Passive Microwave and Hyperspectral Infrared Retrievals of Atmospheric Water Vapor Profiles

by

Jay Brian Hancock

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science and Engineering at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2001

© Massachusetts Institute of Technology 2001. All rights reserved.
DISCLAIMER OF QUALITY

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available. If you are dissatisfied with this product and find it unusable, please contact Document Services as soon as possible.

Thank you.

Some pages in the original document contain text that runs off the edge of the page.

(Appendix section A)
Passive Microwave and Hyperspectral Infrared Retrievals of Atmospheric Water Vapor Profiles

by

Jay Brian Hancock

Submitted to the Department of Electrical Engineering and Computer Science on May 25, 2001, in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science and Engineering

Abstract

Two clear-air relative humidity profile estimators were designed and implemented using neural networks. The microwave estimator is the first to utilize 54-, 118-, and 183-GHz channels for simultaneously retrieving a relative humidity profile. It utilizes 2 separate instruments simultaneously. The first instrument is a medium-resolution dual-band radiometer with one set of 8 double-sideband 118-GHz channels and a second set of 8 single-sideband 54-GHz channels. The other instrument is a high-resolution double-sideband radiometer with a set of 3 183-GHz channels, and additional channels at 89, 220, and 150 GHz. The infrared estimator is among the first to utilize a hyperspectral infrared aircraft instrument for relative humidity profile retrievals. The infrared instrument is a 9000-channel interferometer operative over the wavelength range of 3.8–16.2 microns. Both estimators utilized neural networks of comparable topology and training methods. The training data was generated from the SATIGR set of 1761 RAOBs using a different implementation of the discrete radiative transfer equation for each estimator.

The test data were from two clear-air ER-2 aircraft flights during the tropical CAMEX-3 mission near Andros Island. The retrievals were robust in the face of unknown instrument bias and noise, which introduced a difference between the training data and the flight data. A noise-averaging technique achieved robustness in exchange for a degradation in sensitivity of the retrieval algorithms. Robustness was demonstrated by the retrieval agreement between the microwave and infrared instruments. The theoretical average rms error in relative humidity for the various techniques on the training set was 12% for the microwave estimator, 11% for the infrared, and 10% for a linear regression of the two. In application to two flights, the rms error was 9.4% for the microwave, 7.7% for the infrared, and 7.7% for the combination, based on comparisons with nearby radiosondes.

Thesis Supervisor: David H. Staelin
Title: Professor of Electrical Engineering
Acknowledgments

First and foremost, I thank professor David Staelin for his patience and guidance throughout the past two years. His good humor made an immediate impression upon me in the beginning, and his encouragement helped me to finish this thesis and find overall enjoyment in my Master's work. In the middle of designing an algorithm when nothing seems to be working properly it is like being in a cloud, and since then, that cloud has been cleared.

I thank Jim Wang and colleagues from NASA/Goddard flight center for providing the MIR instrument data, and Bill Smith and Bob Knuteson for providing the NAST-I data.

I thank my colleagues in the Remote Sensing and Estimation group who have helped me at various points: William Blackwell, Scott Bressler, Fredrick Chen, Junehee Lee, and Vincent Leslie; especially Phil Rosenkranz for his atmospheric science help. In the RLE office, thanks go out to Marylin Pierce without whom chaos would take over.

My mother has shown extraordinary strength throughout the years which has been an inspiration to me as I pursue my advanced degree. She deserves more thanks than I could possibly give for her understanding and her support for everything I do. Thanks, Mom!

I thank Rabbi Sally Finestone, Rabbi Norman Janus, and Rabbi Morheim for helping me to navigate onto a spiritual path while my thesis was in a fog; all my other Jewish friends for their warmth and support; and Irit Aharony for teaching me not just about Hebrew, but also about life. Todah rabah lechem!

Finally, thanks to all those who make contributions to my life: Friends of mine, the guy who takes our the trash at 1 a.m., my fish, the sandwich guy at LaVerdes’, the parking clerk for his bright orange “poetry” on my car, those friendly people at the Trident Cafe, and all others.

This work was made possible from the support of NASA Grant #NAG5-7487 and NASA Contract #NAS5-31376, and from the support of Lincoln Laboratory.
3.5 Training Algorithms ........................................ 42
  3.5.1 Gradient Descent ..................................... 42
  3.5.2 Backpropagation ...................................... 43
  3.5.3 Momentum ............................................. 44
  3.5.4 Adaptive Learning Rate ............................... 45
  3.5.5 Speed of Convergence .................................. 45
  3.5.6 A priori Bias .......................................... 46
3.6 Designing Network Architecture ............................. 47
3.7 Noise Averaging ............................................ 48
  3.7.1 Justification .......................................... 48
  3.7.2 Robust Noise Description ............................... 48
  3.7.3 Effective Increase in Training Set Size .................. 50
3.8 Regularization ................................................ 50

4 Data ................................................. 53
  4.1 Introduction ............................................. 53
  4.2 The SATIGR Radiosonde Collection ......................... 53
    4.2.1 Water Vapor Problem .................................. 55
    4.2.2 Modifying SATIGR ..................................... 55
  4.3 CAMEX-3 Flight Data ....................................... 57
    4.3.1 ER-2 .................................................. 57
    4.3.2 RAOBs ................................................ 57
    4.3.3 Microwave Flight Data ................................ 58
    4.3.4 Infrared Flight Data .................................... 63

5 Microwave Retrieval Algorithm ................................. 67
  5.1 Brightness Temperature Simulation ......................... 67
    5.1.1 Instrument Models ..................................... 68
    5.1.2 Microwave Simulation Code ............................. 71
  5.2 Training the Neural Network Estimator ..................... 73
    5.2.1 Surface Model ........................................ 73
    5.2.2 Instrument Noise Model, N_{\text{inst}} .................. 74
    5.2.3 Network Architecture .................................. 75
5.2.4 The Training Process ........................................... 77
5.2.5 Training Analysis .............................................. 78
5.2.6 Scan Angle Note .................................................. 79
5.3 Retrievals ........................................................... 83
  5.3.1 Definitions for Retrieval Figures ............................ 83
  5.3.2 Retrieval Image Processing .................................... 84
  5.3.3 September 13th, 1998 .......................................... 85
  5.3.4 August 13th, 1998 ............................................. 86
  5.3.5 August 8th, 1998 ............................................... 92
5.4 Analysis of Retrieval Accuracy ................................. 104

6 Infrared Retrievals .................................................. 105
  6.1 Radiative Transfer Coefficient Simulation .................... 106
  6.2 Feature Selection ................................................ 106
  6.3 Feature Compression ............................................. 107
  6.4 Network Training ................................................ 109
    6.4.1 Surface Model .............................................. 109
    6.4.2 Instrument Noise Model, N_{inst} .......................... 109
    6.4.3 Network Architecture ...................................... 112
    6.4.4 The Training Process ...................................... 112
    6.4.5 Training Analysis ......................................... 112
    6.4.6 Scan Angle Note ............................................ 113
  6.5 Retrievals ....................................................... 113
    6.5.1 September 13th Flight ..................................... 114
    6.5.2 August 13th Flight ......................................... 115
  6.6 Analysis of Retrieval Accuracy ............................... 115

7 Comparing Microwave and Infrared ............................ 123
  7.1 Comparison ..................................................... 123
    7.1.1 Correlation of Retrievals ................................ 125
  7.2 Combined Linear Retrieval ................................... 127
    7.2.1 Retrievals ................................................ 127
    7.2.2 Analysis of Retrieval Accuracy ......................... 136
8 Conclusion and Future Work

8.1 Summary of Work ........................................ 137
8.2 Conclusion .............................................. 138
8.3 Future Work ............................................. 139

A Software Source Code Selections 141
A.1 Saturation Vapor Pressure ......................... 141
A.2 Simulation ............................................. 143
  A.2.1 Microwave .......................................... 143
  A.2.2 Infrared ............................................ 156
A.3 Neural Network Training .......................... 175
  A.3.1 Network Control Code ......................... 175
  A.3.2 Modified MATLAB Training Code with Momentum and Backpropagation ............................. 197
  A.3.3 Noise-Averaging Routines ..................... 208
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Calculation of $T_D$ with a sum of paths.</td>
<td>24</td>
</tr>
<tr>
<td>2-2</td>
<td>Calculation of $T_D$ with a state space model.</td>
<td>24</td>
</tr>
<tr>
<td>2-3</td>
<td>Calculation of $T_U$ with a sum of paths.</td>
<td>26</td>
</tr>
<tr>
<td>2-4</td>
<td>Calculation of $T_U$ with a state space model.</td>
<td>26</td>
</tr>
<tr>
<td>3-1</td>
<td>A scree plot of the eigenvalues showing the noise floor.</td>
<td>37</td>
</tr>
<tr>
<td>3-2</td>
<td>Effect of removing eigenvectors.</td>
<td>38</td>
</tr>
<tr>
<td>3-3</td>
<td>A single neuron.</td>
<td>39</td>
</tr>
<tr>
<td>3-4</td>
<td>Multilayer neural network.</td>
<td>41</td>
</tr>
<tr>
<td>3-5</td>
<td>Backpropagation schematic.</td>
<td>43</td>
</tr>
<tr>
<td>3-6</td>
<td>A deficient fixed training set.</td>
<td>49</td>
</tr>
<tr>
<td>3-7</td>
<td>A conservative noise model for a robust network.</td>
<td>49</td>
</tr>
<tr>
<td>4-1</td>
<td>SATIGR temperature distribution.</td>
<td>54</td>
</tr>
<tr>
<td>4-2</td>
<td>SATIGR relative humidity distribution.</td>
<td>55</td>
</tr>
<tr>
<td>4-3</td>
<td>Distribution of relative humidity vs. temperature.</td>
<td>56</td>
</tr>
<tr>
<td>4-4</td>
<td>Distribution of relative humidity vs. pressure.</td>
<td>56</td>
</tr>
<tr>
<td>4-5</td>
<td>The ER-2 aircraft.</td>
<td>58</td>
</tr>
<tr>
<td>4-6</td>
<td>Example of NAST-M 54 GHz data over Andros Island.</td>
<td>59</td>
</tr>
<tr>
<td>4-7</td>
<td>Example of NAST-M 118 GHz data over Andros Island.</td>
<td>60</td>
</tr>
<tr>
<td>4-8</td>
<td>Example of MIR data over Andros Island.</td>
<td>62</td>
</tr>
<tr>
<td>4-9</td>
<td>A comparison between simulated RAOB and flight brightness temperatures for August 8th.</td>
<td>64</td>
</tr>
<tr>
<td>4-10</td>
<td>A comparison between simulated RAOB and flight brightness temperatures for September 13th.</td>
<td>64</td>
</tr>
</tbody>
</table>
4-11 The computed bias that is removed from the microwave data.

4-12 Example of NAST-I data over Andros Island.

5-1 MIR frequency passband model.

5-2 Antenna pattern model.

5-3 NAST-M 54 GHz frequency passbands.

5-4 NAST-M 118 GHz frequency passbands.

5-5 Typical surface emissivity vectors.

5-6 MIR modeled sensitivity.

5-7 NAST-M 54 GHz modeled sensitivity.

5-8 NAST-M 118 GHz modeled sensitivity.

5-9 Microwave neural network training curve.

5-10 Linear vs. non-linear retrieval error.

5-11 Single angle vs. noise-averaged network.

5-12 Grouping scan angles for network training.

5-13 GOES infrared imagery on September 13th with the ER-2 flight path superimposed.

5-14 Microwave flight retrieval for September 13th.

5-15 Microwave vertical nadir retrieval over Andros Island, pass #1 for September 13th.

5-16 Microwave vertical nadir retrieval over Andros Island, pass #2 for September 13th.

5-17 Microwave horizontal set of retrievals over Andros Island, Pass #2 for September 13th.

5-18 Microwave RAOB/retrieval-average comparison for September 13th.

5-19 Microwave rms error plot for September 13th.

5-20 GOES infrared imagery on August 13th with the ER-2 flight path superimposed.

5-21 Microwave flight retrieval for August 13th.

5-22 Microwave vertical nadir retrieval for region #1 on August 13th.

5-23 Microwave vertical nadir retrieval for region #2 on August 13th.

5-24 Microwave horizontal set of retrievals for region #1 on August 13th.
5-25 Microwave horizontal set of retrievals for region #2 on August 13th. 96
5-26 Microwave RAOB/retrieval-average comparison for August 13th. 97
5-27 Microwave rms error plot for August 13th. 97
5-28 GOES infrared imagery on August 8th with the ER-2 flight path superimposed. 98
5-29 Microwave flight retrieval for August 8th. 99
5-30 Microwave vertical nadir retrieval for region #1 on August 8th. 100
5-31 Microwave vertical nadir retrieval for region #2 on August 8th where the Land-Sea Boundary occurs at 6:27 P.M. 100
5-32 Microwave horizontal set of retrievals for region #1 on August 8th. 101
5-33 Microwave horizontal set of retrievals for region #2 on August 8th where the Land-Sea Boundary occurs at 6:27 P.M.. 102
5-34 Microwave RAOB/retrieval-average comparison for August 8th. 103
5-35 Microwave rms error plot for August 8th. 103
6-1 Weighting function 1st moment, 2nd moment, and wavenumber scatter plots. 108
6-2 Bounding the distribution of instrument bias. 111
6-3 Training curve for the infrared neural network. 113
6-4 Linear vs. non-linear retrieval error. 114
6-5 Infrared flight retrieval for September 13th. 116
6-6 Infrared vertical nadir retrieval over Andros Island, pass #1 for September 13th. 117
6-7 Infrared vertical nadir retrieval over Andros Island, pass #2 for September 13th. 117
6-8 Infrared RAOB/retrieval-average comparison for September 13th. 118
6-9 Infrared rms error plot for September 13th. 118
6-10 Infrared flight retrieval for August 13th. 119
6-11 Infrared vertical nadir retrieval for region #1 on August 13th. 120
6-12 Infrared vertical nadir retrieval for region #2 on August 13th. 120
6-13 Infrared RAOB/retrieval-average comparison for August 13th. 121
6-14 Infrared rms error plot for August 13th. 121
7-1 Schematic of features for August 13th flight, region #1. 124
7-2 Cross-correlation matrix between infrared and microwave on the training data. 125
7-3 Scatter diagram of 1st principal component correlation for August 13th. . . . 126
7-4 Scatter diagram of 1st principal component correlation for September 13th. . . 126
7-5 Combined linear rms retrieval error. .................................................. 127
7-6 Combined infrared and microwave flight retrieval for September 13th. . . . 130
7-7 Combined infrared and microwave vertical nadir retrieval over Andros Island,
    pass #1 for September 13th. .......................................................... 131
7-8 Combined infrared and microwave vertical nadir retrieval over Andros Island,
    pass #2 for September 13th. .......................................................... 131
7-9 Combined infrared and microwave RAOB/retrieval-average comparison for
    September 13th. .............................................................................. 132
7-10 Combined infrared and microwave rms error plot for September 13th. . . . 132
7-11 Combined infrared and microwave flight retrieval for August 13th. . . . . 133
7-12 Combined infrared and microwave vertical nadir retrieval for region #1 on
    August 13th. ................................................................................. 134
7-13 Combined infrared and microwave vertical nadir retrieval for region #2 on
    August 13th. ................................................................................. 134
7-14 Combined infrared and microwave RAOB/retrieval-average comparison for
    August 13th. ................................................................................. 135
7-15 Combined infrared and microwave rms error plot for August 13th. . . . . 135
### List of Tables

4.1 Distribution of SATIGR throughout the climates. ........................................... 54

5.1 Spectral description of the MIR system. ......................................................... 69

5.2 Spectral description of the NAST-M 54-GHz system. ................................... 70

5.3 Spectral description of the NAST-M 118-GHz system. ................................ 71

5.4 Results from altering training assumptions. ................................................. 83

5.5 Surface features for September 13th, pass #1. .......................................... 88

5.6 Surface Features for September 13th, pass #2. ......................................... 88

5.7 Microwave retrieval accuracies for the flight data. ................................... 104

6.1 Infrared retrieval accuracies for the flight data. ....................................... 122
Chapter 1

Introduction

Retrieving water vapor profiles using the 183 GHz band has been of interest to the community for some time. A study by Wang et al. using both 90-GHz and 183-GHz channels indicated that it is possible to retrieve the general features of the water vapor profiles, but the fine details in rapidly changing profiles is lost [WC90], [WBS93]. MIR was then deployed and showed promise for improved profile retrievals [W+95], [RAW+96].

Around the same time, simulation studies were conducted on retrieving water vapor profiles using temperature channels around 54 GHz in addition to the 183 GHz channels. Kuo et al. designed an iterative statistical estimator which addressed the non-linearity of the radiative transfer function of water vapor. The resulting rms error of 10-12% in relative humidity was encouraging but the estimation can not be performed in real-time and the a priori statistics were based on a local, not global, climate [KS94]. Cabrera et al. continued the simulation study and developed a neural network retrieval algorithm which performs as well as the iterative statistical technique. It also uses global statistics and can perform retrievals in real-time [CM93], [CMS95]. Another simulation study by Li et al. on retrieving vapor, liquid, and ice again shows success for neural networks, and the networks were applied to an actual radiometer [LVCT97].

An algorithm to retrieve water vapor relative humidity profiles was developed by Wang et al. for the MIR instrument with channels near 183 GHz. Real data over ocean shows accurate retrievals near the surface with 0.12 rms error, but the error increases to 0.67 at 10.25 km altitude [WRC97].

This work uses a neural network-based retrieval algorithm for the 183-GHz water-vapor
channels in combination with the 54 GHz- and 118 GHz-temperature channels. The algorithm is global across climate and robust across varying surface types and achieves 11-13% error over the pressure range 1013 – 315 mbar. The algorithm is applied to MIR and NAST-M flight data and performs well for several flights when compared to radiosondes.

The same retrieval algorithm developed for the microwave channels is applied to the infrared and similar results are achieved. The two paradigms produce retrievals which agree in several feature-rich flight segments. A simple linear regression on the outputs of two estimators yields better results than each individual result.

1.1 Thesis Outline

Chapter 2 discusses the necessary physics and mathematics used in this work. Chapter 3 describes neural networks and various techniques employed in conjunction with them such as noise averaging, a priori knowledge incorporation, and regularization. Chapter 4 presents the various types of data used: training data, flight data, and RAOBs used for validation. Chapter 5 presents the algorithm for microwave retrievals using neural networks and validates it using flight data and coincident RAOBs. Several retrievals are presented. Chapter 6 is the counter-part of Chapter 5. It describes the application of the microwave algorithm to the infrared paradigm and performs the same validations and retrievals as in the previous chapter. Chapter 7 examines a linear estimator which combines the retrievals from the microwave and infrared paradigms into one retrieval. Chapter 8 brings it all to a close.

The appendix provide the source code for this thesis.
Chapter 2

Background

2.1 Physics

2.1.1 Remote Sensing

Remote sensing is the art and science of using radiation which passes through a substance to extract useful information from that substance. For the remote sensing of water vapor, an instrument measures radiation which has passed through the atmosphere, originating from various sources, including outer space, the earth’s surface, and from the atmosphere itself. The radiation carries information about the temperature and water vapor content of the atmosphere, and it is our job to extract that information.

The most important equation in this work, the radiative transfer equation, relates the atmospheric temperature and relative humidity profiles to the radiation emitted from the atmosphere. A tractable model of radiative transfer is obtained by discretizing this equation. The resulting model, the radiative transfer forward algorithm, is used to compute the radiances that a downward-looking instrument would see for a certain atmospheric profile. Typically, several different relative humidity profiles produce the same set of radiances at the instrument, so the inverse problem is singular. A priori knowledge is used to decide which profiles most likely occurred. When given the atmospheric profile and the computed radiances values, statistical techniques are able to learn the inverse relationship. In general, this so-called inverse problem is a hard problem for radiative transfer. The accuracy we can achieve is limited by the model and by our knowledge of it.
2.1.2 Radiative Transfer Equation

The assumptions for the radiative transfer equation are stated and the resulting equation is given here, but we omit the derivation. A derivation can be found in [Lio80, Bla95].

The radiative transfer equation is derived from the optical properties of the atmosphere. The transmittance of the atmosphere at a particular frequency is determined by the absorption of radiation by the constituent molecules of the atmosphere.

Two simplifying assumptions are made in a derivation for this equation. The first is that the atmosphere is planar stratified; the temperature and water vapor content are constant for a given pressure level. The second is that the atmosphere and surface behave like a blackbody; the radiation emitted from them is proportional to their temperature. From these assumptions, an instrument looking down from a height \( z \) will observe a brightness temperature according to

\[
TB(f, z) = \int_0^z T(h)\alpha(f, h)\tau_b(f) \, dh + \tau_0(f) \left( \rho \int_0^\infty T(h)\alpha(f, h)\tau_0^b(f) \, dh + \rho\tau_0^{\text{inf}}(f)T_{\text{cosmic}} + \epsilon T_{\text{surf}} \right),
\]

where \( \tau_0^b(f) \) is the transmittance from level \( a \) to level \( b \) given by \( \tau_0^b(f) = \exp(-\int_a^b \alpha(f, x) \, dx) \), \( T(h) \) is the temperature at height \( h \), \( \alpha(f, h) \) is the atmospheric absorption at frequency \( f \) and height \( h \), \( \rho \) is the surface reflectivity, \( \epsilon \) is the surface emissivity which is equivalent to \((1 - \rho)\), \( T_{\text{surf}} \) is the surface temperature, and \( T_{\text{cosmic}} \) is the cosmic background temperature.

The first term in Equation 2.1 represents the atmospheric radiation emitted in an upward direction along the path from the surface to the downward-looking instrument. The second term represents the radiation from the atmosphere in a downward direction along the path from the top of the atmosphere down to the surface. This contribution then reflects from the surface and passes through the atmosphere up to the instrument, being attenuated by the optical transmittance factor. The third term gives the effect of the cosmic background radiation as it travels downward through the entire atmosphere, reflects from the surface, and travels upward again until it reaches the instrument, being attenuated by the transmittance along both paths. The fourth term is due to the radiation from the surface which is emitted in proportion to the surface emissivity and is also attenuated as it passes through
the lower atmosphere up to the instrument.

2.1.3 Radiative Transfer Algorithm

Data sets, which contain discrete profiles of pressure, water vapor, and temperature, are available to use in atmospheric work. Equation 2.1 is discretized to obtain a tractable model that uses these profiles to simulate observations of a downward looking instrument. The discretization is performed by first partitioning the abscissa of the integrand and then using the trapezoidal rule to approximate the integral. The partitioning is determined by the particular profile set that is used.

The calculation of the surface parameters $\epsilon$, $\rho$, $T_{\text{surf}}$, and $T_{\text{cosmic}}$ can be separated from the atmospheric parameters. This separation is used when computing surface parameters on the fly during the neural network training; the computationally intense part of the computation, the integrals, are computed a priori. The remaining part of the computation is performed on the fly using minimal computation for the generation of the surface parameters and their linear combination with the pre-computed atmospheric parameters. Removing these redundancies saves a gargantuan number of computations.

The core radiative transfer algorithm is then the computation of the three terms, which are not surface parameters. Define the notation $T_D$ to represent the term for the downwelling radiation, $\int_{0}^{\infty} T(h) \alpha(f, h) \tau^{h}_{\phi}(f) \, dh$; $T_U$ to represent the term for the upwelling radiation, $\int_{0}^{2} T(h) \alpha(f, h) \tau^{h}_{\phi}(f) \, dh$; and $\tau$ to represent the term for the one-way transmittance of the atmosphere for radiance, $\tau^{h}_{\phi}(f)$. Whenever the term transmittance is used, it refers to either the radiance transmittance, given here, or the equivalent transmittance in brightness temperature; the meaning will be clear from the context. We will use this notation to describe the implementation of the radiative transfer algorithm specific to this thesis.

The algorithm is summarized as follows. The input of the algorithm is a water vapor and temperature profile, the antenna pattern, and a weighting function for the frequency passband. Parameters pre-specified for the physics are the cosmic background radiation and the model of the absorption lines for oxygen and water vapor. The output of the algorithm is the three coefficients: $T_D$, $T_U$, and $\tau$.

To use these three coefficients for a realistic rendition of the total radiation seen by a downward-looking instrument, a source external to the algorithm must provide the temperature and reflectivity (or emissivity) of the surface. In our case, the surface parameters are
generated according to an a priori model.

The atmosphere is modeled as a set of slabs characterized by an optical transmittance parameter and two components of emitted radiation: one component is emitted from the top of the slab, and the other is emitted from the bottom. One might ask why the two components are different. The reason the upward component differs from the downward component is because the cumulative transmittance through several layers is multiplicative over the layers' transmittances. For example, if we divide a slab into two smaller slabs: a top slab with transmittance \( r_A \) and a bottom slab with transmittance \( r_B \). If an emitting molecule lies in the middle of these slabs and emits an amount of radiation \( r \), then the radiation emitted from the top of the slab due to the molecule is \( rr_A \) whereas the radiation emitted from the bottom of the slab is \( rr_B \). Each molecule within a slab effectively divides the slab into two slabs. If the distribution of molecules or the transmittances is asymmetric, then the outputs from the top and bottom will also be asymmetric.

**Frequency and Antenna Weighting Functions**

The algorithm requires a tractable model for the antenna pattern and the frequency passband. The description given below will refer to the antenna pattern and then the corresponding frequency passband description is analogous.

The input to the model is a single antenna angle. The antenna pattern can be described at nadir by a set of small angles that surround nadir, each with an associated weight. The radiative transfer algorithm will compute output coefficients for each angle in the set, and then take a weighted sum of the individual coefficients using the associated weights to obtain one set of output coefficients. For computation of output points that are not at nadir, the algorithm will translate all angles in the set by the value of the single antenna angle before computation. The passband description is analogous, replacing the angle concept with a frequency.

For example, this model can describe both a delta function approximation and a 1-dimensional Gaussian approximation. The former approximation will have a weight of 1 at the center angle, while the latter approximation will have weights corresponding to a sampled Gaussian.

If an instrument has several channels, a separate description may be needed for each channel for both the antenna pattern and the frequency passband. The algorithm will
compute each channel separately, applying the appropriate model to each channel.

We will now restrict our attention to the computation of one specific angle and frequency for the three coefficients.

**Computing the Downwelling Component, $T_D$**

The transmittance for each slab is dependent on the absorption due to oxygen and water vapor content of that slab. Models are available for computation of the transmittances [Ros93].

We are given the set of transmittances of each slab, $\{\tau(i)\}$, the radiation emitted from the bottom of each slab, $\{T_D(i)\}$, and the cosmic background radiation, $T_{\text{cosmic}}$. One method of computing the total atmospheric contribution to the downward path projects each layer’s contribution to the surface and then sums all of the individual contributions. A second method represents the first as a state-space equation and projects the radiation from one layer at a time, cumulating the radiation at each step.

The first method is demonstrated here. Let the atmosphere be divided into $M$ layers. The equation for computing the contribution of one particular layer at the bottom of the atmosphere is

$$T_D(j) = \prod_{i=1}^{j-1} \tau(i) T_D(j), \quad (2.2)$$

and the total contribution is obtained by

$$T_D = \sum_{j=2}^{M} T_D(j) + \prod_{i=1}^{M} \tau(i) T_{\text{cosmic}}. \quad (2.3)$$

Figure 2-1 shows this calculation graphically.

The second method is the same calculation rewritten as the state-space equation

$$X(M) = T_{\text{cosmic}}$$

$$X(n - 1) = X(n) \tau(n) + T_D(n) \quad (2.4)$$

$$T_D = X(0)$$

This computation is shown graphically in Figure 2-2. Of the two approaches, using the state-space model is more efficient because we use about $\frac{N^2}{2}$ multiplies in Equation 2.3 and about $N$ multiplies in Equation 2.4.
Figure 2-1: Calculation of $T_D$ with a sum of paths.

$$X(M) = T_{\text{cosmic}}$$

$$X(M) = X(M - 1)\tau(M)$$

$$X(M - 1) = X(M - 2)\tau(M - 1)$$

$$X(M - 2) = \vdots$$

$$X(j + 1) = \vdots$$

$$X(j) = X(j + 1)\tau(j) + T_D(j)$$

$$X(j) = \vdots$$

$$X(0) = \tau(1)$$

Figure 2-2: Calculation of $T_D$ with a state space model.
Computing the Upwelling Component, $T_U$

The transmittance of each slab for the upwelling component is the same as for the downwelling component since the surface is non-specular.

We are given the set of transmittance of each slab, \{\tau(i)\}, and the radiation emitted from the top of each slab, \{T_U(i)\}. There are two methods for computing the upwelling component that are analogous to the downwelling case.

The first method again projects each layer's contribution upward to the downward-looking instrument, where the instrument is at the upward extremity of the Nth layer. The equation for computing the contribution of one particular layer at the level of the instrument is

$$\tau(j) = \prod_{i=j+1}^{N} \tau(i) T_U(j), \quad (2.5)$$

and the total contribution is obtained by

$$T_U = \sum_{j=1}^{N-1} T_U(j) \quad (2.6)$$

Figure 2-3 shows this calculation graphically.

The first method is rewritten similarly to how $T_D$ was rewritten to obtain the state space equation

$$X(0) = 0$$
$$X(n+1) = X(n)\tau(n+1) + T_U(n+1) \quad (2.7)$$
$$T_U = X(N)$$

This computation is shown graphically in Figure 2-4.

Computing the One-way Transmittance, $\tau$

The coefficient for the transmittance from the surface to the level of the instrument, $\tau$, is given by

$$\tau = \prod_{i=1}^{N} \tau(i). \quad (2.8)$$
Figure 2-3: Calculation of $T_U$ with a sum of paths.

Figure 2-4: Calculation of $T_U$ with a state space model.
Averaging over the Antenna Pattern and Frequency Passbands

The above computations are frequency and angle dependent. The instrument model provides $W_{\text{Ant}}(\theta)$ and $W_{\text{PB}}(f)$, the weighting functions for the antenna pattern and frequency passband, respectively. To obtain the frequency and angle independent equation, we average over all points within a spot using these weighting functions. This averaging is performed by approximating the equations

$$T_D = \int \int W_{\text{Ant}}(\theta)W_{\text{PB}}(f)\tau_N(f,\theta)T_D(f,\theta)d\theta df,$$

(2.9)

$$T_U = \int \int W_{\text{Ant}}(\theta)W_{\text{PB}}(f)T_D(f,\theta)d\theta df,$$

(2.10)

and

$$\tau = \int \int \tau_N(f,\theta)d\theta df$$

(2.11)

with the trapezoidal rule for integral approximation.

Adding Surface Emissivity and Temperature

We delay computation of the brightness temperature until training of the neural network, because we use the noise-averaging technique mentioned in Section 3.7 on the random distribution of surface parameters. Once we have obtained a value for the surface emissivity and temperature, $\varepsilon_{\text{surf}}$ and $T_{\text{surf}}$, respectively, we use the equation

$$TB = T_U + \tau (\varepsilon_{\text{surf}}T_D + (1 - \varepsilon_{\text{surf}})T_{\text{surf}})$$

(2.12)

to compute the total brightness temperature that the instrument sees.

2.1.4 Relative Humidity

A definition for the notion of relative humidity must be specified because it is somewhat ambiguous.

27
In general, relative humidity is defined as

\[ \rho = \frac{P_{H_2O}}{P_{sat}}, \tag{2.13} \]

where \( P_{H_2O} \) is the partial pressure of water vapor and \( P_{sat} \) is the saturation pressure of water vapor.

There are several definitions for relative humidity. This is due to the fact that water vapor can exist in a super-saturated state in the atmosphere without condensing to a solid or liquid.

We assume here that all water vapor will spontaneously form ice crystals below 233.16°K. The corresponding code used to compute saturation vapor pressure is included in Appendix A.1.

2.2 Mathematics

In this thesis, we denote scalars by variables in plain type as in \( x \), vectors in boldface as in \( \mathbf{x} \), and matrices as capital letters as in \( X \).

2.2.1 Linear Regression

Assume there exists a measurement model \( f \) which maps some process vector

\[ \mathbf{y} = [y_1, y_2, \ldots, y_L]^T \]

to a set of measurements \( \mathbf{x} = [x_1, x_2, \ldots, x_M]^T \) via

\( f : \mathbf{y} \rightarrow \mathbf{x} \). Linear regression finds the best linear estimate \( \hat{\mathbf{y}} = g(\mathbf{x}) \) which minimizes the Euclidean cost function. More specifically, if we augment \( \mathbf{x} \) with a bias dimension by

\[ \bar{x} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}, \tag{2.14} \]

then the linear estimate is given by the matrix operation

\[ \hat{\mathbf{y}} = A\bar{x}. \tag{2.15} \]

Given the cost function

\[ J(\hat{\mathbf{y}}) = (\hat{\mathbf{y}} - \mathbf{y})^T(\hat{\mathbf{y}} - \mathbf{y}), \tag{2.16} \]
a linear regression finds the matrix $A$ using

$$A = \arg \min_A J(A\tilde{x}). \quad (2.17)$$

If $\{x_i\}, 1 \leq i \leq K$ is a set of $K$ measurements of a physical process with corresponding truth values $\{y_i\}$, then the solution can be obtained by projecting the measurement space onto the truth space using the normal equations [Str98, p. 178].

Let the measurement matrix $X$ be defined as

$$X = \begin{bmatrix}
\tilde{x}_1^T \\
\tilde{x}_2^T \\
\vdots \\
\tilde{x}_K^T
\end{bmatrix}, \quad (2.18)$$

and let the truth matrix $Y$ be defined as

$$Y = \begin{bmatrix}
y_1^T \\
y_2^T \\
\vdots \\
y_K^T
\end{bmatrix}. \quad (2.19)$$

Each element $y_i$ of $y$, is modeled as a linear combination of the elements of $\tilde{x}$, which includes the bias term, by

$$y_i = a_{0i}\tilde{x}_1 + a_{1i}\tilde{x}_2 + \cdots + a_{Ni}\tilde{x}_N, \quad (2.20)$$

or in matrix form as

$$XA = Y. \quad (2.21)$$

The matrix $A$ is typically not invertible, so the normal equations are used to find the solution. If $A$ has dimensionality $m \times n$, then if $n > m$, its solution is given by

$$A = (X^TX)^{-1}X^TY. \quad (2.22)$$
If the distributions for \( x \) and \( y \) have zero mean, then \( (X^TX)^{-1} \) is equivalent to the sample covariance matrix for \( X \), designated \( K_{xx} \). \( X^TY \) is equivalent to the sample cross-covariance matrix for \( X \) and \( Y \), designated \( K_{xy} \). In this case, the solution for \( A \) is written as

\[
A = K_{xx}^{-1}K_{xy}.
\]  

(2.23)

If the means of \( X \) and \( Y \) are not zero, then simply subtract the means of \( X \) and \( Y \) and apply the above result as follows:

\[
A = K_{xx}^{-1}K_{xy}
\]

\[
y = \bar{y} + (x - \bar{x})^TA
\]

(2.24)

2.2.2 Principal Components Analysis

Let \( x \) be a random vector whose covariance matrix is given by \( K_x \). We want to find a vector \( v \) which maximizes the variance of \( x^Tv \). So that the solution is unique, we let \( |v| = 1 \). The covariance matrix for \( x^Tv \) is \( v^TK_xv \). The maximization equation is

\[
v = \text{arg max}_{|v|=1} v^TK_xv,
\]

(2.25)

By using Lagrange multipliers, we discover that \( v \) is the eigenvector which corresponds to the largest eigenvalue of \( K_x \) [Lee00, pp. 24–26].

In general, the principal components correspond to the set of orthonormal eigenvectors of the covariance matrix \( K_x \). The variance captured in each component is the eigenvalue associated with that component. Since the eigenvectors are orthogonal to one another, the principal components are uncorrelated. We write the eigen equation as

\[
Q^TK_xQ = \Lambda_x,
\]

(2.26)

where the columns of the matrix \( Q \) consist of the eigenvectors of \( K_x \) and the covariance matrix of the principal component space \( \Lambda_x \) is diagonal and consists of the eigenvalues of \( K_x \).
2.2.3 Generating Correlated Random Vectors

Given a covariance matrix $K_x$ for a Gaussian random vector $\mathbf{x}$, we wish to generate subsequent $\mathbf{x}$'s using independent scalar random numbers. We generate a vector $\mathbf{y}$ whose elements consist of zero-mean unit-variance independent Gaussian random variables. The correlated random vector $\mathbf{x}$ is given by

$$\mathbf{x} = Q\Lambda^{\frac{1}{2}} \mathbf{y}, \quad (2.27)$$

where $Q$ and $\Lambda$ are obtained from the eigen equation

$$Q^T K_x Q = \Lambda. \quad (2.28)$$

It is easy to show that $\mathbf{x}$ has the desired covariance matrix $K_x$. By definition,

$$E[\mathbf{xx}^T] = E[(Q\Lambda^{\frac{1}{2}} \mathbf{y})(Q\Lambda^{\frac{1}{2}} \mathbf{y})^T] = E[Q\Lambda^{\frac{1}{2}} \mathbf{y} \mathbf{y}^T \Lambda^{\frac{1}{2}} Q^T] = Q\Lambda^{\frac{1}{2}} E[\mathbf{yy}^T] \Lambda^{\frac{1}{2}} Q^T. \quad (2.29)$$

By definition, $E[\mathbf{yy}^T] = I$. Then

$$E[\mathbf{xx}^T] = Q\Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} Q^T = K_x. \quad (2.30)$$

The final step is from Equation 2.28.
Chapter 3

Machine Learning using Neural Networks

3.1 Machine Learning Approach

Definition 1 Assume we have an input distribution $\mathcal{X}$ and an output distribution $\mathcal{Y}$ related by a function $\mathcal{F}: \mathcal{X} \rightarrow \mathcal{Y}$. A Machine Learning Approach is an approach where given a priori knowledge $K$ of the distributions $\mathcal{X}$ and $\mathcal{Y}$ and given data $X$ and $Y$ that is used to represent $\mathcal{X}$ and $\mathcal{Y}$, respectively, a method exists to discover $F : X \rightarrow Y$ from $X, Y,$ and $K$, and $F$ is used to approximate $\mathcal{F}$.

Using a machine learning approach successfully means that $F$ approximates $\mathcal{F}$ well according to some performance function. The resulting $F$ may not be a good approximation of $\mathcal{F}$ due to a number of reasons, e.g. insignificant data, but $F$ still represents our current belief of the function $\mathcal{F}$.

Either the pair $(X, Y)$ must represent the actual process being modeled, or any degradation must be recovered through an increase in a priori knowledge $K$. Given no knowledge $K$, then we require that $(\mathcal{X}, \mathcal{Y}) \subseteq (X, Y)$. In this case, if the method for determining $F$ has an infinite number of degrees of freedom, then all data pairs from $(X, Y)$ can be represented exactly, and therefore at least all members of $(\mathcal{X}, \mathcal{Y})$ are represented exactly. However, if the method has a finite number of degrees of freedom, then all elements in $(X, Y)$ may not be represented exactly, and the deficiency in this set most likely will translate to a deficiency in the representation of the set $(\mathcal{X}, \mathcal{Y})$. $F$ will do its best to represent all data pairs equally.
well. If there are too many extraneous members in the set of data pairs \((X, Y)\), then representation ability of \(F\) is wasted on those data pairs. In summary, \((X, Y)\) should contain enough examples to represent \((\mathcal{X}, \mathcal{Y})\), but few enough so the limited number of degrees of freedom of \(F\) are not wasted.

Typically, an algorithm iteratively solves for \(F\) using some error function \(\epsilon(F(X), Y)\) as a performance measure. The canonical example of a machine learning approach for this thesis is an artificial neural network (ANN) using a mean-squared error criterion.

**Definition 2** The process of iteratively solving for \(F\) is called training \(F\). The set of data pairs \((X, Y)\) is called the set of training examples and this set forms the training set. A set of examples from \((\mathcal{X}, \mathcal{Y})\) used to determine how well \(F\) approximates \(F\) is called the set of test examples and this set forms the test set. The function \(\epsilon(F(X), Y)\) that is to be minimized during training is called the error function.

For example, we create a training set of atmospheric data by direct measurement of the atmospheric phenomenon we wish to relate, and use models of the atmosphere, the earth’s surface, and the instruments involved to simulate what we believe the instrument will see from the direct measurements. The simulated data and the direct measurements form the training set. Using the training set, we train a neural network to approximate the true relationship between what the instrument records in reality and what the actual atmospheric physics are. We can use actual data the instrument records, along with a direct measurement of the atmosphere to create a test set with which we can evaluate the performance of the neural net.

It is possible that \(F\) could be trained so well that it performs well on examples in the training set but poorly on examples similar to those in the training set but not seen during training. An extreme example is if \(F\) simply saved the set of mappings between \(X\) and \(Y\) so that \(\epsilon(F(X), Y) = 0\), but any example not in the training set is set to 0. In this case, our evaluation of \(F\) using the error function on the training set is invalid because it does not generalize to similar examples. We wish to prevent this from happening, and we may do so using a validation set.

**Definition 3** \(F\) is called overtrained if the performance on the training set is not similar to performance on independent examples. \(F\) generalizes well if a set of examples not included
in the training set perform well. A validation set is a set of training examples which are set aside to determine when to stop training in order to prevent overtraining.

During the training process, it is desirable to have a validation that helps determine when to stop training so that $F$ generalizes well. See section 3.8 for more details.

3.1.1 Relative Humidity Retrievals

We cast the problem of estimating relative humidity profiles as a machine learning problem.

We model the process by input physics being viewed by an instrument and recorded as data. Let $W$ represent the relative humidity profile; $T$, all other atmospheric physics that influence the problem; $I$, the influence of the instrument; $D$, the data recorded by the instrument; and $S$ the surface effects. Then the physical process that relates the state of the atmosphere to the quantity we wish to measure is

$$G : (W, T, I, S) \rightarrow D.$$  \hspace{1cm} (3.1)

To estimate the relative humidity profile, we want to model the function

$$F : D \rightarrow W.$$  \hspace{1cm} (3.2)

We have a model $G$ that represents $G$, the radiative transfer equation. We have atmospheric temperature and relative humidity profile data, $T$ and $W$ that represents $T$ and $W$, respectively. We note that $T$ is only temperature information, while $T$ represents not only temperature information, but also other physics that we are ignoring, including ozone, wind, rain, clouds, etc. Since the only information we have about $T$ besides $W$ is $T$, $T$ is used to represent $T$. We also have a model of the instrument, $I$, and a model of the surface, $S$. We can compute the data recorded at the instrument, $D$ using the data and model. This computation is written as

$$G : (W, T, I, S) \rightarrow D.$$  \hspace{1cm} (3.3)

We wish to train

$$F : D \rightarrow W.$$  \hspace{1cm} (3.4)
to approximate $F$. We will do this using neural networks.

### 3.2 Feature Selection

**Definition 4** A feature is a data element that is utilized for estimating the output $y$.

The feature selection process involves 3 steps:

1. Collection of feature candidates,
2. Selecting useful features, and

We collect features that are available and that are relevant to the training of $F$. In the microwave remote sensing case where we want to estimate water vapor, we design an instrument with frequencies that are sensitive to water vapor and temperature changes. A collection of instruments may be available, including ground instruments, so we may consider all such available data as a candidate feature.

#### 3.2.1 Removing Features

We want to discard features that have no impact on the training of $F$. One way to remove features is to rank order them by some criterion, i.e. by a mutual information metric or a physical property. After rank ordering them, we then determine a threshold for discarding features. For example, when we examine the instrument data and see that some channels look like noise due to instrument failure, we decide to discard these dimensions.

#### 3.2.2 Reducing Dimensionality

If we have removed all irrelevant features from the candidate set, and the dimensionality of the input vector would lead to an unacceptable number of weights in the net architecture, then it is necessary to reduce the dimensionality to return to an acceptable number of weights. One easy way to do this is to return to Section 3.2.1 and continue to discard features. If we have noise in the feature vector that is uncorrelated with the signal, then we can use a Principal Components Analysis (PCA), discussed in Section 2.2.2 to reduce the
A nice property of the PCA is that the dimensions of the principal components are uncorrelated, thus to a large extent the signal components will populate different dimensions than the noise components if the signal components are mostly uncorrelated with the noise and if the training set is large. If we can identify the signal components, we can use them as features. Further, the principal components with the largest corresponding eigenvalues will contain the signal and those with the smallest eigenvalues will contain the noise. In this case, the scree plot, a plot of the ordered eigenvalues, will contain an easily identifiable ridge that separates the noise floor from the signal.

Figure 3-1 shows an example of a scree plot on a data set. Notice the point where the eigenvalues rise above the noise floor. In this case, we can safely throw away PCs whose eigenvalues lie below the noise floor.

If we still retain too many dimensions after discarding noise dimensions, then signal dimensions must be discarded. Figure 3-2 shows the loss of energy as a function of discarding dimensions.

The procedure for dimensionality reduction using PCA is as follows.

1. Normalize the data in one of two ways:

   - Divide each dimension by its standard deviation. This is the standard technique performed on data whose noise is smaller than the signal.
   - Obtain the noise variance using ION[Lee00] or another noise estimation technique and divide each channel by the standard deviation of the noise. This technique will arrange the dimensions according to signal-to-noise ratio instead
of data variance. This ordering is more meaningful when the noise that is present in the data has similar variance to the signal.

2. Perform the PCA on the normalized feature vectors.

3. Examine the scree plot and identify the ridge which forms the boundary between the signal and noise.

4. Safely discard all noise dimensions with minimal loss of information.

5. Discard signal dimensions if necessary with significant loss of information.

3.3 Neural Network Definition

Neural networks are versatile structures that are used for both classification and estimation problems. A nice property of neural networks is that they can learn non-linear functions of data with arbitrary complexity. We use neural networks to learn the inverse mapping from water vapor to radiation.

3.3.1 A Single Neuron

A neural network is comprised of individual neurons whose basic function is to linearly combine the input and pass it through a non-linear warping function. If $\mathbf{x}$ is an input vector, $\mathbf{w}$ is a weight vector, $b$ is a bias scalar, and $y = f(\cdot)$ is the warping function, then
the output $y$ is obtained by the equation

$$y = f(w'x + b)$$

(3.5)

(This has the form of a generalized additive model [DHS01, p. 306].)

3.3.2 Non-linearity

Before we train a neuron, we initialize $w$ and $b$ by setting them to small random numbers between $[-\frac{1}{N}, \frac{1}{N}]$. The region in the input space of $f$ where most of the inputs from a training set lie is called the active region of $f$. If the input is normalized to zero mean and unit variance, then the active region for an initialized net falls near the origin of the input to $f$. If $f$ is a sigmoid, then by a Taylor’s series expansion, the active region is where the linear term is dominant. Thus a network built of individual units that operate in the linear region will be linear.

An alternative to random uniform initialization is the Nuygen-Widrow method, which chooses initial weights to place the input space in the transfer functions’ active region. The details for this algorithm are given in [NW89].

During training, the training algorithm updates the weights $w$ so that the output of the network converges to the desired output according to some convergence criterion, e.g. mean-squared error. As weights are adjusted, the active region widens, and thus more non-linear terms in the Taylor’s series expansion of $f$ become significant. Neurons begin the training as a linear transfer function, and (on average) gradually become more non-linear until training has stopped.
3.3.3 Multiple Neurons

There are several ways of combining neurons to form a network. We consider the most common way, called a multi-layer feed-forward neural network (MFNN). The definition of a MFNN is given below.

We connect individual neurons into a network as in figure 3-4. The input vector $x$ is fed simultaneously into $N$ neurons, forming the first layer. The first layer is called a non-linear layer because the activation functions for the neurons within the layer are sigmoidal. The output of the first layer connects to the input of the second layer. Similarly, the output of the second layer connects to the input of the final layer. The final layer is called a linear layer because the activation functions are linear. The output of the final layer is the output of the network. Since the output from one layer is fed only into the input of the next layer, the network is a MFNN.

A layer is a collection of neurons that share an input. If the layer is directly connected to an output, then it is called an output layer, otherwise it is called a hidden layer.

The collective set of weights from this set of neurons forms a matrix $W = \{W_{ij}\}$, where the $i$th row is the weight vector for the $i$th neuron. Similarly, the set of biases forms a vector $b = \{b_i\}$, and the set of transfer functions is $\{f_i\}$. The input to the layer is

$$v = Wx + b,$$  \hspace{1cm} (3.6)

and the output for the $i$th node of the layer is then

$$y_i = f_i(v_i).$$  \hspace{1cm} (3.7)

The output for the entire layer can be represented by the vector function

$$y = f(v) = \begin{bmatrix} f_1(v_1) \\ f_2(v_2) \\ \vdots \\ f_N(v_N) \end{bmatrix}.$$  \hspace{1cm} (3.8)

We combine several layers to obtain the network shown in Figure 3-4. This composition
Figure 3-4: Multilayer neural network.

takes the form

\[ y = f_2(W_2(f_1(W_1 x + b_1)) + b_2) \]  

(3.9)

In Equation 3.9, if \( \{f_1\} \) is the non-linear tansig function defined as

\[ y = \frac{2}{1 + \exp(-2x)} - 1, \]  

(3.10)

and \( f_2 \) is the linear function defined by \( y = x \), then the neural network can approximate any function to within a finite number of discontinuities arbitrarily well, given a significant number of nodes in the hidden layer. This is seen by a Taylor's series expansion of the network, which can include cross-terms of any combination and degree. A set of weights for the net can be chosen so that the coefficients of all the terms of the expansion match the function of interest. For further discussion, see [NW90] and [DHS01, pp. 287-288].

### 3.4 Training Set Normalization

The training data must be normalized so that when the network is initialized, the active region for \( f \) is linear. There are several normalization techniques. The one utilized in
This thesis is a zero-mean, unit-variance normalization. This is appropriate because the input distribution is not bounded, and we know its mean and variance, thus the maximum entropy assumption for the input distribution is to assume it is Gaussian with first and second moments equivalent to the mean and variance.

We contrast this technique with a min-max normalization that scales and shifts the distributions so that the minimum is $-1$ and the maximum is $1$. The latter technique is appropriate when the input distribution is naturally bounded on both sides. Had the latter technique been applied to a Gaussian-distributed input, then an outlier of the Gaussian distribution would cause an anomaly which would affect the training results.

### 3.5 Training Algorithms

Training is the process of discovering the functional relationship between $X$ and $Y$, $F : X \rightarrow Y$, using an iterative algorithm. There are various techniques for solving for the weights of a Neural Network. The training technique that we use in this work is gradient descent by backpropagation of error.

#### 3.5.1 Gradient Descent

Gradient descent is an iterative technique that attempts to find the minimum of an error function given an initial point. Most gradient descent techniques are only locally optimal, so the final solution depends on the initial point chosen. If $J(x)$ is the error function we wish to optimize, then a gradient descent approach works by taking an initial guess $x(0)$ and improving that guess by the following update equation:

$$x(n + 1) = x(n) - \delta \nabla J(x(n)),$$

where $\delta$ is a parameter called the learning rate, which controls the rate of convergence, and $\nabla$ is the gradient operator, in this case applied to $J$ with respect to the parameters $x$ at time $n$.

A neural network error function can be written as $J(x, \theta)$, where $x$ is the input and $\theta = \{W_1, \ldots, W_N, b_{N_1}, \ldots, b_N\}$, $W_i$ and $b_i$ are the weight matrix and the bias vector for the input to the $i$th layer, and $N$ is the number of layers in the net. Standard gradient
descent begins with an initial set of parameters $\theta(0)$ and updates $\theta$ by

$$\theta(n + 1) = \theta(n) - \delta\nabla_\theta J(x, \theta), \quad (3.12)$$

where $\nabla_\theta$ is the gradient with respect to the parameter $\theta$.

### 3.5.2 Backpropagation

Backpropagation is a common technique used to train neural networks. While the standard gradient descent algorithm does find a minimum, it takes considerable compute power. A more efficient way to compute the gradient is to use the chain rule to avoid redundant computation in the gradient descent. The backpropagation technique described here is not the most efficient algorithm to use, but it is the most intuitive to explain. Given the number of weights in the networks trained here, the backpropagation, while not that efficient, is the only one which would train these nets with the limited amount of memory in the computers.

We will first focus on the update of the weight matrix for a single layer, and then explain how the backpropagation works in a multi-layer network.

Let $z_i$ be the output of the $i$th neuron for a particular layer. The partial derivative between the error function of the network and a particular weight in this layer is $\frac{\partial J}{\partial w_{ij}}$. Using the chain rule, we rewrite the gradient in terms of the output for this layer and obtain

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial f} \frac{\partial f}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}}. \quad (3.13)$$

There are two ways to update the parameters given a training set. First, the gradient
can be computed on all examples from the training set and the updates to the parameters associated with each example can be applied simultaneously to update the weights at each iteration. The total update is computed by a weighted combination of the individual updates; typically all examples are equally weighted. Alternatively, the gradient for a single example can be computed and applied at each iteration, cycling through the examples until training has finished. The former technique is called batch training, while the latter is called adaptive training. In the following equations, assume that either types of updates could be used.

For a multi-layer net, we begin at the output and successively apply the chain rule through the beginning of the net. For example, Figure 3-5 shows a net with two hidden layers and one output layer. Working backwards, the update rule for $W_3$ is:

$$\begin{align*}
W_3(n+1) &= W_3(n) - \delta \nabla W_3(x, \theta) \\
\nabla W_3 &= \frac{\partial J}{\partial W_3} 
\end{align*}$$

The update rule for $W_2$ is:

$$\begin{align*}
W_2(n+1) &= W_2(n) - \delta \nabla W_2(x, \theta) \\
\nabla W_2 &= \frac{\partial J}{\partial W_2} \frac{\partial \delta^{(3)}}{\partial W_2} 
\end{align*}$$

The update rule for $W_1$ is:

$$\begin{align*}
W_1(n+1) &= W_1(n) - \delta \nabla W_1(x, \theta) \\
\nabla W_1 &= \frac{\partial J}{\partial W_1} \frac{\partial \delta^{(2)}}{\partial W_1} \frac{\partial \delta^{(1)}}{\partial W_1} 
\end{align*}$$

Deriving the update rules for the biases is similar.

### 3.5.3 Momentum

One improvement to the standard gradient descent algorithm is momentum. When using momentum, the update equation is changed to

$$\begin{align*}
\Delta(n) &= \alpha \delta \nabla J(x(n)) + (1 - \alpha) \Delta(n - 1) \\
x(n+1) &= x(n) - \Delta(n)
\end{align*}$$

44
where $\alpha \in [0, 1]$ is the momentum parameter. Momentum gives some weight to the previous direction of the update. If the error surface is noisy, the net may get stuck within a wrinkle of the error surface. Momentum allows the training algorithm to ignore some of the randomness in the error surface.

### 3.5.4 Adaptive Learning Rate

One problem with gradient descent is that the learning parameter $\delta$ is fixed. A fixed learning rate is not optimal throughout the entire training. The solution to this problem is to let the algorithm adjust the learning rate so that optimal learning can be maintained.

The learning rate update rule has 3 parts:

1. If the performance in one iteration decreases from the previous iteration, the algorithm decreases the learning rate by a specific factor.

2. If the ratio between the performance in one iteration from the previous iteration increases by more than a pre-determined threshold, the algorithm increases the learning rate by a specific factor.

3. If neither of these criterion are met, then the algorithm does not update the learning rate.

For example, the learning rate update rule used in the MATLAB implementation of the training algorithm that is used in this thesis can be expressed as

$$
\delta(n + 1) = \begin{cases} 
1.1\delta(n) & \text{if } J(n) - J(n - 1) < 0, \\
0.7\delta(n) & \text{if } J(n)/J(n - 1) > 1.2, \\
\delta(n) & \text{otherwise.}
\end{cases}
$$

### 3.5.5 Speed of Convergence

Using gradient descent with backpropagation, adaptive learning rate, and momentum, we see a significant increase in the rate of convergence of the algorithm. As an example, a neural network in Chapter 5 trained using all of the techniques discussed so far converged to a certain performance level in approximately 100,000 iterations, while the same net using only gradient descent with backpropagation converged to the same performance level in approximately 170,000 iterations.
3.5.6 A priori Bias

A priori knowledge can be incorporated into the network training process to speed up training. This can be done by weighting the cost function \( J(x, \theta) \). Weighting specific derivative terms in the backpropagation algorithm can make a similar modification. The reason for doing the latter is that the implementation is easier and the result is the same.

The backpropagation algorithm updates weights according to the gradient of the output. This gradient term is effectively weighting the error with respect to each weight in the net. Using the chain rule, the gradient term is

\[
\begin{align*}
W_2(n + 1) &= W_2(n) - \delta \nabla W_2(x, \theta) \\
\nabla W_2 &= \frac{\partial J}{\partial \theta} \frac{\partial f}{\partial z}.
\end{align*}
\]

(3.19)

We will focus on the term \( \frac{\partial f}{\partial z} \). Each element of the vector can be written as

\[
\frac{\partial f_i}{\partial z_i} = \frac{\partial f_i(y)}{\partial y} \frac{\partial y}{\partial z_i}.
\]

(3.20)

We notice that the term \( \frac{\partial f_i(y)}{\partial y} \) is present in the update equation for every weight in the network and is specific to only the \( i \)th output. We can modify this term by replacing this derivative by a function of the derivative. The modified backpropagation rule becomes

\[
\nabla W_2 = \frac{\partial J}{\partial \theta} g \left( \frac{\partial f_i(y)}{\partial y} \right) \frac{\partial y}{\partial z_i} \frac{\partial z}{\partial W}.
\]

(3.21)

**Relative Humidity Example** Relative humidity values lie between 0 and 1. Values of relative humidity for super-saturated air will lie outside this range, but this will be ignored in this example. Since the output layers of the networks we use in this thesis are linear,

\[
\frac{\partial f_i(y)}{\partial y} = 1.
\]

(3.22)

To aid the training algorithm, we use the function

\[
g \left( \frac{\partial f_i(y)}{\partial y} \right) = \begin{cases} 
1 & y \in [0, 1] \\
2 & \text{otherwise}
\end{cases}
\]

(3.23)
This function provides the normal correction to weight updates due to errors from relative humidity values within the known distribution of \([0, 1]\) and twice the normal correction when the errors are due to relative humidity values outside of \([0, 1]\). The mechanism helps the training algorithm to focus on unreasonable output first, and then, once all values are reasonable, it focuses equally on all outputs.

3.6 Designing Network Architecture

Training Data and Network Size

Careful consideration must be given to how many degrees of freedom are chosen when designing the architecture of the neural network. For the purpose of this discussion, assume we have a net with one hidden and one output layer. The training data typically consists of the signal that we are modeling and various forms of additive, multiplicative, and other noise. We must choose the number of degrees of freedom in the net to allow accurate representation of the training set, but we must avoid having too many degrees of freedom, otherwise we will also represent the outliers.

If the number of weights is much larger than the number of training examples, then the network will not have enough examples to properly train the network. We can understand this by recalling that a system with \(N\) equations and \(M > N\) unknowns has multiple solutions. In a network, this translates to multiple global optima for the weights. Thus the set of weights that the training algorithm has converged to will not necessarily be the true global optimum.

Choosing the Number of Layers While one hidden layer suffices for arbitrarily good approximating power, Cabrera noticed that two hidden layers can increase training speed and reduce the number of weights [CM93]. The back-propagation algorithm is more efficient because a significant number of calculations can be saved.

Choosing the Number of Nodes First, we need enough degrees of freedom to model the relationship between the training set \(X\) and the truth \(Y\). Having too few nodes will force the training algorithm to smooth the data resulting in an increased prediction error. Second, we must not have too many degrees of freedom relative to the number of training
examples. A good rule of thumb is more than 3 training examples for each weight in the network. If the dimensionality of each training example is increased, then the number of training examples per weight decreases proportionally. In general, choosing the correct topology for a problem is considered an art form that requires experience or it is addressed by trial and error.

3.7 Noise Averaging

When the generation of a training set consists of steps where we add noise to some noise-free data, we may benefit from noise averaging during training. Noise-averaging is implemented by generating a different noise vector at each iteration during training. The resulting net will be more robust than without noise averaging. The error on the training set may increase, but the added robustness can translate to a lower error on the test set.

3.7.1 Justification

Some training set distributions \( T \) take the form \( T = S + N \) where \( S \) is the fixed, noise-free part of the training set and \( N \) is the noise distribution. We can produce a fixed training set \( T_f \) by generating one instance of the noise vector \( N_f \) before training begins, and keep the training data constant throughout the training process. This will work in the case where the number of training examples is sufficiently large. However, in the case where the noise model is complex, the number of training examples required to fully represent the noise increases dramatically. \( N_f \) will not necessarily be representative of the noise distribution \( N \). If this happens, then \( T_f \) will not be representative of \( T \). Figure 3-6 shows the effect.

By varying the noise at each iteration, the distributions converge:

\[
\bigcup_{i=1}^{M} F_{T_f}^i \xrightarrow{\text{M}} \infty F_T \quad (3.24)
\]

3.7.2 Robust Noise Description

We require the network to be robust when faced with a true noise distribution \( N \) that does not match our noise distribution \( N_f \). Figure 3-7 shows that our knowledge of \( N \) may not properly represent \( N_f \).
Figure 3-6: A deficient fixed training set.

Figure 3-7: A conservative noise model for a robust network.
To account for this difference, we add a margin of error $\Delta$ to the noise model. The network finds a set of weights that performs well in the face of the noise bound, so if the actual noise falls within this bound, the net will perform well.

### 3.7.3 Effective Increase in Training Set Size

Another benefit of noise averaging is that the effective size of the training set is increased by an amount depending on how significant the noise is. If the training set is small relative to the size of the network, then this factor may provide enough variation to specify the relatively large set of weights.

In the case of relative humidity retrievals, we model the surface emissivity by a random vector. The model is robust because the correlation assumed for the model is lower than the actual correlation, and the emissivity values are constrained within the values of 0.35-1, which is conservative. We consider the surface emissivity to be a multiplicative noise to a set of radiative transfer values. The noise is varied during training, and the resulting network is capable of acting reasonably in all surface situations tested.

### 3.8 Regularization

Regularization is a technique to aid the training algorithm in producing a network that generalizes well to unseen data. Using the notation from Section 3.1, we motivate the need for regularization by observing that if the network has enough degrees of freedom, then the resulting functional relationship learned $F : X \rightarrow Y$, will not only model the general relationship $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$, but it will also model anomalies specific to the training set $X$. This can occur if the network has too many parameters to fit, as discussed in Section 3.6. In this case, the network will use the extra degrees of freedom to model outliers in the data.

The extra degrees of freedom can model outliers in the training set. For instance, if the noise added to the training set contains a rare event, such as a 5-standard-deviation value from a Gaussian noise model, then that event may become a significant feature that the net will model with extra degrees of freedom.

There are two methods for regularizing the data used in this work. The first method is to reduce the number of parameters in the network, which reduces the ability of the network to allocate degrees of freedom to outliers. The second is to use a validation set to detect...
when the network is losing its generalization ability, as stated in Definition 3. As training proceeds, the weights are adjusted to decrease the error on the training set. An assumption that decreasing the error on the training set will produce a decrease in the error on an independent test set can be validated. A net is worthless if it produces zero error on the training set but performs terribly when given an example not seen during training.

One method of helping a network to generalize to unseen data is to set aside a few training examples from the training set. These examples will be used as a validation set, and they will not be used by the backpropagation algorithm to update the weights. These examples are unseen to the network throughout training, and thus can act as a set of "test" examples during training. At each iteration, the weights are updated using the training set and then the error is evaluated on the validation set. If a decrease in the error on the training set produces a corresponding decrease in the error on the validation set, then the network remains robust and the training continues. However, if the error on the training set continues to decrease but the error on the validation set begins to increase, then the network is being tuned to the training set and the network is losing its generalization ability. The best set of weights from the viewpoint of generalization is the one that produced the minimum on the validation set.
Chapter 4

Data

4.1 Introduction

This chapter describes the data sets used in this work. First, we discuss the production of the training set from the SATIGR radiosonde set. Then, we describe the microwave and infrared flight data which we will test on.

4.2 The SATIGR Radiosonde Collection

A radiosonde is a rocket or weather balloon that is launched to collect water vapor, temperature, ozone, wind speed, and other information in the atmosphere. The word RAOB will be used throughout this work, which stands for RAdiosonde OBservation.

The SATIGR data set consists of 1761 atmospheric profiles over 40 pressure levels between 0.05 and 1013 mbar. Each atmospheric profile contains temperature, water vapor density, and ozone readings. Figures 4-1 and 4-2 show the distribution of the temperature and the relative humidity, respectively. Notice that the relative humidity distribution has values that far exceed 1. The atmospheres are distributed over multiple seasons and climates, but there is a concentration in the Polar Regions during winter. Table 4.1 gives the distribution across climates. The surface temperature is equivalent to the temperature reading at 1013 mbar.
Table 4.1: Distribution of SATIGR throughout the climates.

<table>
<thead>
<tr>
<th>Climate type</th>
<th>No. of RAOBs</th>
</tr>
</thead>
<tbody>
<tr>
<td>tropical type atmospheres</td>
<td>322</td>
</tr>
<tr>
<td>mid latitude-1 type atmospheres</td>
<td>388</td>
</tr>
<tr>
<td>mid latitude-2 type atmospheres</td>
<td>354</td>
</tr>
<tr>
<td>polar-1 type atmospheres</td>
<td>104</td>
</tr>
<tr>
<td>polar-2 type atmospheres</td>
<td>593</td>
</tr>
</tbody>
</table>

Figure 4-1: SATIGR temperature distribution.
4.2.1 Water Vapor Problem

SATIGR contains unreasonable water vapor values for many low pressure and low temperature readings, about 1/3 of the data set. The problem can be seen by a scatter plot of the relative humidity values vs. temperature, as seen in Figure 4-3. Notice that as the temperature drops below 240 °K, the number of erroneous reading and the magnitude of the errors increase. Figure 4-4 shows a scatter plot of the relative humidity values vs. the pressure. Most of the erroneous readings occur in the lower pressure levels.

This problem is not due to the definition of relative humidity, as discussed in Chapter 2.1.4. An analysis was done on two cases. The first case is the definition of relative humidity assuming spontaneous ice formation below 233.16 °K. The second case does not assume any spontaneous ice. The partial pressure over water and ice differ. Each case contains a similar number of unreasonable values even though the two methods differ.

4.2.2 Modifying SATIGR

In order to use SATIGR for training a neural network, we must simulate a set of brightness temperature vectors using the radiative transfer code. A modification to the relative humidity problem taken in previous work was to simply clip relative humidity values for the truth
Figure 4-3: Distribution of relative humidity vs. temperature.

Figure 4-4: Distribution of relative humidity vs. pressure.
Y of the training set to 1. However, the simulated brightness temperatures were based on the original water vapor values, while the relative humidity values had been altered. This creates a distortion in the training set.

We want to modify SATIGR so that both the simulated brightness temperatures and the truth reflect this change in relative humidity. We do this by first computing relative humidity values from the temperature, pressure, and water vapor density information in the data set. Then, we clip the relative humidity values to be in the range \([0, 1]\). We perform the inverse calculation to convert relative humidity back to water vapor density. Finally, we simulate new brightness temperatures so that our data set is consistent.

### 4.3 CAMEX-3 Flight Data

The Third Convection and Moisture EXperiment (CAMEX-3) was flown during the summer of 1998 to gather data for development of moisture and hurricane estimation and prediction algorithms. One of the aircraft flown during CAMEX-3 was the ER-2, which carried the three instruments whose data we design retrievals for in this thesis. The three instruments are the NPOESS Aircraft Sounder Testbed for Microwave (NAST-M), the Millimeter-wave Imaging Radiometer (MIR) for microwave, and the NPOESS Airborne Sounder Testbed Interferometer (NAST-I) for infrared.

#### 4.3.1 ER-2

The ER-2 aircraft, shown in Figure 4-5, flew at approximately 20 km at a speed of 200 m/s. The data that the ER-2 gathered contains artifacts from the flight pattern, such as not being at full altitude during take-off and landing, fluctuation of altitude throughout the flight, and turns during the flight, which must be removed from the data before analyzing retrievals. Flight patterns for all CAMEX-3 flights were superimposed on GOES satellite imagery [Whi01]. From this data, we are able to identify clouds, land, and aircraft turns. Navigational data from the ER-2 allows us to determine when the aircraft is at altitude.

#### 4.3.2 RAOBs

There were several weather balloons radiosonde observations, or RAOBs, that were launched during the CAMEX-3 flights. The RAOBs directly measured the temperature and water
vapor over Andros Island for comparison with retrieval algorithms. Since there are several groups who launched RAOBs during this mission, we choose one set that we can use for comparison. The Wallops Island RAOBs were available and were sufficient for our comparison [Sch98].

4.3.3 Microwave Flight Data

NAST-M Data

The NAST-M instrument scans the surface every $7.2^\circ$ for 15 angles per scan, and observes a hot- and cold-load and sky calibration sources each scan. The limb scan angle is $64.8^\circ$. Each scan was completed in 5.5 seconds. The data was calibrated using a 3-point calibration. To smooth calibration noise, several calibration loads were combined with a weighted average using a triangle filter. Some of the calibration issues are quite complex and are dealt with nicely by Leslie et al. [Les00]. Figure 4-7 and Figure 4-6 show the 118 GHz and 54 GHz channels, respectively, from a section of data taken over Andros Island on Sept 13.
Figure 4-6: Example of NAST-M 54 GHz data over Andros Island.
Figure 4-7: Example of NAST-M 118 GHz data over Andros Island.
MIR Data

The MIR instrument scanned the surface every 1.79° for 57 angles per scan, observing a hot- and cold-load calibration source at each scan. The limb scan angle is 50 degrees. Each scan is completed in 3 seconds. The antenna beamwidth varies between 3.2 and 3.6 degrees. The data is calibrated using a 2-point calibration. A more detailed description of the MIR data is found in [W+95]. Figure 4-8 shows data from the same Andros Island pass as Figures 4-7 and 4-6 above for the six MIR channels operational during CAMEX.

Combining NAST-M and MIR

In forming a combined data set for the two instruments, we first align the MIR data to the NAST-M data, and then remove a bias inherent in the data.

We first crop the 57.6° and 64.8° scan angles from the NAST-M data which are outside of MIR’s scan pattern. We down-sample MIR by first spatially filtering it with a convolution kernel specified by

$$
\begin{pmatrix}
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9
\end{pmatrix}.
$$

(4.1)

For the edges, we re-normalize the values by multiplying by $\frac{3}{2}$, because 3 of the 9 kernel spots on an edge are set to zero; for corners, we multiply by a factor of $\frac{9}{4}$ using similar reasoning. This filter is appropriate because the observations are approximately decorrelated every 3 spots, so this filter will prevent aliasing. An advantage we gain from averaging 9 spots is that we obtain an rms noise-reduction factor of 3, which will be utilized in the instrument models in Chapters 5 and 6.

After spatial filtering, we then linearly interpolate MIR data in the single-scan direction to the scan angles of NAST-M. Since the MIR scan direction is opposite to the NAST-M scan-direction, we reverse the scan angle indices. We linearly interpolate MIR in the direction of the flight to the NAST positions by using the timing information recorded by NAST-M and MIR.

Note that the NAST-M data is not averaged, but both it and the MIR data could be further averaged to improve the signal-to-noise ratio.
Figure 4-8: Example of MIR data over Andros Island.
Bias Removal

Inherent in the NAST-M and MIR data is a bias between the flight data and data simulated using the radiative transfer equation that was used to train our estimator. The bias could be due to a limitation in the calibration technique, or because there is an inherent difference between the radiative transfer model and the true physics.

To remove this bias, we use the microwave radiative transfer code explained in Section 5.1.2 to simulate the brightness temperatures that correspond to each RAOB $TB_{\text{sim},j}$. We compare this simulated brightness temperature data with the brightness temperatures that were observed near the RAOB $TB_{\text{flight},i}$ for flight $i$. We compute a bias for each flight,

$$b_i = TB_{\text{flight},i} - \frac{1}{N} \sum_{j=1}^{N} TB_{\text{sim},j},$$

where $N$ is the number of RAOBs for a particular flight, and average over all flights using

$$b = \frac{1}{M} \sum_{i=1}^{M} b_i,$$

where $M$ is the number of flights.

In using this technique, we hope that accidental differences from spatial or temporal displacement between the flight portion and the RAOBs and weather-related anomalies are reduced by averaging. We compute the bias from two flights and average them together to obtain the final bias. One RAOBs was used in the computation of the bias for August 8th; and six for September 13th. Figures 4-9 and 4-10 show the biases computed for each flight. The final bias is given in Figure 4-11 and is removed from all of the flight data.

4.3.4 Infrared Flight Data

The NAST-I instrument scans the surface every 7.4° for 13 angles per scan, and each scan is completed in 12.36 seconds. The limb scan angle is 48°. The image in Figure 4-12 shows the 9000 channels at nadir from the section of the flight on September 13th taken over Andros Island. This flight section is the same section that was presented above in Figures 4-7 and 4-6 for NAST-M, and Figure 4-8 for MIR.
Figure 4-9: A comparison between simulated RAOB and flight brightness temperatures for August 8th.

Figure 4-10: A comparison between simulated RAOB and flight brightness temperatures for September 13th.
Figure 4-11: The computed bias that is removed from the microwave data.
All Channels are zero-mean, unit-variance normalized.

Figure 4-12: Example of NAST-I data over Andros Island.
Chapter 5

Microwave Retrieval Algorithm

The derivation of the water-vapor function from the radiative transfer equation is non-linear. A solution can be obtained either by using physical knowledge of the problem, or by training a neural network to learn that physics given a sufficient set of examples. In this thesis, the latter approach is used.

5.1 Brightness Temperature Simulation

We begin implementing our algorithm by producing a data set with which we can train our net. Specifically, we take each RAOB vector in the SATIGR data set and produce a brightness temperature vector that corresponds to what our instrument would see when recording the flight data. This requires two steps.

1. An instrument model is designed using the instrument characteristics. It describes the antenna and frequency weighting functions for each channel and the scan angles for non-nadir observations is designed using the instrument characteristics.

2. Then, the discrete radiative transfer algorithm discussed in Section 2.1.3 uses the instrument model to simulate three atmospheric parameters for each channel and scan angle.

The surface parameters are computed on the fly at each iteration of the neural network training and combined with the pre-computed atmospheric parameters to produce a set of output brightness temperatures. Pre-computing the atmospheric parameters makes sense because the atmospheric parameters remain constant throughout training. The surface
parameters are described by a stochastic model and the technique of noise-averaging from Section 3.7 is used, therefore they cannot be pre-computed.

5.1.1 Instrument Models

The instrument models must be sufficiently accurate that the simulated brightness temperatures represent the observed brightness temperatures. Since MIR and NAST-M are physically two separate instruments, they are modeled separately.

For each instrument, the frequency and antenna weighting functions are described. Each angle/frequency pair from these weighting functions will be simulated to produce three atmospheric parameters. The set of parameters-triplets are combined using weighting functions and the trapezoidal rule for integration.

MIR Instrument Model

The MIR instrument has 6 double-sideband channels whose characteristics are summarized in Table 5.1[W+95]. Four of the channels measure the 183-GHz and 220 GHz water-vapor lines. The optical depths of the remaining two channels allow them to see the surface.

Frequency Weighting Function Figure 5-1 displays the model of the frequency weighting function. We model each channel as a pair of rectangular frequency passbands consisting of 75 equally-spaced and equally-weighted point-frequencies per sideband. The width of each passband is assigned the value of the bandwidth given in Table 5.1.

Antenna Weighting Function Figure 5-2 shows the one-dimensional antenna pattern weighting function. The values for the antenna gain pattern are obtained from a Gaussian density sampled at 11 equally-spaced points from $-4\sigma$ to $4\sigma$ and normalized so the weights sum to unity. The $\pm 4\sigma$ region under a Gaussian curve contains 99% of its total energy. We use the published value of $\sigma = 3.5^\circ$ for MIR’s antenna pattern.

NAST-M Instrument Model

More information is available on the NAST-M instrument’s characteristics, so we design a more detailed model.
Figure 5-1: MIR frequency passband model.

Table 5.1: Spectral description of the MIR system.

<table>
<thead>
<tr>
<th>MIR System DSB</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chan.</td>
<td>RF (GHz)</td>
<td>BW (GHz)</td>
</tr>
<tr>
<td>1</td>
<td>89 ± 1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>150 ± 1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>183.3 ± 1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>183.3 ± 3</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>183.3 ± 7</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>220 ± 2.5</td>
<td>3</td>
</tr>
</tbody>
</table>
NAST-M consists of 2 groups of channels. The first group consists of 8 single-sideband channels and measures the 54-GHz oxygen line. The other group consists of 9 double-sideband channels and measures the 118-GHz oxygen line. Channel 9 was not operational during CAMEX-3, so is omitted from further discussion. The characteristics of these two channel sets are summarized in Table 5.2 and Table 5.3.

**Frequency Weighting Function** During WINTEX, the IF passbands for NAST-M were measured and recorded, and are used for the frequency weighting function. Figures 5-3

### Table 5.2: Spectral description of the NAST-M 54-GHz system.

<table>
<thead>
<tr>
<th>Chan.</th>
<th>RF (GHz)</th>
<th>IF (MHz)</th>
<th>BW (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50.21-50.39</td>
<td>4210-4390</td>
<td>180</td>
</tr>
<tr>
<td>2</td>
<td>51.56-51.96</td>
<td>5560-5960</td>
<td>400</td>
</tr>
<tr>
<td>3</td>
<td>52.6-53</td>
<td>6600-7000</td>
<td>400</td>
</tr>
<tr>
<td>4</td>
<td>53.63-53.87</td>
<td>7630-7870</td>
<td>240</td>
</tr>
<tr>
<td>5</td>
<td>54.2-54.6</td>
<td>8200-8600</td>
<td>400</td>
</tr>
<tr>
<td>6</td>
<td>54.74-55.14</td>
<td>8740-9140</td>
<td>400</td>
</tr>
<tr>
<td>7</td>
<td>55.335-55.665</td>
<td>9335-9665</td>
<td>330</td>
</tr>
<tr>
<td>8</td>
<td>55.885-56.155</td>
<td>9885-10155</td>
<td>270</td>
</tr>
</tbody>
</table>
Table 5.3: Spectral description of the NAST-M 118-GHz system.

<table>
<thead>
<tr>
<th>Channel</th>
<th>LSB (GHz)</th>
<th>USB (GHz)</th>
<th>IF (MHz)</th>
<th>BW (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>114.75-115.75</td>
<td>121.75-122.75</td>
<td>3000-4000</td>
<td>1000</td>
</tr>
<tr>
<td>2</td>
<td>115.95-116.45</td>
<td>121.05-121.55</td>
<td>2300-2800</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>116.45-116.95</td>
<td>120.55-121.05</td>
<td>1800-2300</td>
<td>500</td>
</tr>
<tr>
<td>4</td>
<td>116.95-117.35</td>
<td>120.15-120.55</td>
<td>1400-1800</td>
<td>400</td>
</tr>
<tr>
<td>5</td>
<td>117.35-117.75</td>
<td>119.75-120.15</td>
<td>1000-1400</td>
<td>400</td>
</tr>
<tr>
<td>6</td>
<td>117.75-118.15</td>
<td>119.35-119.75</td>
<td>600-1000</td>
<td>400</td>
</tr>
<tr>
<td>7</td>
<td>118.15-118.45</td>
<td>119.05-119.35</td>
<td>300-600</td>
<td>300</td>
</tr>
<tr>
<td>8</td>
<td>118.45-118.58</td>
<td>118.92-119.05</td>
<td>170-300</td>
<td>130</td>
</tr>
</tbody>
</table>

and 5-4 show these passband measurements against the absorption curve for oxygen.

**Antenna Weighting Function** The antenna weighting function for NAST-M is identical in structure to that which was used for MIR in (Figure 5-2), except the value of $\sigma$ is 7.5 degrees.

**5.1.2 Microwave Simulation Code**

Given an instrument model and the SATIGR data set, we simulate brightness temperatures as seen by our instrument in simulation space. The entire simulation assumes a planar-stratified atmosphere. We first compute the absorption due to oxygen and water vapor for a given RAOB. Using this and the temperature information of the RAOB, we compute the transmittance of the atmosphere. Using the radiative transfer algorithm, we produce the three atmospheric coefficients: the downwelling component, the upwelling component, and the one-way transmittance. These three terms were described in Section 2.1.3.

**Oxygen and Water Vapor Absorption Coefficients**

We compute the absorption due to oxygen and water vapor molecules in the atmosphere with the current model provided in Appendix A [Ros93, LRH92]. This vector is computed for each of the point-frequencies of the instrument model. For a path through the atmosphere other than nadir, the distance through each layer increases by a factor equivalent to the secant of the angle. Absorption is proportional to the path through the atmosphere, so absorption also increases by the same factor.
Figure 5-3: NAST-M 54 GHz frequency passbands.

Figure 5-4: NAST-M 118 GHz frequency passbands.
Radiative Transfer Coefficient Calculation

The three atmospheric coefficients are calculated using the discrete radiative transfer algorithm by computing parallel paths. We assume that the aircraft flies at about the 50-mbar pressure level for the simulation. The code, implemented in MATLAB, is located in Appendix A.

5.2 Training the Neural Network Estimator

We wish to train a neural network on brightness temperature data that has a distribution similar to the flight data. To do this, we must specify the training set, define the distributions of the surface parameters, define the architecture of the network, and finally, train the net. Afterwards we evaluate the performance of the net with respect to the training and validation performance.

5.2.1 Surface Model

Section 5.1.2 describes how to calculate the three coefficients to the radiative transfer model which will be used in training the neural net. Equation 2.12 states that we must combine these coefficients with a surface temperature, $T_{\text{surf}}$, and surface emissivity, $\epsilon_{\text{surf}}$ in order to obtain the brightness temperature the instrument measures. We define a distribution which describes these two parameters.

Surface Temperature Model, $T_{\text{surf}}$

The surface temperature in the SATIGR data set is defined as the temperature of the lowest radiosonde reading. We produce a model by assuming the surface temperature varies from this value by a Gaussian random variable with a variance of 3 degrees Kelvin. We assume that the cross-covariance function is stationary and exponential, and that the correlation coefficient is approximately 0.5 between 54 GHz and 183 GHz. The cross-covariance function is written as

$$K(f_1, f_2) = \sigma_{T_{\text{surf}}}^2 e^{-\ln\frac{0.5}{130}|f_2-f_1|}$$ (5.1)

We form a covariance matrix from this specification.
To generate these random vectors, we use the technique described in Section 2.2.3. We generate a vector of independent zero-mean Gaussian random variables with variances equivalent to the size of the eigenvalues of the covariance matrix. We then compute the square root of the correlation matrix. The correlated vector is obtained by multiplication of the two.

**Surface Emissivity Model, }_{surf}\**

The }_{surf} model is more complex than the }_{surf} model because the distribution is approximately uniform instead of Gaussian. We have a measurement of the correlation of surface emissivity between frequencies over land [KPR82]. Using this data, we generate a correlation matrix } for the frequencies in the instrument model.

The procedure for generating random emissivity vectors is similar to the procedure given for generating random surface temperature vectors. We generate a vector of uniform random variables } between 0 and 0.65. We then compute } the square root of the correlation matrix }\(K\). We obtain the correlated reflectivity vector by the equation

\[ \rho = Q(v - 0.325) + 0.325. \]  

(5.2)

The resulting vector is not truly uniform, but a histogram show that it is approximately uniform. The emissivity vector is obtained by } = 1 - \rho. We clip any values outside of the interval [0, 1] to the boundary. Figure 5-5 shows a plot of typical emissivity vectors across the frequencies.

**5.2.2 Instrument Noise Model, }_{inst}\**

All realistic instruments introduce noise in the measurement process. After calibration, the dominant noise sources are thermal sources.

To model the instrument noise }_{inst}, we introduce an additive noise vector at each epoch. We assume the noise is uncorrelated across channels and across time. The values of the instruments' sensitivities are given in [RAW+96] and [BBR+00] We then take into account signal processing gains due to the smoothing of the MIR data before down-sampling, which reduced the noise on the MIR channels by a factor of 3. (See Section 4.3.3 for details on the signal processing of the MIR data.) Finally, we multiply in a safety margin factor of
1.3 for noise robustness.

Figures 5-6, 5-7, and 5-8 give the values for these noise sensitivity values.

Given knowledge of the first and second moment of the noise distribution, then the maximum entropy assumption for the distribution is Gaussian. We add the Gaussian noise vector to the calculated brightness temperatures.

5.2.3 Network Architecture

The number of hidden nodes used for the network architecture is obtained from previous work [CM93, CMS95]. This architecture consists of two sigmoidal hidden layers of 30 and 15 nodes, and a linear output layer of 15 nodes. We modified the derivative of the linear output function so that there is a cost of 2 for output values outside the a priori known range of [0, 1] (See Section 3.5.6 for details). The training function is backpropagation of error using gradient descent with momentum and adaptive learning rate. The momentum constant is set to 0.9, and the adaptive learning rate parameters are given in Equation 3.18. We use a validation stopping mechanism that stops training if 20,000 epochs have passed since the validation set has obtained its minimum error. The network is trained using coefficients generated at a scan angle of 7.2° (See Section 5.2.6 for explanation). The code is based on the MATLAB Neural Network Toolbox, version 4.0.
Figure 5-6: MIR modeled sensitivity.

Figure 5-7: NAST-M 54 GHz modeled sensitivity.
5.2.4 The Training Process

Initialization

First, we divide the 1761 SATIGR radiosondes into a training set and a validation set. The process for choosing which RAOBs belong to the training set and which belong to the test set the following. First, the RAOBs are ordered according to climate and season. Then, the first 15 of every 16 RAOBs are assigned to the training set, and the remaining radiosondes are assigned to the validation set. This produces a total of 1651 training vectors and 110 validation vectors that are evenly distributed across all climates.

We then generate the stochastic surface parameters $T_{surf}$ and $\epsilon_{surf}$, and the instrument noise parameter $N_{inst}$ and compute the brightness temperatures for both the training and validation set by the equation

$$TB = T_U + \epsilon_{surf}T_D + (1 - \epsilon_{surf})T_{surf} + N_{inst}.$$  \hspace{0.3cm} (5.3)

Using the training set, we compute a zero-mean and unit-variance normalization. We then apply the normalization to both the training and validation set.

The weights for the neural network are initialized by using the Nguyen-Widrow ini-
tialization method [NW89], which is a MATLAB function that ensures that the input is mapped to the linear active region of the neurons in the network.

**During Training**

The training vectors are used in the backpropagation to update the weights. The training set is re-generated at each epoch by realizing a new set of noise vectors, calculating TB, and applying the same normalization that was computed at initialization. We generate a new realization of noise at each epoch to utilize the noise averaging technique discussed in Section 3.7.

**Stopping Rules**

The training algorithm will stop in two situations: either the maximum number of iterations, set at 300,000, is reached, or the validation stopping rule has occurred. To evaluate if the validation stopping rule has occurred, the training program determines the error on the validation vectors at each epoch to ensure that a continued decrease in training data performance generalizes to a continued increase in validation performance.

The validation set remains fixed throughout training because otherwise the generation of the random noise vectors may produce an outlier that will give unreasonably good performance. This is problematic because of the stopping mechanism used. If we vary the noise in the validation set, and by chance the validation set achieves overly good performance, then it is likely that the validation stopping rule will occur before the performance of the validation set improves upon this unrealistically low value.

If the validation stopping rule occurs, we keep the weights for the network with the minimum error on the validation set so the network generalizes well. Otherwise, a limit to the number of epochs will be reached and we keep the weights of the final iteration.

**5.2.5 Training Analysis**

The network stopped after 300,000 iterations to an average RMS error of 0.1233 on the training set and 0.1268 on the validation set. The error on both data sets was still being decreased when the network stopped training. The performance as a function of the number of iterations is shown in Figure 5-9. To determine how much gain we are obtaining by a non-linear retrieval technique, we compare the network’s performance with that of a linear
regression. The resulting RMS errors at each pressure level retrieved in the profile are shown in Figure 5-10. We see that the training and validation errors are similar, and the network achieves significant gains over the linear regression. Both are well below a priori.

5.2.6 Scan Angle Note

We must explain why we train the network for a scan angle of 7.2°. To address the effects of scan angle that distorts brightness temperatures, we divide the scan pattern into regions. Each region can be approximated by its center point. The effect of approximating a region of angles by its center point is an increased error as we approach the boundary of the region.

Figure 5-11 shows how the error changes across scan angle when a network is trained specific to one scan angle and given data of another scan angle. The three networks trained in this manner used the scan angles 7.2°, 28.8°, and 43.2°. Notice that the error has a minimum at the scan angle the network was trained on and the error increases the further we deviate from this angle.

An interesting experiment is to use the noise-averaging technique for training a neural network by considering that the scan angle effect is a form of noise. When generating the training set at each iteration, assign a random scan angle within instrument’s swath to each training vector and generate its brightness temperature according to its assigned angle. In
Figure 5-10: Linear vs. non-linear retrieval error.

Figure 5-11: Single angle vs. noise-averaged network.
the long term, the network will learn to smooth over all scan angles it has seen. The result is given in Figure 5-11. Notice that the minimum error occurs at a scan angle of 21.6° and that the maximum deviation from that angle is 0.7% at 50.4°.

Figure 5-12 shows how we group the scan angles for network training. For the nadir region, we use the angle 7.2° to approximate the region. Because the effects are symmetric, we consider the boundaries to be 0° and 14.4°. The central region then covers the central 5 spots of a NAST-M scan: −14.4° − 14.4°. The worst-case deviation of 10.6% occurs at 14.4°. If a set of 4 networks was used, the 3 networks whose performance is plotted in Figure 5-11 and a 4th network trained at 50.4°, then the error across the entire scan can be kept below 10.6%, approximately.

Because of these experiments, we admire the fact that one network can cover all scan angles. However, the tradeoff for coverage of all scan angles is a decrease in sensitivity from 10.0% to 12.1%. To have retrievals with minimum error and maximal scan-angle coverage around nadir, the microwave network is trained at 7.2°.

**Comparison with Previous Work**

In comparison, previous neural networks by Cabrera obtained a mean RMS error of 10.3% over land and 9.85% over ocean using the same training data, noise averaging technique for
instrument noise, and network architecture, but without the relative humidity correction on the SATIGR data set. The differences between the previous work and this work are the following.

- The frequencies have been changed by a deletion of the 23.8, 31.4, and 166.0 GHz channels, and addition of the 150.0, 220.0 ± 2.5, and 118.75 ± {3.5, 2.3, 2.1, 1.6, 1.2, 0.8, 0.4, 0.23} GHz channels.

- The measured instrument sensitivities are being used instead of typical 0.3 rms sensitivity figure.

- The land and ocean surface model distributions have been combined into a single distribution which encompasses both models (Section 5.2.1).

- The surface temperature is now decorrelated across frequencies.

- The network is trained at a scan angle of 7.2°.

- The radiative transfer algorithm assumes the instrument is at the 24th pressure level, which is at 200 mbar instead of at the 40th pressure level, which is at 0.05 mbar.

- The SATIGR RAOBs which are used in the radiative transfer algorithm have been adjusted so the values for relative humidity do not exceed 1. Previously the problematic RAOBs were used directly in the radiative transfer equation to generate brightness temperatures for input to the neural network, but the corresponding relative humidity vectors at the output of the neural network were clipped so their maximum value was 1. This created an inconsistency, which has been fixed.

A series of experiments were performed to determine what effect there is from changing the models from this work to the models in our work. We first recreated the original work using the original SATIGR data set without modifications. We then changed the frequencies to match those of the NAST-M and MIR instruments. We changed the assumed instrument sensitivity from 0.3 to the recorded sensitivity of the instruments. We combined the surface models by using the generalized surface emissivity model discussed in Section 5.2.1 The results of these experiments are summarized in Table 5.4. It seems that the addition of the 118 GHz channels and the 220 GHz channel does not make up the loss of the window channels at 23.8 and 31.4 GHz, as seen by an increase in error. The fact that using actual
Table 5.4: Results from altering training assumptions.

<table>
<thead>
<tr>
<th>Exp #</th>
<th>Description</th>
<th>Surface Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Baseline</td>
<td>Land</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.3%</td>
</tr>
<tr>
<td>2</td>
<td>+ NAST-M/MIR frequencies</td>
<td>11.3%</td>
</tr>
<tr>
<td>3</td>
<td>+ Measured Sensitivities</td>
<td>11.0%</td>
</tr>
</tbody>
</table>

sensitivities decreases the error is not a surprise, since many of the channels have better sensitivity than the previously assumed value of 0.3 rms.

Implementing all changes listed above results in a final error on the validation set of 12.68% The removal of certain assumptions, like the assumption of a perfectly correlated surface, causes more variation in the radiances and thus it is more difficult for the network to learn from them.

5.3 Retrievals

Once the network is trained, we test it on three flights from the CAMEX-3 mission: August 8th and 13th, and September 13th, 1998. We will present evidence that the retrieval algorithm achieves an average of 10% rms relative humidity error. In a later chapter, we will demonstrate that the infrared algorithm performs better than the microwave algorithm and that a retrieval resulting from the combination of the two algorithms performs slightly better than the best case of both.

5.3.1 Definitions for Retrieval Figures

For each flight, we present several different types of images.

- A flight retrieval is a set of retrievals for regions where the data is considered good; masked regions are those with clouds or where the plane is turning which our retrieval algorithm is not trained to handle.

- A vertical nadir retrieval for a region is a vertical cross-section of the atmosphere along the flight track for a specific range of time.

- A horizontal set of retrievals for a region consists of a horizontal image using scan angles $-24.5^\circ - 24.5^\circ$ on the y-axes and a range of time during the flight on the
x-axes. Scan angle dependencies affect the retrieval by warping them near the angle extremities.

- A RAOB/retrieval-average comparison is a comparison between the RAOBs launched around the time of the flight and the average of naturally-partitioned retrieval sections from the flight retrieval. Usually a mean is computed across a continuous section of good data from the flight retrieval. Occasionally, as in the case of the Andros Island passes from September 13th, the visible region on the flight retrieval is partitioned into smaller regions which identify surface features. In this case, an average is computed over each smaller region and displayed as a separate line on the plot. Each retrieval-average line is coded to represent either land or water. When observing these plots, keep in mind that the statistics for retrievals which we average over not necessarily stationary.

- An rms error plot demonstrates the best fit between the RAOB and one particular retrieval at nadir. The best fit is determined by using the mean-squared-error cost function to compare the mean of all the RAOBs for the flight to each retrieval from the flight retrieval. In general, the retrieval chosen by this method as being the best corresponds to a retrieval close to the RAOB launch site. There are no exception in this thesis.

Note: the 5 terms defined above, flight retrieval, vertical nadir retrieval for a region, horizontal set of retrievals for a region, RAOB/retrieval-average comparison, and rms error plot, have been italicized throughout the text to aid the reader in identifying when we are referring to a particular type of plot.

Each contour represents a differential of 0.05 relative humidity.

5.3.2 Retrieval Image Processing

The MIR data was averaged when being aligned and resampled to the resolution of the NAST-M data, as discussed in Section 4.3.3. The data was applied to the neural network and the resulting retrievals were analyzed without any further smoothing. In all of the retrieval images shown in this thesis, smoothing was performed.

In preparing the retrieval images below, we first smooth the retrievals along the dimension of the flight track using a 7-point uniformly spaced symmetric Gaussian filter whose
standard deviation is 2 points. Then, we increase the resolution by a factor of four by linearly interpolating in both the flight track dimension, and the other dimension, which can be either parallel or perpendicular to the earth's surface. This processing is only used for displaying the images and not for any other purpose.

![Image of GOES infrared imagery on September 13th with the ER-2 flight path superimposed.](image)

**Figure 5-13:** GOES infrared imagery on September 13th with the ER-2 flight path superimposed.

### 5.3.3 September 13th, 1998

We first examine the flight on September 13th, 1998. During the flight, the ER-2 aircraft flew over Andros Island where the RAOBs that we will use to validate our algorithm were launched. The GOES imagery in Figure 5-13 shows the flight path over Andros Island on a mostly cloud-free path. Figure 5-14 gives the flight retrieval.

During the flight, the ER-2 makes two passes over Andros Island. Figure 5-15 shows the first island pass, whose surface features are listed in Table 5.5. Figures 5-16 and 5-17 show the second island pass, whose surface features are listed in Table 5.6.
Six RAOBs were launched from Andros Island around the time of this flight. Figure 5-18 gives the RAOB/retrieval-average comparison. Notice that the water retrievals fit the RAOB better than the land retrievals, except for the two land retrievals that correspond to the flights over Andros Island. The rms error plot given in Figure 5-19 demonstrates the best retrieval whose rms error in relative humidity is 8.86%. This number is calculated by taking the average of the six RAOBs from this flight and finding the retrieval from the entire flight that best matches the RAOB-mean according to the mean-squared error metric. The mean-RAOB is plotted with the best retrieval, and the corresponding section of the flight where this retrieval occurred is displayed along side.

**High-frequency retrieval noise** Notice the retrievals are somewhat noisy; they have been smoothed by a Gaussian smoothing kernel so that one standard deviation is equivalent to approximately 3.5 km. A NAST-M scan occurs every 5.5 seconds, and the ER-2 flies at a speed of approximately 200 meters per second. Therefore, one second is approximately one kilometer. The high-frequency noise in the retrievals has a period of approximately 90 seconds or 90 kilometers.

5.3.4 August 13th, 1998

Now we examine the flight on August 13th, 1998. The ER-2 does fly past Andros Island where the RAOBs are launched, however the GOES imagery (Figure 5-20) shows clouds over the island. Since we only produce clear-air retrievals, a comparison with a RAOB that observes a cloud will be a poor comparison. Figure 5-21 gives the flight retrieval.

During the flight, the ER-2 passed over a strip of ocean (Figures 5-22 and 5-24) which exhibits a band of water vapor at 560 mbar and a dry band at 880 mbar. A small region near Andros Island where the cross is located in the GOES image produces a relatively boring retrieval (Figures 5-23 and 5-25) which is monotonically decreasing with decreasing pressure.

Only one RAOB was launched in conjunction with this flight. Since the RAOB was launched in a cloudy region where our retrievals do not operate, the comparison is expected to show larger error than in the September 13th flight. Figure 5-26 gives the RAOB/retrieval-average comparison which does indeed show a large error. The rms error plot given in Figure 5-27 demonstrates the best retrieval whose rms error in relative
Figure 5-14: Microwave flight retrieval for September 13th. Contours represent a 0.05 change.
Figure 5-15: Microwave vertical nadir retrieval over Andros Island, pass #1 for September 13th. Contours represent a 0.05 change.

Table 5.5: Surface features for September 13th, pass #1.

<table>
<thead>
<tr>
<th>Time</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:31</td>
<td>Begin Island Pass</td>
</tr>
<tr>
<td>12:33</td>
<td>Andros Island</td>
</tr>
<tr>
<td>12:37</td>
<td>Lake on Island</td>
</tr>
<tr>
<td>12:39</td>
<td>Peninsula on Island</td>
</tr>
<tr>
<td>12:40</td>
<td>End Island Pass</td>
</tr>
</tbody>
</table>

Table 5.6: Surface Features for September 13th, pass #2.

<table>
<thead>
<tr>
<th>Time</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>01:46</td>
<td>Begin Island Pass</td>
</tr>
<tr>
<td>01:50</td>
<td>Andros Island</td>
</tr>
<tr>
<td>01:53</td>
<td>Lake on Island</td>
</tr>
<tr>
<td>01:54</td>
<td>Peninsula on Island</td>
</tr>
<tr>
<td>01:55</td>
<td>End Island Pass</td>
</tr>
</tbody>
</table>
Figure 5-16: Microwave vertical nadir retrieval over Andros Island, pass #2 for September 13th. Contours represent a 0.05 change.
Figure 5-17: Microwave horizontal set of retrievals over Andros Island, Pass #2 for September 13th. Contours represent a 0.05 change.
Figure 5-18: Microwave RAOB/retrieval-average comparison for September 13th.

Figure 5-19: Microwave rms error plot for September 13th. Average of 6 RAOBs (o) vs. best-fit retrieval (x) at 1:52 UTC.
humidity is 11.38%. This number is calculated by taking the only RAOB from this flight and finding the retrieval from the entire flight that best matches the RAOB according to the mean-squared error metric. The RAOB is plotted with the best retrieval, and the corresponding section of the flight where this retrieval occurred is displayed along side.

These retrievals have noise effects similar to those of the September 13th retrievals. The noise appears to have a different frequency in each image, but it does not; each plot covers a different time span.

5.3.5 August 8th, 1998

Finally, we examine the flight on August 8th, 1998. During the flight, the ER-2 aircraft flew several passes over the Florida coastline and then passed nearby Andros Island for validation with the RAOBs. The GOES imagery in Figure 5-28 shows the Florida coast covered with what appears to be a storm system. Figure 5-29 gives the flight retrieval.

During the flight, the ER-2 passes over a clear-air strip (Figures 5-30 and 5-32) which
Figure 5-21: Microwave flight retrieval for August 13th. Contours represent a 0.05 change.
Figure 5-22: Microwave vertical nadir retrieval for region #1 on August 13th. Contours represent a 0.05 change.

Figure 5-23: Microwave vertical nadir retrieval for region #2 on August 13th. Contours represent a 0.05 change.
Figure 5-24: Microwave horizontal set of retrievals for region #1 on August 13th. Contours represent a 0.05 change.
Figure 5-25: Microwave horizontal set of retrievals for region #2 on August 13th. Contours represent a 0.05 change.
Figure 5-26: Microwave RAOB/retrieval-average comparison for August 13th.

Figure 5-27: Microwave rms error plot for August 13th. Single RAOB (o) vs. best-fit retrieval (x) at 8:50 UTC.
Figure 5-28: GOES infrared imagery on August 8th with the ER-2 flight path superimposed.

exhibits a rise in humidity as the aircraft moves closer to the clouds. The 70A second region
is taken from where the aircraft flies over the land/sea boundary on the Florida coast at 6:27
p.m. (Figures 5-31 and 5-33). Because the transition from retrievals over land to ocean is
smooth, the noise-averaging technique used to smooth over surface effects is thus validated.

Two RAOBs were launched for this flight. Figure 5-34 gives the RAOB/retrieval-average
comparison. The rms error plot given in Figure 5-35 demonstrates the best retrieval whose
rms error in relative humidity is 9.81%. This number is calculated by taking the average
of the two RAOBs from this flight and finding the retrieval from the entire flight that best
matches the RAOB-mean according to the mean-squared error metric. The mean RAOB
is plotted with the best retrieval, and the corresponding section of the flight where this
retrieval occurred is displayed along side.
Figure 5-29: Microwave flight retrieval for August 8th. Contours represent a 0.05 change.
Figure 5-30: Microwave vertical nadir retrieval for region #1 on August 8th. Contours represent a 0.05 change.

Figure 5-31: Microwave vertical nadir retrieval for region #2 on August 8th where the Land-Sea Boundary occurs at 6:27 P.M.. Contours represent a 0.05 change.
Figure 5-32: Microwave horizontal set of retrievals for region #1 on August 8th. Contours represent a 0.05 change.
Figure 5-33: Microwave horizontal set of retrievals for region #2 on August 8th where the Land-Sea Boundary occurs at 6:27 P.M.. Contours represent a 0.05 change.
Figure 5-34: Microwave RAOB/retrieval-average comparison for August 8th.

Figure 5-35: Microwave rms error plot for August 8th. Average of 2 RAOBs (o) vs. best-fit retrieval (x) at 6:42 UTC.
5.4 Analysis of Retrieval Accuracy

The rms validation error after training was completed is 13.1%. The mean rms error for the three sets of retrievals is 10.02%. The individual errors for the flight data are summarized in Table 5.7. It makes sense that our flight retrieval errors could be lower than the network validation error because the training set includes RAOBs from all climates and seasons and is generated using a more general surface model whereas the flight data is from a tropical climate with very specific surface parameters, which is an easier problem.

Recall that we obtain each number by first computing the average of all available RAOBs for a particular flight (the number of RAOBs is given in the table), then computing the rms error between the RAOB-mean and all retrievals in the entire flight which do not correspond to aircraft turns, cloudy regions, etc., and finally choosing the retrieval with the lowest rms error. The mean-RAOB and the best retrieval are plotted along side the region where the retrieval originated in the figure that is listed in the table. The time listed in the table is the exact time in this figure from where the retrieval was taken.

Table 5.7: Microwave retrieval accuracies for the flight data.

<table>
<thead>
<tr>
<th>Date</th>
<th>RMS error</th>
<th>Time in flight</th>
<th># of RAOBs</th>
<th>Figure #</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/03/98</td>
<td>9.81%</td>
<td>6:42</td>
<td>2</td>
<td>5-35</td>
</tr>
<tr>
<td>8/13/98</td>
<td>11.38%</td>
<td>8:50</td>
<td>1</td>
<td>5-27</td>
</tr>
<tr>
<td>9/13/98</td>
<td>8.86%</td>
<td>1:52</td>
<td>6</td>
<td>5-19</td>
</tr>
</tbody>
</table>
Chapter 6

Infrared Retrievals

The technique we apply for the infrared retrievals is very similar to the technique that we use for the microwave retrievals. We adapt our technique with several changes.

The first issue is that the flight data is given in radiances, and due to noise and biases inherent in the flight data, some of the radiances are negative. This makes it impossible to convert to brightness temperature without first removing the problem that causes the radiances to go negative. For this reason, we will develop our training set on radiance directly. The data for the microwave case was given in brightness temperatures, so in that case it was more natural to work in those units. Part of this problem is the large biases present on some of the channels. Since accurately estimating this bias is beyond the scope of this work, we apply an alternate technique, which allows retrievals at least as good as the microwave case. We will add to the training data a random bias in addition to the instrument noise and surface models which were present in the microwave retrievals.

The second issue is that there are about 9000 channels, so we cannot apply the infrared data directly to the input of the neural net because the number of weights in the input layer would far imbalance the relatively small size of the training set; we must first perform feature selection and compression. We will do this in two steps. First, we will choose a subset of 3000 channels whose weighting functions are distributed throughout the atmosphere. Second, we will use principal components analysis on the training set to compress the feature space down to a 100-element feature vector. This feature vector will be used to train the network.

The final issue is that simulating radiances for 9000 channels is computationally intense, so a special routine exists for fast computation of infrared radiative transfer coefficients.
These coefficients will be combined with random noise as in the microwave case, except this will be done every 10 epochs instead of every epoch because each generation process takes 10 seconds and this techniques reduces the time for training a network from 18 days to 2 on an Intel Athelon Processor with sufficient memory.

6.1 Radiative Transfer Coefficient Simulation

We compute radiative transfer coefficients for the infrared case similarly to the microwave case. We again assume a planar-stratified atmosphere. There is a difference in the computation of the transmittances for each layer. The transmittances are computed from a pre-calculated transmittance coefficients file. This uses the Strow-Woof-VanDelst regression model based on LBLRTM line-by-line transmittances, which is included in the appendix.

The assumed aircraft height is 50 mbar. The instrument model is degenerate; both the passband and the antenna pattern are delta functions. The computation is performed using a state-space model with $T_{\text{cosmic}} = 3.00$ K.

6.2 Feature Selection

We select 3000 channels from the 9000 available by considering the characteristics of the weighting function of each channel. The channels that we choose have the characteristic of being the sharpest of all weighting functions whose center of gravity is within a specific region. More rigorously, let \{w_1[p], w_2[p], \ldots, w_N[p]\} be a the set of weighting functions which correspond to the channels of the instrument, where each $w$ is a discrete function of pressure, the discrete pressure values are given by vector $P = \{p_1, p_2, \ldots, p_M\}$, and $\sum_{j=1}^{M} w_i[p_j] = 1, \forall i \in [1, N]$. Let $P_i = \sum_{i=1}^{M} p_i$ be the normalizing factor. The first moment is

$$\bar{w}_i = \frac{1}{P_i} \sum_{j=1}^{M} Mp_j w_i[p_j], \quad (6.1)$$

and the second moment is

$$\bar{w}_i = \frac{1}{P_i} \sum_{j=1}^{M} (p_j)^2 w_i[p_j]. \quad (6.2)$$
We see in Figure 6-1 the distribution of the first and second moments across the three bands.

To select channels, we partition the channels into bands: Band 1 is $7.76 - 16.13\mu m$, Band 2 is $4.81 - 7.76\mu m$, and Band 3 is $3.68 - 4.81\mu m$. We perform the following process once on each band.

1. Order the weighting functions within a band according to the first moment.

2. Remove all weighting functions whose first moment is above 200 mbar. These functions peak near the aircraft height.

3. Partition the ordered set of weighting functions into 50 bins, where each bin contains an equal number of channels. This amounts to approximately 60 channels per bin.

4. Select the 20 channels in each bin with the smallest second moment. These weighting functions are the sharpest of each bin. This step amounts to approximately 1000 channels per band.

5. Repeat for all three bands, giving a total of about 3000 channels.

### 6.3 Feature Compression

To facilitate input into the neural network, the feature vector of 3000 channels obtained from feature selection must be reduced to a feature vector of length 100. The full feature vector is rotated into principal components space where the dimensions are uncorrelated with each other. Only the training vectors are used in computation of the covariance matrix whereas the validation vectors are not. The top 100 principal components are kept and the remaining ones are discarded. Since the removal of each additional principal component can only remove information, it is possible to see gains in performance by facilitating more principal components in the input of the neural net. However, since the number of weights in the input matrix increases, more training vectors would be required to properly constrain the weights.
Figure 6-1: Weighting function 1st moment, 2nd moment, and wavenumber scatter plots.
6.4 Network Training

The neural network training is the similar to the training of the microwave network. The differences will be given below.

6.4.1 Surface Model

Surface Temperature Model, \( T_{\text{surf}} \)

The surface temperature is varied using the same model as in the microwave case. Recall that this model assumes that the surface temperature is equivalent to the lowest level of the radiosonde with additive zero-mean Gaussian noise correlated across frequencies according to the model given in Section 5.2.1.

Surface Emissivity Model, \( \varepsilon_{\text{surf}} \)

The surface emissivity model differs from the microwave case. The frequency dependence has previously been modeled by a 7-point piecewise-linear equation which varies according to some distribution [LAC+93]. We do not have the model which generates these vectors, but we do have an ensemble of 12000 examples generated by the model that we can use to derive our own. Using the noise-averaging robustness idea from before, we simply need a model that approximates the actual surface and errs on the conservative side. To create our model, we compute the correlation matrix for the 7 coefficients of the piecewise-linear model given by the AIRS ensemble. We then use a Gaussian distribution with a mean of 0.95 and a standard deviation of 0.25 with this correlation matrix to generate our own set of vectors. The emissivity is clipped at unity. The Gaussian parameters were chosen to allow a good match between the histogram of our model and that of the AIRS data set.

6.4.2 Instrument Noise Model, \( N_{\text{inst}} \)

NAST-I poses a greater challenge for producing retrievals because the noise is difficult to characterize. Some channels contain large amounts of noise due to not only thermal noise and amplification, but also due to vibrations on the interferometer mirrors. In addition, many of the channels contain a large calibration bias.

Our approach uses the neural network's capability for noise-averaging, as discussed in Section 3.7. We simply determine a bound on the noise and add a different instance of this
noise to the training data every 10 epochs. The network learns to be robust in the face of this noise, but the tradeoff is the increase in rms error. Even so, we are able to use the retrievals from the resulting network to validate the microwave retrievals.

**Uncorrelated Noise**

If we assume that the difference between neighboring pixels is due to a change in physics and additive noise, then we can bound the noise by the variance of this difference. Using the data from September 13th, the variance on the first difference of the nadir soundings is used as this bound. This bound is conservative since the noise exists in both of the pixels used to compute each difference. A tighter bound would divide the difference estimate in two. Since training the network with a slightly larger noise figure produces a more robust network, we simply used the variance.

To generate an instance of this noise, we generate zero-mean Gaussian random variables that are independent across frequencies and training examples and use the standard deviation bound computed for each channel. The uncorrelated noise is updated every 10 training iterations.

**Bias Model**

A premise to remote sensing of the atmosphere is that given the state of the atmosphere, the radiative transfer characteristics are uniquely determined. We can think of the concept of a physics-mean $\overline{P}$ which is the average state of the atmosphere over a set of atmospheric states. Each state $P$ within the above set can then be described as the sum of the physics-mean and a perturbation from that mean $\Delta P$, or

$$P = \overline{P} + \Delta P. \quad (6.3)$$

If we have a retrieval algorithm which uses $P$ and not just $\Delta P$, then any error in $\overline{P}$ will translate into a degradation of the retrieval. An instrument bias $\overline{B}$ shifts the physics-mean in this manner.

According to the noise-averaging theory, if we introduce this varying behavior into the training data, then the network will find a locally optimal solution that accounts for this behavior. The desirable way to introduce the instrument bias is to have a description of the
bias in the form of a probability distribution where the variance of the model is as small as possible. For robustness, any distribution of the bias must include the actual unknown bias otherwise the network performance may degrade significantly.

Figure 6-2 shows an estimate of the bias and physics-mean $\bar{P} + \hat{B}$ with an error ball of standard deviation $\Delta_{\bar{P}}$ surrounding it. In this case, we believe our estimate of the bias to the degree of $\Delta_{\bar{P}}$. The system will be robust with some degradation on performance because of the noise averaging.

The same figure also shows the case when we don’t have any specific estimate of the bias, but have an estimate of the distribution of the bias, $\Delta_{\bar{P}}$. The resulting network trained on this distribution of the noise will also be robust, but the degradation in this case will be greater than the previous case because the noise averaging is performed over a larger space.

We do not know the bias, so we assume a random bias with a zero-mean Gaussian distribution and standard deviation of $1.25^\circ$K for all channels. This value was determined by guessing a larger value, training a network, and then relaxing that value. As the value was relaxed, the retrievals became sharper, like putting on a better pair of glasses. We stopped at the value of $1.25^\circ$K because the retrievals were sufficiently clear to depict the atmospheric features.

To generate an instance of the bias noise, we generate a zero-mean Gaussian random
variable independently for each channel and apply the same value to all of the training examples in the training set. Every 10 iterations of the training algorithm, a new bias value is used for each channel.

6.4.3 Network Architecture

Two changes are made in the network architecture from the microwave case. First, the number of inputs has increased from 22 to 100, which produces a corresponding increase in the number of weights in the input layer. In addition, the training data is simulated at nadir instead of 7.2° because we only perform nadir retrievals for NAST-I. There are still 30 nodes in the first hidden layer, 15 nodes in the second hidden layer, and 20 output nodes. We have still modified the derivative of the linear output layer to add the a priori knowledge that relative humidity lies in the range of [0, 1]. The use of backpropagation with adaptive learning rate and momentum with the same validation stopping rules has not changed.

6.4.4 The Training Process

The initialization and stopping rules are identical to those used in the microwave case. A change was required in the training algorithm to the adaptive learning rate mechanism. When the bias and instrument noise is updated every 10 epochs, the adaptation mechanism detected a large increase in error and responded by reducing the learning rate. After a few iterations, the learning rate would converge to zero.

To fix this problem, the learning rate is boosted by a factor of 10 upon each update of the noise in the training set. This allows the training algorithm to have enough extra energy to compensate for the abrupt change in the training set, and since it is a factor and not an additive term, the effect is the same later in the training when the learning rate becomes orders of magnitude smaller.

Since the network trained in this manner takes considerably longer than the microwave network, a third stopping rule was instituted: the long-enough stopping rule which occurs when the human operator of the training algorithm intervenes.

6.4.5 Training Analysis

The network stopped by an occurrence of the long-enough stopping rule after 43200 iterations (3 days) to an average rms error of 0.1095 on the training set and 0.1104 on the
validation set. The performance as a function of the number of iterations is shown in Figure 6-3. To determine how much gain we are obtaining by a non-linear retrieval technique, we compare the network’s performance with that of a linear regression. The resulting rms error at each pressure level retrieved in the profile is shown in Figure 6-4. We see that the training and validation errors are similar, and the network achieves significant gains over the linear regression. Both are well below a-priori.

6.4.6 Scan Angle Note

Unlike the microwave case, we did not include scan angle in the simulation because only data from nadir observations was used. Thus, we do not perform horizontal retrievals.

6.5 Retrievals

Once the infrared network is trained, we test it on two of the flights used in the microwave case for comparison: August 13th and September 13th, 1998. We will present evidence similar to that of the microwave case which demonstrates a 7.66% average rms error in relative humidity. To facilitate comparison, the same regions within each flight that were examined in the microwave case are also examined here.
The definitions for the retrieval figures are given in Section 5.3.1. Recall that the terms flight retrieval, vertical nadir retrieval for a region, horizontal set of retrievals for a region, RAOB/retrieval-average comparison, and rms error plot refer to particular types of plots. Recall from Section 5.3.2 that the images being displayed are smoothed along the flight track. The infrared retrievals extend higher into the atmosphere than the microwave retrievals, so close attention should be paid to this. Each contour represents a differential of 0.05 relative humidity. Only nadir retrievals are provided.

6.5.1 September 13th Flight

The first flight examined with the infrared retrievals is the flight on September 13th, 1998. Figure 6-5 gives the flight retrieval.

Figure 6-6 is the first Andros Island pass whose features are listed in Table 5.5. Figure 6-7 is the second pass whose features are in Table 5.6.

The six RAOBs launched on this date are used to generate the RAOB/retrieval-average comparison in Figure 6-9 and the rms error plot in Figure 6-8. Notice that the RAOB/retrieval-average comparison plot does not show how good our retrievals are exactly; we must examine the rms error plot. The retrievals near the island vary enough that the retrieval-average does not reflect the existence of good matches to the RAOB within the island pass.
The rms error for this flight is 8.44%. This number is calculated by taking the average of the six RAOBs from this flight and finding the retrieval from the entire flight that best matches the RAOB-mean according to a mean-squared error metric. The mean-RAOB is plotted with the best retrieval, and the corresponding section of the flight where this retrieval occurred is displayed along side.

Notice that even though the same image smoothing that was performed on the microwave images is performed here (Section 5.3.2), these retrievals appear smoother than the microwave retrievals.

6.5.2 August 13th Flight

Next, we examine the flight on August 13th, 1998. Recall that for this flight the RAOB was launched when a cloud had covered the launch site, so we cannot get an accurate validation of our retrievals. Figure 6-10 gives the flight retrieval.

The same two segments of the flight as given in Section 5.3.4 are given here. We examine the strip of ocean (Figure 6-11) and the small region Andros Island (Figure 6-12).

The one RAOB launched for this flight is used to generate the RAOB/retrieval-average comparison in Figure 6-14 and the rms error plot in Figure 6-13.

This number is calculated by taking the only RAOB from this flight and finding the retrieval from the entire flight that best matches the RAOB according to a mean-squared error metric. The RAOB is plotted with the best retrieval, and the corresponding section of the flight where this retrieval occurred is displayed along side.

6.6 Analysis of Retrieval Accuracy

The rms validation error after training was completed is 11.04%. The mean rms error for the 2 sets of flight retrievals is 7.66%. The individual errors for the flight data are summarized in Table 6.1.

Again, the flight retrieval errors are much lower than the error on the training set. The same argument holds as in the microwave case.

Recall that we obtain each number by first computing the average of all available RAOBs for a particular flight (the number of RAOBs is given in the table), then computing the rms error between the RAOB-mean and all retrievals in the entire flight which do not correspond...
Figure 6-5: Infrared flight retrieval for September 13th. Contours represent a 0.05 change.
Figure 6-6: Infrared vertical nadir retrieval over Andros Island, pass #1 for September 13th. Contours represent a 0.05 change.

Figure 6-7: Infrared vertical nadir retrieval over Andros Island, pass #2 for September 13th. Contours represent a 0.05 change.
Figure 6-8: Infrared RAOB/retrieval-average comparison for September 13th.

Figure 6-9: Infrared rms error plot for September 13th. Average of 6 RAOBs (o) vs. best-fit retrieval (x) at 1:50 UTC.
Figure 6-10: Infrared flight retrieval for August 13th. Contours represent a 0.05 change.
Figure 6-11: Infrared vertical nadir retrieval for region #1 on August 13th. Contours represent a 0.05 change.

Figure 6-12: Infrared vertical nadir retrieval for region #2 on August 13th. Contours represent a 0.05 change.
Figure 6-13: Infrared RAOB/retrieval-average comparison for August 13th.

Figure 6-14: Infrared rms error plot for August 13th. Single RAOB (o) vs. best-fit retrieval (x) at 8:52 UTC.
to aircraft turns, cloudy regions, etc., and finally choosing the retrieval with the lowest rms error. The mean-RAOB and the best retrieval are plotted along side the region where the retrieval originated in the Figure listed in the table. The time given is the exact time in this figure from where the retrieval was taken.

Table 6.1: Infrared retrieval accuracies for the flight data.

<table>
<thead>
<tr>
<th>Date</th>
<th>RMS error</th>
<th>Time in flight</th>
<th># of RAOBs</th>
<th>Figure #</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/13/98</td>
<td>6.88%</td>
<td>8:52</td>
<td>1</td>
<td>6-14</td>
</tr>
<tr>
<td>9/13/98</td>
<td>8.44%</td>
<td>1:50</td>
<td>6</td>
<td>6-9</td>
</tr>
</tbody>
</table>
Chapter 7

Comparing Microwave and Infrared

7.1 Comparison

In the microwave case, the retrievals which were taken nearby a RAOB launch site matched the retrievals with an rms accuracy of 9.39% in both flights, and the August 13th flight which did not occur nearby the RAOB launch site matched the retrievals with an rms accuracy of 11.38%. In the infrared case, the retrievals matched well in both cases with an rms accuracy of 7.66%.

In the two flight sections over Andros Island on September 13th, the same features appear in both the infrared and the microwave case. The first pass (Figures 5-15 and 6-6) contains several features. Feature A is a mass of water vapor during 12:37-12:40 in the pressure band 450-800 mbar. Feature B is a band of water vapor between pressure levels 310-380 mbar and is clearly present in the infrared case. With careful examination, the microwave case shows faint signs of this feature with the two circular contours at time 12:31-12:33 and pressure 345 mbar. If the RAOB is correct, then the infrared case gives a more accurate appraisal of 0.35 for the relative humidity in this band, while the microwave estimates less near the time 12:31 and is overrun with humidity due to blurring from Feature A.

In the second pass (Figures 5-16 and 6-7, the infrared shows Feature B more distinctly while the microwave case is blurred further. Notice that underneath Feature A, the infrared
case now shows a dry dimple at the time 1:53 and pressure level of 852 mbar which did not exist in the first pass and is similar to what the microwave case estimated in the retrievals for both passes. The first region for August 13th (Figures 5-22 and 6-11) is very rich with features. Figure 7-1 is a schematic diagram of the features for this flight region. Notice that in both the microwave and infrared cases, we can identify all seven features. Feature A is not as saturated in the microwave case as it is in the infrared case. The saturated band which consists of Features C and F are very similar in both images. Feature D is much stronger in the infrared case. The dry band near the surface, Features B and E, is split by Feature D, but appears to be one continuous band in the microwave case because Feature D is not as strong. Finally, the dry column, Feature G, is present in both images and is more distinct in the infrared case.

In the second region for August 13th (Figures 5-23 and 6-12) the microwave and infrared retrievals agree well as to how the relative humidity falls as a function of altitude, but they differ as to the rate in which they fall. In the microwave case, the 0.5 relative humidity contour lies at about 480 mbar, while in the infrared case it lies at 670 mbar.
7.1.1 Correlation of Retrievals

One way to determine agreement between the microwave and the infrared retrievals is to compute a principal components analysis on a data set which contains examples of both retrievals types and observe how the two data point relate in the first principal component. The first principal component for each of the microwave points is plotted against the 1st principal component for each of the corresponding infrared points. This plot is a good visual indication of agreement. Figure 7-3 gives this plot for August 13th, and Figure 7-4 gives this plot for September 13th. In the plot for August 13th, we notice that there does not appear to be a strong correlation between the microwave and the infrared retrievals. In contrast, the plot for September 13th shows a stronger correlation. We suggest, but have not confirmed, that since August 13th is a cloudier day than September 13th, and since the August 13th plot is less correlated than the September 13th plot, the outliers in the former plot result from the effects of clouds.

Another visual method is to compute the cross-correlation matrix between the microwave and infrared retrievals and color-code each entry of the matrix. We do this for the training data in Figure 7-2 and see that the retrievals have strong correlation at all common pressure levels. Notice that the correlation in the upper atmosphere is stronger than that near the
Figure 7-3: Scatter diagram of 1st principal component correlation for August 13th.

Figure 7-4: Scatter diagram of 1st principal component correlation for September 13th.
surface. This most likely indicates that the surface models for the two cases are different, and it is further backed up by observing in Figure 6-4 that the infrared network does not improve over the linear regression at the surface in the training data.

7.2 Combined Linear Retrieval

Since each retrieval technique utilizes different information to obtain its retrieval, a combination of the two techniques can only improve the retrieval quality. A linear regression is performed on the output of each of the retrieval techniques. The inputs to the regression are the 20 relative humidity outputs from the infrared neural network and the 14 outputs from the microwave network. The outputs are a combined relative humidity profile at the 20 pressure levels of the infrared net, a superset of the microwave outputs.

The combined retrieval produces a decrease of as much as 10% relative rms error. Figure 7-5 shows the error curves for the two networks and the combined retrieval.

7.2.1 Retrievals

After computing the linear regression, we test it on the two flights that were used in the infrared and microwave cases for comparison: August 13th and September 13th. We will
present evidence that shows visual improvement in the retrievals, but the actual average rms error remains about the same. One flight is improved slightly and the other is degraded slightly. To facilitate comparison, we again provide the same regions as in both flights of the infrared case and the two corresponding flights in the microwave case.

The definitions for the retrieval figures are given in Section 5.3.1. Recall that the terms flight retrieval, vertical nadir retrieval for a region, horizontal set of retrievals for a region, RAOB/retrieval-average comparison, and rms error plot refer to particular types of plots. Also recall from Section 5.3.2 that the images being displayed are smoothed along the flight track. Each contour represents a differential of 0.05 relative humidity. Only nadir retrievals are provided.

September 13th, 1998

The first flight examined for the combined retrievals is the flight on September 13th, 1998. Figure 7-6 gives the flight retrieval.

Figure 7-7 is the first Andros Island pass whose features are listed in Table 5.5 on page 88. Figure 7-8 is the second pass whose features are in Table 5.6 on page 88.

The six RAOBs launched on this date are used to generate the RAOB/retrieval-average comparison in Figure 7-9 and the rms error plot in Figure 7-10. Notice that the RAOB/retrieval-average comparison plot does not show how good our retrievals are exactly; we must examine the rms error plot.

The retrievals from September 13th are visually improved. A distinct band of moisture that appears in the RAOB at 350 mbar now appears clearly in the retrieval. In the respective microwave and infrared retrievals in Figures 5-14 and 6-5, the moisture band could be picked out from the surrounding features, but it is not as distinct as in the comparable retrieval in Figure 7-6. The rms error for this flight is 8.06%.

This number is calculated by taking the average of the six RAOBs from this flight and finding the retrieval from the entire flight that best matches the RAOB-mean according to a mean-squared error metric. The mean-RAOB is plotted with the best retrieval, and the corresponding section of the flight where this retrieval occurred is displayed along side.

While the moisture band can be seen more easily in the combined retrieval, the retrievals are filled with noise at pressures less than 222 mbar. Recall that in the infrared retrievals starting on page 116, this region also appears noisy. This is most likely due to the removal
of all weighting functions whose first moment is less than 200 mbar in Section 6.2. Most channels that were removed because of this threshold were among the channels whose biases were the largest and whose radiance values were negative. In addition, the effects could also be due to the fluctuations in the aircraft altitude during the flight. At altitude, the aircraft is flying at about 50 mbar, which may be close enough to the highest retrieved pressure level to affect the readings.

August 13th, 1998

Next, we examine the flight on August 13th, 1998. Recall that for this flight the RAOB was launched when a cloud had covered the launch site, so we cannot get an accurate validation of our retrievals. Figure 7-11 gives the flight retrieval.

The same two segments of the flight as given in Sections 5.3.4 and 6.5.2 are given here. We examine the strip of ocean (Figure 7-12) and the small region Andros Island (Figure 7-13).

The one RAOB launched for this flight is used to generate the RAOB/retrieval-average comparison in Figure 7-14 and the rms error plot in Figure 7-15.

The combined retrieval shows a better match for the RAOB/retrieval-average comparison than either microwave or infrared separately. The infrared and combined retrieval curves differ in the upper atmosphere, where the retrieval curves from the infrared only retrievals lie below the RAOB curve while the curves from the combined retrievals lie closer to the line. The microwave retrievals performs even worse on this day, agreeing with the combined retrievals near the surface and then lying above the curve near the mid-atmosphere and diving below the curve for the upper-atmosphere.

According to this visual measure that uses the mean of a large region within the flight, the combined retrieval performs better. However, the analytical measure using the rms error plot gives an rms error of 8.44%, which is worse than the analytical measure in the infrared case.

This number is calculated by taking the only RAOB from this flight and finding the retrieval from the entire flight that best matches the RAOB according to a mean-squared error metric. The RAOB is plotted with the best retrieval, and the corresponding section of the flight where this retrieval occurred is displayed along side.
Figure 7-6: Combined infrared and microwave flight retrieval for September 13th. Contours represent a 0.05 change.
Figure 7-7: Combined infrared and microwave vertical nadir retrieval over Andros Island, pass #1 for September 13th. Contours represent a 0.05 change.

Figure 7-8: Combined infrared and microwave vertical nadir retrieval over Andros Island, pass #2 for September 13th. Contours represent a 0.05 change.
Figure 7-9: Combined infrared and microwave RAOB/retrieval-average comparison for September 13th.

Figure 7-10: Combined infrared and microwave rms error plot for September 13th. Average of 6 RAOBs (o) vs. best-fit retrieval (x) at 1:50 UTC.
Figure 7-11: Combined infrared and microwave flight retrieval for August 13th. Contours represent a 0.05 change.
Figure 7-12: Combined infrared and microwave vertical nadir retrieval for region #1 on August 13th. Contours represent a 0.05 change.

Figure 7-13: Combined infrared and microwave vertical nadir retrieval for region #2 on August 13th. Contours represent a 0.05 change.
Figure 7-14: Combined infrared and microwave RAOB/retrieval-average comparison for August 13th.

Figure 7-15: Combined infrared and microwave rms error plot for August 13th. Single RAOB (o) vs. best-fit retrieval (x) at 8:52 UTC.
<table>
<thead>
<tr>
<th>Date</th>
<th>RMS error</th>
<th>Time in flight</th>
<th># of RAOBs</th>
<th>Figure #</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/13/98</td>
<td>7.40%</td>
<td>8:52</td>
<td>1</td>
<td>7-15</td>
</tr>
<tr>
<td>9/13/98</td>
<td>8.06%</td>
<td>1:50</td>
<td>6</td>
<td>7-10</td>
</tr>
</tbody>
</table>

### 7.2.2 Analysis of Retrieval Accuracy

The rms validation error after training was completed is about 10% as seen in Figure 7-5 on page 127. The mean rms error for the 2 sets of flight retrievals is 7.73%, which is slightly higher than using the infrared retrievals alone. The reason for this is the degradation in the retrievals on August 13th. The September 13th retrievals appear about the same according to the rms error plots. Compare the infrared rms error shown in Figure 6-14 on page 121 with the combined infrared and microwave rms error shown in Figure 7-15 on page 135. In the infrared case, the points from the retrieval line up with all the points from the RAOB except for the two peaks at 430 and 580 mbar. In contrast, the combined retrieval is worse at the surface and seems to lie in between the peaks and valleys between 400-800 mbar while still showing agreement in the upper atmosphere. Notice also that of the two vertical retrieval images shown in these figures, the infrared image is much smoother, so there appears to be a non-linear effect that the linear regression cannot handle. The region where the errors were computed is the region nearby Andros Island, shown by the cross in the GOES imagery on page 92. The most likely reason why the combined retrieval performs worse than the infrared alone according to this measure is that the linear regression is computed on cloud-free training data while the GOES image indicates that the region is not truly cloud-free.

The individual errors for the flight data are summarized in Table 6.1.

Recall that we obtain each number by first computing the average of all available RAOBs for a particular flight (the number of RAOBs is given in the table), then computing the rms error between the RAOB-mean and all retrievals in the entire flight which do not correspond to aircraft turns, cloudy regions, etc., and finally choosing the retrieval with the lowest rms error. The mean-RAOB and the best retrieval are plotted along side the region where the retrieval originated in the Figure listed in the table. The time given is the exact time in this figure from where the retrieval was taken.
Chapter 8

Conclusion and Future Work

8.1 Summary of Work

We were charged with developing a retrieval algorithm for relative humidity profiles using data from the microwave radiometer, NAST-M and MIR, and data from the infrared interferometer, NAST-I. The key problems we have addressed are

- The relative humidity values for the training set were unreasonable.
- Two surface models had been used previously; one for dry land and another for smooth ocean. The surface model incorporated here covers all surface types.
- The microwave data had to be properly simulated. Various simulation parameters are chosen to reflect the actual instrument, such as channel noise, antenna pattern, and aircraft height.
- The infrared data had to be properly simulated. Knowledge of the noise and bias parameters were not available so a bound was estimated from the data.
- The infrared data had to be compressed so that it could be provided as input to the neural network. Feature selection based on 1st and 2nd moments of the water vapor weighting functions was performed and principal components analysis was used to compress the data.
- The infrared data contains excessive noise and large calibration biases. The retrieval algorithm was designed to produce good retrievals despite these problems.
8.2 Conclusion

The microwave algorithm is capable of 9.4% rms error when compared with nearby RAOBs. The neural networks used a noise-averaging technique to train a network that is robust in the face of instrument noise and surface effects. The retrieval algorithms were trained without clouds and retrievals were performed in clear air. Examples of retrievals over the Florida coast, the Atlantic Ocean, and across the boundary of the two and retrievals over Andros Island show meteorologically significant information. We see almost no surface effects in the retrievals which indicates that the technique for smoothing over various surface types during the neural network training performs well. Scan angle effects were present in retrievals not at nadir; we demonstrated that these effects can be addressed either by using several networks each trained on a subset of scan angles or by feeding the neural network training algorithm data generated at randomly distributed angles. Calibration smoothing may help to improve our retrievals. Though the NAST-M instrument calibration already employs a fancier smoothing technique to improve calibration load signal-to-noise ratio, the MIR instrument uses a 2-point calibration and some noise may exist between calibration load observations that can be further smoothed.

The infrared algorithm is capable of 7.7% rms error when compared with nearby RAOBs. This result is certainly preliminary because of the excessive instrument noise and calibration bias present. Examples of retrievals from two flights including retrievals over the Atlantic Ocean and over Andros Island show similar meteorologically significant information as the microwave algorithm. We had in our favor regions of clear air to obtain retrievals from; the infrared instrument would have performed poorly in cloudy regions. The surface emissivity model seems to perform well according to the retrievals, but the theoretical neural network rms error near the surface does not improve significantly on the linear rms error indicating that perhaps the model is not sufficient for obtaining the best retrievals.

Combining the two retrieval algorithms into one retrieval using a linear regression showed a theoretical improvement of up to 1% rms error at some pressure levels, but these results were not realized in application to actual flight data. Retrievals from the two algorithms were more correlated up in the atmosphere than near the surface. A scatter-plot of the top principal component showed that in the flight where there were no clouds the correlation coefficient was higher than in the flight having some clouds. We note that the presence
of clouds has a significant effect on the agreement between the two retrievals; the infrared principal component being more afflicted than the microwave one.

The infrared result demonstrates that good retrieval accuracy and useful meteorologically significant retrievals are possible even in the face of excessive instrument noise and large calibration bias by using the noise-averaging technique to smooth over the distribution of noises and biases. The redundancy inherent in the 9000 channels overcame the correlated noise and is undoubtedly an asset.

Finally, the paradigm using neural networks is a valid means for accommodating both microwave and infrared retrievals with little difference in application from one domain to the other. The robustness can be controlled by properly choosing the stochastic distributions used in the noise-averaging technique.

8.3 Future Work

There are several possible expansions on the present work.

- There are only 2 flights available to compare the microwave domain to the infrared domain. More flights are needed in general to truly ascertain how robust these networks are across various climates, surface types, and seasons. The error may be reduced if more flights are used.

- When dealing with data, one can either normalize the data to fit the algorithm or adapt the algorithm to fit the data. In this work, we have adapted the algorithm to fit the noisy data. By the argument in 6.4.2, if the calibration bias could be better estimated, then the distribution of biases used in the training of the network can be tightened and more sensitive retrievals can be obtained.

- Clouds effect the quality of the retrievals. It would be useful to know how the cloud conditions effect correlation between the microwave and the infrared retrievals as shown in the principal components plot on page 126.

- The correlation matrix in Figure 7-2 shows that the agreement between the two techniques near the surface is less-correlated than in the upper atmosphere according to training data, which is expected and indicates room for improvement. The infrared network is not performing much better than linear near the surface, but the microwave
network performs 25% better than linear in the same pressure level. More work is needed on the infrared surface model.

- The number of principal components chosen for input to the infrared network was determined according to a guess on computational limitation. Alternative techniques for data compression, such as noise-adjusted principal components analysis could better compress the data. More principal components could also be incorporated.

- A linear combination of the microwave and infrared algorithms produces a small improvement. Since neural networks have a high capacity for learning, perhaps the same network structure used in both cases can be applied to train a single neural network for both infrared data and microwave data simultaneously. The network may be able to accommodate clouds and rain either with no change to the net or by adding a few more nodes to the hidden layers. It would be interesting to know if the network can be robust without cloud detection.

- A more detailed examination of the effects of the aircraft altitude fluctuations and of the removal of the infrared channels whose first moment lies above 200 mbar on the retrievals between 100–200 mbar could explain why the retrievals are so noisy in this region. If the ultimate goal is to produce an algorithm which can be used on a satellite, then characterizing differences between an aircraft and a satellite is important.
Appendix A

Software Source Code Selections

A.1 Saturation Vapor Pressure

```
function svp = sat_vap_press(T)
    %[svp]=sat_vap_press(T)
    % function for calculating saturation vapor pressure
    % based on Phil Rosenkranz fortran subroutine
    % T is in Kelvin

    Z=2.302585;
    TS=373.16;
    TICE=273.16;
    spv=zeros(size(T));

    for i=1:size(T,1)
        for j=1:size(T,2)
            if T(i,j)<233.16
                ei1 = -9.0971*(TICE/T(i,j)-1)*Z;
                ei2 = -3.56654*log(TICE/T(i,j));
                ei3 = 0.876793*(1.0-T(i,j)/TICE)*Z;
                svp(i,j) = exp(ei1+ei2+ei3)*6.1071;
            else
                es1 = -7.90298*(TS/T(i,j)-1)*Z;
            end
        end
    end
```
es2 = 5.02808*\log((TS/T(i,j))); 

es3 = (-1.3816e-7)*(10^{-11.344*(1-T(i,j)/TS)})-1)*Z; 

es4 = (8.1328e-3)*(10^{-3.49149*(TS/T(i,j)-1)})-1)*Z; 

svp(i,j) = \exp(es1+es2+es3+es4)*1013.246; 

end; 

end; 

end;
A.2 Simulation

A.2.1 Microwave

function [coeff] = TB_Sim(Inst, outputIndex, satigr)

% function [coeff] = TB_Sim(Inst, outputIndex, satigr)
%
% Inst.freqs vector of frequencies to compute in GHz
% .freqWeights matrix of weights, col is .freq, row is channel
% .scanAngles vector of spot angles (deg) within a scan
% .pencilAngles vector of evenly spaced angles (deg) about a scanAngle to average over
% .pencilWeights vector of weights corresponding to .pencilAngles
%
% outputIndex vector of indices for the raob levels that determine where the outputs 10
% are to be taken from. each index is the top of
% the particular slab. i.e. 23 represents the top of the slab
% computed through raob level 23. The satigr database assumes that
% level 1 is near the ground, and 40 towards infinity. The indices
% must be sorted ground first.
%
% Output coeff.TU surface to radiometer upwelling component
% .TD sky to surface component as seen at output
% .E surface to radiometer atmospheric emissivity
% dimension: chan X raob X scanAngle X outputNumber

version = '/usr/users/vleslie/RETRIEVALS/RADTRANS/v1.1';
addpath(version);

NOutputs = length(outputIndex);
NRaobs = size(satigr,3); %raobs from the satigr database.
NChans = size(Inst.freqWeights,1);
NScanAngles = length(Inst.scanAngles);
NPencilAngles = length(Inst.pencilAngles);
Nlevels = size(satigr,2);                %number of slabs in the raob
W = ones(length(Inst.freqs),1) * Inst.pencilWeights; %weight matrix for antenna pattern
dA = mean(diff(Inst.pencilAngles));             %delta angle for antenna pattern

SEC = sec( (Inst.pencilAngles' * ones(1,NScanAngles) + ones(NPencilAngles,1)) * Inst.sca;

% allocate memory
coeff.E = zeros(NChans,NRaobs,NScanAngles,NOutputs);
coeff.TU = zeros(NChans,NRaobs,NScanAngles,NOutputs);
coeff.TD = zeros(NChans,NRaobs,NScanAngles,NOutputs);

% determine atmospheric regions, ground to sky (1=ground).
lastLevel = 1;
j=1;
for i=1:length(outputIndex)
    if(outputIndex ~= 1 & outputIndex ~= Nlevels)
        region{j} = lastLevel:outputIndex(i);
        lastLevel = outputIndex(i);
        j = j+1;
    end
end
region{j} = lastLevel:Nlevels;

for m = 1:NRaobs
    for n = 1:NScanAngles
        fprintf('Working on Raob %d... Scan Angle %d\r',m,n);
            % Compute coefficients for each region.
        for k = 1:length(region)
            [tbDown{k}, tbUp{k}, cDown{k}, eUp{k}] = ...
            tbararray(satigr(2,region{k},m)',satigr(3,region{k},m)',satigr(4,region{k},m)', ...
                  zeros(length(region{k}),1),zeros(length(region{k}),1),SEC(:,n),SEC(:,n),Inst.freqs');
% Compute Downwelling as seen from upward looking radiometer at the ground
TDa = 0;
for i = 1:length(region)+1
    % compute emissivity
    e = 1;
    for j = 1:length(region)-i+1
        e = e .* eDown{j};
    end

    % compute TB contribution
    if(i == 1)
        TDa = TDa + e .* (ones(NPencilAngles,1) * Tcosmic(Inst.freqs));
    else
        TDa = TDa + e .* tbDown{length(region)+2-i}; % verify
    end
end

% Compute coefficients at output points.
for i = 1:length(outputIndex)
    % Downwelling component from sky to ground, Emissivity of 1.
    e = 1;
    for j = 1:i
        e = e .* eDown{j};
    end
    coeff.TD(:,m,n,i) = Inst.freqWeights * trapz( W .* e' .* TDa', 2) * dA;

    % Upwelling component from surface to output point
    % eqn: $T_b = T(i) + \sum_{j=2}^i \prod_{k=j}^i e(k) \prod_{k=1}^{j-1} e(k) T(j-1)$
\[ T = \text{tbUp}\{i\} \]

\[ \text{for } j = 2:i \]
\[ e = 1; \]
\[ \text{for } k = j:i \]
\[ e = e \cdot \text{eUp}\{k\}; \]
\[ \text{end} \]
\[ T = T + e \cdot \text{tbUp}\{j-1\}; \]
\[ \text{end} \]
\[ \text{coeff.TU}(;m,n,i) = \text{Inst.freqWeights} \cdot \text{trapz}( W \cdot T', 2) \cdot dA; \]

\% Emissivity of atmosphere from surface to output point
\[ e = 1; \]
\[ \text{for } j = 1:i \]
\[ e = e \cdot \text{eUp}\{j\}; \]
\[ \text{end} \]
\[ \text{coeff.E}(;m,n,i) = \text{Inst.freqWeights} \cdot \text{trapz}( W \cdot e', 2) \cdot dA; \]
\[ \text{end} \]

\[ \text{coeff.Inst} = \text{Inst}; \]
\[ \text{coeff.OutputLevels} = \text{outputIndex}; \]
\[ \text{coeff.TbarrayVersion} = \text{version}; \]
function Tcos=Tcosmic(f, T)

%function Tcos=Tcosmic(f, T)

%returns "corrected" high-temp linear fit to Planck Intensity

% f frequency in GHz

% T optional physical temperature (default 2.7K)

if exist('T')~=1, T=2.7; end;

c = 3e8; % (m/s)

h = 6.626e-34; % (Js)

k = 1.3806e-23; % (J/K)

L=c./f;

f=f.*1000000000;

x=h.*f./2./k./T;

Tcos=T.*x./tanh(x);
function [F54, p54, F118, p118] = nastm_passband_load(LO_54GHz, LO_118GHz)
% Load 54-GHz passbands

HEADERLINES = 11;
NUMFREQ = 601;
NUMCHANS = 8;

if (nargin == 0)
    LO_118GHz = 118.7595; % measured 27 mar 99;
    LO_54GHz = 45.995; % measured 27 mar 99;
end

fprintf('Using %6.4f GHz for the 54-GHz LO, and %7.4f GHz for the 118-GHz LO.\n',LO_54
ch54 = zeros(NUMCHANS,2,NUMFREQ);

for j = 1:NUMCHANS
    eval(['fid = fopen('/usr/sounder1/NAST/NAST-M/WINTEX/Passbands/54GHz/c' num2str(j) ']
    for i = 1:HEADERLINES
        fgetl(fid);
    end

    ch54(j,:, :) = fscanf(fid,'%g',[2,NUMFREQ]);

    ch54(j,2,:) = 10.^(squeeze(ch54(j,2,:))/10);
    min_value = min(squeeze(ch54(j,2,:)));
    min_index = find(squeeze(ch54(j,2,:)) == min_value);
    ch54(j,2,min_index) = 0;
    ch54(j,2,:) = ch54(j,2,:)/sum(squeeze(ch54(j,2,:)));

fclose(fid);

148
fprintf('Finished loading passband characteristics of 54-GHz channel %d.
', j);
end

% Get ideal passbands ..
[F54, p54, F118, p54_] = passnastm(LO_54GHz, LO_118GHz);

for j = 1:NUMCHANS
    p54(1,:,:) = interp1(squeeze(ch54(j,1,:)) / lo9 + LO_54GHz, squeeze(ch54(j,2,:)), F54);
    p54(find(isnan(p54(j,:)))) = 0;
    p54(j,:) = p54(j,:)/sum(p54(j,:));  % renormalize
end

% Load 118-GHz passbands
HEADERLINES = 11;
NUMFREQ = 601;
NUMCHANS = 9;

ch118 = zeros(NUMCHANS,2,NUMFREQ);

for j = 1:NUMCHANS
    eval(['fid = fopen(''/usr/sounder1/NAST/NAST-M/WINTEX/Passbands/118GHz/118_ch' num2st for i = 1:HEADERLINES
    fgetl(fid);
end

ch118(j,:,:) = fscanf(fid,'%g',[2,NUMFREQ]);

ch118(j,2,:) = 10.^(squeeze(ch118(j,2,:))/10);
min_value = min(squeeze(ch118(j,2,:)));
min_index = find(squeeze(ch118(j,2,:)) == min_value);
ch118(j,2,min_index) = 0;

ch118(j,2,:) = ch118(j,2,:)/sum(squeeze(ch118(j,2,:)));

fclose(fid);

fprintf('Finished loading passband characteristics of 118-GHz channel %d\n', j);

end

% need to create double sideband

F118_ = F118;

F118_(1:800) = fliplr(F118(802:1601));

for j = 1:NUMCHANS

  p118(j,:) = interp1(squeeze(ch118(j,1,:))/1e9+LO_118GHz,squeeze(ch118(j,2,:)),F118_);

  p118(j,find(isnan(p118(j,:)))) = 0;

  p118(j,:) = p118(j,:)/sum(p118(j,:)); % renormalize

end
function [F54, p54, F118, p118] = passnastm(LO_54GHz, LO_118GHz)
% This function makes passband filter matrices for NAST-M
% p54 and p118 are sparse matrices that can be multiplied with
% matrices in frequency stepped across the bands F118 and F54
% to give channel averages

if (nargin == 0)
    LO_118GHz = 118.7595; % measured 27 mar 99;
    LO_54GHz = 45.995; % measured 27 mar 99;
end

F0_118=LO_118GHz;
F1_118= [.12 .235 .45 .8 1.2 1.6 2.05 2.55 3.5];
Wid118= [.1 .13 .3 .4 .4 .4 .5 .5 1];

F118=LO_118GHz + [-4:0.005:4];
p118=passdsb(F118,F0_118,F1_118,Wid118);

%F54=[50.300 51.760 52.800 54.750 54.400 54.940 55.500 56.020];

% use double-sideband values, then restrict to half band
F54=LO_54GHz + [-11:0.005:11];
F0_54=LO_54GHz;
F1_54= [4.300 5.760 6.800 7.750 8.400 8.940 9.500 10.020];
% tweak last two widths to keep from having getting one endpoint
% but not the other for odd width in steps of 10-MHz
Wid54= [0.180 0.400 0.400 0.240 0.400 0.400 0.331 0.271];

p54=passdsb(F54,F0_54,F1_54,Wid54);
F54=F54(3001:4401);

p54=p54(:,3001:4401)*2;

%nz54=find(sum(p54));

%F54_=F54(nz54);

%p54_=p54(:,nz54);
SUBROUTINE TBARRAY(NLEV, TEMP, PRES, H2OVAPOR, H2OLIQUID, OZONE, & NANG, SECANT1, SECANT2, FREQ, TB1, TB2, E1, E2)

C $Id: tbarray.fv 1.1 2000/08/02 01:26:31 vieslie Exp root$
C COMPUTES MICROWAVE EMISSION AND TRANSMISSION FOR AN ATMOSPHERIC
C PROFILE FOR TWO PATHS, AT MULTIPLE ANGLES FOR EACH PATH:
C PATH 1: PROPAGATION FROM LEVEL NLEV TO LEVEL 1
C PATH 2: PROPAGATION FROM LEVEL 1 TO LEVEL NLEV
C
C REFLECTION AT BOUNDARIES AND THE COSMIC BACKGROUND ARE TO BE
C CONSIDERED IN THE CALLING PROGRAM, E.G. FOR AN OBSERVER AT LEVEL 1
C WITH SURFACE AT LEVEL NLEV,
C TB = TB1 + E1* ( (1.-Refl)*Tsurf + Refl*(TB2 + E2*TBcosmic) );
C OR, FOR AN OBSERVER AT LEVEL NLEV WITH SURFACE AT LEVEL 1,
C TB = TB2 + E2* ( (1.-Refl)*Tsurf + Refl*(TB1 + E1*TBcosmic) );
C
C IMPLICIT NONE
C
C ARGUMENTS
C Note: H2OLIQUID and OZONE can be set to zeros if not needed.
C For specular reflection at the surface, SECANT1 and
C SECANT2 would be identical.
C
C INPUTS

INTEGER NLEV !no. of atmospheric levels
REAL TEMP(NLEV) !temperature (K)
REAL PRES(NLEV) !pressure (hPa)
REAL H2OVAPOR(NLEV) !H2O vapor density (g/m**3)
REAL H2OLIQUID(NLEV)!H2O liquid density (g/m**3)
REAL OZONE(NLEV) !O3 number density (molecules/m**3)
INTEGER NANG !no. of angles
REAL SECANT1(NANG) !secant of propagation angle along path 1
REAL SECANT2(NANG) ! secant of propagation angle along path 2
REAL FREQ    ! frequency (GHz)

C OUTPUTS
REAL TB1(NANG) ! brightness temperature (K) for path 1; e.g.,
c   emerging from the atmosphere at level 1
REAL TB2(NANG) ! brightness temperature (K) for path 2; e.g.,
c   emerging from the atmosphere at level NLEV
REAL E1(NANG) ! total transmittance for path 1
REAL E2(NANG) ! total transmittance for path 2

C Returned transmittance values will always be > 1.e-10

C LOCAL VARIABLES
INTEGER I,J
REAL EM,TAV,PAV,WVAV,WLAV,O3AV,ABSCOEF,OPACITY,TRAN_SLAB
REAL O2ABS,ABH2O,ABLIQ,ABSN2,ABSO3,RG

DO J=1,NANG
      C initialize at each angle
      TB1(J) = 0.
      TB2(J) = 0.
      E1(J) = 1.
      E2(J) = 1.
END DO

DO 20 I=2,NLEV
      C use the 'absorption-of-averages' method to compute optical
c   depth of each slab; see M.J. Schwartz, Ph.D. thesis pp. 84-87.
      TAV = (TEMP(I) + TEMP(I-1))/2.
      PAV = SQRT(PRES(I)*PRES(I-1))
      WVAV = (H2OVAPOR(I) + H2OVAPOR(I-1))/2.
      WLAV = (H2OLIQUID(I) + H2OLIQUID(I-1))/2.

20 CONTINUE
O3AV = (OZONE(I) + OZONE(I-1))/2.

ABSCOEF = O2ABS(TAV,PAV,WVAV,FREQ) +
& ABH2O(TAV,PAV,WVAV,FREQ) + ABSN2(TAV,PAV,FREQ) +
& ABLIQ(WLAV,FREQ,TEMP) + ABSO3(TAV,PAV,O3AV,FREQ)

RG = .0293*(1. + .00174*WVAV*TAV/PAV)

OPACITY = ABSCOEF*RG*TAV*ABS(ALOG(Pres(I)/Pres(I-1)))

C

DO 10 J=1,NANG
C trace path 1 using integral form of RTE

IF(E1(J) .GT. 1.E-10) THEN

EM = E1(J)
E1(J) = E1(J)*EXP(-SECANT1(J)*OPACITY)
TB1(J) = TB1(J) + TAV*(EM-E1(J))

ENDIF
C

C trace path 2 using differential form of RTE

TRAN_SLAB = EXP(-SECANT2(J)*OPACITY)
TB2(J) = TAV + TRAN_SLAB*(TB2(J)-TAV)
IF(E2(J) .GT. 1.E-10) E2(J) = E2(J)*TRAN_SLAB
C
10 CONTINUE
20 CONTINUE

RETURN

END
A.2.2 Infrared

PROGRAM NIRTE2

C NAME: nastirte2.f

C PURPOSE: NAST-I Forward Calculation Routine

C CATEGORY:

C CALLING SEQUENCE: nastirte < nastirte.inp > nastirte.out

C INPUTS: Input file is “output of raob2ac145.f”

C see nastirte.inp “input file for an example”

C OUTPUTS: forward calculation of brightness temperature for a given

C input sounding on the NAST-I wavenumber grid

C jbh: Output is 3 coefficients per raob and frequency.

C TD(Upwelling), TU(Downwelling), and E(one-way transmittance)

C COMMON BLOCKS: none

C CALLS:

C SIDE EFFECTS: none

C RESTRICTIONS: none

C PROCEDURE: n/a

C EXAMPLE:

C MODIFICATION HISTORY:
Modified by: Jay Hancock, Feb 14, 2001
Written by: Jun Li, Feb 7, 1998
based off code by: Hal Wolf and others, Feb 7, 1998

REVISION INFORMATION:

'$Id: nastirte.fv 1.1.1.1 1999/01/25 15:35:36 chriss Exp $'
'$Log: nastirte.f,v $'
'Revision 1.1.1.1 1999/01/25 15:35:36 chriss'
'Imported Source

OTHER INFORMATION:

Res IFGM scan regions (cm) in band 1; band 2; band 3
0-2.07 (4096); 0-2.07(4096); 0-2.07(4096)

Parameter and Dimension declarations

NB = number of bands (3)
NL = number of levels (40)
MAXCHA = maximum number of channels (2968)
TCSM is the background cosmic radiation in degrees K.

PARAMETER (NB=3,NL=40,MAXCHA=2968,TCSM=3.0)
PARAMETER (LENG=145)
frequency list information, double precision

REAL*8 VN,DWN(NB),FREQ(NB,MAXCHA),VXNAST
REAL TU,TD
DIMENSION BUF(LENG)
DIMENSION TAUTUP(NL),TAUTDN(NL),NBUSE(NB),RRMIN(NB)

store spectral information variables

DIMENSION KDO(NB),NCH(NB)
DIMENSION TUOUT(NB,MAXCHA)
DIMENSION TDOUT(NB,MAXCHA)
DIMENSION EMOUT(NB,MAXCHA)
DIMENSION P(NL),T(NL),W(NL),O(NL)

C LUO, LUIA,B output and input LUNs
DIMENSION LUO(NB),LUIA(NB),LUIB(NB)
COMMON /taudwo/TAUD(NL),TAUW(NL),TAUZ(NL)
CHARACTER*64 TBF(NB),PROFF

DATA KDO/NB*1/

C LUO output LUN numbers
DATA LUO/11,12,13/,LUI/15/

C LUIA,B input LUN numbers
DATA LUIA/24,25,26/,LUIB/27,28,29/

C NCH = number channels per band
DATA NCH/2718,2946,2968/
DATA RRMIN/1.,0.01,0.0001/

C Press levels (40 levels altogether)
DATA P/50.,60.,70.,75.,80.,75.,80.,85.,90.,100.,125.,150.,175.,200.,
* 250.,300.,350.,400.,450.,500.,550.,600.,620.,640.,660.,680.,
* 940.,960.,980.,1000./

C Prompt for user input
WRITE(*,'('' ENTER INPUT PROFILE FILE : '' )')
READ(*,'(A)') PROFF
WRITE(*,*) PROFF
WRITE(*,'('' ENTER BEGINING NUMBER OF RECORD TO PROCESS : '' )')
READ(*,*) NBEG
WRITE(*,*) NBEG
WRITE(*,'('' ENTER ENDING NUMBER OF RECORD TO PROCESS : '' )')
READ(*,*) NREC
WRITE(*,*) NREC
write(*,'('' ENTER A/C LEVEL PRESSURE (MB) : '' )')
READ(*,*) ACP
WRITE(*,*) ACP
WRITE(*,')(' ENTER BAND USE FLAG (0 for NOT USED) ':')'
READ(*,*) NBUSE
WRITE(*,*) NBUSE
WRITE(*,')(' ENTER OUTPUT Brightness Tem FILE (all 3 Bands): ')'

READ(*,') TBF(1)
WRITE(*,*) TBF(1)
READ(*,') TBF(2)
WRITE(*,*) TBF(2)
READ(*,') TBF(3)
WRITE(*,*) TBF(3)

C initialize NAST wavenumbers
VN=VXNAST(0,0)

C loop through and retrieve all wavenumbers
DO N=1,NB
  DO K=1,NCH(N)
    FREQ(N,K)=VXNAST(N,K)
  ENDDO
  DWN(N)=FREQ(N,2)-FREQ(N,1)
  WRITE(*,'(' BAND '',12,'','' WNBEG;WNEND;DWN&NCH ''',3F9.2,i6)')
  $ N,FREQ(N,1),FREQ(N,NCH(N)),DWN(N),NCH(N)
ENDDO

C open up the input file to be read in
LENI=LENG*4
OPEN(LUI,RECL=LENI,FILE=PROFF,STATUS='OLD',ACCESS='DIRECT')
DO N=1,NB
IF(NBUSE(N).NE.0) THEN
  C *** fix the length for each record: will change because I am writing more
  C records now. NCH(N)*4 original length, times 3 coefficients
  LENO=NCH(N)*4*3
C open up the output file

    OPEN(LUO(N),RECL=LENO,FILE=TBF(N),STATUS='UNKNOWN',
$ ACCESS='DIRECT')

ENDIF
ENDDO
NRX=0
DO 180 NR=NBEG,NREC
READ(LUI,REC=NR) (BUF(J),J=1,LENG)

C assign profile variables
    DO L=1,NL
         T(L)=BUF(L)
         W(L)=BUF(NL+L)
         O(L)=BUF(2*NL+L)
    ENDDO

C skin press and surface temperature (skin) assignment
    TSKIN=BUF(121)
    PSFC=BUF(122)
    if(PSFC.gt.1000.) PSFC=1000.
    LSFC=lsurface ( nl, p, psfc, 700., 1000. )
    AZEN=BUF(140)

C output profile information
    WRITE(*,'( RECORD='',I8,'' Psfc;TSKIN;ZEN;Lsfc='',3f9.3,i6)')
$ NR,PSFC,TSKIN,AZEN,LSFC
    IF(MOD(NR,100).EQ.1) THEN
        WRITE(*,'( pressure temperature water vapor ozone''))
        DO l=1,nl
            WRITE(*,'(1x ,f8. 1,2x ,f11.2, 2x,f11.3,2x,f11.4)')
$ p(l),t(l),w(l),o(l)
        ENDDO
    ENDIF

C N is the band, K is channel in the band
    DO 150 N=1,NB
IF(NBUSE(N).NE.0) THEN
  DO 130 K=1,NCH(N)
  VN=FREQ(N,K)
  C DO THE RADIATIVE-TRANSFER CALCULATIONS
  CALL TRANNAST(N,K,KDO,ACP,AZEN,T,W,O, TAUUP,*200)
  CALL TRANNAST2(N,K,KDO,ACP,AZEN,T,W,O, TAU TDN,*200)
  CALL WNMRAD2(VN,TAUUP,TAU TDN,T,LSFC, NL, TCSM, TU,
  +   TD, EMOUT(N,K))
  C *** what to do here? this is a lower limit...
  C IF(RR.LE.RRMIN(N)) THEN
  C   RR=RRSAV
  C ELSE
  C   RRS AV=RR
  C ENDIF
  TUOUT(N,K)=TU
  TDOUT(N,K)=TD

if(nr.eq.1) write(*, ' (lx ,2i7 ,F9 .3, F9 .3,F9 .3,F14.10,F14.10)')n,k,VN,
+   TDOUT(N,K), TUOUT(N,K), EMOUT(N,K), TAU TDN(1)

130 CONTINUE
  ENDIF
150 CONTINUE

NRX=NRX+1
  DO 170 N=1,NB
  IF(NBUSE(N).NE.0) THEN
    WRITE(LUO(N),REC=NRX) (TDOUT(N,K), TUOUT(N,K),
  + EMOUT(N,K), K=1, NCH(N))
  IF(MOD(NRX,100).EQ.1) THEN
    WRITE(*,'(RECORDER=','i8,, TSK IN=',f9.3)') NRX,TSK IN
    WRITE(*,'(1x,9f9.3)') (TDOUT(N,K),K=1,9)
  ENDIF

WRITE(*, '(1X,9F9.3)') (TDOUT(N,K), K=NCH(N)-8,NCH(N))
WRITE(*, '(1X,9F9.3)') (TUOUT(N,K), K=1,9)
WRITE(*, '(1X,9F9.3)') (TUOUT(N,K), K=NCH(N)-8,NCH(N))
WRITE(*, '(1X,9F9.3)') (EMOUT(N,K), K=1,9)
WRITE(*, '(1X,9F9.3)') (EMOUT(N,K), K=NCH(N)-8,NCH(N))
ENDIF
ENDIF

170 CONTINUE

180 CONTINUE

GO TO 300

200 CONTINUE

WRITE(*, '<' ERROR IN READING TRANNAST !''>')

300 CONTINUE

DO N=1,NB

IF(NBUSE(N).NE.0) THEN

WRITE(*, '(1X,I6,'' RECORDS WROTE TO FILE ''A48)') NRX,TBF(N)

CLOSE(LUO(N))

ENDIF

ENDDO

STOP

END
subroutine taudoc2(nc,nx,ny,cc,xx,tau)

* Strrow-Woolf model ... for dry, ozo(ne), and wco (water-vapor continuum)

.... version of 02.02.98

revision history

'Revision 1.1.1.1 1999/01/25 15:35:36 chriss

Imported Source

C

dimension cc(nc,ny),xx(nx,ny),tau(*)

data trap/-999.99/

taul=1.

tau(ny+1)=taul

do 100 j=ny,1,-1

if(taul.eq.0.) go to 100

yy=cc(nc,j)

if(yy.eq.trap) then

  taul=0.

  go to 100
endif

do i=1,nx

  yy=yy+cc(i,j)*xx(i,j)
endo
tauy=taul*exp(-yy)
taul=amin1(tauy,taul)

100 tau(j)=taul

return
subroutine wnmrad2(vn,tauup,taudn,tem,ls,
+ siztau,tcosmc,tu,td,em)

* monochromatic radiance calculation

revision history

'$Id: wnmrad.fv 1.1.1.1 1999/01/25 15:35:36 chriss Exp $'

'$Log: wnmrad.fv $

'Revision 1.1.1.1 1999/01/25 15:35:36 chriss

'Imported Source

''

Input arguments:

vn - frequency

tau - vector of one-way transmittances for a slab of the atmosphere.

tem - vector of temperatures corresponding to tau calculations.

ls - calculate upwelling E and TD up to this layer.

siztau - size of tau and tem

tcosmc - cosmic background radiation.

Output arguments:

td - upwelling component terminating at layer ls.

tu - downwelling component from space to surface.

em - one-way transmittance from surface to layer ls.

real*8 vn

real tu, td, em

integer ls, siztau

dimension tauup(*),taudn(*),tem(*)

C need to fix declarations: why isn't ls and ts declared in the original code?
C declare: ls, siztau, tcosmc, tu, td, e

C Compute TD and EM:
taul = tauup(1)
t1 = tem(1)
b1 = wnplan(vn, t1)
td = 0.
do i = 2, ls
    tau2 = tauup(i)
t2 = tem(i)
b2 = wnplan(vn, t2)
    td = td + 0.5*(b1 + b2)*(taul - tau2)
taul = tau2
    bl = b2
endo
cm = taul

C Compute TU:
    taul = taudn(siztau)
t1 = tem(siztau)
b1 = wnplan(vn, t1)
tu = wnplan(vn, tcosmc)
do j = siztau - 1, 1, -1
    tau2 = taudn(j)
t2 = tem(j)
b2 = wnplan(vn, t2)
tu = tu + 0.5*(b1 + b2)*(taul - tau2)
C maybe need a sign?
    taul = tau2
    bl = b2
endo

return
end
subroutine trannast2(iban,ipib,kcom,apre,azen,
  *          temp,wvmr,ozmr, taut,*)

c * NAST-I dry/wet/ozo transmittance

c .... version of 22.12.98

c * band    wbeg  wend  npts

  c 1  645  1300  2718
  c 2 1290 2000  2946
  c 3 1985 2700  2968

  c iban = band
  c ipib = point within band ... if out of range,
  c          all tau-arrays are filled with 1.0
  c kcom = component-transmittance switch ... see NOTE 2
  c apre = aircraft flight-level pressure ... see NOTE 1
  c azen = local zenith angle in degrees
  c temp = temperature profile (degK)
  c wvmr = water-vapor mixing-ratio profile (g/kg)
  c ozmr = ozone mixing-ratio profile (ppmv)
  c taut = total transmittance .............. see NOTE 3
  c * = alternate return, taken if file(s) not available

  c * Strow-Woolf-VanDelst regression model based on LBLRTM line-by-line transmittances.
  c Input temperatures, and water-vapor and ozone mixing ratios, must be defined
  c at the pressure levels in array 'pstd' ... see 'block data acrefatm'.

  c Logical unit numbers 40-99 are used for coefficient files ... see NOTE 1.

  c * NOTE 1
  c There are four sets of coefficient files, corresponding to
  c   'top-of-the-atmosphere' pressures of 50, 60, 70, and 75 mb.
'apre', the pressure in millibars at the aircraft altitude, is used to select the appropriate set of files.

*NOTE 2*

\(kcom = 0 \Rightarrow \) no contribution from component \(\Rightarrow \tau(p) = 1.0\)

\(1 \Rightarrow \) calculate contribution from given distribution

\(kcom(1)\) is for dry, result in 'taud'

\(kcom(2)\) is for wet, result in 'tauw'

\(kcom(3)\) is for ozo, result in 'tauo'

NOTE 3

Component tau's are returned through common;
their product is returned in 'taut'.

Revision history

'$Id: trannast.f, v 1.1.1.1 1999/01/25 15:35:36 chriss Exp$'

'$Log: trannast.f, v$

'Revision 1.1.1.1 1999/01/25 15:35:36 chriss

Imported Source

\begin{verbatim}
parameter (nb=3,nk=3,nk=5,nl=40,nm=nl-1,nt=4,lfac=4)
parameter (nxc=4,ncc=nxc+1,lencc=ncc*nm,lencb=lencc*lfac)
parameter (nxd=8,ncd=nxd+1,lencd=ncd*nm,lencdb=lencd*lfac)
parameter (nxo=9,nco=nxo+1,lencb=nco*nm,lencob=lenco*lfac)
parameter (nxl=2,ncl=nxl+1,lencb=ncl*nm,lencdb=lencl*lfac)
parameter (nxw=nxl+nxs)
common/stdatm/pstd(nl),tstd(nl),wstd(nl),ostd(nl)
common/taudwo/taud(nl),tauw(nl),tauo(nl)
dimension kcom(*),temp(*),wvmr(*),ozmr(*),taut(*)
dimension coefd(ncd,nm),coefo(nco,nm),coeff(ncl,nm)
\end{verbatim}

168
dimension coefs(ncs,nm),coefc(ncc,nm)
dimension pavg(nm),tref(nm),wref(nm),oref(nm)
dimension tavg(nm),wamt(nm),oamt(nm),secz(nm)
dimension tauc(nl),tlas(nl),wlas(nl),olas(nl)
dimension xdry(nxw,nm),xozo(nxo,nm),xwet(nxw,nm),xcon(nxc,nm)
dimension iud(nb,nt),iuc(mk,nb,nt)
character*12 cfile/'nastbcom.tpp'/
character*3 comp(mk)/'dry','ozo','wts','wtl','wco'/
character*2 ctop(nt)/'50','60','70','75'/
character*1 cban
integer kend(nb)/2718,2946,2968/
integer kind(nk)/1,3,2/
integer length(mk)/lencdb,lencob,lencsb,lenclb,lenccb/
data init/1/,iud/40,45,50,55,60,65,70,75,80,85,90,95/
logical here,newang,newatm,openc(mk,nb,nt)
data tlas/nl*0./,wlas/nl*0./,olas/nl*0./,zlas/-999./
secant(z)=1./cos(0.01745329*z)
if(init.ne.0) then
  do k=1,nt
    do j=1,nb
      iux=iud(j,k)-1
      do i=1,mk
        openc(i,j,k)=.false.
        iux=iux+1
        iuc(i,j,k)=iux
      enddo
    enddo
  enddo
enddo
enddo
call conpir(pstd,tstd,wstd,ostd,nl,1,pavg,tref,wref,oref)
init=0
endif
* use aircraft pressure to select coefficient files

```fortran
if(apre.lt.60.) then
   itop=1
elseif(apre.ge.60. .and.apre.lt.70.) then
   itop=2
elseif(apre.ge.70. .and.apre.lt.75.) then
   itop=3
else
   itop=4
endif

write(cban,'(11)') iban
cfile(05:05)=cban

cfile(11:12)=ctop(itop)
do 100 icom=1,mk
   if(openc(icom,iban,itop)) go to 100
   openc(icom,iban,itop)=.true.
cfile(6:8)=comp(icom)
inquire(file=cfile,exist=here)
   if(.not.here) go to 200
   iucc=iuc(icom,iban,itop)
   lenf=length(icom)
WRITE(*,'(1XI6,"opening coefficient file: ",A15)') 0,
cfile(open(iucc,file=cfile,recl=lenf,access='direct',status='old',
   * err=200)
WRITE(*,'(" file successfully opened")')
100 continue

dt=0.
dw=0.
do=0.
```
do j=1, nl
    dt=dt+abs(temp(j)−tlas(j))
    tlas(j)=temp(j)
    dw=dw+abs(wvmr(j)−wlas(j))
    wlas(j)=wvmr(j)
    do=do+abs(ozmr(j)−olas(j))
    olas(j)=ozmr(j)
    taud(j)=1.0
    tauw(j)=1.0
    tauc(j)=1.0
    tauo(j)=1.0
    taut(j)=1.0
  enddo

  datm=dt+dw+do
  newatm=datm.ne.0.
  if(newatm) then
    call conpir(pstd,temp,wvmr,ozmr,nl,1,pavg,tavg,wamt,oamt)
  endif

  newang=azen.ne.zlas
  if(newang) then
    zsec=secant(azen)
    do l=1,nm
      secz(l)=zsec
    enddo
    zlas=azen
  endif

  if(newang.or.newatm) then
    call calpir(tref,wref,oref,tavg,wamt,oamt,pavg,secz,
*     nm,nxd,nxw,nxo,nxc,xdry,xwet,xozo,xcon)
  endif
if(ipib.gt.kend(iban)) return
krec=ipib

* dry
WRITE(*,'(" reading dry ...")')
l=1
if(kcom(l).ne.0) then
  k=kind(l)
  read(iuc(k,iban,itop),rec=krec)((coefd(ij),i=1,ncd),j=1,nm)
call taudoc2(ncd,nxd,nm,coefd,xdry,taud)
endif
WRITE(*,'(" successfully read dry. ")')

* ozo
WRITE(*,'(" reading ozo ...")')
l=3
if(kcom(l).ne.0) then
  k=kind(l)
  read(iuc(k,iban,itop),rec=krec)((coefo(ij),i=1,nco),j=1,nm)
call taudoc2(nco,nxo,nm,coefo,xozo,tauo)
endif
WRITE(*,'(" successfully read ozo. ")')

* wet
WRITE(*,'(" reading wet ...")')
l=2
if(kcom(l).ne.0) then
  k=kind(l)
  read(iuc(k,iban,itop),rec=krec)((coefs(ij),i=1,ncs),j=1,nm)
k=k+1
  read(iuc(k,iban,itop),rec=krec)((coefl(ij),i=1,ncl),j=1,nm)
other than continuum
   call tauwtr2(ncs,ncl,nxs,nxl,nxw,nm,coefs,coefl,xwet,tauw)
   k=k+1
   read(iuc(k,iban,itop),rec=krec)((coefc(i,j),i=1,ncc),j=1,nm)
continuum only
   call tauDOC2(ncc,nxc,nm,coefc,xcon,tauc)
total water vapor
   do j=1,nl
      tauw(j)=tauw(j)*tauc(j)
   enddo
endif
WRITE(*,'(" successfully read wet. ")')
total
   do j=1,nl
      taut(j)=taud(j)*tauo(j)*tauw(j)
   enddo
return
end
C THIS DATA DECLARED GLOBAL IN TRANNAST.F
block data acrfatm
parameter (nl=40)
* Reference Atmosphere is 1976 U.S. Standard
common/stdatm/pstd(nl),tstd(nl),wstd(nl),ostd(nl)
data pstd/50.,60.,70.,75.,80.,85.,90.,100.,125.,150.,175.,200.,
   250.,300.,350.,400.,450.,500.,550.,600.,620.,640.,660.,680.,
   940.,960.,980.,1000./
data tstd/
   + 217.28, 216.70, 216.70, 216.70, 216.70, 216.70, 216.70, 216.70,
$C + 216.70, 216.70, 216.71, 216.72, 220.85, 228.58, 235.38, 241.45,$

$C + 246.94, 251.95, 256.58, 260.86, 262.48, 264.08, 265.64, 267.15,$

$C + 268.61, 270.07, 271.49, 272.87, 274.21, 275.54, 276.85, 278.12,$

$C + 279.36, 280.59, 281.79, 282.97, 284.14, 285.28, 286.40, 287.50/$

$C \text{ data wstd/}$

$C + 0.002, 0.002, 0.002, 0.002, 0.002, 0.002, 0.002,$

$C + 0.003, 0.005, 0.010, 0.014, 0.035, 0.089, 0.211, 0.331,$

$C + 0.500, 0.699, 0.961, 1.348, 1.368, 1.525, 1.678, 1.825,$

$C + 1.969, 2.170, 2.365, 2.556, 2.741, 2.925, 3.105, 3.280,$

$C + 3.456, 3.633, 3.806, 3.975, 4.162, 4.346, 4.525, 4.701/$

$C \text{ data ostd/}$

$C + 2.86792, 2.29259, 1.80627, 1.62277, 1.45112, 1.28988, 1.16673, 0.93973,$

$C + 0.63214, 0.46009, 0.36957, 0.29116, 0.16277, 0.09861, 0.06369, 0.05193,$

$C + 0.04434, 0.03966, 0.03744, 0.03474, 0.03384, 0.03367, 0.03350, 0.0334,$

$C + 0.03319, 0.03301, 0.03283, 0.03266, 0.03249, 0.03196, 0.03145, 0.03095,$

$C + 0.03042, 0.02986, 0.02931, 0.02878, 0.02829, 0.02781, 0.02735, 0.02689/$

$C \text{ end}$
A.3 Neural Network Training

A.3.1 Network Control Code

```matlab
function expXXX(thisRunID, net_, epochs_)

% single argument is string unique to this run. If a previous run uses this
% string and remains in the directory tree, then this script will exit on
% error. The existence of a log file is used to determine this.
%
% An optional use of the script is to supply it with a net that was previously
% trained using this exact script, along with the number of epochs that would
% like to be trained. This is useful for when the training has stopped because
% the maximum number of epochs has been reached, but the net isn't fully
% trained, or when the training has been stopped by outside sources and the
% training should continue. A new validation set is created, and the
% discontinuity in validation sets state may cause early stopping too soon.
%
% see mainExpDir/EXPLOG-???.txt for experimental details relating to this
% file.
%
% Directory tree looks like:
%
% mainExpID
%   + expID_1
%   + expID_2
%     |     + expIDstring1
%     |     + expIDstring2
%   + scripts
%   + EXPLOG
%
% This version is appropriate for use with simulation data generated after
% 11/29/00.
%
```

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

175
% Experiment ID

mainExpID = 'MIR_NAST';
expID = { '009mn', 'fixSat', '7deg', 'GoodRelH' };
%add more subIDs, expID{i+1} ...

expIDstring = makeExpIDstring(expID);

% *** PURPOSE ***************************************************************
% SATIGR values where relH>1 were set to 1 and then the TBs were regenerated.
% This will hopefully lead to a consistant data set, and hopefully will fix
% the problem where retrievals are poor in the upper atmosphere.
% Since the repair of the dataset didn’t help retrievals in the upper
% atmosphere, but it did in the lower atmosphere, and since training only over
% the valid region produced better results before, this experiment trains only
% over the valid region and with the modified SATIGR.
% Experiment specific parameters.

% net topology
firstHiddenLayer = 30;
secondHiddenLayer = 15;
transferFunctions = {'tansig', 'tansig', 'purelin_FakeDeriv'};

% surface effects.
userdata.surface = 'RANDEMISS'; % LAND, WATER, MIXTURE, RANDOM, RANDEMISS,
%ROUGH_OCEAN, DECORR_LAND;
%userdata.mixture = 0.5; % Uncomment if 'MIXTURE' is used.
%userdata.waterRefMean = makeMIR_NAST_Ocean(0);
userdata.randomTsurfFlag = 1; % version of makeExpDataSets determines Ts model.
userdata.TsurfVar = 3; % deg K (actually, stddev even though labeled variance)
userdata.maxTsurfCor = 0.5;  \% required for makeExpDataSets 7 or greater

\%instruments
\% 1.3 factor is for conservitiveness; T/3 in MIR is for signal processing gain.
userdata.instNoise(1:8) = [0.1922; 0.2436; 0.2066; 0.2679; 0.3002; 0.3814; 0.6080; 0.8930] * 1.3;
userdata.instNoise(9:14) = [0.13; 0.13; 0.39; 0.39; 0.31; 0.32] * 1.3 ./ 3;
userdata.instNoise(15:22) = [0.1879; 0.1274; 0.1084; 0.1474; 0.1248; 0.1528; 0.1748; 0.2231] * 1.3;
\%userdata.instNoise(1:22) = 0.3;
userdata.dsb_flag = 0;  \%use only for sim data before Nov26 2000

\%training variations
userdata.emissTruthFlag = 0;  \%flag to add rand emiss to truth.
userdata.changeInstNoiseForValidation = 0;  \%don't change inst noise for validation set.
userdata.raobMatrixName = 'satigr_cor';
userdata.trainSubsetSize = 0;  \%set to zero to disable.
userdata.updateTrainData = 1;
net.userdata.lr_mult = 1;  \% multiply lr every update of Training Data.

\%TRUTH processing
userdata.relHumProcessing = 'NOTHING';  \%GAUSS, CLIP, REMOVE, NOTHING
  \%userdata.relHumGAUSS_STD = 0.2;  \%needed for GAUSS option.
  \%userdata.relHumREMOVE_THRESH = 1.5;  \%needed for REMOVE option.
  \%userdata.relHumREMOVE_THEN = 'CLIP';  \%GAUSS, CLIP, NOTHIGN after remove,then?

\%subspace of simulation data
\%scan angles
userdata.scanAngles = [7.2 28.8 43.2 50.4];  \% a list of angles averaged over in degrees.
userdata.angleIndices = 1;  \% the indices of the angles averaged over.
userdata.removeNAST118chan9 = 1;
userdata.raobFilter = -1;  \%filters coeff & relH before any relH filtering.
  \%set to -1 to disable
  \%SATIGR only: 9557:11317
userdata.relHFilter = 27:40; %(1 - highest atmosphere, 40 - surface)

%atmosphere levels
userdata.coeffLevels = 1; %[1 2] to use 2 both levels

%functions called
userdata.funct.makeTB = 'makeExpDataSets7'; %4 or later required.
userdata.funct.smoothOcean = 'simulate_smoothOcean4';
userdata.funct.water = userdata.funct.smoothOcean;
userdata.funct.land = 'simulate_land4'; %adds emissivity to truth: ONLY!
userdata.funct.decorrLand = 'simulate_decorrLand4';
userdata.funct.mixture = 'simulate_mixture4';
userdata.funct.randomMixture = 'simulate_random_surface4';
userdata.funct.randomEmiss = 'simulate_randomEmiss6'; %if works, then sim data TU and TD

ver = version;
if(ver(1) == '5')
    warning('Using traingdx_jbh16 for Matlab 5');
    userdata.funct.trainFunct = 'traingdx_jbh16';
elseif(ver(1) == '6')
    userdata.funct.trainFunct = 'traingdx_jbh17_v6';
end

% Set up Directories and Dependancies.

% *** Directory Tree Structure:
mainExpDir = sprintf('/usr/users/jkixonia/exps/%s',mainExpID);
expDir = sprintf('%s/%s',mainExpDir,expID{1});
scriptDir = sprintf('%s/scripts',mainExpDir);
resultDir = sprintf('%s/%s',expDir,expIDstring);
resultFile = sprintf('%s/%s-%s.mat',resultDir,expIDstring,thisRunID)
logFile = sprintf('%s/%s-%s.log',resultDir,expIDstring,thisRunID)
simDirROOT = '/usr/users/jkixonia/exps/simulations';

% *** Create subdirs if they do not exist:
if(~exist(expDir,'file'))
    mkdir(mainExpDir,expID{1});
end
if(~exist(resultDir,'file'))
    mkdir(expDir, expIDstring);
end

% *** Dependancies:
sourceDirs{1} = '/usr/users/jkixonia/matlab/nnets';
sourceDirs{2} = '/usr/users/jkixonia/matlab/nnets/training';
sourceDirs{3} = '/usr/users/jkixonia/matlab/general';
sourceDirs{4} = '/usr/users/jkixonia/matlab/radiotransfer';
sourceDirs{5} = '/usr/users/jkixonia/matlab/radiotransfer/Jays';
sourceDirs{6} = scriptDir;
addpath(sourceDirs{:});

% *** Data Files needed in the environment:
dataFiles{1} = sprintf('%s/04-25-01/04_25_01_coeff_multAng.mat',simDirROOT); %coeff data
IsSimDataBeforeNov26_2000 = 0;
dataFiles{2} = sprintf('%s/04-25-01/corMat.mat',simDirROOT);
dataFiles{3} = '/usr/users/jkixonia/exps/SATIGR_CORRECTED.mat';
%dataFiles{4} = '/usr/users/jkixonia/exps/MIRNAST/data/FREQS.mat';
%dataFiles{5} = sprintf('%s/08-18-00/results/MIR_NAST_tb6',simDirROOT); %coeff data
%dataFiles{6} = sprintf('%s/08-18-00/SIM-WORKSPACE',simDirROOT);
%dataFiles{} = 'Initial NET';

%add more like: dataFiles{i+1} = 'datafile'
for i = 1:length(dataFiles)
    load(dataFiles{i});
end

% Prepare for Neural Network Training.

% *** Format data and obtain initial data sets.
userdata.saveStateFile = resultFile;
%userdata.inst_desc = inst_desc; %from dataFile
%userdata.covMat = covMat;
userdata.corMat = corMat; %from dataFile

if(userdata.raobFilter == -1)
    userdata.raobFilter = 1:size(coeff.MIR.E,2);
end
if(userdata.relHFilter == -1)
    userdata.relHFilter = 27:40;
end

if(userdata.removeNAST118chan9)
    coeff.NAST118.E = coeff.NAST118.E(1:8,:,:,:);
    coeff.NAST118.TU = coeff.NAST118.TU(1:8,:,:,:);
    coeff.NAST118.TD = coeff.NAST118.TD(1:8,:,:,:);
    coeff.NAST118.Inst.freqWeights = coeff.NAST118.Inst.freqWeights(1:8,:);
end

% *** format coefficients for dynamic TB rendering using given angles.
if(userdata.dsb_flag)
    numDSB = size(coeff{1}{1},1); userdata.DSB = 1: numDSB;
    numSSB = size(coeff{1}{2},1); userdata.SSB = numDSB + 1: numDSB + numSSB;
end
if(IsSimDataBeforeNov26_2000)
    userdata.coeff = makeCoeff(coeff, userdata); clear coeff;
    error('Must update makeCoeff for userdata.raobFilter\n');
else
    userdata.coeff = makeCoeff2(coeff, userdata); clear coeff;
    userdata.dsb_flag = 0; % make sure this feature is not on.
end
userdata.Tsurf = eval(sprintf(' squeeze (%s (2,1,:))',userdata.raobMatrixName));
userdata.Tsurf = userdata.Tsurf(userdata.raobFilter);

% *** process relH
relH = relH(:, userdata.raobFilter);
t = 1:length(userdata.Tsurf);

if(strcmp(userdata.relHumProcessing, 'CLIP'))
    relH(relH > 1) = 1;
elseif(strcmp(userdata.relHumProcessing, 'NOTHING'))
    ;
elseif(strcmp(userdata.relHumProcessing, 'GAUSS'))
    relH(relH > 1) = random('norm', 1, userdata.relHumGAUSS_STD, length(relH(relH > 1)), 1);
elseif(strcmp(userdata.relHumProcessing, 'REMOVE'))
    % remove raobs that have a value greater than THRESH in levels 21:40.
    [i,j] = find(relH(21:40,:)>userdata.relHumREMOVE_.THRESH);
    relH = relH(:, ~ismember(1:size(relH,2), unique(j)));
    userdata.coeff.E = userdata.coeff.E(:, ~ismember(1:size(relH,2), unique(j)));
    userdata.coeff.TU = userdata.coeff.TU(:, ~ismember(1:size(relH,2), unique(j)));
    userdata.coeff.TD = userdata.coeff.TD(:, ~ismember(1:size(relH,2), unique(j)));
    userdata.Tsurf = userdata.Tsurf(~ismember(1:size(relH,2), unique(j)));

t = 1:length(userdata.Tsurf);
if(strcmp(userdata.relHumREMOVE_THEN,'CLIP'))
    relH(relH>1)=1;
elseif(strcmp(userdata.relHumREMOVE_THEN,'GAUSS'))
    relH(relH>1) = random('norm',1,userdata.relHumGAUSS_STD,length(relH(relH>1)),1);
elseif(strcmp(userdata.relHumREMOVE_THEN,'NOTHING'))
    ;
else
    error('Invalid relH processing.'),
end
else
    error('Invalid relH processing.'),
end
userdata.relH = relH;
clear Ts date latitude longitude satigr relH;

userdata.trainIndices = t(ismember(mod(t,16),1:15));
userdata.validIndices = t(ismember(mod(t,16),0));
clear t;

% *** make data sets, add noise, random angle, normalized, dsb averaged.
if(exist('net_'))
    userdata = net_.userdata;
    userdata.saveStateFile = resultFile; %update in case thisRunID changes.
    UsingOldNet = 1;
else
    UsingOldNet = 0;
end

[userdata, trainDataSet, validDataSet] = feval(userdata.funct.makeTB,userdata, 0, UsingOldNet);

% *** create network and initialize.
range = [min(trainDataSet.P'), max(trainDataSet.P')];

\[ t = \sqrt{\text{var}(\text{trainDataSet.P'})}; \]

range = \([-2t', 2t']\);

layerDef = [firstHiddenLayer, secondHiddenLayer, size(trainDataSet.T, 1)];

net = newff(range, layerDef, transferFunctions, userdata.funct.trainFunct);

net = initnet(net);

**% *** Use pre-existing variables if supplied as inputs.**

if(exist('net_'))
    net = net_;
    fprintf('Using previously trained nnet...
');
end;

if(exist('epochs_')) net.trainParam.epochs = epochs_; end;

**% *** Save variables into net.userdata structure for training.**

userdata.expID = expID;

net.userdata = userdata; %userdata was loaded from net_, may have changed.

**% *** train neural net:**

more off; flops(0);

diary(logFile);

fprintf('Begin experiment %s, %s
', expIDstring, thisRunID);

**% *** Display to log file Experiment setup:**

userdata

**% *** Display to log file Functions Called:**

userdata.funct

[net, tr] = train(net, trainDataSet.P, trainDataSet.T, [], [], validDataSet, validDataSet);

**% *** Error Analysis ***
trainDataSet = errorAnalysis(trainDataSet, net);
validDataSet = errorAnalysis(validDataSet, net);

% *** Save Experiment ***
trainDataSet.expID = expID;
validDataSet.expID = expID;
tr.expID = expID;
save(resultFile,'net','tr','trainDataSet','validDataSet');

% *** exit gracefully ***
fprintf('FLOPPAGE: %d
',flops);
fprintf('Experiment %s Completed!!

',expIDstring);
diary off;
quit;

%*************************************************************************
% helper functions

function [net] = initnet(net)
    net.trainParam.epochs = 100000;
    net.trainParam.show = 100;
    net.trainParam.lr = 0.10;
    net.trainParam.lr_inc = 1.10;
    net.trainParam.lr_dec = 0.7;
    net.trainParam.goal = 0.001;
    net.trainParam.max_fail = 20000;
    net.trainParam.mc = 0.9; %momentum
    net.trainParam.max_perf_inc = 1.20;
    return;

function expIDstring = makeExpIDstring(expID)
expIDstring = expID{1};
for i=1:length(expID)
    if(i==1)
        expIDstring = expID{1};
    else
        expIDstring = sprintf('%s-%s',expIDstring,expID{i});
    end
end
return;

function Coeff = makeCoeff(coeff, userdata)
    %for i=1:length(coeff)
    %    Coeff.TD(userdata.DSB,i,:) = coeff{i}{1}(:,userdata.angleIndices,1);
    %    Coeff.TU(userdata.DSB,i,:) = coeff{i}{1}(:,userdata.angleIndices,2);
    %    Coeff.E(userdata.DSB,i,:) = coeff{i}{1}(:,userdata.angleIndices,3);
    %    Coeff.TD(userdata.SSB,i,:) = coeff{i}{2}(:,userdata.angleIndices,1);
    %    Coeff.TU(userdata.SSB,i,:) = coeff{i}{2}(:,userdata.angleIndices,2);
    %    Coeff.E(userdata.SSB,i,:) = coeff{i}{2}(:,userdata.angleIndices,3);
    %end
    %size(Coeff.[TD|TU|E]) = [chan x raob x angle]
    %return;

function Coeff = makeCoeff2(coeff, userdata)
    firstIndex = 1;
    for i=1:length(coeff.names)
        Inst = eval(['coeff.' coeff.names{i}]);
        lastIndex = firstIndex - 1 + size(Inst.E, 1);
        R = userdata.raobFilter;

        Coeff.E(firstIndex:lastIndex,:,:,:) = Inst.E(:,R,userdata.angleIndices,userdata.coeffLevels);
        Coeff.TD(firstIndex:lastIndex,:,:,:) = Inst.TD(:,R,userdata.angleIndices,userdata.coeffLevels);
Coeff.TU(firstIndex:lastIndex,:,:,:)= Inst.TU(:,R,userdata.angleIndices,userdata.coeffLevels);

Coeff.Freq(firstIndex:lastIndex) = Inst.Inst.freqWeights * Inst.Inst.freqs';

firstIndex = lastIndex + 1;

end

%size(Coeff.TD|TU|E)) = [chan x raob x angle x outputLevel]

return;
function expXXX(thisRunID, net, epochs)

% single argument is string unique to this run. If a previous run uses this
% string and remains in the directory tree, then this script will exit on
% error. The existence of a log file is used to determine this.
%
% An optional use of the script is to supply it with a net that was previously
% trained using this exact script, along with the number of epochs that would
% like to be trained. This is useful for when the training has stopped because
% the maximum number of epochs has been reached, but the net isn't fully
% trained, or when the training has been stopped by outside sources and the
% training should continue. A new validation set is created, and the
% discontinuity in validation sets state may cause early stopping too soon.
%
% see mainExpDir/EXPLOG-???.txt for experimental details relating to this
% file.
%
% Directory tree looks like:
%
% mainExpID
%   + expID_1
%   + expID_2
%     |   + expIDstring1
%     |   + expIDstring2
%   + scripts
%   + EXPLOG
%
% This version is appropriate for use with simulation data generated after
% 11/29/00.
%******************************************************************************
%
% Experiment ID
%
mainExpID = 'NAST-I';
explD = { '001ni', '100pca','1.25noise' };  
%add more subIDs, expID{i+1} ...

explIDstring = makeExpIDstring(expID);

% *** PURPOSE  ****************************************
% The purpose of this script is to determine if I was overkill on the bias
% noise, and how much to reduce it.
%*****************************************
% Experiment specific parameters.

%net topology
firstHiddenLayer = 30;
secondHiddenLayer = 15;
transferFunctions = {'tansig','tansig','purelin_FakeDeriv'};

%training variations:
%userdata.RADinstNoiseStd = see below
userdata.TBrandomBiasStd = 1.25; %degrees Kelvin. indep per channel.
userdata.pca.keep = 100;
userdata.updateTrainData = 10; %critical parameter!

%truth processing
userdata.relHumProcessing = 'NOTHING'; %GAUSS, CLIP, REMOVE, NOTHING
%userdata.relHumGAUSS_STD = 0.2; %needed for GAUSS option.
%userdata.relHumREMOVE_THRESH = 1.5; %needed for REMOVE option.
%userdata.relHumREMOVE_THEN = 'CLIP'; %GAUSS, CLIP, NOTHING after remove,then
userdata.relHFilter = 27:40;

%files and related data:
userdata.file.emiss = '/tmp/jkixonia/emiss.tmp';
userdata.file.tsurf = '/tmp/jkixonia/tsurf.tmp';
userdata.file.tbFile{1} = '/tmp/jkixonia/satigr66-TB.b1';
userdata.file.tbFile{2} = '/tmp/jkixonia/satigr66-TB.b2';
userdata.file.tbFile{3} = '/tmp/jkixonia/satigr66-TB.b3';

userdata.file.coeffFile{1} = '/usr/users/jkixonia/exps2/RAOBS/NASTI/satigr66.b1';
userdata.file.coeffFile{2} = '/usr/users/jkixonia/exps2/RAOBS/NASTI/satigr66.b2';
userdata.file.coeffFile{3} = '/usr/users/jkixonia/exps2/RAOBS/NASTI/satigr66.b3';
userdata.numChannels{1} = 2718; %15 micron %43 weighting functs missing
userdata.numChannels{2} = 2946; %7 micron %64 weighting functs missing
userdata.numChannels{3} = 2968; %4 micron %all weighting functs present.
userdata.file.freqFile{1} = '/usr/sounder2/NAST/NAST-I_simulations/fast_model/little_end'
userdata.file.freqFile{2} = '/usr/sounder2/NAST/NAST-I_simulations/fast_model/little_end'
userdata.file.freqFile{3} = '/usr/sounder2/NAST/NAST-I_simulations/fast_model/little_end'
userdata.file.AIRSprofiles = '/usr/users/jkixonia/exps2/RAOBS/profiles_AIRS.mat';
userdata.file.chanIndices = '/usr/users/jkixonia/exps2/RAOBS/NASTI/3000channelIndices_20C

%functions called
userdata.funct.makeTB = 'makeNASTIDataSets';
userdata.funct.coeff2bt = '/usr/users/jkixonia/src/NASTI_coeff2bt/coeff2bt';

ver = version;
if(ver(1) == '5')
    warning('Using older version for traingdx... Double Check!!!');
    userdata.funct.trainFunct = 'traingdx_jbh16';
elseif(ver(1) == '6')
    userdata.funct.trainFunct = 'traingdx_jbh17_v6';
end

%**********************************************************************
% Set up Directories and Dependancies.

189
% *** Directory Tree Structure:

mainExpDir = sprintf('/usr/users/jkixonia/exps2/%s', mainExpID);
expDir = sprintf('%s/%s', mainExpDir, expID{1});
scriptDir = sprintf('%s/scripts', mainExpDir);
resultDir = sprintf('%s/%s', expDir, expIDstring);
resultFile = sprintf('%s/%s-%s.mat', resultDir, expIDstring, thisRunID);
logFile = sprintf('%s/%s-%s.log', resultDir, expIDstring, thisRunID);

% *** Create subdirs if they do not exist:

if(~exist(expDir, 'file'))
    mkdir(mainExpDir, expID{1});
end
if(~exist(resultDir, 'file'))
    mkdir(expDir, expIDstring);
end
if(~exist('/tmp/jkixonia', 'file'))
    mkdir('/tmp', 'jkixonia');
end
fprintf('Temp files writing to /tmp/jkixonia: DO NOT RUN 2 EXPS on this machine!!
');

% *** Dependancies:

sourceDirs{1} = '/usr/users/jkixonia/matlab/nnets';
sourceDirs{2} = '/usr/users/jkixonia/matlab/nnets/training';
sourceDirs{3} = '/usr/users/jkixonia/matlab/general';
sourceDirs{4} = '/usr/users/jkixonia/matlab/radiotransfer';
sourceDirs{5} = '/usr/users/jkixonia/matlab/radiotransfer/Jays';
sourceDirs{6} = scriptDir;
addpath(sourceDirs{:});

% *** Data Files needed in the environment:
dataFiles{1} = '/usr/users/jkixonia/exps2/RAOBS/profiles_SATIGR.mat';
dataFiles{2} = '/usr/users/jkixonia/exps/SATIGR_READ.mat';
dataFiles{3} = '/usr/users/jkixonia/exps2/NAST-I/data/FLIGHT-RADIANCE-NOISE.mat';
dataFiles{4} = '/usr/users/jkixonia/exps2/NAST-I/data/IR_EMISS_COV.mat';

%add more like: dataFiles{i+1} = 'datafile'
for i = 1:length(dataFiles)
    load(dataFiles{i});
end

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

% Prepare for Neural Network Training.

% *** Format data and obtain initial data sets.
userdata.saveStateFile = resultFile;

if(userdata.relHFilter == -1)
    userdata.relHFilter = 21:40;
end

% data from dataFiles{}:
userdata.Tsurf = profiles.Tsurf; clear profiles;
userdata.RADinstNoiseStd = RADinstNoiseStd; clear RADinstNoiseStd;
userdata.emissModel.covMat = EMISS_COV; clear EMISS_COV;
userdata.emissModel.mean = EMISS_MEAN; clear EMISS_MEAN;

% *** process relH

t = 1:length(userdata.Tsurf);

if(strcmp(userdata.relHumProcessing, 'CLIP'))
    relH(relH>1)=1;
elseif(strcmp(userdata.relHumProcessing, 'NOTHING'))

191
elseif(strcmp(userdata.relHumProcessing,'GAUSS'))
relH(relH>1) = random('norm',1,userdata.relHumGAUSS_STD,length(relH(relH>1)),1);
elseif(strcmp(userdata.relHumProcessing,'REMOVE'))
error('This feature must be updated for NAST-I');

% remove raobs that have a value greater than THRESH in levels 21:40.
% [i,j] = find(relH(21:40,:)>userdata.relHumREMOVE_THRESH);
% relH = relH(:,~ismember(1:size(relH,2),unique(j)));
% userdata.coeff.E = userdata.coeff.E(:,~ismember(1:size(relH,2),unique(j)));
% userdata.coeff.TU = userdata.coeff.TU(:,~ismember(1:size(relH,2),unique(j)));
% userdata.coeff.TD = userdata.coeff.TD(:,~ismember(1:size(relH,2),unique(j)));
% userdata.Tsurf = userdata.Tsurf(~ismember(1:size(relH,2),unique(j)));
% t = 1:length(userdata.Tsurf);
% if(strcmp(userdata.relHumREMOVE_THEN,'CLIP'))
% relH(relH>1)=1;
% elseif(strcmp(userdata.relHumREMOVE_THEN,'GAUSS'))
% relH(relH>1) = random('norm',1,userdata.relHumGAUSS_STD,length(relH(relH>1)),1);
% elseif(strcmp(userdata.relHumREMOVE_THEN,'NOTHING'))
% end
% else
% error('Invalid relH processing.');
% end
else
error('Invalid relH processing.');
end
userdata.relH = relH;

userdata.trainIndices = t(ismember(mod(t,16),1:15));
userdata.validIndices = t(ismember(mod(t,16),0));
clear t;

% *** make data sets, add noise, random angle, normalized, dsb averaged.
if(exist('net_'))
    userdata = net_.userdata;
    userdata.saveStateFile = resultFile; %update in case thisRunID changes.
    UsingOldNet = 1;
else
    UsingOldNet = 0;
end

[userdata, trainDataSet, validDataSet] = feval(userdata.funct.makeTB,userdata, 0, UsingOldNet);

if(~exist('net_'))
    % *** create network and initialize.
    %range = [min(trainDataSet.P'),max(trainDataSet.P')];
    t = sqrt(var(trainDataSet.P'));
    range = [-2*t', 2*t'];
    layerDef = [firstHiddenLayer, secondHiddenLayer,size(trainDataSet.T,1)];
    net = newff(range,layerDef,transferFunctions,userdata.funct.trainFunct);
    net = initnet(net);
else
    % *** Use pre-existing variables if supplied as inputs.
    net = net_; 
    fprintf('Using previously trained nnet...
');
end;

if(exist('epochs_')) net.trainParam.epochs = epochs_; end;

% *** Save variables into net.userdata structure for training.
userdata.expID = expID;
net.userdata = userdata; %userdata was loaded from net_, may have changed.

% *** train neural net:
more off; %flops(0);
diary(logFile);
fprintf('Begin experiment %s, %s\n\n', expIDstring, thisRunID);

% *** Display to log file Experiment setup:
userdata

% *** Display to log file Functions Called:
userdata.funct

[net,tr] = train(net,trainDataSet.P,trainDataSet.T,[],[],validDataSet,validDataSet);

% *** Error Analysis ***
trainDataSet = errorAnalysis(trainDataSet, net);
validDataSet = errorAnalysis(validDataSet, net);

% *** Save Experiment ***
trainDataSet.expID = expID;
validDataSet.expID = expID;
tr.expID = expID;
save(resultFile,'net','tr','trainDataSet','validDataSet');

% *** exit gracefully ***
%fprintf('FLOPPAGE: %d\n',flops);
fprintf('Experiment %s Completed!!\n\n',expIDstring);
diary off;
quit;

%**************************************************************************
% helper functions

function [net] = initnet(net)
    net.trainParam.epochs = 100000;
net.trainParam.show = 10;
net.trainParam.lr = 0.10;
net.trainParam.lr_inc = 1.10;
net.trainParam.lr_dec = 0.7;
net.trainParam.goal = 0.001;
net.trainParam.max_fail = 20000;
net.trainParam.mc = 0.9;  \%momentum

return;

function explDstring = makeExpIDstring(expID)
  explIDstring = expID{1};
  for i=1:length(expID)
    if(i==1)
      explIDstring = expID{1};
    else
      explIDstring = sprintf('%s-%s',explIDstring,expID{i});
    end
  end
  return;

%function Coeff = makeCoeff(coeff, userdata)
%for i=1:length(coeff)
%  Coeff.TD(userdata.DSB,i,:) = coeff{i}{1}(:,userdata.angleIndices,1);
%  Coeff.TU(userdata.DSB,i,:) = coeff{i}{1}(:,userdata.angleIndices,2);
%  Coeff.E(userdata.DSB,i,:) = coeff{i}{1}(:,userdata.angleIndices,3);
%  Coeff.TD(userdata.SSB,i,:) = coeff{i}{2}(:,userdata.angleIndices,1);
%  Coeff.TU(userdata.SSB,i,:) = coeff{i}{2}(:,userdata.angleIndices,2);
%  Coeff.E(userdata.SSB,i,:) = coeff{i}{2}(:,userdata.angleIndices,3);
%end
%size(Coeff.[TD|TU|E]) = [chan x raob x angle]
%return;
function Coeff = makeCoeff2(coeff, userdata)

firstIndex = 1;

for i=1:length(coeff.names)
    Inst = eval(['coeff.', coeff.names{i}]);

    lastIndex = firstIndex - 1 + size(Inst.E, 1);

    R = userdata.raobFilter;

    Coeff.E(firstIndex:lastIndex,:,:,:) = Inst.E(:,R,userdata.angleIndices,userdata.coeffLevels);
    Coeff.TD(firstIndex:lastIndex,:,:,:) = Inst.TD(:,R,userdata.angleIndices,userdata.coeffLevels);
    Coeff.TU(firstIndex:lastIndex,:,:,:) = Inst.TU(:,R,userdata.angleIndices,userdata.coeffLevels);
    Coeff.Freq(firstIndex:lastIndex) = Inst.Inst.freqWeights * Inst.Inst.freqs';

    firstIndex = lastIndex + 1;
end

%size(Coeff.[TD|TU|E]) = [chan x raob x angle x outputLevel]

return;
A.3.2 Modified MATLAB Training Code with Momentum and Back-propagation

function [net,tr,Ac,E1] = traingdx(net,Pd,Tl,Ai,Q,TS,VV,TV)

%TRAINGDX Gradient descent w/momentum & adaptive lr backpropagation.
%
% JBH: Modified for use with MIR_NAST scripts.
% eg. "jkizonia/exps/MIR_NAST/scripts/exp005_tbArray_A5_7degree.m
% this version runs on matlab 6.
%
% Syntax
%
% [net,tr,Ac,E1] = traingdx(net,Pd,Tl,Ai,Q,TS,VV,TV)
% info = traingdx(code)
%
% Description
%
% TRAINGDX is a network training function that updates weight and
% bias values according to gradient descent momentum and an
% adaptive learning rate.
%
% TRAINGDX(NET,Pd,Tl,Ai,Q,TS,VV,TV) takes these inputs,
% NET - Neural network.
% Pd - Delayed input vectors.
% Tl - Layer target vectors.
% Ai - Initial input delay conditions.
% Q - Batch size.
% TS - Time steps.
% VV - Empty matrix [] or structure of validation vectors.
% TV - Empty matrix [] or structure of test vectors.
% and returns,
% NET - Trained network.
% TR - Training record of various values over each epoch:
%  TR.epoch - Epoch number.
%  TR.perf - Training performance.
%  TR.vperf - Validation performance.
%  TR.tperf - Test performance.
%  TR.lr - Adaptive learning rate.
% Ac - Collective layer outputs for last epoch.
% El - Layer errors for last epoch.

Training occurs according to the TRAINGDX's training parameters shown here with their default values:

% net.trainParam.epochs 10 Maximum number of epochs to train
% net.trainParam.goal 0 Performance goal
% net.trainParam.lr 0.01 Learning rate
% net.trainParam.lr_inc 1.05 Ratio to increase learning rate
% net.trainParam.lr_dec 0.7 Ratio to decrease learning rate
% net.trainParam.max_fail 5 Maximum validation failures
% net.trainParam.max_perf_inc 1.04 Maximum performance increase
% net.trainParam.mc 0.9 Momentum constant.
% net.trainParam.min_grad 1e-10 Minimum performance gradient
% net.trainParam.show 25 Epochs between displays (NaN for no displays)
% net.trainParam.time inf Maximum time to train in seconds

Dimensions for these variables are:
% Pd - NoxNixTS cell array, each element P{i,j,ts} is a DijxQ matrix.
% Tl - NlxTS cell array, each element P{i,ts} is an VixQ matrix.
% Ai - NlxLD cell array, each element Ai{i,k} is an SixQ matrix.

Where
% Ni = net.numInputs
% Nl = net.numLayers
% LD = net.numLayerDelays
% Ri = net.inputs{i}.size
\%  \quad Si = net.layers{i}.size
\%  \quad Vi = net.targets{i}.size
\%  \quad Dij = Ri * length(net.inputWeights{i,j}.delays)
\%  
\%  If VV or TV is not [], it must be a structure of vectors:
\%  \quad VV.PD, TV.PD - Validation/test delayed inputs.
\%  \quad VV.Tl, TV.Tl - Validation/test layer targets.
\%  \quad VV.Ai, TV.Ai - Validation/test initial input conditions.
\%  \quad VV.Q, TV.Q - Validation/test batch size.
\%  \quad VV.TS, TV.TS - Validation/test time steps.
\%  Validation vectors are used to stop training early if the network
\%  performance on the validation vectors fails to improve or remains
\%  the same for MAX_FAIL epochs in a row. Test vectors are used as
\%  a further check that the network is generalizing well, but do not
\%  have any effect on training.
\%  
\%  TRAINGDX(CODE) return useful information for each CODE string:
\%  \quad \text{'pnames'} - Names of training parameters.
\%  \quad \text{'pdefaults'} - Default training parameters.
\%  
\%  Network Use
\%
\%  You can create a standard network that uses TRAINGDX with
\%  NEWFF, NEWCF, or NEWELM.
\%
\%  To prepare a custom network to be trained with TRAINGDX:
\%  1) Set NET.trainFcn to 'traingdx'.
\%     This will set NET.trainParam to TRAINGDX's default parameters.
\%  2) Set NET.trainParam properties to desired values.
\%  In either case, calling TRAIN with the resulting network will
\%  train the network with TRAINGDX.
See NEWFF, NEWCF, and NEWELM for examples.

Algorithm

TRAINGDX can train any network as long as its weight, net input, and transfer functions have derivative functions.

Backpropagation is used to calculate derivatives of performance PERF with respect to the weight and bias variables X. Each variable is adjusted according to the gradient descent with momentum.

\[ dX = mc^*dX_{prev} + lr^*mc^*dperf/dX \]

where \( dX_{prev} \) is the previous change to the weight or bias.

For each epoch, if performance decreases toward the goal, then the learning rate is increased by the factor \( lr_{inc} \). If performance increases by more than the factor \( max\_perf\_inc \), the learning rate is adjusted by the factor \( lr\_dec \) and the change, which increased the performance, is not made.

Training stops when any of these conditions occur:

1) The maximum number of EPOCHS (repetitions) is reached.
2) The maximum amount of TIME has been exceeded.
3) Performance has been minimized to the GOAL.
4) The performance gradient falls below MINGRAD.
5) Validation performance has increase more than MAX_FAIL times since the last time it decreased (when using validation).

See also NEWFF, NEWCF, TRAINGD, TRAINGDM, TRAINGDA, TRAINLM.
% Mark Beale, 11-31-97
% ODJ, 11/20/98, added support for user stopping.
% $Revision: 1.7 $ $Date: 2000/06/15 04:20:55$

% FUNCTION INFO
% ==================

if isstr(net)
    switch (net)
        case 'pnames',
            net = fieldnames(traingdx('pdefaults'));
        case 'pdefaults',
            trainParam.epochs = 100;
            trainParam.goal = 0;
            trainParam.lr = 0.01;
            trainParam.lr_dec = 0.7;
            trainParam.lr_inc = 1.05;
            trainParam.max_fail = 5;
            trainParam.max_perf_inc = 1.04;
            trainParam.mc = 0.9;
            trainParam.min_grad = 1.0e-6;
            trainParam.show = 25;
            trainParam.time = inf;
            net = trainParam;
        otherwise
            error('Unrecognized code.')
    end
end

return
end

% CALCULATION
% == = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

% Parameters
epochs = net.trainParam.epochs;
goal = net.trainParam.goal;
lr = net.trainParam.lr;
lr_inc = net.trainParam.lr_inc;
lr_dec = net.trainParam.lr_dec;
max_fail = net.trainParam.max_fail;
max_perf_inc = net.trainParam.max_perf_inc;
mc = net.trainParam.mc;
min_grad = net.trainParam.min_grad;
show = net.trainParam.show;
time = net.trainParam.time;

% Parameter Checking
if (~isa(epochs,'double')) | (~isreal(epochs)) | (any(size(epochs)) ~= 1) | ...
   (epochs < 1) | (round(epochs) ~= epochs)
   error('Epochs is not a positive integer.')
end
if (~isa(goal,'double')) | (~isreal(goal)) | (any(size(goal)) ~= 1) | ...
   (goal < 0)
   error('Goal is not zero or a positive real value.')
end
if (~isa(lr,'double')) | (~isreal(lr)) | (any(size(lr)) ~= 1) | ...
   (lr < 0)
   error('Learning rate is not zero or a positive real value.')
end
if (~isa(lr_inc,'double')) | (~isreal(lr_inc)) | (any(size(lr_inc)) ~= 1) | ...
   (lr_inc < 1)
   error('LR-inc is not a positive real value greater or equal to 1.0.')
end
if (~isa(lr_dec,'double')) | (~isreal(lr_dec)) | (any(size(lr_dec)) ~ 1) | ... (lr_dec < 0) | (lr_dec > 1)
    error('LR_dec is not a positive real value greater or between 0.0 and 1.0.')
end

if (~isa(max_fail,'double')) | (~isreal(max_fail)) | (any(size(max_fail)) ~ 1) | ... (max_fail < 1) | (round(max_fail) ~= max_fail)
    error('Max_fail is not a positive integer.')
end

if (~isa(max_perf_inc,'double')) | (~isreal(max_perf_inc)) | (any(size(max_perf_inc)) ~ 1) | ... (max_perf_inc < 1)
    error('Max_perf_inc is not a positive real value greater or equal to 1.0.')
end

if (~isa(mc,'double')) | (~isreal(mc)) | (any(size(mc)) ~ 1) | ...
    (mc < 0) | (mc > 1)
    error('MC is not real value between 0.0 and 1.0.')
end

if (~isa(min_grad,'double')) | (~isreal(min_grad)) | (any(size(min_grad)) ~ 1) | ... (min_grad < 0)
    error('Min_grad is not zero or a positive real value.')
end

if (~isa(show,'double')) | (~isreal(show)) | (any(size(show)) ~ 1) | ...
    (isfinite(show) & ((show < 1) | (round(show) ~= show)))
    error('Show is not ''NaN'' or a positive integer.')
end

if (~isa(time,'double')) | (~isreal(time)) | (any(size(time)) ~ 1) | ...
    (time < 0)
    error('Time is not zero or a positive real value.')
end

% Constants
this = 'TRANGDX_JBH17_v6';
doValidation = isempty(VV);
doTest = isempty(TV);

% Initialize
flag_stop=0;
stop = '\';
startTime = clock;
X = getx(net);
[perf,El,Ac,N,Zb,Zi,Zl] = calcperf(net,X,Pd,T1,Ai,Q,TS);
[gX,normgX] = calcgx(net,X,Pd,Zb,Zi,Zl,N,Ac,El,perf,Q,TS);
dX = lr*gX;
if (doValidation)
    VV.net = net;
vperf = calcperf(net,X,VV.Pd,VV.T1,VV.Ai,VV.Q,VV.TS);
    VV.perf = vperf;
    VV.numFail = 0;
end
tr = newtr(epochs,'perf','vperf','tperf','lr');

% Train
for epoch=0:epochs

    %save state often and print time.
    if(mod(epoch,net.trainParam.show*10) == 0)
        net.saveState = net;
        save(net.userdata.saveStateFile,'net_saveState','tr');
        s = fix(clock);
        fprintf('*** Time is: %d/%d/%d %d:%d:%d\n',s(2),s(3),s(1),s(4),s(5),s(6))
    end

    %add noise every epoch and perform processing. Just change training data.
    if(mod(epoch,net.userdata.updateTrainData) == (net.userdata.updateTrainData-1))
        [net.userdata, trainTB, validTB] = feval(net.userdata.funct.makeTB,net.userdata, 1);
Pd{1} = trainTB.P;
Tl{3} = trainTB.T;
VV.Pd{1} = validTB.P;
VV.Tl{3} = validTB.T;
end

% *** Change made to gradient calculation!

if(epoch==50)
    warning('Epoch is 50');
end

% Training Record
epochPlus1 = epoch+1;
tr.perf(epochPlus1) = perf;
tr.lr(epochPlus1) = lr;
if (doValidation)
    tr.vperf(epochPlus1) = vperf;
end
if (doTest)
    tr.tperf(epochPlus1) = calcperf(net,X,TV.Pd,TV.T1,TV.Ai,TV.Q,TV.TS);
end

% Stopping Criteria
currentTime = etime(clock,startTime);
if (perf <= goal)
    stop = 'Performance goal met.';
elseif (epoch == epochs)
    stop = 'Maximum epoch reached, performance goal was not met.';
elseif (currentTime > time)
    stop = 'Maximum time elapsed, performance goal was not met.';
elseif (normgX < min_grad)
    stop = 'Minimum gradient reached, performance goal was not met.';
elseif (doValidation) & (VV.numFail > max_fail)
  stop = 'Validation stop.';
elseif flag_stop
  stop = 'User stop.';
end

% Progress
if isfinite(show) & (~rem(epoch,show) | length(stop))
  fprintf(this);
  if isfinite(epochs) fprintf(', Epoch %g/%g',epoch, epochs); end
  if isfinite(time) fprintf(', Time %g/%g',currentTime/time/100); end
  if isfinite(goal) fprintf(', %s %g/%g',upper(net.performFen),perf,goal); end
  if isfinite(mingrad) fprintf(', Gradient %g/%g',normgX,mingrad); end
  fprintf('
')
  flag_stop=plotperf(tr,goal,this,epoch);
  if length(stop) fprintf('%s, %s

',this,stop); end
end

% Stop when criteria indicate its time
if length(stop)
  if (doValidation)
    net = VV.net;
  end
  break
end

% Gradient Descent with Momentum and Adaptive Learning Rate
dX = mc*dX + (1-mc)*lr*gX;
X2 = X + dX;
net2 = setx(net,X2);
[perf2,El,Ac,N,Zb,Zi,ZI] = calcperf(net2,X2,Pd,Tl,Ai,Q,TS);
if (perf2/perf) > max_perf_inc
lr = lr*lr_dec;
    dX = lr*gX;
else
    if (perf2 < perf)
        lr = lr*lr_inc;
        end
    X = X2;
    net = net2;
    perf = perf2;
    [gX,normgX] = calcgx(net,X,Pd,Zb,Zi,Zl,N,Ac,El,perf,Q,TS);
    norm_g = sqrt(sum(sum(gX.^2)));
end

% *******************************************************
if(mod(epoch,net.userdata.updateTrainData) == (net.userdata.updateTrainData-1))
    lr = min(lr*10,net.trainParam.lr);
end
% *******************************************************

% Validation
if (doValidation)
    vperf = calcperf(net,X,VV.Pd,VV.T1,VV.Ai,VV.Q,VV.TS);
    if (vperf < VV.perf)
        VV.perf = vperf; VV.net = net; VV.numFail = 0;
    elseif (vperf > VV.perf)
        VV.numFail = VV.numFail + 1;
    end
end
end
end

% Finish
tr = cliptr(tr,epoch);
A.3.3 Noise-Averaging Routines

Microwave

function [userdata, trainDataSet, validDataSet] = makeExpDataSets(userdata, FlagInsideTraining, FlagInside Training = 1 if being called from the nnet training function, 0 if being called from outside the program.

% ver 3: predicts emiss for land as a test. Vince says it speeds up training.
% validation set, emiss of truth is changed, but input TBs are not.
%
% ver 4: uses new simulation format, post-Nov 26, 2000. Supports several
% atmospheric levels for airplane simulation.
%
% ver 5: option to choose random subset of raobs from entire training set each
% call. Good for larger training sets. set userdata.trainSubsetSize.
%
% ver 6: Tsurf is now decorrelated across frequencies. Tsurf is now a vector
% and the appropriate simulate_surface function must be called.
%
% ver 7: 1)userdata.relHFilter defines the subset of relH levels to train on:
% For the typical 21:40, set to -1.
% 2)userdata.maxTsurfCor defines the decorrelation function used
% on Tsurf across 130GHz. Exponential decorrelation is assumed.
%
% additional option: FlagOldNetUserdata, used when the userdata input comes
% from a previously trained net. Specifically, normalization factors are not
% recomputed, so this flag overrides the FlagInsideTraining=0 setting. Errors
% will occur if the input userdata is not from a net that was previously being
% trained.

% size(userdata.coeff) = [chan x raob x scanAngle x outputNumber]
if(exist('FlagOldNetUserdata') & FlagOldNetUserdata)
    FlagInsideTraining = 1;
end

if(~FlagInsideTraining)
    userdata.freqs = userdata.coeff.Freq; %getFreqs(userdata.inst_desc);
end

if(userdata.trainSubsetSize ~= 0)
    userdata.trainSubsetIndices = pickRandomIndex(length(userdata.trainIndices),userdata.trainIndices);
else
    userdata.trainSubsetIndices = 1:length(userdata.trainIndices);
end

dataIndices = [userdata.validIndices, userdata.trainIndices(userdata.trainSubsetIndices)];

% *** Choose random scan angle and randomly interpolated output point for each epoch.
if(length(userdata.angleIndices) > 1 | length(userdata.coeffLevels) > 1)
    %randomCoeff2 not debugged!!!
    [userdata.index, userdata.TD, userdata.TU, userdata.E] = randomCoeff2(userdata.coeff.TD(1:length(userdata.angleIndices),userdata.coeffLevels(1:length(userdata.coeffLevels))));
elseif(FlagInsideTraining == 0)
    %userdata.index = userdata.angleIndices;
    userdata.TD = userdata.coeff.TD(:,dataIndices);
    userdata.TU = userdata.coeff.TU(:,dataIndices);
    userdata.E = userdata.coeff.E(:,dataIndices);
elseif( FlagInsideTraining & (userdata.trainSubsetSize ~= 0) )
    userdata.TD = userdata.coeff.TD(:,dataIndices);
    userdata.TU = userdata.coeff.TU(:,dataIndices);
    userdata.E = userdata.coeff.E(:,dataIndices);
end

% *** Decorrelating the surface temperature from the lowest level of raob.
% correlation: \( \exp(-\ln(10/8)/130.*f) \) derived by assuming 0.8 correlation bet. 54 & 183 and exp

```matlab
if(userdata.randomTsurfFlag == 1)
    persistent L V;
    if(isempty(L) | isempty(V))
        K = \exp(-\log(userdata.maxTsurfCor)/130*abs(ones(22,1)*userdata.freqs - ...
            userdata.freqs'*ones(1,22)))*userdata.TsurfVar;
        [V,D] = eig(K);
        L = eig(K);
        l = find(L>0.001);
        L = L(l);
        V = V(:,l);
    end
    Tsurf = V*random('norm',0,L*ones(1,length(dataIndices)))+...
        + ones(size(V,1),1)*userdata.Tsurf(dataIndices);
    else
        Tsurf = userdata.Tsurf(dataIndices);
    end

% *** implement surface
if(strcmp(userdata.surface, 'WATER'))
    TB = feval(userdata.funct.smoothOcean,userdata.TD, userdata.TU, userdata.E, Tsurf, use:
elseif(strcmp(userdata.surface, 'LAND'))
    [TB,emiss] = feval(userdata.funct.land,userdata.TD, userdata.TU, userdata.E, Tsurf);
elseif(strcmp(userdata.surface, 'MIXTURE'))
    error('check water ref mean')
    TB = feval(userdata.funct.mixture, userdata.TD, userdata.TU, userdata.E, Tsurf, userdat
elseif(strcmp(userdata.surface, 'RANDOM'))
    error('check water ref mean')
    TB = feval(userdata.funct.randomMixture, userdata.TD, userdata.TU, userdata.E, Tsurf,
elseif(strcmp(userdata.surface, 'ROUGHOCEAN'))
    error('check water ref mean')
    TB = feval(userdata.funct.roughOcean, userdata.TD, userdata.TU, userdata.E, Tsurf, user

elseif(strcmp(userdata.surface, 'DECORRLAND'))
    TB = feval(userdata.funct.decorrLand, userdata.TD, userdata.TU, userdata.E, Tsurf, user

elseif(strcmp(userdata.surface, 'RANDEMISS'))
    TB = feval(userdata.funct.randomEmiss, userdata.TD, userdata.TU, userdata.E, Tsurf, us
else
    error('Surface type not supported in experiment description file.');
end

% *** average double sidebands:
if(userdata.dsb-flag ==1)
    userdata.TB = [ ( TB(userdata.DSB(mod(userdata.DSB,2)==1),:) + TB(userdata.DSB(mo
else
    userdata.TB = TB;
end

% *** add instrument noise:
userdata.TB = addInstrumentNoise(userdata.TB, userdata.instNoise);

% *** add secant if training several angles.
if(length(userdata.angleIndices) > 1)
    userdata.TB = [ userdata.TB; sec(userdata.index * userdata.scanAngles(userdata.angleIndi
end

if(userdata.emissTruthFlag == 1 & strcmp(userdata.surface, 'LAND'))
    userdata.truth = [userdata.relH(userdata.relHFilter,dataIndices); emiss];
elseif(FlagInsideTraining == 0)
userdata.truth = userdata.relH(userdata.relHFilter, dataIndices);

elseif (FlagInsideTraining & (userdata.trainSubsetSize ~= 0))
    userdata.truth = userdata.relH(userdata.relHFilter, dataIndices);
end

% *** divide up data sets and normalize
TBvalidIndices = 1:length(userdata.validIndices);
TBtrainIndices = (length(userdata.validIndices) + 1):size(userdata.TB, 2);

if (FlagInsideTraining == 0)
    trainDataSet.P = userdata.TB(:, TBtrainIndices);
    validDataSet.P = userdata.TB(:, TBvalidIndices);
    trainDataSet.T = userdata.truth(:, TBtrainIndices);
    validDataSet.T = userdata.truth(:, TBvalidIndices);

    [trainDataSet.P, userdata.meanp, userdata.stdp] = prestd(trainDataSet.P);
    validDataSet.P = trastd(validDataSet.P, userdata.meanp, userdata.stdp);

    userdata.validDataSet = validDataSet.P;
else

    trainDataSet.P = trastd(userdata.TB(:, TBtrainIndices), userdata.meanp, userdata.stdp);
    if (userdata.changeInstNoiseForValidation == 0)
        validDataSet.P = userdata.validDataSet; % one instance of inst noise.
    else
        validDataSet.P = trastd(userdata.TB(:, TBvalidIndices), userdata.meanp, userdata.stdp);
    end

    trainDataSet.T = userdata.truth(:, TBtrainIndices);
    validDataSet.T = userdata.truth(:, TBvalidIndices); % only changes w/emiss.
end
function f = getFreqs(inst_desc)

for i = 1:length(inst_desc{1}{1})

    f(i) = inst_desc{1}{1}{i}{2}(1);

end

if(length(inst_desc)>1)

    for j = 1:length(inst_desc{2}{1})

        f(i+j) = inst_desc{2}{1}{j}{2}(1);

    end

end

return;
function TBnoiseless = simulate_smoothOcean4(TD, TU, E, Tsurf, corMat)
% Usage: TBnoiseless = simulate_smoothOcean4(TD, TU, E, Tsurf, ref_mean);
% Acoeff > TD, E, and TU, where each data is (Chan x Raob) in dimension.
% Tsurf is a vector of size (lxRaob).
% Only 1 spot considered per raob. if more than one spot is needed, then replace
% all dimensions Raob by (Raob*numSpots).
% uses simulation data as of 11/29/00
% Change in definition: TD is now the old TD * E.

numChan = size(TD,1);
numRaob = size(TD,2);

persistent q;
%similarity transform using sqrt of cor mat.
if(isempty(q))
    q = real(sqrtm(corMat));
end

%check corMat:

%if(find(imag(q)) ~= 0)
%    warning('corMat has negative eigenvalues');
%end
persistent corMatOKflag;
if(isempty(corMatOKflag))
    if(sum(sum(imag(sqrtm(corMat)))) > 1e-6 )
        error('corMat imaginary parts greater than 1e-6');
    end
    corMatOKflag = 1;
\begin{verbatim}
end

a = unifrnd(0,0.65,numChan,numRaob);

y = q*(a-0.325) + 0.325;

y(y<0) = 0;

y(y>1) = 1;

if(size(Tsurf,1) == 1)

    TBnoiseless = TU + y.*TD + E.*(1-y).*(ones(numChan,1)*Tsurf);

elseif(size(Tsurf) == size(E))

    TBnoiseless = TU + y.*TD + E.*(1-y).*Tsurf;

else

    error('Tsurf and E do not agree in function simulate_randomEmiss');

end
\end{verbatim}
function TB = addInstrumentNoise(TBnoiseless, instNoise)

% Usage: TB = addInstrumentNoise(TBnoiseless, instNoise)
%
% Takes TBnoiseless and adds noise in 1 of two ways:
% - If instNoise is a scalar, then a gaussian rv with instNoise stddev
% is added to all data points.
% - If instNoise is a vector of length size(TBnoiseless,1) (the number of
% channels), then to each channel is added a gaussian rv vector with
% stddev for the ith channel given by the ith element in instNoise.

numChan = size(TBnoiseless,1);
numRaob = size(TBnoiseless,2);

%generate instrument noise

if(0)

%obsolete code

if(size(instNoise,1) == 1 & size(instNoise,2) == 1)
    noise = random('norm',0,instNoise,numChan, numRaob);
else
    for i=1:length(instNoise)
        noise(i,:) = random('norm',0,instNoise(i),1,numRaob);
    end
end

TB = TBnoiseless + noise;
else

TB = random('norm',TBnoiseless,instNoise' * ones(1,size(TBnoiseless,2)));
end
Infrared

function [userdata, trainDataSet, validDataSet] = makeExpDataSets(userdata, FlagInsideTraining

% FlagInsideTraining = 1 if being called from the nnet training function,
% 0 if being called from outside the program.
%
% additional option: FlagOldNetUserdata, used when the userdata input comes
% from a previously trained net. Specifically, normalization factors are not
% recomputed, so this flag overrides the FlagInsideTraining=0 setting. Errors
% will occur if the input userdata is not from a net that was previously being
% trained.

if(exist('FlagOldNetUserdata') & FlagOldNetUserdata)
    FlaglnsideTraining = 1;
end

% *** implement surface
fid = fopen(userdata.file.tsurf,'w');
fwrite(fid, userdata.Tsurf, 'real*4');
fclose(fid);

% *** choose random coeffs.
if(~FlagInsideTraining)
    load(userdata.file.AIRSprofiles);
    userdata.emissModel.IRemiswn = profiles.IRemiswn(:,1)*ones(1,length(userdata.Tsurf));
    clear profiles
end

% *** emiss model, gaussian, clipped at 1,
% correlated according to AIRS dataset model.
userdata.emissModel.IRemis = simulate_randomEmissGauss(userdata.emissModel.covMat,userdata.
% *** write to temp file.

fid = fopen(userdata.file.emiss, 'w');
fwrite(fid, [userdata.emissModel.IRemiswn;userdata.emissModel.IRemis;\1 -userdata.emissModel.I{1}];
fclose(fid);

% *** generate TBs.

for i=1:3
runstring = sprintf('!s %s %s %s %s', userdata.funct.coeff2bt, ...
    userdata.file.coeffFile{i}, userdata.file.emiss, ...
    userdata.file.tsurf, userdata.file.freqFile{i}, ...;
    userdata.file.tbFile{i});
    eval(runstring);
end

if(~FlagInsideTraining)
    load(userdata.file.chanIndices);
    userdata.finalI = finalI;
    for i=1:3
        fid = fopen(userdata.file.freqFile{i}, 'rb');
        userdata.wavenumber{i} = fread(fid, userdata.numChannels{i}, 'real*4');
        fclose(fid);
    end
end

for i=1:3
    fid = fopen(userdata.file.tbFile{i}, 'r');
    Temp = fread(fid, [userdata.numChannels{i}, length(userdata.Tsurf)], 'real*4');
    indicesChan = ((i-1)*1000+1):(i*1000);
    indicesRaob = 1:length(userdata.Tsurf);
    userdata.TrainData(indicesChan,indicesRaob) = Temp(userdata.final{i},:);
end
% add random bias:
if(userdata.TBrandomBiasStd > 0 )
    userdata.TrainData = userdata.TrainData + userdata.TBrandomBiasStd*randn(size(userdata.T)
end

% convert to radiances:
if(size(userdata.TrainData,1)~=3000)
    error(sprintf('Expected 3000 channels, got %d',size(userdata.TrainData,1)));
end

for i=1:3000
    if(i<=1000)
        j=1;
    elseif(i<=2000)
        j=2;
    else
        j=3;
    end
    userdata.TrainData(i,:) = brit2rad(userdata.wavenumber{j})(userdata.finalI{j}(mod(i-1,100
end

% *** add instrument noise:
userdata.TrainData = userdata.TrainData + randn(size(userdata.TrainData)).*(userdata.RADinst

if(FlagInsideTraining == 0)
    trainDataSet.T = userdata.relH(userdata.relHFilter,userdata.trainIndices);
    validDataSet.T = userdata.relH(userdata.relHFilter,userdata.validIndices);
    [trainDataSet.P, userdata.meanp, userdata.stdp] = prestd(userdata.TrainData(:,userdata.trainI
    validDataSet.P = trastd(userdata.TrainData(:,userdata.validIndices), userdata.meanp, userdata

fprintf('Computing PCA (may take a while)...');
%%*** implement efficient PCA:
%cov matrix:
Ldd = trainDataSet.P*trainDataSet.P'./size(trainDataSet.P,2);

%eigenvalues/vectors:
[V,D] = eig(Ldd);

%sort largest to smallest:
E = diag(D);
[E,mEig] = sort(E);
E = E(length(E):-1:1);
mEig = mEig(length(E):-1:1);
V = V(:,mEig);

fprintf('Finished PCA.
');

userdata.pca.eig = E;
userdata.pca.Q = V;

trainDataSet.P = V(:,1:userdata.pca.keep)'*trainDataSet.P;
validDataSet.P = V(:,1:userdata.pca.keep)'*validDataSet.P;

userdata.validDataSet = validDataSet.P;
else

trainDataSet.P = userdata.pca.Q(:,1:userdata.pca.keep)'*trastd(userdata.TrainData(:,us
validDataSet.P = userdata.validDataSet; % one instance of inst noise.

trainDataSet.T = userdata.relH(userdata.relHFilter,userdata.trainIndices);
validDataSet.T = userdata.relH(userdata.relHFilter,userdata.validIndices);
end
Notes on Noise:

Statistics of rms over each band for flight data in radiance domain.

<table>
<thead>
<tr>
<th>Band</th>
<th>Min</th>
<th>Mean</th>
<th>Std</th>
<th>Max</th>
<th>Range of Data:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.67</td>
<td>4.17</td>
<td>1.64</td>
<td>5.96</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.21</td>
<td>0.83</td>
<td>0.55</td>
<td>2.23</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.033</td>
<td>0.175</td>
<td>0.116</td>
<td>0.445</td>
<td></td>
</tr>
</tbody>
</table>

Statistics of rms over each band for train data in radiance domain.

<table>
<thead>
<tr>
<th>Band</th>
<th>Min</th>
<th>Mean</th>
<th>Std</th>
<th>Max</th>
<th>Range of Data:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.58</td>
<td>15.31</td>
<td>6.22</td>
<td>21.45</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.23</td>
<td>0.97</td>
<td>0.66</td>
<td>5.34</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.016</td>
<td>0.47</td>
<td>0.37</td>
<td>1.32</td>
<td></td>
</tr>
</tbody>
</table>

R = 2001:3000;

\[\min(\text{std}(\text{TrainData}(R,:'))),\text{mean}(\text{std}(\text{TrainData}(R,:'))),\text{std}(\text{std}(\text{TrainData}(R,:'))),\max(\text{std}(\text{TrainData}(R,:')))\]
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

int interp(float *x1, float *y1, float *x2, float *y2, int n2);
int brit2rad(float *vn, float *Trad, int n, float Tbrit);
int rad2brit(float *vn, float *Trad, int n, float *Tbrit);
FILE* fileOpen(const char *filename, const char *mode);
int checkFile(const char *filename);
int fileExists(const char *filename);
int checkLength(const char *filename, int compare, int size);
int fileLength(const char *filename, int size);

#define nIR 7
#define SIZE 4

// Usage: coeff2bt coeffFile EnRFile tsurfFile wnFile outFile

void main(int argc, char *argv[])
{

    // declarations:
    char coeffFile[256], wnFile[256], outFile[256], EnRFile[256], tsurfFile[256]; // inputs

    int numRaobs, numChans; // determined from input files
    float *SurfReflect, // [1 x numChans] x SIZE bytes | surface reflectivity vector for one itera
    *SurfEmiss, // [1 x numChans] x SIZE bytes | surface emissivity vector for one itera
    *Tsurf, // [1 x numChans] x SIZE bytes | surface temperature
    *TupLooking, // [1 x numChans] x SIZE bytes | radiance of an upward-looking instru
    *TdownLooking, // [1 x numChans] x SIZE bytes | radiance of a downward-looking instru
    *oneWayTrans, // [1 x numChans] x SIZE bytes | transmittance from the ground to the
    *waveNum, // [1 x numChans] x SIZE bytes | wave number used in plank calculation
```
*T,                 // [1 x numChans] x SIZE bytes | final calculated BT
*memory;           // one memory mass for the above variables: [8 x numChans] x SIZE byt

float *buffer;     // [3 x numChans]

float Ts,           // single Ts used to calculate Tsurf vector.
TempBrightness;    // output scalar

struct surfaceModel {
    float IRemiswn[nIR], IRemis[nIR], IRrefl[nIR];
} SurfModel;

// parse input string and check for existence of files:
if(argc == 6) {
    strcpy(coeffFile,argv[1]);
    strcpy(EnRFile,argv[2]);
    strcpy(tsurfFile,argv[3]);
    strcpy(wnFile,argv[4]);
    strcpy(outFile,argv[5]);
} else {
    printf("Usage: coeff2bt coeffFile EnRFile tsurfFile wnFile outFile\n");
    exit(0);
}

// check for file existence
checkFile(coeffFile);
checkFile(EnRFile);
cHECKFILE(tsurfFile);
cHECKFILE(wnFile);

// open and read tsurfFile and wnFile to obtain numRaobs and numChans:
numChans = fileLength(wnFile,SIZE);
numRaobs = fileLength(tsurfFile,SIZE);

// check lengths of files for consistancy:
checkLength(coeffFile, 3*numChans*numRaobs, SIZE);
checkLength(EnRFile, nIR*3*numRaobs, SIZE);

// allocate memory:
memory = (float*)malloc(11*numChans*SIZE);
SurfReflect = memory;
SurfEmiss = memory + numChans;
Tsurf = memory + numChans*2;
TupLooking = memory + numChans*3;
TdownLooking = memory + numChans*4;
oneWayTrans = memory + numChans*5;
waveNum = memory + numChans*6;
T = memory + numChans*7;
buffer = memory + numChans*8; //8,9,10 next is 11!

// read in wave numbers once:
wnF = fileOpen(wnFile,"rb");
if(fread(waveNum, SIZE, numChans, wnF) != numChans) {
    printf("Error in reading file %s\n",wnFile);
    exit(1);
}
fclose(wnF);

// open output file:
outF = fileOpen(outFile,"wb");

// open input files:
cF = fileOpen(coeffFile,"rb");
erF = fileOpen(EnRFile,"rb");
tsF = fileOpen(tsurfFile,"rb");

\textbf{int} i, j;
\textbf{for}(i=0; i<numRaobs; i++)
{
    //read in SurfModel, TupLooking, TdownLooking, oneWayTrans.
    fread(SurfModel.IRemiswn, SIZE, nIR, erF);
fread(SurfModel.IRemis, SIZE, nIR, erF);
    fread(SurfModel.IRrefl, SIZE, nIR, erF);
    fread(buffer, SIZE, 3*numChans, cF);
    fread(&Ts, SIZE, 1, tsF);

    \textbf{for}(j=0; j<numChans; j++)
    {
        TdownLooking[j] = buffer[3*j];
        TupLooking[j] = buffer[3*j+1];
        oneWayTrans[j] = buffer[3*j+2];
    }

    //calculate SurfReflect, SurfEmiss, Tsurf.
    interp(SurfModel.IRemiswn, SurfModel.IRemis, waveNum, SurfEmiss, numChans);
    interp(SurfModel.IRemiswn, SurfModel.IRrefl, waveNum, SurfReflect, numChans);
    brit2rad(waveNum, Tsurf, numChans, Ts);

    //calculate T
    \textbf{for}(j=0; j<numChans; j++)
    {
    }
    rad2brit(waveNum, T, numChans, T);

    //write to output.
    if(fwrite(T,SIZE,numChans,outF) != numChans) 

printf("Error writing output to file for output number %d!!\n", i);

}

}

// clean up
fclose(cF);
fclose(erF);
fclose(tsF);
fclose(outF);
}

/******************************
Functions
*******/

// extrapolates outside, assumes x1 is sorted, assumes nIR is small
int interp(float *x1, float *y1, float *x2, float *y2, int n2) {
    int i, j;
    float m[nIR-1], b[nIR-1];

    // calculate slope and offset for each segment.
    for(i=0; i<(nIR-1); i++) {
        m[i] = (y1[i+1]-y1[i])/(x1[i+1]-x1[i]);
        b[i] = y1[i] - m[i]*x1[i];
    }

    for(i=0; i<n2; i++) {
        if(x2[i] < x1[0]) {
            y2[i] = m[0]*x2[i] + b[0]; // extrapolate
        } else if(x2[i] >= x1[0] && x2[i] < x1[nIR-1]) {
            for(j=0; j<(nIR-1); j++) {
               // code...
            }
        }
    }
}

226
if (x2[i] >= x1[j] && x2[i] < x1[j+1]) {
    y2[i] = m[j] * x2[i] + b[j]; //interpolate
    break;
}
}
else if (x2[i] >= x1[nIR-1]) {
    y2[i] = m[nIR-1] * x2[i] + b[nIR-1]; //extrapolate
}
else {
    printf("Error in interpolation routine\n");
    exit(1);
}
return 0;

int brit2rad(float *vn, float *Trad, int n, float Tbrit) {
    /* rad = wnbrit(vn,Tb)
       as written in wnbrit.f */
    static float h = 6.626176e-27;
    static float c = 2.997925e+10;
    static float b = 1.380662e-16;
    static float c1 = 2*h*c*c;
    static float c2 = h*c/b;

    float f1, f2;
    int i;

    for(i=0; i<n; i++) {

/ * \( tnb(x,y,z) = \frac{y}{d \log(x/z + 1.0)} \) *

\[
f_1 = c_1 \cdot \text{pow}(v_n[i], 3);
\]

\[
f_2 = c_2 \cdot v_n[i];
\]

\[
\text{Trad}[i] = \frac{f_1}{\exp(f_2/T_{brit}) - 1};
\]

\[
\text{return 0;}
\]

```c
int rad2brit(float *vn, float *Trad, int n, float *Tbrit)
{
    /* Tb = \text{wnbrit}(vn, rad) 
    as written in \text{wnbrit.f} */

    static float h = 6.626176e-27;
    static float c = 2.997925e+10;
    static float b = 1.380662e-16;
    static float c1 = 2*h*c*c;
    static float c2 = h*c/b;

    /* \( tnb(x,y,z) = \frac{y}{d \log(x/z + 1.0)} \) */
    float f1, f2;
    int i;

    for(i=0; i<n; i++) {
        f1 = c1 * \text{pow}(v_n[i], 3);
        f2 = c2 * v_n[i];

        /* tbb = tnb(f1, f2, r) */
        Tbrit[i] = f2 / \log(f1/Trad[i] + 1);
    }
}
```

FILE* fileOpen(const char *filename, const char *mode) {

FILE *f;
if((f = fopen(filename,mode)) == NULL) {
    printf("File %s cannot be opened\n", filename);
    exit(1);
}
if(ferror(f)) {
    printf("File %s encountered error when opening\n",filename);
    exit(1);
}
return f;

int checkFile(const char *filename)
{
    if(fileExists(filename)) {
        return 0;
    }
    else {
        printf("File %s does not exist!\n",filename);
        exit(1);
    }
    return 1;
}

int fileExists(const char *filename)
{
    fclose(fileOpen(filename,"r"));
    return 1;
}

int checkLength(const char *filename, int compare, int size)
{

int n = fileLength(filename, size);
if(n != compare) {
    printf("Error: File %s is not consistent with numRaobs and numChans determined from ;
    exit(1);
}
return 0;

/* returns length of file assuming "size" byte units. */
int fileLength(const char *filename, int size)
{
    FILE *f = fileOpen(filename,"rb");
    void *t = malloc(1000*size);
    int n = 0;

    while(!feof(f))
    {
        n += fread(t,size,1000,f);
        if(ferror(f))
        {
            printf("Error in file read %s\n",filename);
            exit(1);
        }
    }
    fclose(f);
    free(t);
    return n;
}
Bibliography


231


