrightarrow$ $H_2 + NO_2 + 2H_2O + OH$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
315	$H_3O^+ + CO_4^- \longrightarrow$ $H_2 + CO_2 + O_2 + OH$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$
316	$H_3O^+(H_2O) + e^- \longrightarrow$ $H_2 + OH + H_2O$	$k = (2.8\text{e-}6)(T/300.0)^{-0.15}$
317	$H_3O^+(H_2O) + O_2^- \longrightarrow$ $H_2 + O_2 + OH + H_2O$	$k = (2.0\text{e-}7)(T/300.0)^{-0.5}$

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	$H_3O^+ + CO_3^- \longrightarrow$ $H_2 + CO_2 + O(^3P) + OH + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
77	$H_3O^+ + NO_2^-(H2O)_2 \longrightarrow$ $H_2 + NO_2 + 2H_2O + OH + M$	$k_0 = M(1.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
78	$H_3O^+ + CO_4^- \longrightarrow$ $H_2 + CO_2 + O_2 + OH + M$	$k_0 = M(3.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
79	$H_3O^+(H_2O) + O^- \longrightarrow$ $H_2 + O(^3P) + OH + H_2O + M$	$k_0 = M(1.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
80	$H_3O^+(H_2O) + O_2^- \longrightarrow$ $H_2 + O_2 + OH + H_2O + M$	$k_0 = M(1.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

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	$H_3O^+(H_2O) + CO_4^- \longrightarrow$ $H_2 + CO_2 + O_2 + OH + H_2O + M$	$k_0 = M(1.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
87	$O_2^+O_2 + O^- \longrightarrow$ $O_2 + O(^3P) + O_2 + M$	$k_0 = M(1.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
88	$O_2^+O_2 + O_2^- \longrightarrow 3O_2 + M$	$k_0 = M(1.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
89	$O_2^+O_2 + NO_3^- \longrightarrow$ $2O_2 + NO_3 + M$	$k_0 = M(1.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$
90	$O_2^+O_2 + CO_3^- \longrightarrow$ $2O_2 + CO_2 + O(^3P) + M$	$k_0 = M(1.0\text{e-}25)(T/300.0)^{\text{-}2.5}$ $k_\infty = (T/300.0)^{0.0}$ $f_c = (0.0)$

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

Otter. `https://www.maritimerobotics.com/otter`.

(2021, May). Introducing alta x. `https://freeflysystems.com/alta-x`.

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.

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.

Blaom, A. D., F. Kiraly, T. Lienart, Y. Simillides, D. Arenas, and S. J. Vollmer (2020). MLJ: A julia package for composable machine learning. *Journal of Open Source Software 5*(55), 2704.

Blaom, A. D. and S. J. Vollmer (2020). Flexible model composition in machine learning and its implementation in MLJ.

Boldi, R. A. (1994). A model of the ion chemistry of electrified convection.

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 ….

Huang, S., L. Tang, J. P. Hupy, Y. Wang, and G. Shao (2020). A commentary review on the use of normalized difference vegetation index (ndvi) in the era of popular remote sensing. *Journal of Forestry Research 32*, 1–6.

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*.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetics 43*(1), 59–69.

Lamqadem, A. A., H. Saber, and B. Pradhan (2018). Quantitative assessment of desertification in an arid oasis using remote sensing data and spectral index techniques. *Remote. Sens. 10*, 1862.

Lary, D. J., D. Schaefer, J. Waczak, A. Aker, A. Barbosa, L. O. Wijeratne, S. Talebi, B. Fernando, J. Sadler, T. Lary, et al. (2021). Autonomous learning of new environments with a robotic team employing hyper-spectral remote sensing, comprehensive in-situ sensing and machine learning. *Sensors 21*(6), 2240.

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.

Neumark, D. M. (1992). Transition state spectroscopy of bimolecular chemical reactions. *Annual Review of Physical Chemistry 43*(1), 153–176.

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.

Rasmussen, C. E., C. K. Williams, et al. (2006). *Gaussian processes for machine learning*, Volume 1. Springer.

Strumbelj, E. and I. Kononenko (2013). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems 41*, 647–665.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks 5*, 241–259.

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).

Yu, X., D. J. Lary, and C. S. Simmons (2021). Pm2. 5 modeling and historical reconstruction over the continental usa utilizing goes-16 aod. *Remote Sensing 13*(23), 4788.

Zheng, Q., W. Huang, X. Cui, Y. Shi, and L. Liu (2018). New spectral index for detecting wheat yellow rust using sentinel-2 multispectral imagery. *Sensors (Basel, Switzerland) 18*.

BIOGRAPHICAL SKETCH

UPDATE REQUIRED!!!

CURRICULUM VITAE

UPDATE REQUIRED!