STOCHASTIC MODELS FOR PRECIPITABLE WATER IN CONVECTION

BY

KIMBERLY LEUNG

A DISSERTATION SUBMITTED TO THE FACULTY OF CLAREMONT GRADUATE UNIVERSITY AND SAN DIEGO STATE UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTATIONAL SCIENCE

JULY 2016

© Copyright Kimberly Leung, 2016. All rights reserved.
APPROVAL OF THE DISSERTATION COMMITTEE

This dissertation has been duly read, reviewed, and critiqued by the Committee listed below, which hereby approves the manuscript of Kimberly Leung as fulfilling the scope and quality requirements for meriting the degree of Doctor of Philosophy.

Samuel Shen, Chair
San Diego State University
Department of Mathematics & Statistics
Professor

John Angus
Claremont Graduate University
Institute of Mathematical Sciences
Professor

Barbara Bailey
San Diego State University
Department of Mathematics & Statistics
Associate Professor

Marina Chugunova
Claremont Graduate University
Institute of Mathematical Sciences
Associate Professor

Bo-Wen Shen
San Diego State University
Department of Mathematics & Statistics
Associate Professor
Abstract

Stochastic Models for Precipitable Water in Convection

by

Kimberly Leung

Claremont Graduate University and San Diego State University, 2016

Atmospheric precipitable water vapor (PWV) is the amount of water vapor in the atmosphere within a vertical column of unit cross-sectional area and is a critically important parameter of precipitation processes. However, accurate high-frequency and long-term observations of PWV in the sky were impossible until the availability of modern instruments such as radar. The United States Department of Energy (DOE)’s Atmospheric Radiation Measurement (ARM) Program facility made the first systematic and high-resolution observations of PWV at Darwin, Australia since 2002. At a resolution of 20 seconds, this time series allowed us to examine the volatility of PWV, including fractal behavior with dimension equal to 1.9, higher than the Brownian motion dimension of 1.5. Such strong fractal behavior calls for stochastic differential equation modeling in an attempt to address some of the difficulties of convective parameterization in various kinds of climate models, ranging from general circulation models (GCM) to weather research forecasting (WRF) models. This important observed data at high resolution can capture the fractal behavior of PWV and enables stochastic exploration into the next generation of climate models which considers scales from micrometers to thousands of kilometers. As a first step, this thesis explores a simple stochastic differential equation model of water mass balance for PWV and assesses accuracy, robustness, and sensitivity of the stochastic model. A 1000-day simulation allows for the determination of the best-fitting 25-day period as compared to data from the TWP-ICE field campaign conducted out of Darwin, Australia in early 2006. The observed data and this portion of the simulation had a correlation coefficient of 0.6513 and followed similar statistics and low-resolution temporal trends. Building on the point model foundation, a similar algorithm was applied to the National Center for Atmospheric Research (NCAR)’s existing single-column model as a test-of-concept for eventual inclusion in a general circulation model. The stochastic scheme was designed to be coupled with the deterministic single-column simulation by modifying results of the existing convective scheme (Zhang-McFarlane) and was able to produce a 20-second resolution time series that effectively simulated observed PWV, as measured by correlation coefficient (0.5510), fractal dimension (1.9), statistics, and visual examination of temporal trends.
Results indicate that simulation of a highly volatile time series of observed PWV is certainly achievable and has potential to improve prediction capabilities in climate modeling. Further, this study demonstrates the feasibility of adding a mathematics- and statistics-based stochastic scheme to an existing deterministic parameterization to simulate observed fractal behavior.
Acknowledgments

I would never have been able to finish my dissertation without the guidance of my adviser and committee members, and support from my husband and family.

I would like to express my deepest gratitude to my adviser Samuel Shen for his constant guidance, patience, and encouragement. I especially appreciate his support when it came to my desire to start a family during graduate school.

I would also like to thank my committee members, both at CGU and SDSU (John Angus, Barbara Bailey, Marina Chugunova, Bo-Wen Shen), for their insightful comments and suggestions throughout my research.

Thank you to Guang Zhang for all the atmospheric knowledge that I was missing, and to Aneesh Subramanian for the knowledge, encouragement, support, collaboration, and all kinds of other help he generously provided. Thank you to Xiaoliang Song for helping me get SCAM running (a crucial step in my research!), and to James Otto for insisting I liberally and comprehensively comment my code - this has made my life easier on so many occasions.

Many thanks to Samuel Shen, the Computational Science Research Center at SDSU, the SDSU Department of Mathematics and Statistics, and National Science Foundation (NSF) Grant No. AGS-1015926 for helping to fund my education and research.

Data was provided by the U.S. Department of Energy Atmospheric Radiation Measurement (ARM) Climate Research Facility (precipitable water vapor at Darwin, Australia) and the Global Historical Climatology Network. I also acknowledge high-performance computing support from Yellowstone (ark:/85065/d7wd3xhc) provided by NCAR’s Computational and Information Systems Laboratory, sponsored by the National Science Foundation.

I would like to acknowledge some of the other students in my CGU cohort (Colette Smirniotis, Jamie
Turtle, Bill Spinella) - this journey would have been so much lonelier and more difficult without their company, support, and vegan donuts. Also, I am incredibly grateful to Michelle Weirathmueller and Colette Smirniotis for getting me through the most difficult times, and for understanding what it’s like to be in my shoes.

Finally, my family. Jason - well, thanks to Jason I learned to manage my time efficiently and to keep my priorities straight as an arrow. Baby C - there’s nothing like an impending birth to light a fire under your... backside. Thank you to my parents and in-laws for sending me home with delicious home-cooking and for taking care of Jason when needed, and to my sister Kristel for making some of my graphics.

Most of all, thank you to Ben. Thank you for taking such tender care of me, and for clearing the way for me to focus on school... for making it possible for me to achieve my dreams.

- KDL, 2016
# Contents

Acknowledgments .......................................................... v

List of Figures ............................................................ ix

List of Tables .............................................................. xi

1 Overview ................................................................. 1

2 Introduction ............................................................... 5

2.1 Atmospheric Water Vapor ........................................... 6

2.2 Darwin Long-Term Observed Data ................................. 7

2.2.1 ARM Facility, Instrumentation, and Data ................... 8

2.2.2 Quantitative Analysis ............................................ 10

2.3 Motivation - Fractal Behavior ..................................... 18

2.4 Convection and Convective Parameterization .................. 21

2.4.1 Convection Physics .............................................. 22

2.4.2 Parameterization of Convection ............................... 25

3 Mass Balance Model ......................................................... 27

3.1 Stochastic Calculus and Differential Equations ................. 27

3.2 Model Development .................................................. 29

3.3 Numerical Solution of the SDE Model ........................... 32

4 Simulation of Observed Data - Point Model ......................... 35

4.1 TWP-ICE Field Campaign Data ..................................... 35

4.2 Description of Code .................................................. 38

4.2.1 Model Code ....................................................... 39

4.2.2 Correlation Code ................................................ 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.3 Model Parameters</td>
<td>44</td>
</tr>
<tr>
<td>4.3 Observed-Model Correlation and Statistics</td>
<td>44</td>
</tr>
<tr>
<td>4.4 Robustness Test</td>
<td>52</td>
</tr>
<tr>
<td>4.5 Sensitivity Test</td>
<td>55</td>
</tr>
<tr>
<td>4.6 Levy Motion</td>
<td>58</td>
</tr>
<tr>
<td>4.7 Conclusions</td>
<td>61</td>
</tr>
<tr>
<td>5 Single-Column Modeling</td>
<td>63</td>
</tr>
<tr>
<td>5.1 SCM Background</td>
<td>63</td>
</tr>
<tr>
<td>5.2 NCAR: SCAM</td>
<td>64</td>
</tr>
<tr>
<td>5.3 Coupling the SCM and the Mass Balance Model</td>
<td>66</td>
</tr>
<tr>
<td>5.4 Details of Proposed Scheme</td>
<td>66</td>
</tr>
<tr>
<td>6 Simulation of Observed Data - Single-Column Model</td>
<td>73</td>
</tr>
<tr>
<td>6.1 Description of Code</td>
<td>73</td>
</tr>
<tr>
<td>6.1.1 Interfacing Requirements</td>
<td>74</td>
</tr>
<tr>
<td>6.1.2 Module: Global Variables</td>
<td>76</td>
</tr>
<tr>
<td>6.1.3 Modifications to Existing Routines</td>
<td>76</td>
</tr>
<tr>
<td>6.1.4 Module: Additional Subroutines for the Stochastic Scheme</td>
<td>77</td>
</tr>
<tr>
<td>6.1.5 Call Structure</td>
<td>85</td>
</tr>
<tr>
<td>6.1.6 Running SCAM on Yellowstone</td>
<td>87</td>
</tr>
<tr>
<td>6.1.7 Post-processing</td>
<td>89</td>
</tr>
<tr>
<td>6.2 Simulation Results</td>
<td>90</td>
</tr>
<tr>
<td>7 Conclusions</td>
<td>94</td>
</tr>
</tbody>
</table>

References                                                                98

Appendices                                                               106
List of Figures

1.1 TWP-ICE observed precipitable water with mode simulation and GHCN daily precipitation. ............................................. 1
1.2 Darwin, Australia map ......................................................... 2

2.1 Illustration of the hydrologic cycle. ........................................... 6
2.2 20-second observed precipitable water in cm at Darwin, Australia, 12 March 2002 - 28 February 2011 ................................................. 10
2.3 Precipitable water vapor daily climatology and standard deviation. ............... 13
2.4 Precipitable water vapor anomalies .......................................... 14
2.5 Cumulative monthly precipitation at Darwin and corresponding monthly Southern Oscillation Index ............................................. 15
2.6 Histogram of precipitable water vapor anomaly at Darwin, Australia. ............... 16
2.7 Precipitable water vapor anomalies, daily extremes, and daily precipitation. ....... 17
2.8 Precipitable water vapor anomaly spectra ..................................... 18
2.9 Temperature profile depicting an idealized dry adiabatic lapse rate. .................. 23
2.10 Temperature profile (CAPE) .................................................. 24

3.1 Illustration of a diffusion process ............................................ 29
3.2 Stretched and shifted tanh function ......................................... 32
3.3 Numerical solution program flow ............................................ 34

4.1 TWP-ICE 20-second observed PWV data ..................................... 36
4.2 ARM instrumentation map .................................................... 37
4.3 The microwave radiometer at the Darwin ARM facility ............................... 38
4.4 Call structure for MATLAB point model code .................................. 39
4.5 Probability distribution of observed and best simulated model data ................. 47
4.6 Point model and observed PWV ............................................. 48
List of Tables

4.1 Model parameters for the SDE point model simulations. . . . . . . . . . . . . . . . . . 44
4.2 Statistical moments - model and observed. . . . . . . . . . . . . . . . . . . . . . . . . 45
4.3 Initial condition statistics. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 52

6.1 Single-column model stochastic module parameters. . . . . . . . . . . . . . . . . . . 91
6.2 Single-column model statistics and other results. . . . . . . . . . . . . . . . . . . . . 92
1 Overview

Atmospheric precipitable water vapor (PWV) can be defined as the amount of water vapor in the atmosphere within a vertical column of unit cross-sectional area. It is usually measured in mm and represents the amount of water that would be condensed to liquid from water vapor for the given column. PWV is an important measure of several convective precipitation processes (Sherwood et al., 2010); however, because convective processes occur on such a small time scale (Davies et al., 2013), it is difficult to obtain observed data at high enough resolution to capture the fractal behavior that it displays. Most available observed datasets are of resolution too low to capture its
fractal behavior and data at longer time scales tend to oversmooth.

For this study, we take advantage of high resolution microwave radiometer data and a data aggregation method described in Shen et al. (2013). We have obtained an observed PWV time series at a uniform 20-second grid which allows us to examine the fractal behavior as well as verify model results.

Figure 1.1 (black curve) displays the US Department of Energy (DOE) Atmospheric Radiation Measurement (ARM) program TWP-ICE (Tropical Warm Pool International Cloud Experiment) PWV data time series at 20 seconds temporal resolution for 25 days from January 20, 2006 February 13, 2006, located at the TWP-ICE facility in Darwin, northern Australia (Figure 1.2). The green curve of Figure 1.1 is the daily precipitation data observed at the Darwin airport
located at (12 25' 26.04" S, 130 53' 33.00" E), approximately 100 meters away from the TWP-ICE site (Station WMO ID: ASN00014015). The data are archived at the US National Climatic Data Centers Global Historical Climatology Network-Daily (GHCN-D), and can be retrieved from http://www.ncdc.noaa.gov/oa/climate/ghcn-daily. The precipitation data show a major precipitation process between January 20 - January 26, and a minor precipitation process from February 7 - February 13. These two precipitation process periods are in general agreement with observed data retrieved from precipitation radar that covers an area with radius of 150 km around Darwin (see Figure 3 of Lin et al. (2012)), but the precipitation intensities are different because of the differences of station and radar observations. The January major precipitation process led a major PWV decrease. A clear period free of precipitation beginning February 3 led a sharp PWV increase during February 4 - February 6. Of course, the convergence and divergence from the surrounding environment should also contribute to this low frequency PWV variation.

The observed PWV time series displays self-similar patterns, mathematically called fractal behavior, of PWV temporal variation (Breslin and Belward, 1999). The classical Weierstrass function of a variable, which is continuous everywhere but nowhere differentiable, is an example of this fractal behavior and serves as an excellent test of stochastic modeling (Falconer, 2003).

As the fractal behavior of PWV makes it difficult to model with traditional deterministic parameterization schemes, we build the case for stochastic parameterization of convection with a theoretical and observation-based framework, and propose the use of a stochastic differential equation model for PWV. We start with a simple point model, then apply the SDE model to an existing single-column model as a bridge toward implementation in a general circulation model.

This study aims to address the inadequacy of climate models in their ability to simulate such observed data with fractal characteristics. We use a stochastic differential equation (SDE) model for the mass balance to effectively simulate a time series of observed column PWV data (blue curve in Figure 1.1), then implement a similar algorithm in a single column model, thereby incorporating the vertical structure of the atmosphere.
Organization of the Thesis

Chapter 2 of the thesis (the Introduction) introduces the concepts and significance of atmospheric water vapor in convection, and includes the majority of an observational analysis paper that has been submitted for publication (Section 2.2). The remainder of Chapter 2 discusses the motivation for the thesis as well as the physics background and discussion of previous studies.

Chapter 3 describes the development of the simple SDE model and Chapter 4 discusses the model implementation and results of the model simulations. Chapter 5 discusses the application of the SDE model to an existing single-column model, and Chapter 6 discusses the results of the single-column model simulations. Model code can be found in the Appendices (Appendix A: point model; Appendix B: single-column model). Parts of Chapters 2, 3, and 4 are also included in a separate article submitted for publication.
2 Introduction

The effect of moist convection on large scale fields in numerical global climate and weather models is not adequately represented mainly due to limitations in computational resources to resolve convection (Arakawa, 2004). The bulk mean effect of convection on resolved scales are represented as tendencies of moisture and dry static energy in conventional climate models (Franzke et al., 2015). Representing uncertainty and under-represented variability in convective processes in climate models is key to derive their effect on the mean state (Christensen et al., 2015).

We are particularly interested in water vapor due to its significant role in convection. It is well-known that increased convective precipitation is associated with more atmospheric humidity (Bretherton et al., 2004), and in particular, more column water vapor (Holloway and Neelin, 2010). Holloway and Neelin (2009) note that the behavior of convection as a function of water vapor is still not well-represented in global climate models; part of this may be due to the difficulty in parameterizing the subgrid processes - both spatial and temporal - involved in convection.

Since it is usually computationally prohibitive to represent convective processes explicitly (Franzke et al., 2015), there has been much focus on developing convective parameterization schemes that represent the bulk mean effect of sub-grid scale convection on the large scale fields.

Stochastic modeling is a relatively new paradigm in climate modeling to better represent sub-grid scale processes. Various weather forecasting centers around the world are exploring the use of stochastic parameterization over the last decade to represent uncertainty in model formulations. Yet, there have not been very many rigorous quantifications of the stochastic nature of observed variables, especially in relation to convection.
2.1 Atmospheric Water Vapor

Water vapor is one of the most important elements in the atmosphere. Not only is it the most abundant of the greenhouse gases ("Greenhouse Gases", NOAA/NCEI), it is also both a driver and product of atmospheric processes. With the addition of solar energy, water vapor is a principal part of the hydrologic cycle (Figure 2.1), the process by which water circulates through the earth-atmosphere system. Evaporation and transpiration, which convert liquid water into vapor, require solar energy; when this vapor condenses and some eventually falls as precipitation, this energy is released as the latent heat of condensation. This heat transfer drives major processes such as convection and convective storms (Ahrens, 2012).

![The Hydrologic Cycle](image)

Figure 2.1: Illustration of the hydrologic cycle.

Although water vapor is the dominant greenhouse gas (Held and Soden, 2000), discussions of global warming due to the "greenhouse effect" focus on the implications of other chemical species, such as carbon dioxide (CO$_2$), methane (CH$_4$), and nitrous oxide (N$_2$O). This is because water vapor is not a forcing, but a feedback: as other greenhouse gases contribute to temperature increase, the temperature increase causes water vapor to increase and therefore amplify the warming initiated
by the other gases. Numerous studies have described the significance of the water-vapor feedback; Dessler et al. (2008) strongly demonstrated that the magnitude of the feedback can be as much as 2.04 W/m²/K, a factor that doubles the contribution of the forcing.

One aspect of the behavior of water vapor that could hinder its ability to force a temperature increase on its own is its relatively short atmospheric residence time. While carbon dioxide may remain in the atmosphere for approximately 100 years, methane for approximately 12 years, and nitrous oxide for more than 100 years (Blasing, 2006), water vapor cycles in and out of the atmosphere in about 10 days on average, due to the nature of the hydrologic cycle (evaporation and precipitation) and depending on the particular weather present.

This leads to the notion that the amount of water vapor in the atmosphere is in fact highly temporally variable. Unfortunately, most current observed data sets are of resolution too low to capture this high-resolution variability.

2.2 Darwin Long-Term Observed Data

Holloway and Neelin (2010) and Lintner et al. (2011) analyzed 1- and 5-minute (respectively) resolution water vapor data at Nauru, while Sobel et al. (2004) examined daily water vapor at the Marshall Islands; both datasets were at lower resolution. The uniquely long record of high resolution (20-second) data at the ARM facility in Darwin, Australia is of marked interest due to the many studies that have used data collected as part of the 25-day TWP-ICE experiment based at ARM Darwin.

We have examined a number of aspects of the data: statistics, extreme distribution, fractal dimension, and spectra. Many of these can be useful in informing the design of parameterization schemes, as well as in evaluating the accuracy of such schemes. While classical statistics are commonly used in designing stochastic models both in climate (Stechmann and Neelin, 2001) and in other fields (Lavallée and Archuleta, 2003; Chen et al., 2008), extreme value statistics acknowledge that trends in the frequency and amplitude of extreme events may differ from those estimated by averages (Katz, 2010); the occurrence of extreme events is of significant interest in stochastic modeling of convection.
Spectral analysis can provide time-independent information about the cyclical nature of the water vapor data; the fractal dimension can help quantify the chaotic nature of the time series data (Theiler, 1990).

2.2.1 ARM Facility, Instrumentation, and Data

The U.S. Department of Energy (DOE) created The Atmospheric Radiation Measurement (ARM) Program in 1989 to study cloud formation processes and their influence on radiative transfer. ARM developed multiple highly instrumented ground stations with the objective of improved scientific understanding of the fundamental physics related to radiative feedback processes in the atmosphere, particularly the interactions between clouds and aerosols. These stations focus on obtaining continuous long-term measurements and providing data that can aid in the development of climate models (DOE ARM, web). Additionally, field campaigns that use both the fixed sites and mobile instruments provide shorter periods of more intense data collection.

For approximately two decades ARM has maintained a network of 2-channel ground based microwave radiometers (MWR) which provide long-term time series of PWV, with the goal of providing scientists with reliable calibrated continuous datasets. Brightness temperature is the fundamental parameter measured by passive microwave radiometers (Remote Sensing Systems, web). There are two general retrieval methods used to retrieve water vapor from the observed brightness temperatures: statistical methods and physical retrievals. Statistical methods use a radiative transfer method to derive a set of retrieval coefficients, usually from a set of nearby climatological thermodynamic profiles, while physical retrievals involve iteratively computing an implicit PWV using the radiative transfer model until the computed brightness temperatures are within the observational uncertainty (Cadeddu et al., 2013).

The Darwin facility is part of the ARM Tropical Western Pacific fixed site, which also includes facilities at Manus Island and Nauru. Darwin MWR retrievals began in early 2002, and data up to 2011 is currently available.

Raw PWV data are at nominal 20-second resolution, although in reality the resolution is fairly
irregular. PWV is measured in cm height. When possible, both PWV is retrieved using a physical-iterative approach where the atmospheric state (structure) is specified by interpolated radiosonde profiles. When the physical retrieval is of poor quality, the data are retrieved using a statistical method where the retrieval coefficients utilized are predicted from surface meteorological observations. Where both methods result in poor quality, values are set to -9999.

The time series data spans the period 12 March 2002 - 28 February 2011, with a proportion of valid to missing values of approximately 4:7. To interpolate some of the missing values as well as to convert the data to a uniform 20-second resolution, we employed an aggregation method (Shen et al., 2013) that essentially averages all valid data values in the interval with time marked by the left side of the 20-second interval. For each 20-second interval containing at most two observations, the following method was applied:

**One valid observation:** Use the single numerical value;

**Two valid observations:** Compute the mean;

**No valid observations:** Compute the mean of any valid observations in the adjacent intervals (either side). If there are none, consider the data missing for the interval.

The resulting time series (Figure 2.2) is at a uniform 20-second resolution, containing approximately 4.8 million valid interpolated observations of PWV, and spans almost 9 years.
2.2.2 Quantitative Analysis

Figure 2.3 shows the PWV daily climatology and standard deviation. Northern Australia, where Darwin is located, has a tropical savanna climate, which consists of distinct wet and dry seasons, although the average temperature remains fairly constant all year (Australian BOM, 2016). The wet season, mainly between November and April, includes cyclones and monsoons (Nicholls et al., 1982). The climatology (Figure 2.3) shows high PWV during this time (summer), after which it steadily decreases into the driest months of June through August (winter). It can be seen that summer PWV is twice as high as winter, so, as is expected for a tropical climate in the southern hemisphere, precipitation occurs mainly during the summer. The monthly standard deviation trend (broken line) stays fairly steady, with a mild increase as the level of PWV increases in the last third of the year.

Looking at the time series anomaly data for 2002-2011 (Figure 2.4), we see that there are sustained low anomalies in summer 2002-03, 2004-05, and 2006-07. Similarly, there are sustained high
anomalies in summer 2007-08, 2008-09, and 2010-11. Most of these correspond to established El Niño (dry) and La Niña (wet) events. It is well-known that in Australia, Indonesia, and the general Tropical Western Pacific area, El Niño results in drought (Chiew et al., 1998), and this is reflected in the data signal. Figure 2.5 shows the cumulative monthly precipitation for each year (July-June) and the corresponding time series of the Southern Oscillation Index, where sustained negative indicates El Niño and sustained positive indicates La Niña. Although the signals for some years are clear (2005: negative SOI and less precipitation; 2008 and 2009: positive SOI and more precipitation), others are less so (2003 and 2010: negative SOI but more precipitation). It is important to remember that other factors, such as East Asian monsoon and Australia summer monsoon, contribute to precipitation patterns as well (Zheng et al., 2014).

The histogram of PWV anomaly (Figure 2.6) is mostly symmetric, but slightly negative skewed, with a stronger kurtosis as it is more peaked than the theoretical normal distribution shown in red. The extra peakedness and lack of long tails indicates there are mostly normal events and few extreme events. To further examine this behavior, we examined the daily extremes.

For the extreme value statistics we are interested in the occurrence and sizes of extreme high and low values of PWV. We use a block extrema approach, taking the daily maxima and daily minima and fitting each series to a Generalized Extreme Value Distribution, where the parameters (shape, location, and scale) are estimated by Maximum Likelihood. Then, using the inverse cumulative distribution function (CDF), we estimate the return values as a function of return period (in days). Figure 2.7 shows the PWV anomaly with extremes indicated by red (positive) and green (negative) triangles, along with the corresponding GHCN daily precipitation data. It can be seen that many of the extreme highs occur before periods of increased precipitation, while extreme lows often occur right after. The return period plots show that the PWV anomaly is fairly stable, with not many outliers; hence an absence of real strong events (also indicated by the distribution in Figure 2.6).

Fourier spectral analysis, using the Fast Fourier Transform (Figure 2.8), shows that most of the power is in the annual cycle of 365 days ($3.18 \times 10^{-8}$ Hz), with a rather insignificant peak for the diurnal (daily) cycle of 24 hours ($1.16 \times 10^{-5}$ Hz). Aside from the annual cycle, few other prominent peaks appear. However, although it appears that in Australia there is no diurnal PWV cycle, this
does not imply a nonexistent precipitation diurnal cycle (May et al., 2012), which can also depend on temperature, radiation, wind, and other factors (Nesbitt and Zipser, 2003).

Finally, the high-frequency volatility of the observed data suggests fractality, which can be measured by its Hausdorff fractal dimension (Falconer, 2003). Here, we use the box counting method in MATLAB (Moisy, 2006) to calculate the Hausdorff dimension of the time series; the box-counting method has been used previously in the meteorology community to calculate the fractal dimension of precipitation time series (Hubert et al., 1989; Olsson et al., 1993; Breslin and Belward, 1999). This method yields a fractal dimension of 1.9, which implies more complexity than the path of Brownian motion, which has fractal dimension of 1.5. The fractal dimension close to 2 indicates very high complexity, with the complexity most likely coming from nonlinearity of atmospheric dynamics - demonstrated by strong convection. Darwin does show some sudden precipitation events, indicating the complexity of climate dynamics compared to Brownian motion. This gives rise to the difficulty of numerical simulation, justifying the notion that classical differential equations are not sufficient to model this data, as differentiability implies a fractal dimension equal to 1.
Figure 2.3: Precipitable water vapor daily climatology and daily standard deviation at Darwin, Australia. The broken line shows the standard deviation monthly moving average.
Figure 2.4: Precipitable water vapor anomalies. Red lines represent one and two standard deviations from the mean.
Figure 2.5: Cumulative monthly precipitation at Darwin (from GHCN daily observed data) and corresponding monthly Southern Oscillation Index. SOI Data Source: https://www.ncdc.noaa.gov/teleconnections/enso/indicators/soi/
Figure 2.6: Histogram of precipitable water vapor anomaly at Darwin, Australia. Red line depicts the theoretical normal distribution based on anomaly mean and standard deviation.
Figure 2.7: Precipitable water vapor anomalies (top) and daily extremes return periods (bottom), along with corresponding GHCN daily precipitation data. Red and green triangle point out some extreme highs (red) and lows (green).
Figure 2.8: Precipitable water vapor anomaly spectra. The main peak is at 365 days, or, the annual cycle frequency.

### 2.3 Motivation - Fractal Behavior

The complexity of a highly volatile time series may be measured by a Hausdorff fractal dimension, which is an asymptotic value of the ratio of the logarithmic change in details to the logarithmic change in scales: \( \ln(N)/(-\ln(\epsilon)) \) (Falconer, 2003), where \( N \) square boxes of edge \( \epsilon \) cover the entire time series. Visually, the PWV data of Figure 1.1 imply that the time series is likely to have a fractal dimension greater than one, where the dimension of a smooth curve is one since it can be covered by \( N \) boxes of length \( \epsilon = 1/N \). It is known that the Hausdorff dimension is 1.5 for the path of Brownian motion (Falconer, 2003). The greater-than-1 dimensionality of the PWV time series demonstrates its complexity and shows the difficulty of simulating PWV using conventional partial differential equation (PDE)-based parameterized climate models that yield everywhere-differentiable solutions except for some possible jump discontinuities, called shocks. Fridlind et al. (2012) compared the
TWP-ICE C-band polarimetric (C-POL) weather radar precipitation rate, ice water path (IWP), and other high temporal resolution parameters with simulations from a cloud resolving model (CRM). They found that although this model could not demonstrate the self-similarity and fractal dimension of the precipitation rate and IWP, the observed and model data follow the same temporal trends of increasing and decreasing. They also identified another fundamental difference between model output and observed data: the spatial scale mismatch between the grid box-based CRM PDE model discretization and the point-based radar scanning. Both the randomness and the spatial mismatch call for using proper stochastic models to simulate the observed data, which in turn can help validate models from the perspective of self-similarity, probability, and quantified uncertainty, rather than the traditional perspective of considering simple differences between observed and model data.

PWV variations are inevitably related to precipitation. The fractal dimensions of PWV and precipitation should be close to each other, if not the same, since the PWV decrease at any given time is usually synchronized with precipitation and convergence (or divergence). Breslin and Belward (1999) analyzed the monthly rain data from 52 land gauges in Queensland, Australia with 100 years of records. Olsson et al. (1993) analyzed six rain-intensity data time series of 1-minute temporal resolution from 1979-1980 observed by tipping-bucket gauges with 0.035 mm/min resolution at Lund, Sweden. Both studies found the rainfall fractal dimensions to lie between 1 and 2, despite their different temporal resolutions. Rodriguez et al. (2013) studied the rainfall data recorded in the metropolitan area of Barcelona, Spain for the period of 1994-2001 with 8-minute time resolution and also found fractal behavior. The common fractal behavior commonly found in precipitation time series may be attributable to the self-similarity of stochastic and nonlinear precipitation processes, closely related to the PWV's stochastic fluctuations (Schertzer and Lovejoy, 1987). We justify the need for stochastic PWV modeling from three perspectives.

First, the nonlinearity, fractal dimension and self-similarity of the PWV time series or datasets call for stochastic modeling of water vapor. Entekhabi and Brubaker (1995) used an SDE model and Itô calculus (Grigoriu, 2002) to study the water vapor content for the land-atmosphere interaction. Their SDE model did a reasonable job of partitioning the net heat gain on the surface into sensible and latent heat fluxes for certain soil conditions and seasons. The model also set up a framework for
introducing random forcing in the coupling of water-energy and land-atmosphere in a climate system. Stechmann and Neelin (2011), referred to as SN11 hereafter, used a noise-forced differential equation model to simulate the water vapor and precipitation relationship. They successfully simulated several statistical characteristics of a strong convective precipitation process, including a large precipitation variance near the critical PWV state and exponential tails of PWVs probabilistic distribution.

Another important property of a PWV dataset is its probability distribution. Lintner et al. (2011) used a simple local model to simulate quantities like PWV and analyzed the 5-min PWV data time series of ARM passive radiometer-derived PWV at Nauru (0.5 S, 167 E) in the western equatorial Pacific for the period of November 20, 1998 to August 16, 2006. They delineated the PWV pdf differences between observed and simulated data and illustrated how to use the pdf-based diagnostics from NCEP-NCAR reanalysis for model inter-comparison and validation. Our SDE simulation, by default, considers the PWV probability distribution function (pdf) as model output. Our SDE simulations for PWV can provide remedies for the pdf differences between simulations and observations. The flexibility of the random trigger and the adjustable transition function from rain to no-rain and vice versa makes it relatively easy to produce simulated PWV pdfs that are very close to the observed ones, as measured by histograms.

Finally, an inevitable problem when simulating observed PWV data by models is the mismatch of point measurement and either an area measurement or area- or volume-based model results, such as a grid box output of a PDE-based climate model using the conventional continuum mechanics method. Fridlind et al. (2012) compared the TWP-ICE observed data with cloud-resolving models and concluded that one could not expect a close agreement between the observed and modeled PWV and other atmospheric data on hourly or shorter time scales due to domain mismatch and topographic drifts. Lin et al. (2012) compared atmospheric general circulation model (GCM) simulations of tropical convection during the TWP-ICE observational period. They found that although the ensemble mean of the modeled water vapor on daily time scales agree well with the observations during the TWP-ICE wet (January 19 - January 24, 2006) and dry (January 25 - February 1) periods, the observations are outside of the model spread in the TWP-ICE precipitation clear period (February 2 - February 4) and break period (February 5 - February 12). Both their conclusion
that an increase in model resolution does not necessarily improve the aforementioned model bias and their identification of the importance of convection parameterization suggest an alternative: introducing a stochastic mechanism in modeling strong convection in order to improve the convection responsiveness of a model. Habib and Krajewski (2002)’s comparison of time-integrated point-based rain gauge measurements with instantaneous precipitation radar measurements at 2-km horizontal resolution is another example of point value vs. grid box value mismatch. The spatial scale mismatch is an important reason why the intrinsic nonlinear, non-stationary and stochastic fractal behavior is not shown in CRM and GCM model simulations. The lack of any intrinsic stochastic nature of the PDE-based CRM or GCM must surely be another reason for this type of model failing. Conventional PDE-based climate models will typically over-smooth climate variables in both space and time and cannot adequately simulate the fractal dimensions and self-similarity of highly volatile atmospheric variables.

Our question is: how can this data be modeled? We seek to address the above three problems in simulating the nonlinear, non-stationary and random climate data and in validating climate models. This study starts with simulating PWV in convection using an SDE-based point model and then builds by incorporating the atmospheric vertical structure available in a single column model.

Our primary objectives are to:

- Introduce a stochastic model to simulate the stochastic nature of precipitable water in convective processes.

- Apply this algorithm to a single column model to further improve the parameterization and to form the basis for using this type of parameterization in a general circulation model.

2.4 Convection and Convective Parameterization

In general circulation models (GCM), the region of interest must be divided into a grid over which the governing equations can be solved computationally. There is a limit on the resolution of such a grid which depends on the computational resources available. However, some atmospheric processes
occur on much smaller scales than model resolution, and must be parameterized. In other words, the effects of subgrid-size processes are represented as functions of grid-level variables. Convection is one such process that occurs at a subgrid scale. Before discussing its parameterization however, we describe the basic physics involved.

### 2.4.1 Convection Physics

Convection in general occurs when a fluid of sufficiently small viscosity and diffusivity is heated from below and develops overturning motions. The fluid at the bottom is warmed and therefore becomes less dense, so it rises due to the pressure gradient from the surrounding fluid. Essentially, the heat destabilizes the fluid and convection results as a mechanism to return the fluid to equilibrium (Marshall and Plumb, 2008).

In meteorology, convection refers primarily to vertical movement of air in the troposphere (the lowest 8-16 km of the atmosphere) due to such instability. As the earth is heated by the sun, convection may occur where the surface heats up very rapidly and consequently warms the overlying air, which gradually becomes less dense than the surrounding air and begins to rise.

Let us first define the adiabatic lapse rate. An adiabatic process is one in which neither heat nor matter transfers between a system and its environment. In the atmosphere, the adiabatic lapse rate (ALR) is the rate at which an air parcel’s temperature decreases with height under adiabatic displacement. In a dry atmosphere, if the ambient temperature decreases with height faster than the ALR, the state is unstable and dry convection can occur. Figure 2.9 depicts a stable system, in which if a parcel of air is lifted by an external force, it follows the dry adiabat until the force ceases. At this point the parcel would return to its origin, without convection occurring. In reality, the lower tropospheric lapse rate in the tropics is \(\simeq -4.6 \text{ K/km}\), about 50% of the adiabatic value. Thus we would expect little or no convection based on the stability criteria. Yet, convection does occur over most of the world, so we must consider the influence of moisture in the air (Marshall and Plumb, 2008).

If a moist air parcel is lifted, it cools adiabatically as with a dry parcel; however, if the cooling is
Figure 2.9: Temperature profile depicting an idealized dry adiabatic lapse rate. In this figure, the solid line is the ambient temperature profile, or environmental lapse rate. Since the environmental lapse rate is less than that of the adiabatic lapse rate, the system is stable, and dry convection is unlikely to occur.

Moisture is quantified in terms of humidity. $q$ is the specific humidity, and is defined as the ratio of water vapor mass to the total mass of air (dry air plus water vapor) per unit volume:

$$ q = \frac{\rho_v}{\rho} \text{, where } \rho = \rho_d + \rho_v $$  \hspace{1cm} (2.1)

where $\rho_d$ denotes the density of dry air and $\rho_v$ denotes the density of the water vapor. Relative humidity, $U$, is the ratio of the specific humidity $q$ to the saturation-specific humidity, $q_s$, the specific humidity at which saturation occurs. Figure 2.10 shows the temperature profile of the saturated ALR, which is smaller than the dry ALR.
If a moist air parcel is lifted adiabatically by some large scale forcing, it cools rapidly as it follows the dry adiabat, increasing $U$ until saturation is achieved at the lifting condensation level (LCL), after which it follows the saturated adiabat. As it rises along the saturated adiabat, $q_s$ decreases, and some of the vapor in the parcel must condense to keep $q = q_s$. This condensation releases latent heat which warms the parcel and slows the rate of temperature decrease. Eventually, if the parcel reaches the level of free convection (LFC), it has become as warm or warmer than its environment, and can accelerate on its own due to the temperature/pressure differential (Figure 2.10). The parcel will continue to rise until its temperature once again becomes equal to that of its environment at the level of neutral buoyancy (LNB).

![Figure 2.10: Temperature profile](image)

This ability to rise without external forcing is quantified by the convective available potential energy (CAPE). CAPE can be thought of as the positive buoyancy of a parcel, and as an indicator of instability (lack of CAPE means the system is stable to perturbations, and will return to its stable state once forcing is ceased). CAPE is depicted in Figure 2.10 as the shaded area between the dry and saturated adiabats.
2.4.2 Parameterization of Convection

Most of the convection parameterization schemes currently utilized in GCMs are of the deterministic variety. That is, they approximate the mean state of the subgrid-scale processes. Some of the most widely-known and studied schemes are described in Kain and Fritsch (1990), Betts and Miller (1993), Arakawa and Schubert (1974), and Zhang and McFarlane (1995); all utilize either a mass flux type approach or a plume ensemble approach.

As standard as these schemes are considered, however, Lin and Neelin (2003) note that it may important to include higher-order moments of convective variables, as they can generate pronounced effects on intraseasonal variability. Furthermore, as convective fluctuations have been seen to strongly affect large-scale dynamics (Plant and Craig, 2008), it is insufficient to rely on the mean state. One approach that has been considered in recent years is super-parameterization, which embeds a cloud model to serve as the parameterization; however this is computationally expensive. Thus, there is a need for stochastic parameterizations which can effect the variability associated with higher moments, but at reasonable computational expense.

Buizza et al. (1999) attempted to represent the random errors associated with parameterization of subgrid-scale convective processes by applying multiplicative noise to the parameterized tendencies and found improvement in the ensemble spread and performance. Lin and Neelin have implemented a number of schemes: introduce a random contribution to CAPE (often used as a closure term for the onset of convection) in a deep convective scheme (2000); choose precipitation input from a distribution tuned with observations to reproduce certain statistical properties (2002); and modify the Zhang-McFarlane scheme to add a stochastic component to the removal of CAPE by convection (2003). Plant and Craig (2008) used a random plume model to replicate the known variability about convective equilibrium.

Jakob et al. (2011) observed that most of these studies have relied on ad-hoc empirical relationships or on the use of other models, rather than on observations, and attempt to address this shortcoming...
with a close examination of the relationship between large and convective scales. Stechmann and Neelin (2011) also take this approach with a scheme that relies heavily on theory and empirical relationships. It is this scheme (SN11) that we wish to examine more closely and build upon in this study.

SN11 developed a simple point model for the change in precipitable water vapor (the total amount of water vapor in a column) based on several characteristics of the transition to strong convection. By examining the relationship between precipitation and water vapor in the context of this transition, they were able to approach convective parameterization in terms of the resolved water vapor as the large-scale variable.

The first of these characteristics is the increase in precipitation with an increase in precipitable water vapor (PWV); more specifically, there is small mean precipitation with low PWV, large mean precipitation with high PWV, and a sharp, highly variable transition near a critical value of PWV (denoted by \( q_c \)). Secondly, the occurrence of extreme events (high PWV, high precipitation rate, or high total precipitation) is more frequent than Gaussian statistics would suggest. Finally, observed temporal variability suggests that the onset of convective precipitation is not necessarily a result of a fixed critical PWV threshold, i.e. precipitation does not always commence once the value of PWV reaches \( q_c \).

The point model is based on the water vapor mass balance, and makes use of the above traits in both model design and parameter choice. While it is common to trigger precipitation when \( q > q_c \) and to turn off precipitation when \( q < q_c \) this is a deterministic criterion, and SN11 noted that there are many unresolved factors that influence the onset of convection; thus they introduced a stochastic trigger to model the onset and demise of convective precipitation.
3 Mass Balance Model

The mass balance model applies conservation of mass to the volume of the air column under consideration; the change in water vapor in the volume must be equal to the difference between the source(s) and sink(s) rates:

\[
\frac{dq}{dt} = \Sigma \text{sources} - \Sigma \text{sinks} \tag{3.1}
\]

where \(\frac{dq}{dt}\) represents the change in water vapor with time. Worden et al. (2007) discuss the limitations of accurately assessing atmospheric humidity, which calls for stochastic closures in the computation of humidity; here sources include evaporation and convergence, where evaporation and part of convergence are deterministic and the remainder of convergence is random. Similarly, sinks include precipitation and divergence, where part of each is deterministic and the remainder is random.

This study uses SN11’s PWV model of mass balance with an equivalent SDE mathematical formulation. The SDE formulation introduces an intrinsic probabilistic framework and the convergence of a numerical solution is in the sense of probability equal to one. The SDE approach has the flexibility to facilitate a more general formation of SDE models that may include either Brownian motion or Levy motion (Grigoriu, 2002). This does not assume the normal distribution of an increment of a random variable with respect to time and admits jumps in the discontinuous paths of an SDE solution.

3.1 Stochastic Calculus and Differential Equations

Stochastic differential equation (SDE) models can be used to simulate systems with randomly varying influences as they combine deterministic and stochastic components. Richards (2004) observed that volatility diffusions with various types of Levy motion are capable of simulating fractal behavior found in the real world (economics, in this particular case). SDEs vary from ordinary differential equations (ODE) in that they are driven by a stochastic process, which is not necessarily
differentiable. Thus, an SDE requires the use of a stochastic calculus, one of which is Itô stochastic calculus.

Itô calculus broadens the methods of classical calculus such that they can be applied to integrals where both the integrand and integrator are stochastic processes. An Itô integral takes the form

\[ \int_{t_0}^{t} g(s) dW(s) \]  

(3.2)

where \( W \) is the Wiener process. Similar to a Riemann sum, it may be approximated by a sum of the form

\[ \sum_{i=1}^{n} g(t_{i-1}) \Delta W_i \]  

(3.3)

where \( \Delta W_i = W(t_i) - W(t_{i-1}) \), a step of Brownian motion.

A typical diffusion process includes deterministic (drift) and random (diffusion) terms (Figure 3.1), and can be modeled by the differential equation

\[ dX(t) = f(X(t))dt + g(X(t))dW(t), \quad X(0) = X_0, \quad 0 \leq t \leq T \]  

(3.4)

where \( W(t) \) denotes a Wiener or Brownian motion process and \( f \) and \( g \) are deterministic functions (Gard, 1998). The first term on the right hand side represents the drift while the second term represents the diffusion, or volatility. Equivalent to Equation 3.4 is the integral equation

\[ X(t) = X_0 + \int_{0}^{t} f(X(s))ds + \int_{0}^{t} g(X(s))dW(s), \quad 0 \leq t \leq T \]  

(3.5)

where the last integral on the right hand side is an Itô integral.
Figure 3.1: Illustration of a diffusion process. The yellow curve is the deterministic process, the blue curve is the random process, and the orange curve is the diffusion process consisting of the sum of the random and deterministic curves.

3.2 Model Development

Given this stochastic calculus basis, the mass balance system is modeled with the stochastic differential equation for a diffusion process:

\[
\begin{align*}
    dq &= \begin{cases} 
    E(q)dt + D_0(q)dL & \iff \text{non-precipitating} \\
    -[P(q)dt + D_P(q)dL_P + D_F(q)dL_F] & \iff \text{precipitating}
    \end{cases} 
\end{align*}
\]  

(3.6)

where \(q(t)\) is the time-varying PWV of a single column, \(E(q)\) parameterizes mean evaporation along with deterministic convergence and divergence, and \(P(q)\) parameterizes mean precipitation also with deterministic convergence and divergence. \(D_P\) represents the random part of precipitation, \(D_0\) and \(D_F\) represent the random part of convergence or divergence (reflecting the randomness of wind, multiscale turbulence, and sub-grid scale physics), and \(L\) is a Levy motion, a general class of stochastic processes. Equation 3.6 is a general SDE model formulation useful for various
applications, but in this study the driving stochastic process will be limited to a sub-class of Levy motion with a normal distribution. Specifically, $dL$ is replaced with a Brownian motion whose standard deviation is proportional to the square root of the time interval:

$$dL \to dW \sim N(0, 1)\sqrt{dt} \quad (3.7)$$

where $N(0, 1)$ denotes the standard normal distribution and $dt$ is the time step size.

The switch between precipitation and non-precipitation states is modeled by a random trigger and transition rate function. This allows random precipitation to occur occasionally when PWV is not saturated and, likewise, non-precipitation to occur occasionally when PWV is oversaturated. We make use of statistical distributions to mathematically describe this trigger.

The Poisson distribution is commonly used to model the probability distribution function (pdf) for the number of occurrences of a random event per unit time (Wackerly et al., 2002). Assuming that past observations imply that precipitation events occur $\lambda$ times per unit time on average, the probability of $n$ precipitation events occurring in an interval of unit time follows a Poisson distribution with rate $\lambda$, i.e.,

$$p(n; \lambda) = \frac{\lambda^n \exp(-\lambda)}{n!}. \quad (3.8)$$

Then, an exponential distribution can be used to model the pdf for the length of time between each pair of consecutive events. Since $\lambda$ is the precipitation occurrence rate, $\lambda dt$ precipitation events on average will occur during $\Delta t$. The probability of zero precipitation events occurring during $dt$ is then

$$p(n; \lambda) = \frac{(\lambda dt)^0 \exp(-\lambda dt)}{0!} = \exp(-\lambda dt). \quad (3.9)$$

Thus, the probability of a precipitation event occurring during $dt$ is $1 - p(0; \lambda dt)$, i.e.,

$$p_e(dt) = 1 - \exp(-\lambda dt). \quad (3.10)$$

This is the CDF of an exponential distribution with $p_e(0) = 0$ (zero events occur in a time interval of zero length) and $p_e(\infty) = 1$ (an event must occur during an interval of infinite length). This
justifies the use of the exponential distribution to model the probability of the state transition between time steps.

To determine the precipitation state at each time step, a stochastic trigger in the form of a Markov jump process is used:

\[
\sigma(t) = \begin{cases} 
0 & \iff \text{non-precipitating} \\
1 & \iff \text{precipitating.}
\end{cases}
\]  

(3.11)

The probability of a jump from \( \sigma = 0 \) to \( \sigma = 1 \) (onset of precipitation) or from \( \sigma = 1 \) to \( \sigma = 0 \) (demise of precipitation) in any given time interval \( \Delta t \) is determined by the transition rate \( r_{ij}(q) \) where \( r_{01}(q) \) denotes the transition rate between precipitation and non-precipitation, and \( r_{10}(q) \) denotes the opposite. Then the transition probability is determined by the CDF of the exponential distribution

\[
prob_{ij} = 1 - e^{-r_{ij} \Delta t} \in [0, 1).
\]  

(3.12)

All model parameters (transition rate and source parameterizations) were derived from observations described in Neelin et al. (2009) and theory developed in SN11; they are functions of \( q \) and are defined by a stretched and shifted tanh function (Figure 3.2)

\[
f(q) = f_- + (f_+ - f_-) \cdot \frac{1}{2} \left[ 1 + \tanh \left( \frac{q - q_{mid}}{q_{width}} \right) \right] \]  

(3.13)

where \( f_- \) is the value of \( f \) at very low PWV, and \( f_+ \) is the value of \( f \) at very high PWV. \( q_{mid} \) is the value of \( q \) at the midpoint of the smooth step transition, and \( q_{width} \) is the width of the transition, which determines the steepness as a function of \( q \). The parameters of the transition rate function are location-specific; this study will appropriately select parameters to reflect the particular location to be modeled.
3.3 Numerical Solution of the SDE Model

Recall the derivation of the stochastic differential equation for $q(t)$ in Section 3:

$$dq = \begin{cases} 
E(q)dt + D_0(q)dW & \leftrightarrow \text{non-precipitating} \\
-[P(q)dt + D_P(q)dW_P + D_F(q)dW_F] & \leftrightarrow \text{precipitating.}
\end{cases} \quad (3.14)$$

where $q(t)$ is the time-varying PWV of a single column.

As with an ordinary differential equation (ODE) model, we are interested in the numerical solution at discrete time steps. While ODEs may be numerically solved using Euler’s method for numerical integration, an SDE requires a more general scheme, one of which is the Euler-Maruyama method. Higham (2001) describes the algorithm as derived from the integral form of the SDE:

$$X(t) = X_0 + \int_0^t f(X(s))ds + \int_0^t g(X(s))dW(s), \quad 0 \leq t \leq T. \quad (3.15)$$

Discretizing the interval $[0, T]$ with $\Delta t = T/L$ for some positive integer $L$ and $\tau_j = j\Delta t$, the numerical solution can be computed at each point using

$$X_j = X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})(W(\tau_j) - W(\tau_{j-1})), \quad j = 1, 2, \ldots, L. \quad (3.16)$$
Note that, from the integral form defined above,

\[ X(\tau_j) = X(\tau_{j-1}) + \int_{\tau_{j-1}}^{\tau_j} f(X(s))ds + \int_{\tau_{j-1}}^{\tau_j} g(X(s))dW(s) \]  

(3.17)

so the Euler-Maruyama method approximates each of the terms on the right side. Although Euler’s method for ODEs has strong convergence order 1, the Euler-Maruyama method has only order 1/2 (Sauer, 2012). This can be increased to 1 with the use of the Milstein method, which may be addressed in future work.

The numerical solution (see Figure 3.3) uses a time step of 20 seconds (1/1800 h) and updates at each step in two stages: (1) the PWV \( q(t) \) is updated to the next time step \( q(t + dt) \) with \( \sigma(t) \) held fixed, and (2) \( \sigma(t) \) is updated to \( \sigma(t + dt) \) using the value of \( q(t + dt) \). For both steps pseudo-random numbers uniformly distributed between 0 and 1 are generated using the Merseenne-Twister algorithm, and converted to Gaussian noise using the Box-Muller method. Box-Muller converts two uniform random numbers, \( u_1 \) and \( u_2 \), into a Gaussian random number using

\[ g = \sqrt{-2 \ln u_1 \cos(2\pi u_2)}. \]

To step \( q \) forward in time using the Euler-Maruyama method,

\[ q(t + dt) = \begin{cases} 
q(t) + E(q(t))dt + D_0(q(t))dW_t & \text{if } \sigma(t) = 0 \\
q(t) - [P(q(t))dt + D_P(q(t))dW_{P,t} + D_F(q(t))dW_{F,t}] & \text{if } \sigma(t) = 1.
\end{cases} \]  

(3.18)

where \( dW_t \) is Gaussian white noise with standard deviation \( \sqrt{dt} \).

\( \sigma(t) \) is then updated to \( \sigma(t + dt) \) by computing the probability of switching states - using \( r_{01} \) if \( \sigma(t) = 0 \) and \( r_{10} \) if \( \sigma(t) = 1 \) - and converting this probability to a Z-score from the standard normal distribution. A Gaussian random number \( p_{\text{crit}} \) is generated and if it is less than or equal to the probability value, the state switches. If it is greater, the state remains the same.
Figure 3.3: Numerical solution program flow. The algorithm is initialized with random values of \( q \) (of appropriate magnitude) and \( \sigma \), then repeats the computations at 20-second intervals as many times as the user desires.
4 Simulation of Observed Data - Point Model

SN11 were able to achieve reasonable statistics that resembled theoretical solutions. However; this was a simple, idealized model designed to investigate the transition to strong convection as well as possible stochastic parameterizations. We sought to simulate observed time-series data as a first step to improving predictive capabilities.

For model validation, we chose to use a 25-day segment of the Darwin long-term data. These 25 days were a part of the TWP-ICE field campaign that took place in Darwin in early 2006, and the various datasets that resulted from this campaign have been used in many studies (Fridlind et al, 2012; Lin et al, 2012; Wang et al, 2009; Xie et al., 2010). Thus we consider this data to be a benchmark dataset for various modeling studies, mechanism studies, and climate model improvements.

4.1 TWP-ICE Field Campaign Data

Figure 4.1 displays the US Department of Energy (DOE) Atmospheric Radiation Measurement (ARM) program Tropical Warm Pool - International Cloud Experiment (TWP-ICE) PWV time series at 20 seconds temporal resolution for 25 days from 20 January 2006 to 13 February 2006. The aim of this experiment was to acquire a dataset that would provide as complete a picture as possible of the evolution of oceanic convection during the Australian monsoon season (May et al., 2004). It was the first field program in the tropics that attempted to describe the evolution of tropical convection, including the large-scale heat, moisture, and momentum budgets at 3-hourly time resolution, while at the same time obtaining detailed observations of cloud properties and the impact of the clouds on the environment (May, et al., 2008). The experiment took place in and around Darwin, Northern Australia (Figure 1.2) and utilized a network (Figure 4.2) of ground-based observations (soundings, active and passive remote sensors) combined with a large range of low-, mid-, and high-altitude aircraft for in situ and remote-sensing measurements (May, et al., 2008).
Data were collected via microwave radiometer (Figure 4.3, method described in Section 2.2.1) using a mean interval of approximately 20 seconds; the actual data intervals are non-uniform, though usually close to 20 seconds. The aggregation method described above for the long-term Darwin data was also applied to this data set, resulting in 108,000 temporal grid points at 20-second intervals.

Visually, Figure 4.1’s observed PWV time series appears to have high frequency fluctuation patterns repeating themselves when zooming in and is more complex than a simple smooth curve. This demonstrates fractal properties due to its high volatility and self-similarity. The complexity is measured by its Hausdorff fractal dimension (Falconer, 2003). Higher fractal dimension indicates an object that fills more space than an object with a lower dimension. For instance, a curve with a fractal dimension of $D = 1.3$ fills more space than a smooth one-dimensional curve but less than a two-dimensional object. Here, we use the box counting method in MATLAB (Moisy, 2006) to calculate the Hausdorff dimension of the time series in Figure 4.1; the box-counting method has been used previously in the meteorology community to calculate the fractal dimension of precipitation time series (Hubert et al., 1989; Olsson et al., 1993; Breslin and Belward, 1999). This method yields
a fractal dimension of 1.9 for the PWV time data shown in Figure 4.1, which is larger than the
dimension 1.5 of the path for a Brownian motion. This implies more complex PWV and precipitation
dynamics and associated physical behaviors than white noise. The PWV fractal dimension of 1.9
also displays an anti-persistent property (Dauphine, 2013), meaning that a decrease in the amplitude
at one time is likely to lead to an increase at a later time. This property maintains an overall
balance of the PWV in the atmosphere, and also indicates great irregularity because of non-periodic
ups and downs.

The fractal behavior of the time series in Figure 4.1 calls for stochastic modeling of water vapor.
Previous studies have shown that stochastic differential equation modeling in conjunction with Itô calculus can reasonably simulate water vapor for land-surface interaction (Grigoriu, 2002).

4.2 Description of Code

Model code was written in MATLAB and consists of five functions for the actual model plus an additional two for computing correlation coefficients. The call structure, depicted in Figure 4.4, shows that \texttt{pwvmodel()} is the main function that initiates and runs the model. It calls the three routines in the second column, and of those three, \texttt{pwvtrigger()} calls the two routines in the third
column. Separately, \texttt{pwvcor()} and \texttt{corrcfcomp()} take the model simulation data as input and compute a time series of moving-window correlation coefficients between the observed data and each 25-section of the simulation.

### 4.2.1 Model Code

![Call structure for MATLAB point model code.](image)

**PWVMODEL** This function simulates the change in column precipitable water vapor with a stochastic trigger for convective precipitation and stochastic closures for precipitation and other external forcings. First, random initial conditions are set, with \( q \) between 30 and 80, and \( \sigma \) equal to either 0 or 1. Following the scheme described by Equation 3.18, the model steps forward in time by 1) calling \texttt{fpwv()} to compute the parameters \( E \) and \( D_0 \) (for \( \sigma = 0 \)) or \( P \) and \( D_1 \) (for \( \sigma = 1 \)), 2) generating a stochastic noise component using \texttt{prng()} (or alternatively, \texttt{levyrn()}), and 3) summing the elements of Equation 3.18 to compute \( q \) for the next time step. \( \sigma \) is stepped forward in time by calling \texttt{pwvtrigger()} with the new \( q \) and previous \( \sigma \) as arguments. Finally, \( q \) is converted from mm to cm height water.

**Input:**

- \texttt{lenrun} - run length in days
- \texttt{intsize} - model time step in hours
Output:

\( q \) - time series of column water vapor (cm)

\( \sigma \) - time series of precipitation state (0 or 1)

Routines called:

\( \text{fpwv()} \)

\( \text{prng()} \)

\( \text{levyrn()} \) (alt)

\( \text{pwvtrigger()} \)

**FPWV**   \( q \)-dependent parameters are computed using \( \text{fpwv()} \). Each parameter is identified by an integer 1-8 (\( \text{id} \)) and is evaluated using a stretched and shifted tanh function of \( q \). The values in \( q_{\text{mid}}, q_{\text{width}}, f_{-\infty}, \) and \( f_{+\infty} \) define the specific function for each of the different parameters:

\[
    f(q) = f_{-} + (f_{+} - f_{-}) \cdot \frac{1}{2} \left[ 1 + \tanh \left( \frac{q - q_{\text{mid}}}{q_{\text{width}}} \right) \right]
\]  \hspace{1cm} (4.1)

Input:

\( q \) - PWV (cm)

\( \text{id} \) - integer identifying which parameter is requested

Output:

\( f \) - evaluated parameter

**PRNG**   \( \text{prng()} \) generates a pseudo-random Gaussian number using the Box-Muller transform. Box-Muller is a pseudo-random number sampling method that generates pairs of independent, standard, normally distributed (zero expectation, unit variance) random numbers, given a pair of
uniformly distributed random numbers. It takes two samples from the uniform distribution on the interval $(0, 1]$ and maps them to two standard, normally distributed samples:

Given $u_1, u_2$ from the uniform distribution,

$$Z_1 = \sqrt{-2 \ln(u_1)} \cos(2\pi u_2) \quad (4.2)$$

$$Z_2 = \sqrt{-2 \ln(u_1)} \sin(2\pi u_2) \quad (4.3)$$

Then $Z_1$ and $Z_2$ are independent random variables with a standard normal distribution.

Output:

- $gr$: pseudo-random Gaussian number

LEVYRN  An alternate to prng(), levyrn() generates a random number from the standard (standard $\sim \mu = 0, \sigma = 1$) Levy $\alpha$-stable family. Using Weron’s (1996) algorithm which is based on the formulation by Chambers, Mallows, and Stuck (1976), two random numbers are generated: one from Unif(-pi/2,pi/2) and one from the exponential distribution with mean one. These are then transformed into one standard Levy variable, and then translated to a Levy variable with $\alpha = 1.9$ and $\beta = 0$ (see section 4.6).

Output:

- $lr$: Levy random number

PWVTRIGGER  pwvtrigger() simulates the stochastic trigger for precipitation using a markov jump process. A transition probability is computed as a function of $q$, then compared to a randomly generated critical probability. If the transition probability is greater than the critical probability, the precipitation state switches. Otherwise it remains the same.

Given a precipitation state $\sigma, r_{ij}$, where $i$ is the input precipitation state and $j$ is the new state, is computed with a call to fpwv(). The transition probability is computed: $p_{\text{trans}} = \text{the normal} \ldots$
inverse of $1 - e^{r_{ij}dt}$. Then a call to `prng()` generates a random critical probability $p_{crit}$, to which $p_{trans}$ is compared.

Input:

- $q$ - precipitable water vapor (cm)
- $\sigma$ - input precipitation state (0 or 1)
- $dt$ - model time step (hr)

Output:

- $\sigma$ - new precipitation state (0 or 1)

Routines called:

- `fpwv()`
- `prng()`

### 4.2.2 Correlation Code

**PWVCORR** `pwcorr()` computes and plots a time series of correlation coefficients between a long model PWV time series and the 25-day observed PWV time series using a moving window. Model and observed data are compared only at the points where valid observation data is available. The correlation coefficient is plotted vs time step of beginning of interval (25 days), and the maximum value is output, as well as an array of the coefficients. The correlation coefficient $r$ is computed as follows:

For two variables $x$ and $y$, and number of data points, $N$,

$$r = \frac{N \sum x y - \sum x \sum y}{\sqrt{(N \sum x^2 - (\sum x)^2)(N \sum y^2 - (\sum y)^2)}}.$$  \hspace{1cm} (4.4)
$r$ will be in the interval $[-1,1]$. The closer to +/- 1, the stronger the relationship. A negative value indicates a negative correlation.

Input:

$q$ - precipitable water vapor (cm)

$\text{obs}$ - 20 second observed data, valid values only (from file)

$\text{ind}$ - array of indices for which valid observed data is available (from file)

$\text{len}$ - length of full observation data (from file)

Output:

$\text{corrcoeff}$ - array of correlation coefficients

$maxcorr$ - maximum positive correlation coefficient

$maxcorr_t$ - time step corresponding to $maxcorr$

Routines called:

$\text{corrcfcomp}()$

$\text{CORRCFCOMP}$ This function computes the correlation coefficient between two time series, $\text{obs}$ and $q$, both of which contain data only at time steps for which valid observed data is available. The coefficient is computed using Equation 4.4.

Input:

$q$ - model PWV, at valid observed data time steps only (cm)

$\text{obs}$ - 20 second observed data, valid values only

Output:
4.2.3 Model Parameters

During initial testing, we found that we could not achieve the amount of variance found in the observed data using the model parameters defined by SN11. After considering the location upon which their parameters were based (Nauru), along with the quantitative analysis of the Darwin long-term data, we concluded that it would be appropriate to add a measure of multiplicative noise to the non-precipitating state; hence $D_0 dW_t$ was multiplied by a factor of 1.5 for $\sigma = 0$. Table 4.1 summarizes model parameters for the simulations described in this chapter.

Table 4.1: Model parameters for the SDE point model simulations.

<table>
<thead>
<tr>
<th>Parameter [units]</th>
<th>$f_{-\infty}$</th>
<th>$f_{\infty}$</th>
<th>$q_{mid}$ (cm)</th>
<th>$q_{width}$ (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{01}$ (h$^{-1}$)</td>
<td>0</td>
<td>1.0</td>
<td>6.1</td>
<td>0.2</td>
</tr>
<tr>
<td>$r_{10}$ (h$^{-1}$)</td>
<td>4.0</td>
<td>0</td>
<td>6.3</td>
<td>0.2</td>
</tr>
<tr>
<td>$E$ (mm$^2$ h$^{-1}$)</td>
<td>0.2</td>
<td>0.2</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>$P$ (mm$^2$ h$^{-1}$)</td>
<td>2.0</td>
<td>10.0</td>
<td>6.45</td>
<td>0.1</td>
</tr>
<tr>
<td>$D_0^2$ (mm$^2$ h$^{-1}$)</td>
<td>1.5</td>
<td>1.5</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>$D_1^2$ (mm$^2$ h$^{-1}$)</td>
<td>16.0</td>
<td>64.04</td>
<td>6.45</td>
<td>0.1</td>
</tr>
<tr>
<td>$D_{1F}$ (mm$^2$ h$^{-1}$)</td>
<td>16.0</td>
<td>64.00</td>
<td>6.45</td>
<td>0.1</td>
</tr>
<tr>
<td>$D_{1P}$ (mm$^2$ h$^{-1}$)</td>
<td>0.0</td>
<td>0.04</td>
<td>6.45</td>
<td>0.1</td>
</tr>
<tr>
<td>$D_0 dW_t$ factor</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>time step (s)</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_0$ (cm)</td>
<td>6.0414</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sigma_0$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.3 Observed-Model Correlation and Statistics

Using a 20-second time step to match the temporal resolution of the observations, we ran the SDE model for 1,000 days. It is unreasonable to expect a single 25-day simulation to conform to the observations, given the randomness of precipitation and the lack of data assimilation input into the model. Instead we examine statistical properties and temporal patterns from a long simulation to determine whether the model has the ability to reproduce the physical characteristics of the 25-day observed data. We began with a random initial state of $q = 6.0414$ cm and $\sigma = 1$ (non-precipitating), and looked at each 25-day section of simulations. We then looked for the section that best fits the
observed data, in addition to comparing the statistical properties of the simulated and observed
data. To find this section, we used the method of maximum correlation: for every 25-day time
window, we computed the correlation coefficient between the simulation and observed data.

To evaluate the quality of the SDE simulation over the entire 1,000 days we compare the first
four statistical moments of the observations and of the 40 non-overlapping 25-day sections of the
simulation. Table 4.2 lists the mean, standard deviation, skewness, and kurtosis each; the values for
the model reflect the mean and standard deviation of each moment over the 40 sections.

<table>
<thead>
<tr>
<th></th>
<th>Observed</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20-second</td>
<td>20-second</td>
</tr>
<tr>
<td>Mean (cm)</td>
<td>5.493</td>
<td>5.37 ± 0.29</td>
</tr>
<tr>
<td>SD (cm)</td>
<td>0.875</td>
<td>0.704 ± 0.240</td>
</tr>
<tr>
<td>Skewness</td>
<td>-0.5558</td>
<td>-0.5832 ± 0.3278</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>0.0316</td>
<td>0.0842 ± 0.7670</td>
</tr>
</tbody>
</table>

The standard deviation (SD) of the means from the 40 sections (0.29 cm) is about 5% of the average
of the mean (5.37 cm). This average is close to the observed mean (5.493 cm), the average of the
PWV in the entire TWP-ICE observation period from January 20 - Feb 13, 2006. This SD is quite
small, which may be a consequence of preset convergence, divergence, and evaporation parameters.
Despite the fact that these preset parameters largely determine the water mass balance over the
simulated region, our simulations are still able to replicate major precipitation processes which
demonstrates the effectiveness of both the random trigger and the stochastic forcing modeled by
Brownian motion.

The second moment of the simulation has much larger variations from section to section. The
standard deviation (0.240 cm) of the SD from the 40 sections is about 35% of the average SD (about
0.7 cm) from the 40 sections. This implies that the preset parameters can control the overall water
mass balance, but cannot regulate precipitation events, particularly extreme precipitation, due to
the randomness of realistic weather events and also to the intrinsic randomness of a stochastic model
such as this one.
The observed PWV data show noticeable skewness to the left (-0.5558), as do the 40 sections of the model (about -0.58). This shows that there are more negative deviations from the mean.

The kurtosis of the observed data is very small, 0.0316. Thus, the peakedness of the observed data is not very different from that of a Gaussian distribution, and non-Gaussianity is due to the left skewness, implying a long tail of small PWV values. The model kurtosis also has a large volatility 0.77 compared to the mean 0.08. Thus, the simulated kurtosis is not significantly different from zero, which may again be a consequence of the central limit theory.

For longer time averages of 20 minutes (a typical GCM time step) and one hour, the mean does not change significantly (see the two right-most columns of Table 4.2), nor does the standard deviation. The standard deviation of the means of the 40 sections does not change either. The SD of the SDE simulations also varies little from 20-second data to hourly data for our SDE model output. The skewness and kurtosis are again not significantly different from zero, as one would expect.

In addition to the indices of correlation and the first four statistical moments of the observed and model data, we also assessed the simulations using probability distribution. Figure 4.5 shows the histograms of the observed PWV and the model data from the section of best simulation. To explore the sensitivity of the probability distribution to the temporal average, we calculated the histograms for three time scales: 20 seconds, 20 minutes and one hour. Both the observed data and the simulated data are skewed left and they are not sensitive to the temporal average up to one hour. The histograms from observed data and model data resemble each other in these three time scales.
Figure 4.5: Histograms of the 25-day observed data (left column) and the best simulated PWV data (right column) in three time scales: 20 sec, 20 min, and 1 hour.
We identified the section that best fits the observed data by examining each 25-day section of simulations, i.e., a section of 108,000 time steps of 20 seconds. The section that corresponds to the first high peak in correlation compared to the observed TWP-ICE data is the 25-day interval starting from time step 124,828 on Day 29 of the model, shown in Figure 4.6 along with the observed data. This interval had one of the highest correlation coefficients between observations and simulation ($r = 0.6513$), based on a moving window of 25 days. It can be seen that the temporal variations are well-synchronized, implying similar precipitation properties. The fractal dimension of this section of the simulation is about 1.9, computed using the box-counting method (Moisy, 2006), which is comparable to the fractal dimension also of about 1.9 for the observed data and also larger than the fractal dimension 1.5 of Brownian motion.

![Figure 4.6: PWV observed (black curve) at TWP-ICE site 20 January 2006 - 13 February 2006 with time resolution of 20 seconds. The blue curve is the SDE model simulated data.](image)

We also examined the behavior of 25-day sections from the 1,000-day PWV simulations that disagree with the observed data to a large extent, selected according to large negative correlations and zero correlation. Figure 4.7 shows two sections with the correlation coefficients between observed and simulated data equal to -0.6752 (the largest magnitude of the negative correlation in Figure 4.8 at...
time step 3,916,168 on day 907) and 0.0 (which is non-unique). The red time series corresponds to the -0.6752 correlation coefficient and demonstrates that the SDE model can simulate almost the exact opposite phase in the major precipitation process, particularly between January 31st and February 7th: when the observed PWV increases (decreases), the simulated PWV decreases (increases), and when one oscillates around a constant, the other does the same. The green time series shows a section of zero correlation (starting time at time step 2,208,298 on day 512). As expected, the green time series resembles white noise.

In response to a suggestion that a time series of any kind, of sufficient length, would likely produce similar sections of high correlation, we generated a long time series of random noise of similar amplitude. Figure 4.9 shows the correlation coefficient time series between the observed data and the random noise data. It is clear that the model is able to reproduce the characteristics of the observed data, while random noise is not.

![Figure 4.7: The simulated section of maximum negative correlation (red curve) shows the opposite phase between the SDE model data and the observed PWV data, while the simulated section of zero correlation (green curve) resembles white noise.](image)

Observed (black)  
Model 1 (green)  
Model 2 (red)
Figure 4.8: Time series of moving window correlation coefficients between observed data and model simulation. Circles denote the most positive, most negative, and zero correlation.
Figure 4.9: Time series of moving window correlation coefficients between observed data and random time series.
4.4 Robustness Test

To test the robustness of our simulations, we made three 1000-day runs with different initial conditions from the observed data with initial time step at 00791, 23929 and 75057, all within the 25 observed days. The initial PWV value is from the corresponding observation. The initial precipitation indicator was determined by the increase or decrease of PWV in the next time step: $\sigma = 1$ (i.e., precipitation) for a decreasing PWV trend and $\sigma = 0$ (i.e., no precipitation) for an increasing PWV trend. Although the exact time series for each simulation is different due to the intrinsic randomness, the statistical properties are similar and do not ”remember” the initial conditions. This robustness property is desirable for the success of our idealized SDE model. See Table 4.3 for the statistics of these three runs and compare them with those in Table 4.2. The first four moments in Table 4.3 show little variation among the four simulations and hence demonstrate robustness of the simulations. The main statistical properties are determined by the preset model parameters. Initial conditions are quickly forgotten. The precipitation processes, driven by Brownian motion and random triggers, can be very different from one section of 25 days to another, which gives the variety of weather, while the climate, prescribed by statistics, mainly depends on the preset environmental conditions.

<table>
<thead>
<tr>
<th></th>
<th>Random IC</th>
<th>IC 00791</th>
<th>IC 23929</th>
<th>IC 75057</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean (cm)</td>
<td>5.37 ± 0.29</td>
<td>5.29 ± 0.50</td>
<td>5.38 ± 0.32</td>
<td>5.29 ± 0.41</td>
</tr>
<tr>
<td>SD (cm)</td>
<td>0.704 ± 0.240</td>
<td>0.722 ± 0.251</td>
<td>0.681 ± 0.183</td>
<td>0.785 ± 0.325</td>
</tr>
<tr>
<td>Skewness</td>
<td>-0.5832 ± 0.3278</td>
<td>-0.5730 ± 0.4111</td>
<td>-0.6948 ± 0.4621</td>
<td>-0.5793 ± 0.3878</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>0.0842 ± 0.7670</td>
<td>0.0976 ± 0.9505</td>
<td>0.3288 ± 1.5776</td>
<td>0.0135 ± 0.6781</td>
</tr>
</tbody>
</table>

Figure 4.10 summarizes the observed and modeled time series of the maximum correlation, minimum correlation, and zero correlation with the observed data in all the four simulations. Figures 4.10a-d correspond to the four columns of Table 4.3. Among the four simulations, all the blue time series, corresponding to the maximum correlation, can simulate the observed major precipitation process, although the onset and ending time of the process vary. All the green time series, corresponding to the zero correlation, display the nature of white noise, which still allows slow temporal waves for a
particular realization, such as in Figure 4.10c. All the red time series, corresponding to the largest negative correlation, show the oscillation phases opposite to the observed data, as discussed earlier. This level of robustness demonstrates the feasibility of using stochastic models forced by Brownian motion.
Figure 4.10: Summary of the four 1000-day model runs, for maximum (blue), minimum (red), and zero (green) correlation between observed and model data. Vertical blue lines indicate the location of the initial condition in the observed time series.
4.5 Sensitivity Test

The best 25-day simulation (shown in Figure 4.6) corresponds to the global maximal correlation of 0.6513. To confirm that other local extremes also correspond to good SDE simulations of the observed PWV data, we selected five local maxima, five local minima and a zero of the correlation coefficients between the simulated data and the observed data. The corresponding time simulations and the observed data are shown in Figure 4.11, which demonstrates the fidelity of the SDE model: the most positive correlations (in red and yellow) are well synchronized with the observed data with similar lower frequency variations, while the most negative correlations (in deep blue and sky blue) are in opposite phases of lower frequency oscillations with observations: the observed decrease corresponding to the simulated increase.

Figure 4.12 shows the time series of moving-window correlation coefficients with the 5 positive extremes and 5 negative extremes denoted with circles. There appears to be a significant level of periodicity in the time series, which leads us to conclude that 1,000 days of simulation are not necessary to produce a good 25-day section of simulation. The periodicity of the correlation coefficients in Figure 4.12 and the above sensitivity test suggest that a simulation of about 30 days should produce a good 25-day simulation to be synchronized with the observed data.
Figure 4.11: Sensitivity test for different local correlation maxima and minima, using extremes of varying magnitude. Results show that the positive extremes still follow the lower-resolution trends of the observed data, and the negative extremes follow opposing trends.
Figure 4.12: Correlation coefficient time series for sensitivity test. Circles denote the extremes used in Figure 4.11.
4.6 Levy Motion

While we were able to achieve good results with the toy model utilizing Brownian motion, there is further interest in generalizing the stochastic process, so that other distributions might be used to fine tune the statistics of the model, e.g. for extreme events or varied event frequency. The normal distribution used in the original model may not allow large fluctuations and therefore limits high-variability solutions (Weron, 1996).

This suggests that with various parameter modification and fine-tuning, Levy motion can be used to improve model simulations in the future.

The normal (Gaussian) distribution is one family of Levy $\alpha$-stable distributions, so if a generalization of the original distribution were to produce sufficient results, there would be a large family of distributions that could be potentially be used to improve model simulations. Levy $\alpha$-stable distributions are defined by four parameters:

$$\alpha \text{ - stability} \in (0, 2]$$
Figure 4.14: Correlation coefficient time series for Levy motion-driven sensitivity test. Circles denote the extremes used in Figure 4.13.

\[ \beta \text{ - skewness } \in [-1,1] \]

\[ \sigma \text{ - scale } \in (0, \infty) \]

\[ \mu \text{ - location } \in (-\infty, \infty) \]

For normal distributions, \( \alpha = 2.0 \) and \( \beta = 0 \); the distribution becomes more peaked as \( \alpha \) decreases and \( \beta \neq 0 \) results in a skewed distribution (right for \( \beta > 0 \) and left for \( \beta < 0 \)).

Normal (pseudo-)random variables are simple to generate; however in general stable distributions have no analytic expressions for the inverse \( F^{-1} \) (Weron, 1996). Chambers et al. (1976) developed a direct method which transforms two independent random variables - one \( \sim Unif(-\frac{\pi}{2}, \frac{\pi}{2}) \) and one \( \sim Exp(1) \) - into one standard stable random variable \( X \sim S_\alpha(1, \beta, 0) \). This algorithm, clarified and proved by Weron (1996) can be applied in the following steps, given \( \alpha \) and \( \beta \):

- Generate a random variable \( V \) uniformly distributed on \( (-\frac{\pi}{2}, \frac{\pi}{2}) \) and an independent exponen-
tial random variable $W$ with mean 1.

- For $\alpha \neq 1$ compute

\[
B_{\alpha, \beta} = \frac{\tan^{-1} \left( \beta \tan \frac{\pi \alpha}{2} \right)}{\alpha} \quad (4.5)
\]

\[
S_{\alpha, \beta} = \left[ 1 + \beta^2 \tan \frac{\pi \alpha}{2} \right]^{1/2\alpha} \quad (4.6)
\]

Then,

\[
X = S_{\alpha, \beta} \frac{\sin(\alpha(V + B_{\alpha, \beta}))}{(\cos V)^{1/\alpha}} \left( \frac{\cos(V - \alpha(V + B_{\alpha, \beta}))}{W} \right)^{1-\alpha/\alpha} \quad (4.7)
\]

- For $\alpha = 1$ compute

\[
X = \frac{2}{\pi} \left[ (\frac{\pi}{2} + \beta V) \tan V - \beta \log \left( \frac{\pi}{2} W \cos V \right) \right] \quad (4.8)
\]

This produces a standard random variable, $X \sim S_{\alpha}(1, \beta, 0)$. For any permitted values of $\mu$ and $\sigma$, if $X \sim S_{\alpha}(1, \beta, 0)$ then

\[
Y = \begin{cases} 
\sigma X + \mu & \alpha \neq 1 \\
\sigma X + \frac{2}{\pi} \beta \sigma \log \sigma + \mu & \alpha = 1.
\end{cases} \quad (4.9)
\]

follows $S_{\alpha}(\sigma, \beta, \mu)$.

With this random number generator, Gaussian noise was replaced with noise from the stable distribution $S_{1.9}(\sigma, 0, \mu)$. Additionally, since this replacement increased fluctuations and therefore increased the likelihood of computed PWV becoming negative, the value of $D_0(q)$ was decreased (from 2.0 to 0.20) in the stochastic differential equations based on the mass balance.

We again performed a simulation of 1,000 days to investigate the feasibility of using a general Levy motion. As with the Brownian motion-driven model, we selected five local maxima, five local minima and a zero of the correlation coefficients between the simulated data and the observed data. The corresponding time simulations and the observed data are shown in Figure 4.13, which demonstrates the feasibility of the Levy-driven SDE model: the most positive correlations (in red and yellow) are well synchronized with the observed data with similar lower frequency variations, while the most negative correlations (in deep blue and sky blue) are in opposite phases of lower frequency

60
oscillations with observations: the observed decrease corresponding to the simulated increase.

Although the simulations are of lower quality than our Brownian motion-driven simulations, this serves as a test-of-concept for the use of additional stochastic processes that might be able to specify other aspects of the random behavior.

4.7 Conclusions

We have used an SDE model for the mass balance, driven by Brownian motion, to simulate the 20-second temporal resolution TWP-ICE PWV data from January 20 - Feb 13, 2006. The time series of the observed PWV data appears to be governed by a stochastic process with precipitation controlled by a random precipitation trigger. In the model, this trigger is determined by an exponential cumulative distribution function of a rate parameter and the time step size in an SDE simulation, compared with a random critical precipitation parameter uniformly distributed over [0,1]. The simulated and the observed datasets are similar, measured by various statistical indices, including the mean, standard deviation, skewness, kurtosis, extremes, fractal dimensions and probability distributions. The statistics of our simulations imply that the preset parameters of evaporation, precipitation, convergence and divergence can control the overall water mass balance and hence the mean PWV, but cannot regulate precipitation events, particularly extreme precipitations and droughts, due to both the randomness of realistic weather events and the intrinsic randomness of a stochastic model. The statistics also demonstrate the effectiveness of both random trigger and the stochastic forcing modeled by Brownian motion.

Robustness and sensitivity tests further indicate that the statistical indices from long-term averages are determined by the preset environmental conditions, including evaporation, convergence and divergence, and form the climate, while the precipitation events are random in simulations and hence result in a variety of weather conditions. Our successful simulation may be regarded as a feasibility test of using an SDE model to simulate precipitation processes in more complex climate models. The robustness in the simulations also supports using stochastic models in SCM and GCM forced by Brownian motion or Levy motion.
The fractal dimension of about 1.9 from our best simulation data is similar to that of the observed data, demonstrating physically reasonable anti-persistence. The persistence and anti-persistence are related to similarity of current and future patterns and hence to the auto-correlation of a times series and may also be measured by Hurst exponent computed from auto-correlation analysis and other multifractal indices (Olsson et al., 1993; Schertzer and Lovejoy, 1987). For realistic simulations in a general sense, we can further control the complexity of the SDE model by considering Levy motion and time varying control of convergence and divergence, as well as the vertical structure of the atmosphere. The research knowledge gained from the analysis of the high resolution observed data for tropical or extratropical convection, such as multiscale analysis of Tropical Ocean Global Atmosphere Coupled Ocean-Atmosphere Response Experiment (TOGA-COARE) time series of precipitation and other variables (Sui et al., 1997), should be helpful to identify the appropriate complexity for the SDE model to improve PWV and precipitation simulations.

This mass balance point model has demonstrated its unique ability to simulate the PWV observed time series. Thus the next step toward implementing this parameterization in a global model is by starting with a single-column model.
5 Single-Column Modeling

5.1 SCM Background

A single column model represents a single vertical array over a gridpoint of a GCM (Figure 5.1), where since the column is unable to interact dynamically with its surrounding columns, the large-scale forcing and boundary conditions must be prescribed at every time step (depicted by blue arrows). Tompkins and Emanuel (2000) found that two single-column models were sensitive to vertical resolution, indicating the need to consider the vertical structure of the water vapor profile; thus a single-column model is the logical next step. As a SCM is isolated from the global model, the dynamics must be specified in the form of large-scale forcing derived from observations or objective analyses. Although this lack of feedback between the large-scale dynamics and the column simulation can result in solution drift (Xie and Zhang, 2000), this simplified environment allows for simpler and faster testing of parameterization schemes than would be possible in a full GCM (i.e. 1-dimensional versus 3-dimensional).

Although we had initially considered using a simple standalone SCM (MIT SCM) (Emanuel, 2010), we ultimately choose to work with a single column model that is part of a global model, specifically NCAR’s SCAM and CAM, respectively. This will allow for a simpler transition to the GCM as part of a future study (Hack et al., 1999).
5.2 NCAR: SCAM

The National Center for Atmospheric Research (NCAR) has developed a single column model as a part of their Community Atmosphere Model (CAM). The Single-column Community Atmosphere Model (SCAM) takes as forcing input CAM2-generated initial and boundary condition datasets as well as selected data from various field experiments.

SCAM is a one-dimensional time-dependent version of the NCAR CAM. A single column model will predict the local time-rate-of-change of the large-scale state variables (e.g., temperature, moisture, momentum, cloud water, etc.) using specified horizontal flux divergences, a specified vertical motion field (from which the large-scale vertical advection terms are evaluated), and subgrid-scale sources, sinks and eddy transports. The subgrid-scale contributions are determined by an arbitrary collection of user-selected subgrid-scale physics parameterizations (NCAR SCAM, web).
The SCAM-2 distribution will build a single column version of the CAM2.0 model released on May 17, 2002. The single column framework was designed to use much of the original CAM2 code and data structures as possible. Therefore all physics parameterizations and surface components are identical in coding structure to their respective counterparts in the 3D global model making it trivial to migrate components between the two modeling frameworks (NCAR SCAM, web). SCAM was chosen for this particular reason, so that successful modifications to the convection scheme in the SCM can easily be applied to the GCM. Specifically, SCAM parameterizes convection with the Zhang-McFarlane Scheme, which uses the presence of positive CAPE as the criteria for the onset of convection.

The overall design includes the provision of an optional graphical user interface (GUI) to the model. The GUI provides the user with control of code flow including: dataset selection; column location (latitude/longitude) selection; modification of control variables (such as termination conditions, update frequencies, specification of history data, etc.); modification of initial data and the associated large-scale forcing (e.g., modification of vertical structures, amplitudes, etc.); and the visualization of output data (NCAR SCAM, web).

Access to SCAM is via Yellowstone, NCAR’s 1.5-petaflops high-performance IBM iDataPlex cluster, which features 72,576 Intel Sandy Bridge processors and 144.6 TB of memory. The HPC cluster’s login nodes give users access to the entire Yellowstone environment, which includes the Geyser and Caldera clusters, the 16-PB GLADE central disk resource, and the High-Performance Storage System (https://www2.cisl.ucar.edu/resources/yellowstone).

Despite certain limitations, SCAM has proven to be an effective numerical experimentation platform using minimal computing resources to simplify the investigation and improvement of parameterizations of radiative and moist processes in atmospheric general circulation models (NCAR SCAM, web).
5.3 Coupling the SCM and the Mass Balance Model

For this study, we couple the SCM and the mass balance model by considering the SCM-computed value of PWV to be the deterministic (or average) part of a total \( q \), and we compute the variation in \( q \) using the SDE formulation of the mass balance model. CAPE is computed as a function of the vertical distributions of humidity and temperature, so in adding variation to the humidity as in the algorithm described earlier, we implicitly incorporate a stochastic trigger for the transition to convection. In order to ensure that mean state and variation are coupled, the variation at a given time step is computed as a function of the total \( q \) (which includes both deterministic and random parts), while evaporation and precipitation are considered to be functions of the average (deterministic) \( q \).

Consider the change in total PWV for a given time step to be

\[
\frac{dq}{dt} = \text{(evaporation or precipitation) + large-scale forcing} \tag{5.1}
\]

where evaporation and precipitation are functions of the mean state from the SCM, and large-scale forcing is random from the mass balance model. Then,

\[
\frac{dq}{dt} = f_1(\overline{q}) + f_2(q) \tag{5.2}
\]

where \( \overline{q} \) is the (deterministic) mean state computed by the SCM and \( q \) is the total PWV computed by coupling the SCM and mass balance model; and

\[
q(t + dt) = \begin{cases} 
q(t) + E(\overline{q}(t))dt + D_0(q(t))dW_t & \text{if } \sigma(t) = 0 \\
q(t) - [P(\overline{q}(t))dt + D_P(q(t))dW_{P,t} + D_F(q(t))dW_{F,t}] & \text{if } \sigma(t) = 1.
\end{cases} \tag{5.3}
\]

5.4 Details of Proposed Scheme

To implement such a scheme as described above, we have designed a module of subroutines that can be included in the SCAM’s source code and called from the convection routine at each time...
step. Once the humidity profile has been computed for a given time step in SCAM, the stochastic module will be called and a new humidity profile (which includes the addition of variation) will replace the deterministic one.

Input for the stochastic module must include humidity profiles $q_{m-1}$ and $q_m$ (each of which is a function of pressure at their respective time steps) and the value of convective precipitation at time step $m - 1$. We consider $t_{m-1}$ to be the “previous” time step, and $t_m$ to be the “current” time step - the step for which we want to compute the variation\(^1\). Note, for each time step, the previous value of $q$ must be retained in addition to the newly-computed value.

Since the mass balance model uses a column-integrated value of water vapor (in mm) and SCAM computes a humidity profile, we must first convert the humidity profiles to PWV. We achieve this by vertically integrating the humidity profile according to the method documented by Solot (1939). For a column of water vapor with a with horizontal cross-section of area unity:

$$M_v = -\frac{1}{g} \int_p q(p)dp$$

(5.4)

where $M_v$ is the mass of the water vapor per unit area [kg/m\(^2\)], $g$ is gravitational acceleration [m/s\(^2\)], $q$ is specific humidity [kg/kg], and $p$ is pressure [Pa]. $M_v$ can then be converted to height by dividing by the density of water (999.97 kg/m\(^3\)) and converting from m to mm. We utilize the Trapezoid Rule for numerical integration, which states that:

$$\int f(x)dx \approx \frac{\Delta x_01}{2}(f(x_0) + f(x_1)) + \frac{\Delta x_{12}}{2}(f(x_1) + f(x_2)) + \cdots$$

(5.5)

Then, we can compute $M_v$ with

$$M_v = -\frac{1}{g} \sum_i \left[ \frac{p_i - p_{i-1}}{2}(q(p_i) + q(p_{i-1})) \right]$$

(5.6)

where $i$ denotes the level ($i = 0$ is at the surface). Figure 5.2 illustrates the Trapezoid Rule for vertical integration, where the integrated value is the area between the vertical axis and the $q$ curve. The Trapezoid Rule computes the area inside each trapezoid formed by the four points $p_i, p_{i-1}$, $p_{i+1}, p_i$.

\(^1\)m denotes SCAM time steps, while $n$ will be used to denote mass balance model time steps.
\( q(p_i), \) and \( q(p_{i-1}) \), and sums to result in the total area.

Figure 5.2: Illustration of the trapezoid rule for vertical integration. \( q \), the specific humidity as a function of pressure, is integrated over the pressure profile to result in a value of total column precipitable water vapor. PWV then, is the area between the vertical axis and the \( q \) curve.

The next step, and the heart of the scheme, resolves the disparate temporal resolutions of the two models, which in turn leads to the coupling mechanism. SCAM and the mass balance model utilize time steps of 20 minutes and 20 seconds, respectively. An ostensible solution would involve simply modifying the mass balance model to use 20-minute time steps; however, to capture the variation that occurs at the smaller time scales, it is necessary to retain the small time steps and couple the two models. We propose to address this by considering the separation of the SDE into \( f_1(\eta) \) and \( f_2(q) \) described in Equations 5.1-5.3.

As illustrated in Figure 5.3, the 20-minute SCM time step is divided into a uniform grid consisting of 60 20-second time steps (denoted by \( n \)), where \( t_{n=0} \sim t_{m-1} \) and \( t_{n=60} \sim t_m \). We consider this time series to be a series of \( \eta \) values, the deterministic version of each \( n \)-time step. Then, stepping through the time series, the total PWV is computed at each step as a function of both the deterministic and the total (deterministic + stochastic) values at the previous step.
At $t_0$, let $q_0 = \bar{q}_0$, and let $\sigma$ be initialized by the value of precipitation at $t_m$. Then compute

$$f_1(\bar{q}_0) = \begin{cases} E(\bar{q}_0)dt & \text{if } \sigma(t) = 0 \\ -P(\bar{q}_0)dt & \text{if } \sigma(t) = 1. \end{cases}$$ \hfill (5.7)$$

and

$$f_2(q_0) = \begin{cases} D_0(q_0)dW_t & \text{if } \sigma(t) = 0 \\ -[D_P(q_0)dW_{P,t} + D_F(q_0)dW_{F,t}] & \text{if } \sigma(t) = 1, \end{cases}$$ \hfill (5.8)$$

summing with $q_0$ to get

$$q_1 = q_0 + f_1(\bar{q}_0) + f_2(q_0).$$ \hfill (5.9)$$

Thereafter in general,

$$q_{n+1} = q_n + f_1(\bar{q}_n) + f_2(q_n).$$ \hfill (5.10)$$

Once $q_n$ has been computed, the trigger function described in Section 3 determines the precipitation state for the next time step. After stepping through all $n$-time steps, we have a total value of $q$ for $t_{60}$ (or $t_m$) which includes both the mean state as a function of the SCM computed PWV and the accumulation of variation over the 60 20-second time steps.

After the total PWV has been computed for $t_m$, it is then a matter of redistributing the vapor through the pressure profile as a modified humidity profile to be returned to the convection scheme. In order to maintain the shape of the original humidity profile at time step $t_m$, we wish to add the variation in a way that maintains the proportion of water vapor at each pressure level.

We compute a redistribution constant, $C_r$, as the ratio of the total PWV to the deterministic part:

$$C_r = \frac{q_{60}^{\text{tot60}}}{\bar{q}_{60}}.$$ \hfill (5.11)$$

If both sides of Equation 5.6 are multiplied by this constant, it is clear that $C_r$ can be moved inside the summation and distributed to the sum of $q(p_i)$ and $q(p_{i-1})$. Thus, we can multiply the elements of the deterministic humidity profile by $C_r$ to produce a new profile with the variation distributed proportionally.
This new profile is returned to the convection scheme replacing the deterministic profile computed for time step $t_m$. The SCM continues through its time steps, calling the stochastic module at each step from the convection routine. Figure 5.4 illustrates the process, starting from the computation of the humidity profile in the convection scheme, and ending with the return of the new total humidity profile.
where $q_n = q_{n-1} + f_1(q_{n-1}) + f_2(q_{n-1})$

Figure 5.3: Model coupling. The inset shows a subset of the 60 20-second time steps and illustrates the process of accumulating the variation. The variation at each time step is computed as a function of the previous time step’s total water vapor, then added to the deterministic value. This total can then be used to compute the variation at the next time step. The final value at the last 20-second time step is returned to the SCM.
Figure 5.4: Stochastic module program flow. The scheme starts with $q$ for two consecutive time steps from the SCM; each humidity profile is integrated and then 60 20-second time steps are interpolated between them. The variation is computed at each time step consecutively, and the final value is returned to the SCM as a redistributed humidity profile.
6 Simulation of Observed Data - Single-Column Model

6.1 Description of Code

The SCAM source code consists of a collection of .F90 files containing modules of variables and subroutines, in a hierarchy of directories. Figure 6.1 shows that the relevant files, denoted in gray, can be found in two main directories, bld (contains scripts to build and compile model) and src (contains all source code), both found in cesm1_2.0/models/atm/cam/. To modify code, a modifications directory must be created, and any .F90 file which will be modified must be copied from the source directory to this directory. Once modifications are made on these copies, the directory is specified in the run script; during compilation, the model will check this directory, and if it finds any files with the same name as original source code, it will use the new version instead. If there are additional files, it will compile these as well (See Appendix B for relevant code).
Figure 6.1: Directory structure of CESM files relevant to implementation of stochastic scheme modifications. Gray denotes the files that require modification in order to implement stochastic scheme.

### 6.1.1 Interfacing Requirements

While the structure of the SDE modification routines is similar to that of the point model routines, implementation is complicated by the need to interface with NCAR’s existing single-column model (in addition to porting the code from MATLAB to Fortran 90). The general requirements for this aspect are:

- Access SCAM-computed precipitation state and pressure and humidity profiles at each step
- Call the stochastic module of routines
- Output new fields
SCAM-computed pressure and humidity (mixing ratio) profiles are stored in a structure called *state*. Pressure at the \( pver + 1 \) interfaces of each of \( pver \) vertical layers are stored in *state\%p*, while humidity at \( pver \) vertical layers is stored in *state\%q*. Precipitation from the Zhang-McFarlane convection scheme is *prec\_dp*, and the SCAM (20-minute) time step number is *nstep*.

The stochastic module is called right after the moist convection scheme via the subroutine *sconvect()*, and new output fields produce the 20-second PWV and precipitation state data.

New output fields are sent to the output file using the CESM *addfld()* and *outfld()* routines. Six arguments are passed to *addfld()* (CSEG, 2014):

1. Field name: 8-character field name, left-justified, alphanumeric or spaces only.
2. Field units: 8-character units description.
3. Number of vertical levels in the field. 1 for surface, \( pver \) or \( \text{pverp} \) for multi-level.
4. Default averaging flag. The flags available are: Instantaneous (I), Average (A), Maximum (X), and Minimum (M).
5. Field descriptor: up to 128-characters.
6. Parallel decomposition type (i.e. is this a physics or dynamics variable). \( \text{phys\_decomp} \) or \( \text{dyn\_decomp} \).

Four arguments are passed to *outfld()*:

1. 8-character field name.
2. Variable array in which the field is stored.
3. First (longitude) dimension of the data array.
4. Chunk (physics) or latitude (dynamics) index.
6.1.2 Module: Global Variables

Variables needed between time steps are included in a separate module of global variables, stochvar.F90. pstate stores the precipitation state (0 for non-precipitating, or 1 for precipitating), while pwv1 stores the (column-integrated) PWV to be carried to the next time step.

6.1.3 Modifications to Existing Routines

physpkg.F90, located in src/physics/cam/, contains the subroutines phys_init() and tphysbc(), which both required modification. phys_init() initializes the physics package. It is necessary to add a number of USE statements to ensure access to certain global variables and subroutines:

use cam_history, only: addfld, phys_decomp (for adding output fields)

use ppgrid, only: pver (number of vertical levels)

use stochproc, only: init_random_seed (initialization of random number generator).

phys_init() is also where the new addfld calls are inserted, and where the random number generator is initiated with a call to the subroutine init_random_seed().

In tphysbc, we add USE statements as well as calling the stochastic module and assigning a precipitation state to a global variable:

use ppgrid, only: pcols, pver (dimensions for output fields)

use stochvar (module containing global variables for stochastic scheme)

use stochproc (module containing routines for stochastic scheme).

forecast.F90, located in src/dynamics/eul/ contains the subroutine forecast(), which includes the scheme to relax the SCAM to observed and reanalyzed data. Relaxation is a type of data assimilation and helps keep the model from drifting too far from the observed data by picking up
any additional tendency the model schemes may have missed. Relaxation time was reduced from
the default of 3 hours to 40 minutes (two SCAM time steps).

6.1.4 Module: Additional Subroutines for the Stochastic Scheme

Subroutines for the stochastic scheme (listed below) are contained in a module named stochproc.F90.
Aside from init_random_seed(), they are all part of the call to the stochastic scheme, the call
structure of which is illustrated in Figure 6.2 and described in Section 6.1.5.

sconvect(): main calling program; computes stochastic humidity profile

vertint(): vertically integrates water vapor

tsinterp(): interpolates PWV at 20-second resolution

mbvar(): mass balance model implementation

tanhfn(): computes parameters for mass balance model

trigger(): stochastic trigger for precipitation

prng(): pseudorandom number generator (Gaussian)

init_random_seed(): initializes random number generator

norminv(): computes inverse to normal distribution cumulative distribution function (CDF)

ratappx(): approximation of normal inverse; part of norminv()

redist(): redistributes PWV among vertical levels

SCONVECT  sconvect() is the main routine called by SCAM to simulate stochastic parameterization of convection. It takes the humidity arrays from each end of a 20-minute time step from SCAM, along with the associated pressure profiles, and adds variation computed via the stochastic mass balance model. The humidity ($q$) arrays are vertically integrated to produce a value
of precipitable water vapor (PWV) at each end of the time step, between which PWV is interpolated at 20-second intervals $(n = 0,1,2,...,60)$. Using a model derived from the mass balance, each value of PWV is used to compute the variation for the next step, which is then added to the interpolated value of PWV and used for the next 20-second time step calculation. Calls to the SCAM subroutine `outfld()` send the 20-second PWV and $\sigma$ series for each SCAM time step to output.

At the end, the total value of PWV for $n = 60$ is redistributed among the levels according the original proportional distribution of $q$ at the right-hand time step and this array is returned to the SCM convection scheme as a replacement for the deterministic profile of $q$. Note that SCAM computes $q$ in units of kg/kg. We compute PWV in kg/m$^2$ but then convert to mm height of water to be compatible with the mass balance model parameters. Additionally, $\Delta t$ is the 20-minute time step used by SCAM, while $dt$ is the 20-second resolution time step used by the mass balance model.

Input:

- $m$ - current time step

- $p_2$ - pressure profile at current time step (Pa)

- $q_{det2}$ - mixing ratio profile at current time step (kg/kg)

- $lchnk$ - chunk (physics) index for `outfld()` call

Output:

- $q_{det2}$ - updated mixing ratio profile reflecting stochastic mass balance (kg/kg)

Routines called:

- `vertint()`

- `tsinterp()`

- `mbvar()`

- `redist()`
VERTINT  vertint() vertically integrates the specific humidity in a column, to result in the precipitable water vapor (PWV) for that column. It numerically integrates using the Trapezoid Rule:

\[ \text{PWV} = \sum_{i=1}^{n} \frac{1}{g}(p_{i+1} - p_{i})q_{\text{avg}} \]  

(6.1)

where \( q_{\text{avg}} \) is the average \( q \) between \( i \) and \( i + 1 \) levels, \( p \) is the pressure at the interface between two levels (SCAM indexes \( p \) from the highest elevation to the surface), and \( g \) is the acceleration of gravity. We then convert from the resulting units of \( \text{kg/m}^2 \) to mm height of water.

Input:

- \( n \) - number of vertical levels
- \( p \) - pressure profile (Pa)
- \( q \) - mixing ratio profile (kg/kg)

Output:

- \( \text{pwvreturn} \) - PWV for the column

TSINTERP  tsinterp() linearly interpolates between the two values of PWV at either end of a 20-minute interval to create a uniform 20-second grid.

Input:

- \( \text{intsteps} \) - total number of steps (61) over which to interpolate
- \( \text{pwvl} \) - PWV at \( t \) (mm), stored as a global variable
- \( \text{pwvr} \) - PWV at \( t + \Delta t \) (mm), one 20-minute time step later

Output:

- \( \text{pwvint} \) - \( \text{intsteps} \)-long array of interpolated PWV (mm)
MBVAR \texttt{mbvar()} simulates the change in precipitable water vapor with a stochastic trigger for precipitation and stochastic closures for precipitation and other large-scale external forcings. Based on the point model described in Stechmann and Neelin’s 2011 paper, this function utilizes the mass balance to compute the change in PWV at each time step, as a function of PWV from the previous time step. To couple the point model to SCAM, we assume that the deterministic values provided by the SCM and our interpolation are equivalent to the average value, and compute \( E \) and \( P \) as functions of the average PWV from the previous time step. The stochastic trigger and closures are based on the total PWV - including variation - from the last time step.

Given \( \bar{q} = \) average (deterministic) PWV, and \( q_{\text{tot}} = \) total (stochastic) PWV, we compute

\[
q_{t+1} = \begin{cases} 
q_{\text{tot},t} + E(\bar{q}_t)dt + D_0(q_{\text{tot},t})dW_t & \leftrightarrow \sigma = 0 \\
q_{\text{tot},t} - [P(\bar{q}_t)dt + D_P(q_{\text{tot},t})dW_{P,t} + D_F(q_{\text{tot},t})dW_{F,t}] & \leftrightarrow \sigma = 1. 
\end{cases}
\]  

(6.2)

where \( P \) and \( E \) are precipitation and evaporation rates; \( D_0, D_P, \) and \( D_F \) are the stochastic noise coefficients for forcing and precipitation; and \( dW \) is Gaussian noise.

The precipitation state of the next step is then determined with a call to the stochastic trigger function \texttt{trigger()} which takes as input the PWV just computed and the value of \( \sigma \) (stored as a global variable) from the previous step.

Input:

\begin{itemize}
  \item \texttt{sigma1} - precipitation state from previous time step
  \item \texttt{pwvtot1} - total (stochastic) PWV from previous time step (mm)
  \item \texttt{pwvdet1} - average (deterministic) PWV from previous time step (mm)
  \item \texttt{dt} - time step interval size (20 seconds)
\end{itemize}

Output:

\begin{itemize}
  \item \texttt{sigma2} - new precipitation state
\end{itemize}
**totpwv** - total PWV for current step (mm)

Routines called:

- prng()
- tanhfn()
- trigger()

**PRNG**  This subroutine generates pseudo-random Gaussian numbers using the Box-Muller transform. Box-Muller is a pseudo-random number sampling method that generates pairs of independent, standard, normally distributed (zero expectation, unit variance) random numbers, given a pair of uniformly distributed random numbers. It takes two samples from the uniform distribution on the interval (0, 1] and maps them to two standard, normally distributed samples:

Given $u_1, u_2$ from the uniform distribution,

\[
Z_1 = \sqrt{-2 \ln(u_1)} \cos(2\pi u_2) \quad (6.3)
\]
\[
Z_2 = \sqrt{-2 \ln(u_1)} \sin(2\pi u_2) \quad (6.4)
\]

Then $Z_1$ and $Z_2$ are independent random variables with a standard normal distribution.

Output:

- \( r_n \) - pseudo-random Gaussian number

**INIT_RANDOM_SEED**  This subroutine (from https://gcc.gnu.org/onlinedocs/gcc-4.2.1/gfortran/RANDOM_005fSEED.html) initializes the seed for the random number generator.

**TANHFN**  \texttt{tanhfn()} computes the various parameters needed for the mass balance model, using a stretched and shifted tanh function of \texttt{pwv}. The values in $q_{mid}$, $q_{width}$, $f_{-\infty}$, and $f_{+\infty}$ define
the specific function for each of the different parameters, and \texttt{fnid} identifies the specific function requested by the calling program.

Given a PWV $q$,

$$f(q) = f_{-\infty} + (f_{+\infty} - f_{-\infty}) \cdot \frac{1}{2} \left[ 1 + \tanh \left( \frac{q - q_{mid}}{q_{width}} \right) \right]$$

where $q_{mid}$ is the midpoint of the tanh curve (transition), $q_{width}$ is the width of the transition $f_{-\infty}$, and $f_{+\infty}$ are the asymptotic values of the function at $-\infty$ and $+\infty$, respectively.

Input:

\texttt{pwv} - PWV for current step (mm)

\texttt{fnid} - identifies which parameter to compute

Output:

\texttt{tanhfn} - resulting function value

\textbf{TRIGGER} \hspace{1em} \texttt{trigger()} simulates the stochastic trigger for precipitation. It takes as input the current value of PWV and $\sigma$, the precipitation state from the previous time step. The “transition rate”, $r$, is computed as a function of the PWV by calling \texttt{tanhfn()}. Depending on the previous precipitation state we compute either $r_{01}$ or $r_{10}$, i.e. the transition rate from 0 to 1 or 1 to 0, 0 denoting non-precipitating and 1 denoting precipitating. Next, a transition probability is computed, where $p_{trans} = 1 - e^{rdt}$, and then the inverse normal of $p_{trans}$ is computed in order to get the corresponding Gaussian value. Finally, a Gaussian random critical probability ($p_{crit}$) is generated with the \texttt{prng()} function, and the two Gaussians are compared: if $p_{trans}$ is greater than or equal to $p_{crit}$, the precipitation state switches, otherwise it remains the same.

Input:

\texttt{pwv} - current value of pwv

\texttt{sig1} - precipitation state for previous step
dt - time step size

Output:

\[ \text{sig2} \] - precipitation state for current time step

Routines called:

\[ \text{tanhfn()} \]

\[ \text{prng()} \]

\[ \text{norminv()} \]

**NORMINV** The \text{norminv()} function computes a rational approximation of the inverse normal CDF. Since the analytical function is not trivial to compute, we use an algorithm described in *The Handbook of Mathematical Functions* (Abramowitz and Stegun, 1965) as a “rational approximation for \( x_p \), where \( Q(x_p) = p \)” and documented by John D. Cook (http://www.johndcook.com/blog/normal_cdf_inverse/).

The probability \( p \) is first transformed into a parameter \( t \) which is:

\[
\begin{align*}
  t_1 &= \sqrt{-2 \ln p} \quad \text{if } p < 0.5 \\
  t_2 &= \sqrt{-2 \ln(1 - p)} \quad \text{if } p \geq 0.5
\end{align*}
\]  

(6.6)

Then, the inverse normal is:

\[
\begin{align*}
  F &= -\text{ratappx}(t_1) \quad \text{if } p < 0.5 \\
  F &= \text{ratappx}(t_2) \quad \text{if } p \geq 0.5
\end{align*}
\]  

(6.7)

Input:

\[ p \] - input probability

Output:

83
norminv - inverse of normal CDF

Routines called:

ratappx()

RATAPPX  ratappx() is a supplementary function to norminv() (see above).

Input:

t - input parameter

Output:

ratappx - approximation of normal inverse

REDIST  This subroutine redistributes the column-integrated water vapor (PWV) among the profile levels of the single-column model. It computes a proportional coefficient - the ratio of the deterministic PWV and the total PWV (computed with the stochastic mass balance model) - and uses this coefficient to add the variation to the humidity vertical profile (q). Multiplying the deterministic values of q from the \( t + \Delta t \) time step by this coefficient proportionally distributes the variation among the levels while also conserving the column water vapor.

Input:

nlev - number of vertical layers

pwvfin - final value of total PWV at \( t + \Delta t \)

qdet - original SCAM-computed (deterministic) vertical q profile

pwvdet - deterministic value of PWV at \( t + \Delta t \)

Output:

qvar - final distribution of PWV variation over vertical profile
6.1.5 Call Structure

Figure 6.2: Call structure for stochastic module subroutines. `sconvect()` is the main calling subroutine, which calls the four routines in the second column. In turn, `mbvar()` calls the three routines in the third column and `trigger()` calls the first three routines in the fourth column. `norminv()` calls `ratappx()`.

Immediately following the SCAM moist convection scheme in the subroutine `tphysbc()`, a call to

```
sconvect(nstep, state%pint(1, :pver+1), state%q(1,:pver,1), state%lchnk)
```

is inserted. The input variables are, respectively: the SCAM 20-minute time step; the vertical pressure profile at the level interfaces; the vertical humidity profile; and the chunk index needed to output data. Inside `sconvect()`, the corresponding local input variables are `m`, `p2`, `qdet2`, and `lchnk`.

For the first time step (`nstep = 1`), `sconvect()` only calls `vertint()` and vertically integrates the humidity, storing the resulting value in the global variable `pwv1` so that it will be available at the next time step. This value is also assigned to the last element of the 60-element array `pwvtot` (total
PWV) as well as \textbf{pwvdet} (deterministic PWV), which is otherwise padded with zeros. The same is done for \textbf{pstate} and the array \textbf{sigma}. These are sent to output with calls to \texttt{outfld()}\footnote{The output variables are 60 in length to accommodate the 30 vertical levels.}. Each array is split into two output variables; although the fact that there are 30 vertical levels limits the size of the output array accordingly, we take advantage of the fact that our result arrays are of length 60, or twice the 30 levels. Back in \texttt{tphysbc()}, the global variable \textbf{pstate} is assigned a 0 or 1 based on the current value of \texttt{prec.dp} which represents the convective precipitation. This is saved to be used as the initial precipitation state for the next time step.

For every subsequent time step, \texttt{sconvect()} calls \texttt{vertint()} and vertically integrates the current humidity profile. Then, \texttt{tsinterp()} is called with the globally stored \textbf{pwv1} and the just-integrated \textbf{pwv2} as the left- and right-hand ends of the interpolating interval. The interpolated array is assigned to the array \textbf{pwvdet}. \textbf{pwv1} is assigned to the first element of the array \textbf{pwvtot1} and \textbf{pstate} to the first element of the array \textbf{sigma}, then the subroutine \texttt{mbvar()} is called to compute values for each of the remaining elements.

The resulting arrays are sent to output with calls to \texttt{outfld()} as before, and then \texttt{redist()} is called, with the last element of \textbf{pwvtot} as input, to distribute the new total (stochastic) value of PWV among the levels of the vertical humidity profile. This profile is returned to SCAM to replace the deterministic profile, and the last element of \textbf{pwvtot} is assigned to the global variable \textbf{pwv1} to be the new left-hand value for the next time interval. Again, back in \texttt{tphysbc()}, the global variable \textbf{pstate} is assigned a 0 or 1 based on the current value of \texttt{prec.dp}.

Every time \texttt{mbvar()} is called, it takes as arguments \texttt{totpwv, sigma2, sigma1, pwvtot1, pwvdet1}, and \texttt{dt}. \texttt{totpwv} and \texttt{sigma2} are output variables that will hold the new total (stochastic) PWV and the new precipitation state for the current time step. \texttt{sigma1, pwvtot1, and pwvdet1} are the precipitation state, total PWV, and deterministic PWV, respectively, from the previous time step. \texttt{dt} is the time step size (20 seconds). Depending on the previous precipitation state, \texttt{tanhfn()} is called to compute the appropriate parameters, based on Equation 6.2. The previous deterministic PWV is used to compute the $E$ and $P$ parameters, which are based on average PWV, and the previous stochastic PWV is used to computed the large-scale $D$ forcing parameters. \texttt{prng()} produces the necessary random Gaussian(s), and the new total is computed and assigned to \texttt{totpwv}. Finally,
trigger() is called with the new total PWV and previous precipitation state as input, and the output is assigned to sigma2.

prng() uses the Box-Muller transform to generate a pseudo-random Gaussian number. Two uniform random variables in the interval (0, 1] are transformed into a Guassian using Equation 6.3 and this number is returned to the calling program.

tanhfn() computes the various parameters in Equation 6.2. The input arguments are the PWV, pwv, of which each parameter is a function, and fnid, an integer identifying which parameter is to be computed by the particular call. fnid is either 1 (r01), 2 (r10), 3 (E), 4 (P), 5 (D02), 7 (D2F), or 8 (D2P). If fnid is 3 or 5, the result is equal to f−∞ = f+∞ since E and D0 are constants independent of PWV. Otherwise, the parameter is computed as a function of PWV using Equation 6.5.

The stochastic trigger is simulated by the routine trigger(), the last subroutine called by mbvar(). trigger() takes as input the current PWV, the previous precipitation state, and the time step, and assigns the new precipitation state to sig2. After calling tanhfn() to compute r01 or r10 (depending on the previous precipitation state) and using this value to compute pt (the inverse normal of 1 − ertdt), prng() generates pcrit, a random Guassian. pt and pcrit are compared: if pt is greater than or equal to pcrit, the precipitation state switches, otherwise it remains the same.

To acquire the inverse normal, trigger() calls norminv(), which in turn calls ratappx(). These two functions use the algorithm described in Equations 6.6 and 6.7 to compute the inverse normal.

6.1.6 Running SCAM on Yellowstone

Once CESM has been built in a user account on Yellowstone, a run script is used to run SCAM (See Appendix B for sample). Several variables in the first section of the run script must be set by the user:

- CAM_ROOT - path for the directory of model files
- CSMDATA - path for the directory of input data
• **USR_SRC** - path for the directory containing user modifications

• **IOP_CASES_TO_RUN** - SCM pre-set cases based on field campaigns (here, **twp06**)

• **CASE** - name of case

• **WORKDIR** - path for output directory.

Finally, the default namelist must be appended with additional fields to specify the run conditions of SCAM:

• **history_budget = .true.** - diagnostics, budget calculations

• **scm_iop_srf_prop = .true.** - take observed surface properties as input

• **scm_relaxation = .true.** - relaxes the model to observed/reanalyzed data

• **single_column = .true.** - turns on single column model

• **mfilt = 2500** - filtering dynamics

• **nhtfreq = 1** - write frequency in time steps

• **dtime = $tarray** - length (in seconds) of coupling interval between dynamics and physics

• **ndens = 1** - specifies the precision of real data written to history file

• **use_64bit_nc = .false.** - use 32-bit netCDF format for cam history files

• **fincl1 = 'TMQdeta','TMQdetb','TMQstcha','TMQstchb','Sigmaa','Sigmab'** - list of fields to include on first history file.

Output files can be found in the directory specified by the user in **WORKDIR**. There will be three sub-directories in **WORKDIR**: **mods**, **CASENAME_bld_L30_trop_mam3**, and **CASENAME_twp06_L30_T1200**. In **CASENAME_twp06_L30_T1200**, there will be a run log if needed (**scam_output.txt**) and a file **camrun.cam.h0.2006-01-17-10800.nc** which contains all the output fields specified in the namelist (both default and user-appended).
6.1.7 Post-processing

Output is produced as a NetCDF file (.nc in this case), a machine-independent format for representing scientific data. These were read into MATLAB using the functions `ncdisp(filename)` and `ncread(filename)`.

There are six output fields added for the purpose of the stochastic scheme: `TMQdeta`, `TMQdetb`, `TMQstcha`, `TMQstchb`, `Sigmaa`, and `Sigmab`. `TMQdet-` fields hold the series of interpolated/deterministic PWV values, while `TMQstch-` and `Sigma-` fields hold the stochastic PWV and precipitation state series. As mentioned earlier, each variable is split into two fields for each 20-minute SCAM time step (60 20-second time steps), with `a` fields holding the first 30 values and `b` fields holding the second 30 values.

To produce a single series for each variable, MATLAB function `ncproc_stoch.m` (or `ncproc_sigma.m`) reads in `a` and `b` fields with `ncread(filename)`. The raw data are in 4-dimensional matrices (1x1x30x1927), so the first step is to convert to 2-dimensional matrices (30x1927). Then, `a` and `b` fields are spliced into one series by alternating 30-element columns from `a` and `b`. Finally, since the first SCAM time step was padded with zeros at the beginning, the first 59 elements are cut, leaving a 115561-element time series which spans the period 1/17/2006 3:00:00 - 2/12/2006 21:00:00.

Although the observed data ends at 2/13/2006 23:47:20, only the portion that coincides with model data is used for comparison. MATLAB function `plot_twp_stoch.m` plots observed and (stochastic) model PWV time series, and `plot_sigma.m` produces a stem plot of the precipitation state time series.
Figure 6.3: Illustration of data splicing process from two fields (A and B) to one time series. For each of 1927 20-minute SCAM time steps (60 stochastic time steps), A and B alternate contributing 30 stochastic time steps each.

### 6.2 Simulation Results

SDE model parameters for the stochastic module in the single-column model can be seen in Table 6.1. As we found with the point model, we could not achieve the amount of variance found in the observed data using the originally defined parameters. In this case, there were two aspects to consider: not only is our observed data from a different location (further from the equator) than the original model based on characteristics found at Nauru, but also the stochastic module is constrained by its coupling to the deterministic values computed by the SCM. Therefore, to pick up the additional variance, we widened the width of the \( r_{ij} \) transitions and increased the maximum value of \( r_{10} \) to 22 h\(^{-1}\), along with including the \( D_0 dW_t \) multiplicative factor of 1.5. To relax the
model to observed/reanalyzed data as much as possible, we set the relaxation time $r_{\tau}$ to 40 minutes (2 20-minute SCAM time steps).

Table 6.1: Single-column model stochastic module parameters.

<table>
<thead>
<tr>
<th>Parameter [units]</th>
<th>$f_{-\infty}$</th>
<th>$f_{\infty}$</th>
<th>$q_{mid}$ (cm)</th>
<th>$q_{width}$ (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{01}$ (h$^{-1}$)</td>
<td>0</td>
<td>1.0</td>
<td>6.1</td>
<td>30.0</td>
</tr>
<tr>
<td>$r_{10}$ (h$^{-1}$)</td>
<td>22.0</td>
<td>0</td>
<td>6.3</td>
<td>30.0</td>
</tr>
<tr>
<td>$E$ (mm$^2$ h$^{-1}$)</td>
<td>0.2</td>
<td>0.2</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>$P$ (mm$^2$ h$^{-1}$)</td>
<td>2.0</td>
<td>10.0</td>
<td>6.45</td>
<td>0.1</td>
</tr>
<tr>
<td>$D_0^g$ (mm$^2$ h$^{-1}$)</td>
<td>1.5</td>
<td>1.5</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>$D_1^g$ (mm$^2$ h$^{-1}$)</td>
<td>16.0</td>
<td>64.04</td>
<td>6.45</td>
<td>0.1</td>
</tr>
<tr>
<td>$D_2^g$ (mm$^2$ h$^{-1}$)</td>
<td>16.0</td>
<td>64.00</td>
<td>6.45</td>
<td>0.1</td>
</tr>
<tr>
<td>$D_2^f$ (mm$^2$ h$^{-1}$)</td>
<td>0.0</td>
<td>0.04</td>
<td>6.45</td>
<td>0.1</td>
</tr>
<tr>
<td>$D_0dW_t$ factor</td>
<td></td>
<td></td>
<td></td>
<td>1.5</td>
</tr>
<tr>
<td>time step (s)</td>
<td></td>
<td></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>$r_{\tau}$ (min)</td>
<td></td>
<td></td>
<td></td>
<td>40</td>
</tr>
</tbody>
</table>

Figure 6.4 shows the resulting observed and model time series from a single run of the stochastic SCM. Although we were not able to achieve extremely consistent results, approximately half of our simulations were very good, and more than 75% were able to follow the general temporal trend of the observed data. Statistics of the selected run are listed in Table 6.2, and display remarkable similarity to those of the corresponding observed data. While the model mean of 5.081 cm is slightly smaller than that of the observed (5.640 cm), the standard deviations are very similar, demonstrating the extent to which we were able to achieve a volatility close to the observed data. The similar fractal dimensions further support this conclusion. Skewness for both observed and model data is left-tailed, meaning more negative deviations for both; kurtosis is negative but small for both as well, meaning slightly less prone to extremes than a normal distribution would predict.

Visually, we see that model PWV is able to follow the general trends of the observed data. Although it starts a bit lower than the observed ($\sim$6 cm vs. $\sim$6.5 cm) and decreases around January 27 when the observed PWV is increasing (and similarly around February 7), model and observed PWV are similar temporally, and the volatility appears reasonably similar as well.

The correlation coefficient of 0.5510 is smaller than that of our best point model 25-day sections, but still significant considering the single-simulation nature of the experiment. Finally, we look at the ratio of precipitating time steps ($\sigma = 1$) vs. non-precipitating time steps ($\sigma = 0$). If we
consider the GHCN daily data at Darwin, shown in Figure 1.1, there were 2-4 significant rain days, translating to 8-16\% rain time. Our model data (\( \sigma \) time series shown in Figure 6.5) has a \( \sigma \) 1:0 ratio of approximately 0.1, or 10\%, which falls within this range, so the model precipitation prediction can be considered reasonable.

Table 6.2: First four statistical moments of the observed (1/17/2006 - 2/12/2006) and single-column model data, along with fractal dimension, observed-model correlation and \( \sigma \) 1:0 ratio.

<table>
<thead>
<tr>
<th></th>
<th>Observed (20 sec)</th>
<th>SCM (20 sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean (cm)</td>
<td>5.640</td>
<td>5.081</td>
</tr>
<tr>
<td>SD (cm)</td>
<td>0.917</td>
<td>0.901</td>
</tr>
<tr>
<td>Skewness</td>
<td>-0.6807</td>
<td>-0.4855</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>-0.1034</td>
<td>-0.4532</td>
</tr>
<tr>
<td>Fractal Dim.</td>
<td>1.9</td>
<td>1.9</td>
</tr>
<tr>
<td>Correlation</td>
<td></td>
<td>0.5510</td>
</tr>
<tr>
<td>1:0 Ratio</td>
<td></td>
<td>0.1</td>
</tr>
</tbody>
</table>

Figure 6.4: PWV observed (black curve) at TWP-ICE site 17 January 2006 - 12 February 2006 with time resolution of 20 seconds. The blue curve is the SCM model simulated data.
Figure 6.5: Single-column model precipitation time series, where $\sigma = 0$ means non-precipitating and $\sigma = 1$ means precipitating.
7 Conclusions

In this study we have sought to introduce a stochastic model to simulate the highly volatile nature of precipitable water in convective processes and apply this algorithm to a single column model to further improve the parameterization and to form the basis for using this type of parameterization in a general circulation model.

After quantitatively analyzing a long-term high resolution observed time series of precipitable water vapor, we concluded that the fractal behavior (Hausdorff dimension of 1.9) of the observed data calls for stochastic modeling, specifically in the form of a stochastic differential equation model for the mass balance. As in a typical diffusion process which consists of a deterministic drift term and a random diffusion term, this model combined a deterministic process with both stochastic closures and a stochastic trigger for the transition to and from convective precipitation.

With 25 days of observed data and a 1000-day simulation, we computed a time series of moving-window correlation coefficients to determine the best-fitting 25-day section of the simulation (correlation $r = 0.6513$, see Figure 4.6), along with comparing the statistics of the 40 non-overlapping 25-day sections with the those of the observed data.

The statistics of the simulated and the observed datasets are similar, as measured by various indices, including the mean, standard deviation, skewness, kurtosis, extremes, fractal dimensions and probability distributions. This similarity suggests that the prescribed physical parameters of evaporation, precipitation, convergence and divergence can control the overall water mass balance and hence the mean PWV and variation, although specific precipitation events cannot be predicted, particularly extreme precipitations and droughts, due to both the randomness of realistic weather events and the inherent randomness of a stochastic model. The fractal dimension of about 1.9 from our best simulation data is similar to that of the observed data, demonstrating physically reasonable anti-persistence and justifying the use of the stochastic model.
We then applied a similar algorithm to an existing single-column model by adding a stochastic module that is called immediately following the deep convection scheme in NCAR’s Single-column Community Atmosphere Model (SCAM). This module incorporated the vertical structure of atmospheric water vapor and coupled the stochastic SDE algorithm to the deterministic model computed to SCAM using 20-second time steps interpolated between the 20-minute SCAM time steps.

A single simulation (Figure 6.4) resulted in a correlation coefficient of 0.5510, smaller than the best-fitting section of the point model, but still an important result considering that each SCM simulation is a single 27-day simulation. Again, the statistics and fractal dimension are similar, and the ratio of precipitating to non-precipitating time steps is within the reasonable range suggested by daily observed precipitation data.

While we were able to successfully simulate PWV with this scheme much of the time, it was not fully consistent. We suspect that there is an unresolved issue with the way our stochastic module interacts with the native relaxation scheme in SCAM, as the simulations did not always sufficiently relax to the prescribed data, leading to dissimilar temporal trends.

Model relaxation, considered a type of data assimilation, incorporates observations into a model state. SCAM uses Newtonian Relaxation, or nudging, which consists of adding a term to the model equations which is proportional to the difference between the model-computed variable and an observed or prescribed value of that variable at a given time (Kooperman et al., 2012). This can be useful for counteracting the effects of unconstrained variables, although Lohmann et al. (1999) acknowledge that there is some difficulty in finding a nudging coefficient with the strength to force the model sufficiently close to the prescribed values but weak enough to allow the model to develop its independent physical state.

Since our stochastic module is minimally constrained, the relaxation scheme is crucial to its capability to simulate the lower-resolution temporal trends. Given a relaxation time $\tau_r$, we would expect that, even with drift during this time period, at every $\tau_r$ in the model, the simulated value would be forced back toward the observed/prescribed value. Upon examination of the model and observed time series in Figure 7.1 (one of our poorest simulations), we see that this is very much not the case for $\tau_r = 40$ min. While the initial conditions appear to be correct, by January 19 the model
PWV has begun a drastic downward trend that continues through January 24. It is not until about February 2 that model PWV returns to somewhat following the observed trends.

It is so far unclear what mechanism of the relaxation scheme is interacting with the stochastic module incorrectly, but rectifying this issue would likely lead to extensive improvement in simulation capabilities.

![Figure 7.1: PWV observed (black curve) at TWP-ICE site 17 January 2006 - 12 February 2006 with time resolution of 20 seconds. The blue curve is the SCM model simulated data. Here we demonstrate the ineffectiveness of the relaxation scheme.](image)

Despite falling slightly short of our goals for this study, this work addresses a notable gap in the current approaches to stochastic modeling: parameterizations based on observations, rather than relying on ad-hoc relationships or cloud-resolving model simulations (Jakob et al., 2011), as well as based on solid mathematics and statistics (Franzke et al., 2015).

Finally, we consider where the results of this study can be useful in future model development. Ball and Plant (2008) highlight the usefulness of testing stochastic schemes in an SCM as a bridge to implementation in a full GCM. It is for this reason we chose to work with SCAM, the SCM
developed as part of NCAR’s general circulation model, CESM. A next step would be to configure the module to be used in any column of the model, which would likely entail a different, more global approach to defining some of the model parameters. Additionally, some research into the use of data assimilation in the GCM would conceivably also be necessary.
References


99
http://ocw.mit.edu/courses/earth-atmospheric-and-planetary-sciences/12-340-global-warming-


23. Franzke, C. L. E., T. J. O’Kane, J. Berner, P. D. Williams, and V. Lucarini, 2015: Stochastic
climate theory and modeling. *WIREs Climate Change*, 6, 63-78.

R. Jones, M. M. Khaiyer, G. Liu, P. Minnis, H. Morrison, L. Nguyen, S. Park, J. C. Petch,
A comparison of TWP-ICE observational data with cloud-resolving model results. *Journal of
Geophysical Research*, 117, D05204.


744 pp.

radar-rainfall products: Application to the TEFLUN-B field campaign. *Journal of Applied
Meteorology*, 41(5), 558572.


Appendices

Appendix A - Point Model Code

pwvmodel.m

function [q, sigma] = pwvmodel(lenrun, intsize)

% This function simulates column water vector with stochastic trigger for
% precipitation and stochastic closures for precipitation and other
% external forcings.
%
% q(t+1) = q(t) + $E(q(t)) + D_0(q(t))dW$ if non-precipitating
% q(t+1) = q(t) - $[P(q(t)) + D_1(q(t))dW$ if precipitating
%
% The stochastic trigger is modeled with a Markov jump process.
%
% Input: lenrun - run length in days
% intsize - model time step in hours
%
% Output: q - time series of column water vapor
% sigma - time series of precipitation state
%
% Calls: fpwv.m
% prng.m or levyrn.m
% pwvtrigger.m

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

days = lenrun; % Run length in days

dt = intsize; % Time step in hours

% Time vector in days (for plotting)
q = zeros(1, length(t));
sigma = zeros(1, length(t));

q(1) = 30 + (80-30) * rand; % Random initialization of q
rsig = rand; % Random precipitation state (sigma)
if rsig <= 0.5
    sigma(1) = 0;
else
    sigma(1) = 1;
end
% For each time step: 2 --> length(q)
for k = 2:length(t)

% Update q(t) to q(t+dt), keeping sigma constant (Euler-Maruyama):
% If non-precipitating:
if sigma(k-1) == 0,
    % Pass q, ids for parameters to fpwv, return values
    E = fpwv(q(k-1),3);
    D0 = sqrt(fpwv(q(k-1),5));
    gr = prng(); % Gaussian random number using Box-Muller
    dW = 1.4*sqrt(dt)*gr;
    q(k) = q(k-1) + E*dt + D0*dW;
end

% If precipitating:
elseif sigma(k-1) == 1,
    % Pass q, ids for parameters to fpwv, return values
    P = fpwv(q(k-1),4);
    D1 = sqrt(fpwv(q(k-1),6));
    gr = prng(); % Gaussian random number using Box-Muller
    dW = 1*sqrt(dt)*gr;
    q(k) = q(k-1) - (P*dt + D1*dW);
end

% Update sigma to sigma(t+dt) using q(t+dt):
sigma(k) = pwvtrigger(q(k),sigma(k-1),dt);
end

% Convert to cm:
q = q/10;
function f = fpwv(q,id)
%------------------------------------------------------------------------
% This function evaluates a parameter (identified by id) as a function of
% q.
%
%
% Input:  q - PWV
%        id - indicates which parameter to evaluate
%
% Output: f - requested parameter
%
% ID:
% 1: r01
% 2: r10
% 3: E
% 4: P
% 5: D0sq
% 6: D1sq
% 7: DFsq
% 8: DPsq
%------------------------------------------------------------------------

% Empirical values for parameter functions:
f_inf = [0 4 0.2 2 1.5 16 16 0];
finf = [1 0 0.2 10 1.5 64.04 64.0 0.04];
qmid = [61 63 0 64.5 0 64.5 64.5 64.5];
qwidth = [2 2 0 1 0 1 1 1];

% Constant functions:
if id == 3 || id == 5
    f = f_inf(id) + 0.5*(finf(id)-f_inf(id));
% Tanh functions:
else
    f = f_inf(id) + 0.5*(finf(id)-f_inf(id))*(1+tanh((q-qmid(id))/qwidth(id)));
end
end
function gr = prng()
%------------------------------------------------------------------------
% This function generates pseudorandom Gaussian noise (Box-Muller)
%
% Output: gr - Gaussian random number
%------------------------------------------------------------------------
%
% Two values from uniform random distr (0,1)
u1 = rand;
u2 = rand;

% Convert to Gaussian random number using Box-Muller
gr = sqrt(-2*log(u1))*cos(2*pi*u2);
end
function lr = levyrn()

% This function generates pseudorandom numbers from the standard Levy distribution (standard ~ mu = 0, sigma = 1). Using Weron’s (1996) algorithm which is based on the formulation by Chambers, Mallows, and Stuck (1976), two random numbers are generated: one from Unif(-pi/2,pi/2) and one from the exponential distribution with mean one. These are then transformed into one Levy (alpha = 0.5, beta = 1) alpha-stable variable.

% Segments are computed separately then multiplied to produce the final r.v.
% X = S*seg1*seg2, where seg1 = num1/den1, seg2 = num2/den2
%
% Output: lr - Levy random number
%------------------------------------------------------------------------

% Distribution parameters:
alpha = 1.9;
beta = 0;

% Generate uniform random variable by transforming (0,1) to (-pi/2,pi/2):
u = rand;
U = pi*(u-0.5);

% Generate exponential random variable:
E = exprnd(1);

% Compute S:
S = (1 + (beta*tan(pi*alpha/2))^2)^(1/(2*alpha));

% Compute B:
B = atan(beta*tan(pi*alpha/2))/alpha;

% Compute seg1:
um1 = sin(alpha*(U+B));
den1 = (cos(U))^(1/alpha);
seg1 = num1/den1;

% Compute seg2:
um2 = cos(U-alpha*(U+B));
den2 = E;
seg2 = (num2/den2)^((1-alpha)/alpha);

% Compute lr, the Levy r.v.:
lrs = S*seg1*seg2;
% Transform for mu and sigma:
mu = 0;
sigma = 1;
lr = mu + sigma*lrs;
function sigma = pwvtrigger(q,sigma,dt)

% This function simulates the stochastic trigger for precipitation using
% a markov jump process. A transition probability is computed as a
% function of q, then compared to a randomly generated critical probability.
% If the transition probability is greater than the critical probability,
% the precipitation state switches. Otherwise it remains the same.

% Input: q - current value of cwv
% sigma - previous value of sigma

% Output: sigma - new value of sigma (for current time step)

% Calls: fpv.m
% prng.m

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

% If non-precipitating:
if sigma == 0
    r01 = fpv(q,1); % Pass q, id for r01 to fpv, return value of r01
    prob01 = 1 - exp(-r01*dt); % Calc probability of switching
    z = norminv(prob01,0,1); % Calc cutoff for switching based on std normal
    gr = prng(); % Gaussian random number using Box-Muller
    if gr <= z % Compare gr to z to determine switch or not
        sigma = 1;
    end

% If precipitating:
elseif sigma == 1
    r10 = fpv(q,2); % Pass q, id for r10 to fpv, return value of r10
    prob10 = 1 - exp(-r10*dt); % Calc probability of switching
    z = norminv(prob10,0,1); % Calc cutoff for switching based on std normal
    gr = prng(); % Gaussian random number using Box-Muller
    if gr <= z % Compare gr to z to determine switch or not
        sigma = 0;
    end
end
function [corrcoeff, maxcorr, maxcorr_t] = pwvcorr(q)

% This script computes the correlation between the observation and model
% cwv for consecutive time steps. The data is compared only at the points
% where valid observation data is available. The correlation coefficient
% is plotted vs time step of beginning of interval (25 days), and the
% maximum value is output, as well as an array of the coefficients.

% Correlation coefficient is calculated as follows:
% For two variables x and y, and number of data points, N:
% r = \frac{N*\text{sum}(x*y) - \text{sum}(x)*\text{sum}(y)}{\sqrt{[N*\text{sum}(x^2)-(\text{sum}(x))^2][N*\text{sum}(y^2)-(\text{sum}(y))^2]}}
% r will be in the interval [-1,1]. The closer to +/- 1, the stronger the
% relationship. A negative value indicates a negative correlation.

% Input: obs - observed data (20 sec, in cm, only valid data)
% q - model data (20 sec, in cm)
% ind - array of indices for which obs data is available
% len - length of full observation data

% Output: corrcoeff - array of correlations coefficients
% maxcorr - max correlation coefficient
% maxcorr_t - time step corresponding to max
% Calls: corrcfcomp.m

% Load data from mat-file:
load('pwv20sec.mat','obs','ind','len');

obss = obs;
indd = ind;
lenn = len;

% Number of time steps at which to compute corr coeff:
numcomp = (length(q)-len) + 1;

% Initialize array for corr coeff:
r = zeros(numcomp,1);

parfor k = 1:numcomp
    endts = (k+lenn)-1;
    qint = q(k:endts);  % interval of q used to compute corr coeff

qcomp = qint(indd);  % only q data corresponding to obs data
r(k) = corrcf_comp(obss, qcomp);  % compute r
%
% Find max value and corresponding time step:
[maxcorr, maxcorr_t] = max(r);
%
% Assign r to output array:
corrcoeff = r;
%
% Plot
xvec = 1: numcomp;
plot(xvec, r, 'b.'),
xlabel('time step of model data interval start'),
ylabel('correlation coefficient'),
title('Correlations coefficients between observations and model data'),
axis([0 numcomp +1 -1 1])
function r = corrcfcomp(obs, q)

% This function computes the correlation coefficient between two time
% series, obs and q.

% Correlation coefficient is calculated as follows:
% For two variables x and y, and number of data points, N:
% 
% \[ r = \frac{N \sum(xy) - \sum(x)\sum(y)}{\sqrt{\left[N \sum(x^2) - (\sum(x))^2\right] \left[N \sum(y^2) - (\sum(y))^2\right]}} \]

% Input: obs - observed data (20 sec, in cm, only valid data)
% q - model data (20 sec, in cm), at same indices as obs
% Output: r - correlation coefficient

% Number of data points:
N = length(obs);

% Computing each term in equation above:
sumxy = sum(obs.*q);
sumx = sum(obs);
sumy = sum(q);
sumx2 = sum(obs.^2);
sumy2 = sum(q.^2);

% Putting terms together:
um = N*sumxy - sumx*sumy;
denom1 = N*sumx2 - sumx^2;
denom2 = N*sumy2 - sumy^2;
denom = sqrt(denom1*denom2);

% Finally, computing r:
r = num/denom;
Appendix B - Single-Column Model Code

run–scam.csh

# Script to make a community-available SCAM run library for ARM standard IOP cases
# Lin Su and John Truesdale (please contact Lin Su (linsu@ucar.edu) for SCAM diagnostic package)).

set VERSION=12072012
echo "***** _Version_ - _VERSION_ *****"

#########################################################################
### Set vars needed for this script code dir, case, data dir, mods, wrkdir
#########################################################################
set CAM_ROOT = /glade/p/work/kimleung/cesm1_2_0
set CSMDATA = /glade/p/cesmdata/cseg/inputdata

#set USR_SRC = /glade/u/home/xsong/cesm1_2_1/models/atm/cam/src/user_code/memorys
# Test mods dir.
set USR_SRC = /glade/p/work/kimleung/cesm1_2_0/scripts/SourceMods/testmod7

###set IOP_CASES_TO_RUN = 'arm95 arm97 gateIII mspace sparticus togaII twp06'
set IOP_CASES_TO_RUN = 'twp06'

###set CAM_TIMESTEPS_TO_RUN = '60 300 600 900 1200'
set CAM_TIMESTEPS_TO_RUN = '1200'

### set CAM_LEVELS # options are 26,27,30,60,90,120,150,180,210,240
# you must have initial condition files for number of levels
set CAM_LEVELS_TO_RUN = 30

set CASE = SCAMstoch2

set WRKDIR = /glade/scratch/kimleung/SCAM && if (! -e $WRKDIR) mkdir -p $WRKDIR

#########################################################################
### Select compiler+libraries env vars and set paths depending on machine.

116
#set FC_DIR=/usr/local/pgi/linux86-64
set USER_FC = /glade/apps/opt/modulefiles/ys/cmpwrappers/ifort
set NCHOME = /glade/apps/opt/netcdf/4.3.0/intel/12.1.5
set DBUG = "-debug"

### Shouldn’t have to modify below here
### Set some case specific parameters here
### Here the boundary layer cases use prescribed aerosols while the deep convection
### and mixed phase cases use prognostic aerosols. This is because the boundary layer
### cases are so short that the aerosols do not have time to spin up.
if (iopname == 'arm95' || iopname == 'arm97' || iopname == 'mpace' || iopname == 'twp06' || iopname == 'sparticus' || iopname == 'togaII' || iopname == 'gateIII' || iopname == 'IOPCASE') then
    set aero_mode = 'trop_mam3'
else
    set aero_mode = 'none'
endif

set SCAM_MODS = \$WRKDIR/\$CASE/mods && if (! -e \$SCAM_MODS) mkdir -p \$SCAM_MODS
rm -rf \$SCAM_MODS/*
/bin/cp -f \$USR_SRC/* \$SCAM_MODS

set BLDDIR = \$WRKDIR/\$CASE/\{\$CASE\}_bld_L\{levarr\}_\$\{aero_mode\} && if (! -e \$BLDDIR) mkdir -p \$BLDDIR
cd \$BLDDIR
set IOPDESC = 'grep IOP: \$CAM_ROOT/models/atm/cam/bld/namelist_files/use_cases/scam_\$\{iopname\}.xml'

## Configure for building
$CAM_ROOT/models/atm/cam/bld/configure -s -chem \$aero_mode -nlev \$levarr -dyn eul -res 64x128 -nospmd -nosmp -cppdefs -DDISABLE_TIMERS -scam -usr_src \$SCAM_MODS -fc \$USER_FC -DDEBUG -ldflags "-llapack -lblas -Mnobounds" -lapack_libdir $lapackdir # -cice_bsize1 = 1 -cice_bsizey 1 # -microphys mg1.5

## compile

#--------------------------
## Build the namelist with extra fields needed for scam diagnostics
cat <<EOF > tmp_namelistfile
&camexp
history_budget = .true.,
scm_iop_srf_prop = .true.,
scm_relaxation = .true.,
single_column = .true.,
mfilt = 2500,
nhtfrq = 1,
dtime = $tarray,
ndens = 1,
use_64bit_nc = .false.,
fincl1 = 'CLDST','ZMDLF','TMQdeta','TMQdetb','TMQstcha','TMQstchb','Sigmaa','Sigmab','QRELAX','TAURELAX'
/
EOF

$CAM_ROOT/models/atm/cam/bld/build-namelist -s -runtype startup -infile tmp_namelistfile -use_case scam_${iopname} -csmdata $CSMDATA \\ || echo "build-namelist failed" & exit 1

set RUNDIR = $WRKDIR/$CASE/$EXPNAME & if (! -e $RUNDIR)
    mkdir -p $RUNDIR
    cd $RUNDIR

### RUN

cp -f $BLDDIR/docn.stream.txt $RUNDIR
cp -f $BLDDIR/*_in $RUNDIR
cp -f $BLDDIR/cam $RUNDIR

echo ""
echo "_Running in $RUNDIR"
echo ""
./cam >& scam_output.txt

echo 
end #foreach iopname
end #foreach tarray
end #foreach levarr

exit 0
MODULE stochvar

USE shr_kind_mod, only: r8 => shr_kind_r8

implicit none
SAVE

! DECLARE VARIABLES:
!-------------------
integer :: pstate
real(r8) :: pwv1

END MODULE stochvar
This function module takes the mixing ratio arrays from each
end of a 20-minute time step from SCAM, along with the associated
pressure profiles, and computes new profiles via a stochastic
SDE model for the mass balance.

Input for main subroutine sconvect():

- m: number of vertical levels
- qdet2: array of mixing ratio from current time step (z x 1) (kg/kg)
- p2: array of pressure for current time step (z x 1) (Pa or N/m²)

Output:
- qdet2: array of total mixing ratio including accumulated
  variation from stochastic model (z x 1) (kg/kg)

Subprograms (full descriptions at the beginning of each):
- vertint: vertically integrates water vapor
- tsinterp: interpolates pwv at 20-second resolution
- mbvar: mass balance model implementation
- tanhfn: computes parameters for MB model
- trigger: stochastic trigger for precipitation
- prng: pseudorandom number generator (Gaussian)
- init_random_seed: initializes RNG
- ratappx: (part of norminv)
- redist: redistributes pwv among vertical levels

This subroutine is the main routine to simulate stochastic
parameterization of convection. The function module takes the
specific humidity arrays from each end of a 20-minute time step
from SCAM, along with the associated pressure profiles, and adds
! variation computed via a stochastic model. The specific humidity 
!(q) arrays are vertically integrated to produce a value of 
! precipitable water vapor (PWV) at each end of the time step, between 
! which we interpolate at 20-second intervals (n = 0,1,2,...,60).
! Using a model derived from the mass balance, we use each value of 
! pwv to compute the variation for the next step, which is then added 
! to the interpolated value of pwv and used for the next 20-second 
! time step calculation. At the end, the total value of PWV for 
! n = 60 is redistributed among the levels according to the original 
! proportional distribution of q at the right-hand time step and this 
! array is returned to the SCM convection scheme as a replacement for 
! the deterministic value of q.
!
! Notes: o delt is the 20-minute time step used by SCAM. dt is the 
! 20-second resolution used by our mass balance model.
! o SCAM returns q in units of kg/kg. We compute pwv in 
! kg/m^2 but then convert to mm height of water to 
! be compatible with the mass balance model parameters.
!
! Input: o m - current time step 
! o p2 - pressure arrays at current time step 
! o qdet2 - mixing ratio array at current time step 
! o lchnk - chunk index (for output)
!
! Output: o qdet2 - updated array of mixing ratio (incl var)
!
! Routines called: o vertint() - vertically integrates water vapor 
! o tsinterp() - interpolates pwv between t and t+delt 
! o mbvar() - computes pwv variation and trigger 
! o redist() - redistributes pwv among levels
!
SUBROUTINE sconvect(m,p2,qdet2,lchnk)
USE stochvar
USE cam_history, only: outfld, phys_decomp
USE ppgrid, only: pcols, pver
USE shr_kind_mod, only: r8 => shr_kind_r8
implicit none

integer, intent(in) :: m, lchnk
real(r8), intent(in) :: p2(pver+1)
real(r8), intent(inout) :: qdet2(pver)
real(r8) :: qvar(pver)
real(r8) :: precip
real(r8) :: dt
real(r8) :: pwv2
integer, parameter :: intsteps = 61 ! for 20-second resolution
real(r8), dimension(intsteps) :: pwvdet, pvvtot, sigmad
integer, dimension(intsteps) :: sigma
integer :: k

! Define time step as 20 sec (in hours):
dt = 1d0/180d0

! Vertically integrate water vapor:
IF (m == 0) THEN  ! If right after first time step
   CALL vertint(pver,p2,qdet2,pwv1);
   DO k = 1,intsteps-1
      pwvdet(k) = 0d0
      pwvtot(k) = 0d0
      sigma(k) = 0
   END DO
   pwvdet(intsteps) = pwv1
   pwvtot(intsteps) = pwv1
   sigma(intsteps) = pstate

   ! Send to history file:
   CALL outfld('TMQdeta␣', pwvdet(2:31), pcols, lchnk)
   CALL outfld('TMQdetb␣', pwvdet(32:intsteps), pcols, lchnk)
   CALL outfld('TMQstcha', pwvtot(2:31), pcols, lchnk)
   CALL outfld('TMQstchb', pwvtot(32:intsteps), pcols, lchnk)
   CALL outfld('Sigmaa␣␣', sigmad(2:31), pcols, lchnk)
   CALL outfld('Sigmab␣␣', sigmad(32:intsteps), pcols, lchnk)
   RETURN  ! Return to tphysbc without further computation

ELSE  ! If any subsequent time step
   CALL vertint(pver,p2,qdet2,pwv2);  ! Vertically integrate profile
   sigma(1) = pstate  ! Use precip state from previous time step
END IF

! Interpolate between SCM 20-minute time steps:
CALL tsinterp(intsteps,pwv1,pwv2,pwvdet)

! Accumulate variation at each 20-second time step:
pwvtot(1) = pwv1
DO k = 2,intsteps
   CALL mbvar(pwvtot(k),sigma(k),sigma(k-1),pwvtot(k-1),pwvdet(k-1),dt)
END DO

! Convert sigma to double for output:
DO k = 1,intsteps
  sigmad(k) = DBLE(sigma(k))
END DO

! Send to history file:
CALL outfld('TMQdeta', pwvdet(2:31), pcols, lchnk)
CALL outfld('TMQdetb', pwvdet(32:intsteps), pcols, lchnk)
CALL outfld('TMQstcha', pwvtot(2:31), pcols, lchnk)
CALL outfld('TMQstchb', pwvtot(32:intsteps), pcols, lchnk)
CALL outfld('Sigmaa', sigmad(2:31), pcols, lchnk)
CALL outfld('Sigmab', sigmad(32:intsteps), pcols, lchnk)

! Redistribute total pwv (det + stoch) among levels:
CALL redist(pver,pwvtot(intsteps),qdet2,pwv2,qvar)

! Add q tendency:
DO k = 1,pver
  qdet2(k) = qdet2(k) + qvar(k)
END DO

! Assign pwv2 to be new pwv1:
pwv1 = pwvtot(intsteps)

END SUBROUTINE sconvect

!----------------------------------------------------------------------
!---------------!
! VERTINT() !
!---------------!
!
! This subroutine vertically integrates the specific humidity in a
column, to result in the precipitable water vapor (pwv) for that
column. It numerically integrates using the midpoint rule:
!
\[ pwv = \text{sum} [\text{from } 2 \text{ to } n] (qavg(n,n-1) \times \Delta p / g) \]

! where qavg is the average q between n and n-1 levels, \Delta p is the
difference in pressure between two levels, and g is the acceleration
of gravity. We then convert from the resulting units of kg/m^2 to
mm height of water.
!
! Input: o n - number of vertical levels
! o p - pressure profile
! o q - specific humidity profile
! Output:  
! pwvreturn - precip water vapor for the column

SUBROUTINE vertint(n,p,q,pwvreturn)
USE shr_kind_mod, only: r8 => shr_kind_r8

implicit none

integer :: n, k
real(r8) :: pwvreturn
real(r8) :: p(n+1), q(n)
real(r8) :: gaccel, pwv, qavg, dp, rhowater

! Define gravitational acceleration:
gaccel = 9.80665

! Define density of water:
rhowater = 999.97d0

! Initialize pwv to 0:
pwv = 0d0

! Sum pwv for all layers:
DO k = 1,n
  qavg = q(k)
  dp = (p(k+1) - p(k))/gaccel
  pwv = pwv + qavg*dp
END DO

! Converting from kg/m^2 to mm
pwvreturn = pwv*(1000d0/rhowater)

END SUBROUTINE vertint

!----------------------------------------------------------------------
!----------------!
! TSINTERP() !
!----------------!
!
! This subroutine interpolates between two values of pwv to create a
! uniform grid with (intsteps) points.
!
! Input:  
! o intsteps - total # of steps to interpolate over
SUBROUTINE tsinterp(intsteps,pwvl,pwvr,pwvint)
USE shr_kind_mod, only: r8 => shr_kind_r8
implicit none
integer :: intsteps, k
real(r8) :: pwvl, pwvr
real(r8) :: pwvint(intsteps)
real(r8) :: intsize

! Compute interval size:
intsize = (pwvr-pwvl)/DBLE(intsteps-1)

! Set endpoints:
pwvint(1) = pwvl
pwvint(intsteps) = pwvr

! Interpolate at inner gridpoints:
DO k = 2,intsteps-1
   pwvint(k) = pwvint(k-1) + intsize
END DO
END SUBROUTINE tsinterp

!-------------------------------------------------------------------------------
!-------------!
! MBVAR() !
!-------------!
!
! This subroutine simulates the change in precipitable water vapor with
! a stochastic trigger for precipitation and stochastic closures for
! precipitation and other large-scale external forcings. Based on
! Stechmann and Neelin's 2011 paper, this function utilizes the mass
! balance to compute the change in pwv at each time step, as a function
! of pwv from the previous time step. We assume that the deterministic
! values provided by the SCM and our interpolation are equivalent to
! the mean state/average value, and wish to compute a new pwv that is
! based on a stochastic algorithm.
!
! Assuming qbar is the average/deterministic pwv, and q is the total
computed via the stochastic model:

\[ f_1(q_{bar}) = E(q_{bar})dt \quad \text{if non-precipitating} \]
\[ f_1(q_{bar}) = -P(q_{bar})dt \quad \text{if precipitating} \]

and

\[ f_2(q) = D_0(q)dW \]
\[ f_2(q) = -D_P(q)dW - D_F(q)dW \]

where \( P \) and \( E \) are the average precipitation and evaporation rates, based on the mean state, or deterministic pwv; \( D_0, D_P, \) and \( D_F \) are the stochastic noise coefficients for forcing and precipitation; and \( dW \) is Gaussian noise. We then compute \( q \) for the next time step:

\[ q(t+1) = q(t) + f_1(q_{bar},t) + f_2(q,t) \]

We then determine the precipitation state of the next step with the stochastic trigger function \( \text{trigger()} \) which takes as input the pwv we just computed, and the value of sigma (precip state) from the last step.

Input:
- \( \sigma_1 \) - previous precipitation state
- \( \text{pwvtot}_1 \) - total pwv from last step
- \( \text{pwvdet}_1 \) - deterministic pwv from last step
- \( dt \) - time step

Output:
- \( \sigma_2 \) - new precipitation state
- \( \text{totpwv} \) - total pwv for current step

Routines called:
- \( \text{prng()} \) - pseudo-random number generator
- \( \text{tanhfn()} \) - function to compute MB parameters
- \( \text{trigger()} \) - stochastic trigger function

SUBROUTINE mbvar(totpwv,sigma2,sigma1,pwvtot1,pwvdet1,dt)

USE shr_kind_mod, only: r8 => shr_kind_r8
implicit none

integer :: sigma1, sigma2
real(r8) :: pwvtot1, pwvdet1, dt
real(r8) :: E, P, D0, gr, dW, pwvvar, DF, DP, dWF, dWP, totpwv

! Update pwv, keeping sigma constant (Euler-Maruyama):
! If non-precipitating:
IF (sigma1 == 0) THEN
  E = tanhfn(pwvdet1,3)
  D0 = DSQRT(tanhfn(pwvtot1,5)) ! Pass pwv, parameter id to tanhfn.
  CALL prng(gr) ! Gaussian random number.
  dW = DSQRT(dt)*gr ! Noise.
  pwvvar = 1.5*D0*D0*dW ! Compute pwv variation.
totpwv = pwvtot1 + E*dt + pwvvar

! If precipitating:
ELSE IF (sigma1 == 1) THEN
    P = -tanhfn(pwvdet1,4)
    DF = DSQRT(tanhfn(pwvtot1,7)) ! Pass pwv, parameter id to tanhfn.
    DP = DSQRT(tanhfn(pwvtot1,8)) ! Pass pwv, parameter id to tanhfn.
    CALL prng(gr) ! Gaussian random number.
    dWF = DSQRT(dt)*gr ! Noise.
    CALL prng(gr) ! Gaussian random number.
    dWP = DSQRT(dt)*gr ! Noise.
    pwvvar = -(DP*dWP + DF*dWF) ! Compute pwv variation.
    totpwv = pwvtot1 + P*dt + 1.0d0*pwvvar

! If invalid sigma value:
ELSE
    STOP "Error: sigma invalid"
END IF

! Update sigma using new value of totpwv:
CALL trigger(totpwv,sigma1,sigma2,dt)

END SUBROUTINE mbvar

!----------------------------------------------------------------------
!------------!
! PRNG() !
!------------!
!
! This subroutine generates pseudo-random Gaussian numbers using the
! Box-Muller transform. Box-Muller "is a pseudo-random number
! sampling method for generating pairs of independent, standard,
! normally distributed (zero expectation, unit variance) random
! numbers, given a source of uniformly distributed random numbers."
! (Wikipedia) It takes two samples from the uniform distribution on
! the interval (0, 1] and maps them to two standard, normally
! distributed samples:
! Given u1, u2 from the uniform distribution,
! \( Z1 = \sqrt{-2\ln(u1)}\cos(2\pi u2) \)
! \( Z2 = \sqrt{-2\ln(u1)}\sin(2\pi u2) \)
! Then Z1 and Z2 are independent random variables with a standard
! normal distribution.
!
! Output: rn - pseudo-random Gaussian number
!
SUBROUTINE prng(rn)
USE shr_kind_mod, only: r8 => shr_kind_r8
implicit none

real(r8) :: rn
real(r8) :: urand1, urand2, pi

! Compute pi:
pi = DACOS(0d0)

! Get random numbers from uniform distribution on [0,1):
CALL RANDOM_NUMBER(urand1)
CALL RANDOM_NUMBER(urand2)

! Transform to Gaussian using Box-Muller method:
rn = DSQRT(-2d0*DLOG(urand1)) * DCOS(2d0*pi*urand2)

END SUBROUTINE prng

!----------------------------------------------------------------------

!------------------------!
! INIT_RANDOM_SEED() !
!------------------------!

! This subroutine initializes the seed for the random number generator.
!
! Code taken from:
! https://gcc.gnu.org/onlinedocs/gcc-4.2.1/gfortran/RANDOM_005fSEED.html
!
SUBROUTINE init_random_seed()

implicit none

integer :: i, n, clock
integer, dimension(:), allocatable :: seed

CALL RANDOM_SEED(size = n)
ALLOCATE(seed(n))

CALL SYSTEM_CLOCK(COUNT=clock)
seed = clock + 37 * (/ (i - 1, i = 1, n) /)
CALL RANDOM_SEED(PUT = seed)
ALLOCATE(seed)
END SUBROUTINE init_random_seed

!----------------------------------------------------------------------

!--------------!
! TANHFN() !
!--------------!
!
! This function computes the various parameters needed for the mass
! balance model, using a stretched and shifted tanh function of pwv.
! The values in qmid, qwid, f_inf, and finf define the specific
! function for each of the different parameters, and fnid identifies
! the specific function requested by the calling program. f_inf and
! finf are the
!
! Input: o pwv - pwv for current step
! o fnid - id for which parameter to compute
!
! Output: o tanhfn - resulting function value
!
real(r8) FUNCTION tanhfn(pwv,fnid)
USE shr_kind_mod, only: r8 => shr_kind_r8
implicit none
real(r8) :: pwv
integer :: fnid, k
real(r8), dimension(8) :: qmid, qwid, f_inf, finf
!
! Set parameters:
qmid = (/ 61d0,63d0,0d0,64.5d0,0d0,64.5d0,64.5d0,64.5d0 /)
qwid = (/ 30d0,30d0,0d0,1d0,0d0,1d0,1d0,1d0 /)
f_inf = (/ 0d0,22.0d0,0.2d0,2d0,1.5d0,16d0,16d0,0d0 /)
finf = (/ 2d0,0d0,0.2d0,10d0,1.5d0,64.0d0,64.00d0,0.04d0 /)
!
! Compute transition rate:
IF (fnid == 3 .OR. fnid == 5) THEN
  tanhfn = finf(fnid)
ELSE
  tanhfn = f_inf(fnid) + 0.5d0*(finf(fnid)-f_inf(fnid))*(1d0+DTANH((pwv-qmid(fnid))/qwid(fnid)))
END IF
END FUNCTION tanhfn

!----------------------------------------------------------------------
This subroutine simulates the stochastic trigger for precipitation. It takes as input the current value of pwv and sigma, the precipitation state from the previous time step. The "transition rate", r, is computed as a function of the pwv by calling tanhfn(). Depending on the previous precipitation state we compute either r01 or r10, i.e. the transition rate from 0 to 1 or 1 to 0, 0 denoting non-precipitating and 1 denoting precipitating. Next, a transition probability is computed, where ptrans = 1-exp(r*dt), and then we compute the inverse normal of ptrans in order to get the corresponding Gaussian value. Finally, a Gaussian random critical probability (pcrit) is generated with the prng() function, and the Gaussians are compared: if ptrans is greater than or equal to pcrit, the precipitation state switches, otherwise it remains the same.

Input:
- pwv - current value of pwv
- sig1 - precipitation state for previous step
- dt - time step size

Output:
- sig2 - precipitation state for current time step

Routines called:
- tanhfn() - computes transition rate parameters
- prng() - pseudo-random number generator
- norminv() - computes inverse dist to normal cdf

SUBROUTINE trigger(pwv,sig1,sig2,dt)
USE shr_kind_mod, only: r8 => shr_kind_r8
implicit none
real(r8), intent(in) :: pwv, dt
integer, intent(in) :: sig1
integer, intent(out) :: sig2
real(r8) :: r01, r10, prob01, prob10, ptrans, pcrit

! Compute transition probability based on new value of pwv:
! If non-precipitating:
IF (sig1 == 0) THEN
  r01 = tanhfn(pwv,1) ! Pass pwv, parameter id to tanhfn.
  prob01 = 1 - DEXP(-r01*dt) ! Compute transition probability.
  ptrans = norminv(prob01) ! Compute cutoff for switch.
ENDIF
ELSE IF (sig1 == 1) THEN
  r10 = tanhfn(pwv,2) ! Pass pwv, parameter id to tanhfn.
  prob10 = 1 - DEXP(-r10*dt) ! Compute transition probability.
  ptrans = norminv(prob10) ! Compute cutoff for switch.

  ! If invalid sigma value:
  ELSE
    STOP "Error: sigma invalid"
  END IF

! Generate random critical Gaussian:
CALL prng(pcrit)

! Compare transition and critical, update sigma:
IF (ptrans >= pcrit) THEN
  sig2 = ABS(sig1-1)
ELSE
  sig2 = sig1
END IF

END SUBROUTINE trigger

!---------------------------------------------------------------
!---------------!
! NORMINV() !
!---------------!
!
! This function computes a rational approximation of the inverse
! normal CDF. Since the analytical function is not trivial to compute,
! we use an algorithm documented in The Handbook of Mathematical
! Functions (Abramowitz and Stegun, 1965). The probability p is first
! transformed into a parameter t which is:
! t1 = sqrt(-2*ln(p)) if p < 0.5
! t2 = sqrt(-2*ln(1-p)) if p >= 0.5
! Then, the inverse normal is:
! F = -ratappx(t1) if p < 0.5
! F = ratappx(t2) if p >= 0.5
!
! Based on algorithm documented by John D. Cook.
! http://www.johndcook.com/blog/normal_cdf_inverse/
!
! Input: o p - input probability
!
! Output: o norminv - inverse of normal cdf
!
! Routines called: o ratappx() - approximation of normal inverse
real(r8) FUNCTION norminv(p)
USE shr_kind_mod, only: r8 => shr_kind_r8
implicit none
real(r8) :: p

IF (p < 0.5) THEN
  norminv = -ratappx(DSQRT(-2d0*DLOG(p)))
ELSE
  norminv = ratappx(DSQRT(-2d0*DLOG(1d0-p)))
END IF
END FUNCTION norminv

END FUNCTION norminv

!----------------------------------------------------------------------
!---------------!
! RATAPPX()    !
!---------------!
!
! This function computes a rational approximation of the inverse
! normal CDF. Since the analytical function is not trivial to compute,
! we use an algorithm documented in The Handbook of Mathematical
! Functions (Abramowitz and Stegun, 1965) which describes the method as
! a rational approximation for xp where Q(xp) = p.
!
! Based on algorithm documented by John D. Cook.
! http://www.johndcook.com/blog/normal_cdf_inverse/!
!
! Input:    o t - input parameter
!
! Output:   o ratappx - approximation for normal inverse

real(r8) FUNCTION ratappx(t)
USE shr_kind_mod, only: r8 => shr_kind_r8
implicit none
real(r8) :: t
real(r8) :: c(3), d(3)

c = (/2.515517, 0.802853, 0.010328/)
d = (/1.432788, 0.189269, 0.001308/)
ratappx = t - ((c(3)*t + c(2))*t + c(1))/(((d(3)*t + &
d(2))*t + d(1))*t + 1d0)

END FUNCTION ratappx

!----------------------------------------------------------------------
!--------------!
! REDIST() !
!--------------!

! This subroutine redistributes the column-integrated water vapor (pwv)!
! among the levels of the SCM. It computes a proportional
! coefficient - the ratio of the deterministic pwv and the total pwv
! (which includes the deterministic and the variation) - and uses this
! coefficient to add the variation to the specific humidity vertical
! profile (q). Multiplying the deterministic values of q from the
! t+delt time step by this coefficient proportionally distributes the
! variation among the levels while also conserving the column water
! vapor.
!
! Input:
! o nlev - number of vertical layers
! o pwvfin - final value of total pwv at t+delt
! o qdet - det distribution of q among levels
! o pwvdet - deterministic value of pwv at t+delt
!
! Output:
! o qvar - final distr of pwv variation over levels

SUBROUTINE redist(nlev,pwvfin,qdet,pwvdet,qvar)
USE shr_kind_mod, only: r8 => shr_kind_r8
implicit none
integer :: nlev, k
real(r8) :: pwvfin, pwvdet
real(r8), dimension(nlev) :: qdet, qfin, qvar
real(r8) :: pcoeff

! Compute proportion coefficient:
pcoeff = pwvfin/pwvdet

! Distribute proportionally by multiplying det pwv by pcoeff:
DO k = 1,nlev
  qfin(k) = qdet(k)*pcoeff
  qvar(k) = qfin(k) - qdet(k)
END DO
END SUBROUTINE redist
!
END MODULE stochproc
In subroutine phys_init():

```fortran
subroutine phys_init( phys_state, phys_tend, pbuf2d, cam_out )

!-----------------------------------------------------------------------
!
! Initialization of physics package.
!
!-----------------------------------------------------------------------

use physics_buffer, only: physics_buffer_desc, pbuf_initialize,
pbuf_get_index
use physconst, only: rair, cpair, cpwv, gravit, stebol, tmelt, &
latvap, laticce, rh2o, rhoh2o, pstd, zvir, &
karman, rhodair, physconst_init
use ref_pres, only: pref_edge, pref_mid
use aerosol_intr, only: aerosol_init
use carma_intr, only: carma_init
use cloud_rad_props, only: cloud_rad_props_init
use cam_control_mod, only: nsrest ! restart flag
use check_energy, only: check_energy_init
use chemistry, only: chem_init
use prescribed_ozone, only: prescribed_ozone_init
use prescribed_ghg, only: prescribed_ghg_init
use prescribed_aero, only: prescribed_aero_init
use aerodep_flx, only: aerodep_flx_init
use aircraft_emit, only: aircraft_emit_init
use prescribed_volcaero, only: prescribed_volcaero_init
use cloud_fraction, only: cldfrc_init
use co2_cycle, only: co2_init, co2_transport
use convect_deep, only: convect_deep_init
use convect_shallow, only: convect_shallow_init
use cam_diagnostics, only: diag_init
use gw_drug, only: gw_inti
use cam3_aero_data, only: cam3_aero_data_on, cam3_aero_data_init
use cam3_ozone_data, only: cam3_ozone_data_on, cam3_ozone_data_init
use radheat, only: radheat_init
use radiation, only: radiation_init
use cloud_diagnostics, only: cloud_diagnostics_init
use stratiform, only: stratiform_init
use phys_control, only: phys_getopts, waccmx_is
use wv_saturation, only: wv_sat_init
use microp_driver, only: microp_driver_init
use microp_aero, only: microp_aero_init
use macrop_driver, only: macrop_driver_init
```

136
use conv_water, only: conv_water_init
use tracers, only: tracers_init
use aoa_tracers, only: aoa_tracers_init
use rayleigh_friction, only: rayleigh_friction_init
use pbl_util, only: pbl_util_init
use vertical_diffusion, only: vertical_diffusion_init
use dycore, only: dycore_is
use phys_debug_util, only: phys_debug_init
use rad_constituents, only: rad_cnst_init
use aer_rad_props, only: aer_rad_props_init
use qbo, only: qbo_init
use iondrag, only: iondrag_init
endif
if ( defined OFFLINE_DYN )
use metdata, only: metdata_phys_init
endif
use ionosphere, only: ionos_init ! Initialization of ionosphere module (WACCM-X)
use majorsp_diffusion, only: mspd_init ! Initialization of major species diffusion module (WACCM-X)
use clubb_intr, only: clubb inici_cam
use sslt_rebin, only: sslt_rebin_init
use tropopause, only: tropopause_init
use solar_data, only: solar_data_init
use rad_solar_var, only: rad_solar_var_init

!===================================================
!********* For Stochastic Module *********
!===================================================
use cam_history, only: addfld, phys_decomp
use ppgrid, only: pver
use stochproc, only: init_random_seed
!===================================================

! Input/output arguments
type(physics_state), pointer :: phys_state(:)
type(physics_tend ), pointer :: phys_tend(:)
type(physics_buffer_desc), pointer :: pbuf2d(:, :)
type(cam_out_t), intent(inout) :: cam_out(begchunk:endchunk)

! local variables
integer :: lchnk
character(len=16) :: microp_scheme
logical :: do_clubb_sgs

!-----------------------------------------------------------------------
!===================================================
!********* For Stochastic Module *********
!===================================================
call addfld('TMQdeta', 'mm', pver, 'A', 'Precipitable water vapor (deterministic)', phys_decomp)
call addfld('TMQdetb', 'mm', pver, 'A', 'Precipitable water vapor (deterministic)', phys_decomp)
call addfld('TMQstcha', 'mm', pver, 'A', 'Precipitable water vapor (total incl. stochastic component)', phys_decomp)
call addfld('TMQstchb', 'mm', pver, 'A', 'Precipitable water vapor (total incl. stochastic component)', phys_decomp)
call addfld('Sigmaa', 'none', pver, 'A', 'Precipitation state', phys_decomp)
call addfld('Sigmab', 'none', pver, 'A', 'Precipitation state', phys_decomp)
call init_random_seed

!====================================================================
In subroutine tphysbc():

```fortran
!=================================================================
!* For Stochastic Module *********
!=================================================================
use ppgrid, only: pcols, pver
use stochvar
use stockproc
!=================================================================
```
In subroutine tphysbc():

```
physpkg.F90

!=========================================================================
! Moist convection
!=========================================================================

1944 call t_startf('moist_convection')
1945
!
1946 ! Since the PBL doesn't pass constituent perturbations, they
1947 ! are zeroed here for input to the moist convection routine
1948 !
1949 call t_startf ('convect_deep_tend')
1950 call convect_deep_tend( &
1951    cmfmc, cmfcme, &
1952    dlf, pflx, zdu, &
1953    rliq, &
1954    ztodt, &
1955    state, ptend, cam_in%landfrac, pbuf)
1956 call t_stopf('convect_deep_tend')
1957
1958 call physics_update(state, ptend, ztodt, tend)
1959
1960 call pbuf_get_field(pbuf, prec_dp_idx, prec_dp )
1961 call pbuf_get_field(pbuf, snow_dp_idx, snow_dp )
1962 call pbuf_get_field(pbuf, prec_sh_idx, prec_sh )
1963 call pbuf_get_field(pbuf, snow_sh_idx, snow_sh )
1964 call pbuf_get_field(pbuf, prec_str_idx, prec_str )
1965 call pbuf_get_field(pbuf, snow_str_idx, snow_str )
1966 call pbuf_get_field(pbuf, prec_sed_idx, prec_sed )
1967 call pbuf_get_field(pbuf, snow_sed_idx, snow_sed )
1968
!
1969 ! Check energy integrals, including "reserved liquid"
1970 flx_cnd(:,ncol) = prec_dp(:,ncol) + rliq(:,ncol)
1971 call check_energy_chng(state, tend, "convect_deep", nstep, ztodt, zero,
1972    flx_cnd, snow_dp, zero)
1973
!=========================================================================
!********* Stochastic Module *********
!=========================================================================

1976 call sconvect(nstep,state%pint(1,:pver+1),state%q(1,:pver,1),state%lchnk)
1977
1978 if (prec_dp(1) <= 0.0) then
1979     pstate = 0
1980 else
1981     pstate = 1
1982 end if
1983
!=========================================================================
```

140
In subroutine `forecast()`:

```fortran
if (scm_relaxation) then
  ! THIS IS WHERE WE RELAX THE SOLUTION IF REQUESTED
  ! The relaxation can be thought of as a part of the "adjustment" physics
  !
  ! Another way to do this is to estimate the error at t3m2, q3m2 and
  ! include it in the prediction equations (e.g., sum it with the t2
  ! term from the tendency physics). This is numerically stable, but
  ! can not provide a "hard relaxation" because the adjustment physics
  ! then operates on the forecast value. In order to minimize changes
  ! to the code we move the outfld calls for the relaxed variables
  ! (in this case T and q) from linemsbc into this routine after the
  ! relaxation terms have been applied.

  do k=1,plev
    relaxt(k) = 0.0_r8
    relaxq(k) = 0.0_r8
  end do

  if(scm_relaxation) then
    ! dist = 300000. ! distance across the ARM domain
    do k=1,plev
      denom = 2.0*sqrt(u3(k)**2 + v3(k)**2)
      rtau(k) = dist/denom
      ! set relaxation time to constant here if desired
      rtau(k) = 2400._r8 ! time scale in sec
      relaxt(k) = max(ztTd,tobs(k))/rtau(k)
      relaxq(k) = -(q3(k,1) - qobs(k))/rtau(k)
    end do

    t3(k) = t3(k) + relaxt(k)*ztTd
    q3(k,1) = q3(k,1) + relaxq(k)*ztTd

  end do

  call outfld('TRELAX',relaxt,plon,lat )
  call outfld('QRELAX',relaxq,plon,lat )
```

call outfld('TAURELAX', rtau, plon, lat )

! end if
end if
!
!
evaluate the difference in state information from observed
!
do k = 1, plev
  tdiff(k) = t3(k) - tobs(k)
  qdiff(k) = q3(k,1) - qobs(k)
  udiff(k) = u3(k) - uobs(k)
  vdiff(k) = v3(k) - vobs(k)
end do
!
!
Copy observations into time n-1 storage (has diagnostics utility only)
!
tobsml(:,)=tobs(:)
qobsml(:,)=qobs(:)
function pwvstoch = ncproc_stoch(filename)
%------------------------------------------------------------------------
% This function extracts the TMQstch variable from the SCAM netcdf output
% and formats it to be plotted. TMQstch is the 20-second data computed by
% the mass balance SDE scheme.
% %
% Input:  o filename - name of .nc history file with stoch model output
% %
% Output: o pwvstoch - time series of stochastic model output (20sec)
% 1/17/2006 3:00:00 - 2/12/2006 21:00:00
%------------------------------------------------------------------------

% Read field from .nc file
qsta = ncread(filename,'TMQstcha');
qstb = ncread(filename,'TMQstchb');

% Initialize interim arrays:
qsai = zeros(30,1927);
qsbi = zeros(30,1927);

% Convert from 4D arrays to 2D:
for j = 1:1927
    for i = 1:30
        qsai(i,j) = qsta(1,1,i,j);
        qsbi(i,j) = qstb(1,1,i,j);
    end
end

% Initialize 1D array:
qsi = zeros(60*1927,1);

% Splice both 2D arrays into one 1D array:
for k = 1:1927
    for l = 1:30
        ind1 = (k-1)*2*30 + l;
        ind2 = (k-1)*2*30 + 30 +l;
        qsi(ind1) = qsai(l,k);
        qsi(ind2) = qsbi(l,k);
    end
end

% Cut off initial zeros:
qs = qsi(60:end);

% Convert to cm:
pwvstoch = qs/10;
function sigstoch = ncproc_sigma(filename)

%------------------------------------------------------------
% This function extracts the TMQstch variable from the SCAM netcdf output
% and formats it to be plotted. TMQstch is the 20-second data computed by
% the mass balance SDE scheme.
%
% Input: o filename - name of .nc history file with stoch model output
%
% Output: o pwvstoch - time series of stochastic model output (20sec)
% 1/17/2006 3:00:00 - 2/12/2006 21:00:00
%------------------------------------------------------------

% Read field from .nc file
qsta = ncread(filename,'Sigmaa');
qstb = ncread(filename,'Sigmab');

% Initialize interim arrays:
sigai = zeros(30,1927);
sigbi = zeros(30,1927);

% Convert from 4D arrays to 2D:
for j = 1:1927
    for i = 1:30
        sigai(i,j) = qsta(1,1,i,j);
        sigbi(i,j) = qstb(1,1,i,j);
    end
end

% Initialize 1D array:
sigi = zeros(60*1927,1);

% Splice both 2D arrays into one 1D array:
for k = 1:1927
    for l = 1:30
        ind1 = (k-1)*2*30 + l;
        ind2 = (k-1)*2*30 + 30 +l;
        sigi(ind1) = sigai(l,k);
        sigi(ind2) = sigbi(l,k);
    end
end

% Cut off initial zeros:
sigstoch = sigi(60:end);
function plot_twp_stoch(scamstoch)
%------------------------------------------------------------------------
% This function takes the twp (observed) time series and the scam
% stochastic time series, both at 20 second time steps, and plots against
% the time series of date/time. It formats to be ready for inclusion in
% report. 1/17/2006 3:00:00 - 2/12/2006 21:00:00.
%------------------------------------------------------------------------
load('scamplotdata2.mat');

% Plot:
plot(Tpwv20sec,pwvscm20sec,'k',Tpwv20sec,scamstoch,'b')

% Formatting:
ylabel('PWV (cm)')
legend('Observed (black)','Model (blue)');
set(gca,'FontSize',14)

% Format y-axis:
ylim([1 9])
ax = gca;
ax.YTick = [2 4 6 8];

% Format x-axis:
xlim([datenum(Tpwv20sec(1)) datenum(Tpwv20sec(end))])
x1_18 = datenum(2006,1,18);
x1_22 = datenum(2006,1,22);
x1_26 = datenum(2006,1,26);
x1_30 = datenum(2006,1,30);
x2_03 = datenum(2006,2,3);
x2_07 = datenum(2006,2,7);
x2_11 = datenum(2006,2,11);
ax.XTick = [x1_18 x1_22 x1_26 x1_30 x2_03 x2_07 x2_11];
function plot_sigma(sigstoch)

% This function takes the precipitation state time series, consisting of
% 0s and 1s, at 20 second time steps, and plots against
% the time series of date/time. It formats to be ready for inclusion in
% report.
%
% Input:
%  o sigstoch - precipitation state time series (20sec)
%  o Tpwv20sec - date/time series (20sec)
%
% Output: o stem plot of precipitation state output

load('scamplotdata2.mat','Tpwv20sec')

stem(datenum(Tpwv20sec),sigstoch,'k','Marker','none')

% Formatting:
ylabel('\sigma')
set(gca,'FontSize',18)

% Format y-axis:
ylim([-0.5 1.5])
ax = gca;
adax.YTick = [0 1];

% Format x-axis:
xlim([datenum(Tpwv20sec(1)) datenum(Tpwv20sec(end))])
x1_18 = datenum(2006,1,18);
x1_22 = datenum(2006,1,22);
x1_26 = datenum(2006,1,26);
x1_30 = datenum(2006,1,30);
x2_03 = datenum(2006,2,3);
x2_07 = datenum(2006,2,7);
x2_11 = datenum(2006,2,11);
ax.XTick = [x1_18 x1_22 x1_26 x1_30 x2_03 x2_07 x2_11];
datetick('x','mmm\_dd','keeplimits','keepticks')