Computational Model for Atmospheric Propagation Losses in the Radiofrequency Through Millimeter Wave Range

JUDITH V. HUTSON

Formely, Advanced Concepts Group
Radar Division

CHRISTOPHER T. RODENBECK

Advanced Concepts Group
Radar Division

February 5, 2021

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.
# Computational Model for Atmospheric Propagation Losses in the Radiofrequency Through Millimeter Wave Range

Judith V. Hutson* and Christopher T. Rodenbeck

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

**Abstract**

This report describes a MATLAB program for calculating atmospheric propagation losses at radiofrequency (RF) and millimeter-wave (MMW) frequencies. The program computes calculates the path integral of the individual atmospheric loss mechanisms along a line of sight from an elevated platform to a point on the earth’s surface. The integral is performed numerically by discretizing the atmospheric column into a finite number of layers. Earth curvature is neglected.

**Subject Terms**

Radar    ISR    Propagation modeling
This page intentionally left blank.
CONTENTS

INTRODUCTION ................................................................................................................................. 1

INTERFACE ........................................................................................................................................... 1
  1.1 Graphical interface ......................................................................................................................... 1
  1.2 Data files ......................................................................................................................................... 5

CALCULATIONS ............................................................................................................................... 5
  1.3 Ground conditions .......................................................................................................................... 5
  1.4 Extrapolated atmospheric gas conditions ..................................................................................... 6
    1.4.1 Temperature profile ..................................................................................................................... 6
    1.4.2 Pressure profile ........................................................................................................................... 7
    1.4.3 Water vapor density profile ......................................................................................................... 8
  1.5 Specific attenuation due to atmospheric conditions ........................................................................ 9
    1.5.1 Loss from gas .............................................................................................................................. 9
    1.5.2 Loss from clouds/fog .................................................................................................................... 15
    1.5.3 Loss from rain ............................................................................................................................. 16
  1.6 Path length ...................................................................................................................................... 20
  1.7 Dispersion ...................................................................................................................................... 22

CONCLUSION ..................................................................................................................................... 25

APPENDIX 1 ......................................................................................................................................... 26

APPENDIX 2 ......................................................................................................................................... 56
This page intentionally left blank.
COMPUTATIONAL MODEL FOR ATMOSPHERIC PROPAGATION LOSSES AT RADIOFREQUENCY AND MILLIMETER WAVE FREQUENCIES

INTRODUCTION

This report describes a MATLAB program for calculating atmospheric propagation losses at radiofrequency (RF) and millimeter-wave (MMW) frequencies. The program computes calculates the path integral of the individual atmospheric loss mechanisms along a line of sight from an elevated platform to a point on the earth’s surface. The integral is performed numerically by discretizing the atmospheric column into a finite number of layers. Earth curvature is neglected.

The atmospheric model is based on the most recent International Telecommunications Union (ITU) recommendations [1]-[2], [4], [6], [7], and the numeric approach is based on an earlier MathCad program provided to Naval Research Laboratory (NRL) by H. Bruce Wallace at the Defense Advanced Research Projects Agency (DARPA) in 2015.

The source code is documented in Appendix 1, with an alternate version that bypasses the user interface described in Appendix 2.

INTERFACE

1.1 Graphical interface

Figure 1 shows the script’s user interface, with a diagram of a few of the inputs shown in Figure 2. Given the input information regarding atmospheric conditions and the transmitted electromagnetic wave, the program calculates attenuation and phase dispersion to the surface and the angle at which the beam will strike the target. If the ‘Save File’ box is checked, it will also prompt the user for a file name and location in which to record attenuation and dispersion data. This will be saved in an Excel file. An important note is the fact that the dispersion and path bending calculations take into account only atmospheric gases. Any effect due to condensed water, such as clouds or rain, will not appear in the result, and a note stating this will appear if relevant. Additionally, at high altitudes, the ionosphere will reflect and attenuate radio frequencies. As this effect is dependent upon space weather, rather than surface weather, it is not calculated. This will be noted by the program as well.
Fig. 1 – User interface with example settings

Table 1 - GUI variable description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground Temperature</td>
<td>Temperature at surface</td>
<td>K</td>
</tr>
<tr>
<td>Ground Pressure</td>
<td>Pressure at surface</td>
<td>mb</td>
</tr>
<tr>
<td>Relative Humidity</td>
<td>Relative humidity at surface</td>
<td>%</td>
</tr>
<tr>
<td>Clouds: Altitude</td>
<td>Altitude of bottom of cloud</td>
<td>m</td>
</tr>
<tr>
<td>Clouds: Height</td>
<td>Extension of cloud above bottom altitude</td>
<td>m</td>
</tr>
<tr>
<td>Clouds: Water Concentration</td>
<td>Liquid water density in cloud</td>
<td>g/m³</td>
</tr>
<tr>
<td>Fog: Height</td>
<td>Extension of fog above surface</td>
<td>m</td>
</tr>
<tr>
<td>Fog: Water Concentration</td>
<td>Liquid water density in fog</td>
<td>g/m³</td>
</tr>
<tr>
<td>Rain: Rate</td>
<td>Rain rate</td>
<td>mm/hr</td>
</tr>
<tr>
<td>Frequency</td>
<td>Transmitted frequency</td>
<td>GHz</td>
</tr>
<tr>
<td>Polarization</td>
<td>Transmitted polarization; degrees above horizontal</td>
<td>°</td>
</tr>
<tr>
<td>Frequency Sweep: Start</td>
<td>Lowest frequency in sweep</td>
<td>GHz</td>
</tr>
<tr>
<td>Frequency Sweep: Step</td>
<td>Frequency sweep step value</td>
<td>GHz</td>
</tr>
<tr>
<td>Frequency Sweep: Stop</td>
<td>Highest frequency in sweep</td>
<td>GHz</td>
</tr>
<tr>
<td>Altitude</td>
<td>Altitude of transmitter</td>
<td>km</td>
</tr>
<tr>
<td>Grazing Angle</td>
<td>Angle of transmission; degrees below horizontal (negative)</td>
<td>°</td>
</tr>
<tr>
<td>Target Altitude</td>
<td>Position of the target above surface</td>
<td>km</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------</td>
<td>----</td>
</tr>
<tr>
<td>Interval</td>
<td>Resolution of calculation</td>
<td>m</td>
</tr>
<tr>
<td>Results: Attenuation</td>
<td>Attenuation of transmission to surface</td>
<td>dB</td>
</tr>
<tr>
<td>Results: Dispersion</td>
<td>Phase dispersion of transmission to surface</td>
<td>°</td>
</tr>
<tr>
<td>Results: Incidence Angle</td>
<td>Incidence angle of transmission upon surface; degrees above horizontal</td>
<td>°</td>
</tr>
<tr>
<td>Notes</td>
<td>Program text output</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2 – Clarification on grazing and incidence angles, as well as the difference between cloud altitude and cloud height

The application also generates two plots. The first, demonstrated in Figure 3, shows specific attenuation and dispersion, the loss and phase shift produced at each point. The other shows total attenuation varying with altitude, and is shown in Figure 4. This plot also includes a visualization of weather; a gray represents clouds, blue signifies rain, and a reddish hue indicates fog.
Fig. 3 – Program output figure showing specific attenuation and dispersion as functions of altitude.

Both the interface outputs and the generated plots represent the wave entered into the upper ‘Frequency’ field. The data generated as part of a frequency sweep will be saved only to the file. Within the output file, data is written into four sheets. Sheet 1 displays specific attenuation, Sheet 2 total...
attenuation along the path of the beam, Sheet 3 shows specific dispersion, and Sheet 4 total dispersion. Each row represents a frequency, and each column an altitude.

1.2 Data files

If ‘Save File’ is checked, the user will be prompted to select a file location, and an Excel file will be created to store the results for later retrieval and comparison. Data is stored in four worksheets, each with one table: ‘Specific Attenuation’, ‘Total Attenuation’, ‘Specific Dispersion’, and ‘Total Dispersion’. Figure 5 shows an example of a ‘Specific Attenuation’ sheet. Each table is organized by altitude in meters on the vertical axis, and frequency in gigahertz along the horizontal axis, with data units specified in the sheet label in field A1.

![Figure 5 – Example of saved data file for one frequency at 20 GHz](image)

CALCULATIONS

1.3 Ground conditions

The amount of water vapor in the air is input as relative humidity, because this quantity is generally more easily obtained from weather data. However, water vapor density, expressed as grams per cubic meter, is the more relevant quantity in these calculations. Relative humidity must therefore be converted. This can be done by using temperature and pressure to find the maximum possible water vapor partial pressure, which is the vapor pressure if relative humidity were 100% [1]. This can be used as a reference to find actual vapor pressure, which along with temperature can be used to determine water vapor density [1].

The atmospheric water vapor density $\rho$ is given by

$$\rho = \frac{216.7 \cdot P_{wv}}{T}$$  \hspace{1cm} (1)

$\rho = $ water vapor density (g/m$^3$)  
$T =$ temperature (K)  
$P_{wv} =$ vapor pressure (mb)

$$P_{wv} = \frac{RH}{100} \cdot P_{wvs}$$  \hspace{1cm} (2)

$RH =$ relative humidity (%)  
$P_{wvs} =$ saturation vapor pressure (mb)
\[ P_{wvs} = EF \cdot a \cdot e^{\left(\frac{b \cdot t}{c + t}\right)} \]  

\( t = \) temperature (°C)  

\[ EF = \begin{cases} 
1 + 10^{-4} \cdot [7.2 + P \cdot (0.0320 + 5.9 \cdot 10^{-6} \cdot t^2)] & t > 0 \\
1 + 10^{-4} \cdot [2.2 + P \cdot (0.0383 + 6.4 \cdot 10^{-6} \cdot t^2)] & t < 0 
\end{cases} \]

\( P = \) pressure (mb)

The coefficients used in this calculation depend upon the state of condensed water, and therefore upon the ambient temperature as it relates to 0°C [1].

\[ a = \begin{cases} 
6.1121 & t > 0 \\
6.1115 & t < 0 
\end{cases} \]

\[ b = \begin{cases} 
18.678 & t > 0 \\
23.036 & t < 0 
\end{cases} \]

\[ c = \begin{cases} 
257.14 & t > 0 \\
279.82 & t < 0 
\end{cases} \]

\[ d = \begin{cases} 
234.5 & t > 0 \\
333.7 & t < 0 
\end{cases} \]

### 1.4 Extrapolated atmospheric gas conditions

The atmospheric conditions at the surface are input by the script’s user, and extrapolated upwards using reference atmospheric data from the ITU’s recommendation P.835-6. This data is given up to an altitude of 86 km. Above that point, pressure is extremely low. Because attenuation depends upon pressure, it becomes negligible. Therefore temperature and pressure are not calculated above 86 km.

The extrapolation calculations for both temperature and pressure utilize geopotential height, rather than standard geometric height [2]. This is altitude renormalized to a constant gravitational constant, which generally varies slightly [3]. The post-correction variable and unit are denoted with the ‘prime’ symbol (‘) [2], and can be found using

\[ h' = \frac{6356.766 \cdot h}{6356.766 + h} \]

\( h' = \) geopotential height (km’)

\( h = \) height (km)

#### 1.4.1 Temperature profile

The atmosphere can be divided into several zones based upon temperature gradient [2].

<table>
<thead>
<tr>
<th>Lower Boundary (km’)</th>
<th>Upper Boundary (km’)</th>
<th>Temperature Change (K/km’)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11</td>
<td>-6.5</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>47</td>
<td>2.8</td>
</tr>
<tr>
<td>47</td>
<td>51</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>71</td>
<td>-2.8</td>
</tr>
</tbody>
</table>
The temperature at a particular altitude depends upon the starting surface temperature, which zone the desired altitude lies inside, and its position within that zone [2]. The script works by first finding temperature at each zone boundary, then accounting for change within a zone. It can therefore be described by

\[ T = T_b + m \cdot (h' - h'_b) \]  

where:
- \( T \) = temperature (K)
- \( T_b \) = temperature of highest exceeded boundary (K)
- \( m \) = current zone temperature change (K/km’)
- \( h'_b \) = geopotential height of greatest exceeded boundary

Because the temperature gradient \( m \) is a constant whose value depends upon the current altitude zone, temperature with respect to altitude is a piecewise function. Sample temperature profiles with differing initial values are displayed in Figure 6.

![Surface Temperature](image)

**Fig. 6 – Temperature profile with different surface values**

### 1.4.2 Pressure profile

Pressure change with increasing altitude depends largely on temperature, and therefore uses the same set of atmospheric divisions and coefficients. However, pressure changes exponentially [2]. Below the thermosphere it is expressed by the piecewise function

\[
P = \begin{cases} 
  p_b \cdot \left( \frac{T_b}{T} \right)^{\frac{34.1632}{m}} & m \neq 0 \\
  p_b \cdot e^{-\frac{34.1632 \cdot (h' - h'_b)}{T}} & m = 0
\end{cases}
\]  

(11)
P = pressure (mb)

Pₜ = pressure of highest exceeded boundary (mb)

Sample pressure profiles are shown in Figure 7.

Fig. 7 – Pressure profile for different surface conditions. Pressure depends less on atmospheric layers than temperature does.

1.4.3 Water vapor density profile

The proportion of total pressure caused by water vapor, called the mixing ratio and given by $P_{wv}/P$, decreases with altitude to a minimum value of $2 \cdot 10^{-6}$, then remains constant [2]. After that point, water vapor density can be calculated using the altitude-dependent pressure and temperature [2]. The exponential relationship is

$$\rho = \rho_0 \cdot e^{-\frac{h}{2}}$$

(12)

$\rho = $ vapor density (g/m$^3$)

$\rho_0 = $ vapor density at surface (g/m$^3$)

If water vapor density is known, another method of finding vapor partial pressure is

$$P_{wv} = \frac{\rho \cdot T}{216.7}$$

(13)

Example vapor density profiles at varying values of surface humidity are plotted in Figure 8.
1.5 Specific attenuation due to atmospheric conditions

If attenuation caused by any specific factor is expressed in decibels, loss for a given sector of air may be determined by simply adding the specific attenuation due to each cause. For example, attenuation inside a cloud will be the sum of loss from atmospheric gases and loss from condensed water droplets, given that both are expressed in decibels. Various potential loss sources are summarized below, and Figures 8-12 show specific attenuation components for an example atmosphere. Specific attenuation above 86 km is assumed to be zero.

1.5.1 Loss from gas

Electromagnetic attenuation due to gas is divided into components caused by dry air and water vapor [4].

\[ \gamma_G = \gamma_O + \gamma_W \]  \hspace{1cm} (14)

Dry attenuation

Each component is a function of the imaginary part of the complex refractivity of the gas. The refractivity of dry air depends mostly upon oxygen, but is also influenced by nitrogen at particular frequencies and pressures [4]. Figure 9 is a visualization of this quantity as it changes with frequency and altitude in an example case.

\[ \gamma_O = 0.1820 \cdot f \cdot N_O'' \]  \hspace{1cm} (15)
\( \gamma_0 \) = attenuation due to dry air (dB/km)

\( f = \) frequency (GHz)

\( N_0'' = \) imaginary part of complex refractivity of dry air

The refractivity, given by Equation 16, is based upon oxygen spectroscopic data across all frequencies, and also includes the dry air continuum, a correction factor that accounts for the continuum spectrum of oxygen and attenuation caused by nitrogen as a function of pressure [4,5].

\[
N_0'' = \sum_i S_{iO} \cdot F_{iO} + N_D''
\]  

(16)

\( S_{iO} = \) strength of \( i^{th} \) oxygen line

\( F_{iO} = \) \( i^{th} \) oxygen line shape factor

\( N_D'' = \) imaginary part of dry air continuum

\[
S_{iO} = a_1 \cdot 10^{-7} \cdot P_d \cdot \theta^3 \cdot e^{a_2(1-\theta)}
\]  

(17)

\( a_i = \) value of \( j^{th} \) column of spectroscopic data table for oxygen

\( P_d = \) partial pressure of dry air (mb)

\( \theta = \) normalized temperature

\[
P_d = P - P_{wv}
\]  

(18)

\[
\theta = \frac{300}{T}
\]  

(19)

\[
F_{iO} = \frac{f}{f_{iO}} \cdot \left[ \frac{\Delta f_o - \delta \cdot (f_{iO} - f)}{(f_{iO} - f)^2 + \Delta f_o^2} + \frac{\Delta f_o - \delta \cdot (f_{iO} + f)}{(f_{iO} + f)^2 + \Delta f_o^2} \right]
\]  

(20)

\( f_{iO} = \) center frequency of \( i^{th} \) oxygen line (GHz)

\( \Delta f_o = \) oxygen line width (GHz)

\( \delta = \) correction factor for oxygen line interference effects

\[
\Delta f_{o,int} = a_3 \cdot 10^{-4} \cdot P_d \cdot \theta^{(\theta-a_4)} + 1.1 \cdot P_{wv} \cdot \theta
\]  

(21)

Here the line width is corrected for Zeeman splitting [4].

\[
\Delta f_o = \sqrt{\Delta f_{o,int}^2 + 2.25 \cdot 10^{-6}}
\]  

(22)

\[
\delta = (a_5 + a_6 \cdot \theta) \cdot 10^{-4} \cdot P \cdot \theta^8
\]  

(23)

\[
N_D'' = f \cdot P_d \cdot \theta^2 \cdot \left[ \frac{6.14 \cdot 10^{-5}}{d} \cdot \frac{1.4 \cdot 10^{-12} \cdot P_d \cdot \theta^{1.5}}{1 + 1.9 \cdot 10^{-5} \cdot f^{1.5}} \right]
\]  

(24)

d = width parameter for non-resonant oxygen spectrum
Table 3 - Oxygen spectroscopic data [4]

<table>
<thead>
<tr>
<th>f_o</th>
<th>a_1</th>
<th>a_2</th>
<th>a_3</th>
<th>a_4</th>
<th>a_5</th>
<th>a_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.474</td>
<td>0.975</td>
<td>9.651</td>
<td>6.69</td>
<td>0</td>
<td>2.566</td>
<td>6.85</td>
</tr>
<tr>
<td>50.988</td>
<td>2.529</td>
<td>8.653</td>
<td>7.17</td>
<td>0</td>
<td>2.246</td>
<td>6.8</td>
</tr>
<tr>
<td>51.503</td>
<td>6.193</td>
<td>7.709</td>
<td>7.64</td>
<td>0</td>
<td>1.947</td>
<td>6.729</td>
</tr>
<tr>
<td>52.021</td>
<td>14.32</td>
<td>6.819</td>
<td>8.11</td>
<td>0</td>
<td>1.667</td>
<td>6.64</td>
</tr>
<tr>
<td>52.542</td>
<td>31.24</td>
<td>5.983</td>
<td>8.58</td>
<td>0</td>
<td>1.388</td>
<td>6.526</td>
</tr>
<tr>
<td>53.067</td>
<td>64.29</td>
<td>5.201</td>
<td>9.06</td>
<td>0</td>
<td>1.349</td>
<td>6.206</td>
</tr>
<tr>
<td>53.596</td>
<td>124.6</td>
<td>4.474</td>
<td>9.55</td>
<td>0</td>
<td>2.227</td>
<td>5.085</td>
</tr>
<tr>
<td>54.13</td>
<td>227.3</td>
<td>3.8</td>
<td>9.96</td>
<td>0</td>
<td>3.17</td>
<td>3.75</td>
</tr>
<tr>
<td>54.671</td>
<td>389.7</td>
<td>3.182</td>
<td>10.37</td>
<td>0</td>
<td>3.558</td>
<td>2.654</td>
</tr>
<tr>
<td>55.221</td>
<td>627.1</td>
<td>2.618</td>
<td>10.89</td>
<td>0</td>
<td>2.56</td>
<td>2.952</td>
</tr>
<tr>
<td>55.784</td>
<td>945.3</td>
<td>2.109</td>
<td>11.34</td>
<td>0</td>
<td>-1.172</td>
<td>6.135</td>
</tr>
<tr>
<td>56.265</td>
<td>543.4</td>
<td>0.014</td>
<td>17.03</td>
<td>0</td>
<td>3.525</td>
<td>-0.978</td>
</tr>
<tr>
<td>56.363</td>
<td>1331.8</td>
<td>1.654</td>
<td>11.89</td>
<td>0</td>
<td>-2.378</td>
<td>6.547</td>
</tr>
<tr>
<td>56.968</td>
<td>1746.6</td>
<td>1.255</td>
<td>12.23</td>
<td>0</td>
<td>-3.525</td>
<td>6.451</td>
</tr>
<tr>
<td>57.612</td>
<td>2120.1</td>
<td>0.91</td>
<td>12.62</td>
<td>0</td>
<td>-5.416</td>
<td>6.056</td>
</tr>
<tr>
<td>58.324</td>
<td>2363.7</td>
<td>0.621</td>
<td>12.95</td>
<td>0</td>
<td>-1.932</td>
<td>0.436</td>
</tr>
<tr>
<td>58.447</td>
<td>1442.1</td>
<td>0.083</td>
<td>14.91</td>
<td>0</td>
<td>6.768</td>
<td>-1.273</td>
</tr>
<tr>
<td>59.164</td>
<td>2379.9</td>
<td>0.387</td>
<td>13.53</td>
<td>0</td>
<td>-6.561</td>
<td>2.309</td>
</tr>
<tr>
<td>59.591</td>
<td>2090.7</td>
<td>0.207</td>
<td>14.08</td>
<td>0</td>
<td>6.957</td>
<td>-0.776</td>
</tr>
<tr>
<td>60.306</td>
<td>2103.4</td>
<td>0.207</td>
<td>14.15</td>
<td>0</td>
<td>-6.395</td>
<td>0.699</td>
</tr>
<tr>
<td>60.435</td>
<td>2438</td>
<td>0.386</td>
<td>13.39</td>
<td>0</td>
<td>6.342</td>
<td>-2.825</td>
</tr>
<tr>
<td>61.151</td>
<td>2479.5</td>
<td>0.621</td>
<td>12.92</td>
<td>0</td>
<td>1.014</td>
<td>-0.584</td>
</tr>
<tr>
<td>61.8</td>
<td>2275.9</td>
<td>0.91</td>
<td>12.63</td>
<td>0</td>
<td>5.014</td>
<td>-6.619</td>
</tr>
<tr>
<td>62.411</td>
<td>1915.4</td>
<td>1.255</td>
<td>12.17</td>
<td>0</td>
<td>3.029</td>
<td>-6.759</td>
</tr>
<tr>
<td>62.486</td>
<td>1503</td>
<td>0.083</td>
<td>15.13</td>
<td>0</td>
<td>-4.499</td>
<td>0.844</td>
</tr>
<tr>
<td>62.998</td>
<td>1490.2</td>
<td>1.654</td>
<td>11.74</td>
<td>0</td>
<td>1.856</td>
<td>-6.675</td>
</tr>
<tr>
<td>63.569</td>
<td>1078</td>
<td>2.108</td>
<td>11.34</td>
<td>0</td>
<td>0.658</td>
<td>-6.139</td>
</tr>
<tr>
<td>64.128</td>
<td>728.7</td>
<td>2.617</td>
<td>10.88</td>
<td>0</td>
<td>-3.036</td>
<td>-2.895</td>
</tr>
<tr>
<td>64.679</td>
<td>461.3</td>
<td>3.181</td>
<td>10.38</td>
<td>0</td>
<td>-3.968</td>
<td>-2.59</td>
</tr>
<tr>
<td>65.224</td>
<td>274</td>
<td>3.8</td>
<td>9.96</td>
<td>0</td>
<td>-3.528</td>
<td>-3.68</td>
</tr>
<tr>
<td>66.302</td>
<td>80.4</td>
<td>5.2</td>
<td>9.06</td>
<td>0</td>
<td>-1.66</td>
<td>-6.091</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>dB/km</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>66.837</td>
<td>39.8</td>
<td>5.982</td>
<td>8.58</td>
<td>0</td>
<td>-1.68</td>
<td></td>
</tr>
<tr>
<td>67.37</td>
<td>18.56</td>
<td>6.818</td>
<td>8.11</td>
<td>0</td>
<td>-1.956</td>
<td></td>
</tr>
<tr>
<td>67.901</td>
<td>8.172</td>
<td>7.708</td>
<td>7.64</td>
<td>0</td>
<td>-2.216</td>
<td></td>
</tr>
<tr>
<td>68.431</td>
<td>3.397</td>
<td>8.652</td>
<td>7.17</td>
<td>0</td>
<td>-2.492</td>
<td></td>
</tr>
<tr>
<td>68.96</td>
<td>1.334</td>
<td>9.65</td>
<td>6.69</td>
<td>0</td>
<td>-2.773</td>
<td></td>
</tr>
<tr>
<td>118.75</td>
<td>940.3</td>
<td>0.01</td>
<td>16.64</td>
<td>0</td>
<td>-0.439</td>
<td></td>
</tr>
<tr>
<td>368.498</td>
<td>67.4</td>
<td>0.048</td>
<td>16.4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>424.763</td>
<td>637.7</td>
<td>0.044</td>
<td>16.4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>487.249</td>
<td>237.4</td>
<td>0.049</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>715.393</td>
<td>98.1</td>
<td>0.145</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>773.84</td>
<td>572.3</td>
<td>0.141</td>
<td>16.2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>834.145</td>
<td>183.1</td>
<td>0.145</td>
<td>16.2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

*Fig. 9 – Example dry air specific attenuation. Based upon a surface temperature of 15°C and a surface pressure of 1013mb.*
Water vapor attenuation

The refractivity of water vapor, an example of which is displayed in Figure 10, is calculated in much the same way as that for dry air. However, it is slightly simpler as the vapor continuum spectrum is included within the data table, and does not require a separate calculation [4]. It is given in Equation 25.

\[ \gamma_w = 0.1820 \cdot f \cdot N'_w \]  \hspace{1cm} (25)

\( \gamma_w \) = attenuation due to water vapor (dB/km)

\( N'_w \) = imaginary part of complex refractivity of water vapor

\[ N'_w = \sum_i S_{iw} \cdot F_{iw} \]  \hspace{1cm} (26)

\( S_{iw} \) = strength of \( i \)th vapor line

\( F_{iw} \) = \( i \)th vapor line shape factor

\[ S_{iw} = b_1 \cdot 10^{-1} \cdot P_{wv} \cdot \theta^{3.5} \cdot e^{b_2 \cdot (1-\theta)} \]  \hspace{1cm} (27)

\( b_j \) = value of \( j \)th column of spectroscopic data table for water vapor

\[ F_{iw} = \frac{f}{f_{iw}} \cdot \frac{\Delta f_w}{(f_{iw} - f)^2 + \Delta f_w^2} + \frac{\Delta f_w}{(f_{iw} + f)^2 + \Delta f_w^2} \]  \hspace{1cm} (28)

\( f_{iw} \) = center frequency of \( i \)th vapor line (GHz)

\( \Delta f_w \) = vapor line width (GHz)

\[ \Delta f_{wint} = b_3 \cdot 10^{-4} \cdot (P_d \cdot \theta^{b_4} + b_5 \cdot P_{wv} \cdot \theta^{b_6}) \]  \hspace{1cm} (29)

The line width is corrected here for Doppler broadening [4].

\[ \Delta f_w = 0.535 \cdot \Delta f_{wint} + \sqrt{0.217 \cdot (\Delta f_{wint})^2 + \frac{2.1316 \cdot 10^{-12} \cdot f_{iw}^2}{\theta}} \]  \hspace{1cm} (30)

<table>
<thead>
<tr>
<th>( f_{iw} )</th>
<th>( b_1 )</th>
<th>( b_2 )</th>
<th>( b_3 )</th>
<th>( b_4 )</th>
<th>( b_5 )</th>
<th>( b_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.235</td>
<td>0.1079</td>
<td>2.144</td>
<td>26.38</td>
<td>0.76</td>
<td>5.087</td>
<td>1</td>
</tr>
<tr>
<td>67.814</td>
<td>0.0011</td>
<td>8.732</td>
<td>28.58</td>
<td>0.69</td>
<td>4.93</td>
<td>0.82</td>
</tr>
<tr>
<td>119.996</td>
<td>0.0007</td>
<td>8.353</td>
<td>29.48</td>
<td>0.7</td>
<td>4.78</td>
<td>0.79</td>
</tr>
<tr>
<td>183.31</td>
<td>2.273</td>
<td>0.668</td>
<td>29.06</td>
<td>0.77</td>
<td>5.022</td>
<td>0.85</td>
</tr>
<tr>
<td>321.226</td>
<td>0.047</td>
<td>6.179</td>
<td>24.04</td>
<td>0.67</td>
<td>4.398</td>
<td>0.54</td>
</tr>
<tr>
<td>325.153</td>
<td>1.514</td>
<td>1.541</td>
<td>28.23</td>
<td>0.64</td>
<td>4.893</td>
<td>0.74</td>
</tr>
<tr>
<td>336.187</td>
<td>0.001</td>
<td>9.825</td>
<td>26.93</td>
<td>0.69</td>
<td>4.74</td>
<td>0.61</td>
</tr>
<tr>
<td>380.197</td>
<td>11.67</td>
<td>1.048</td>
<td>28.11</td>
<td>0.54</td>
<td>5.063</td>
<td>0.89</td>
</tr>
<tr>
<td>390.135</td>
<td>0.0045</td>
<td>7.347</td>
<td>21.52</td>
<td>0.63</td>
<td>4.81</td>
<td>0.55</td>
</tr>
<tr>
<td>437.347</td>
<td>0.0632</td>
<td>5.048</td>
<td>18.45</td>
<td>0.6</td>
<td>4.23</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>--------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>439.151</td>
<td>0.9098</td>
<td>3.595</td>
<td>20.07</td>
<td>0.63</td>
<td>4.483</td>
<td>0.52</td>
</tr>
<tr>
<td>443.018</td>
<td>0.192</td>
<td>5.048</td>
<td>15.55</td>
<td>0.6</td>
<td>5.083</td>
<td>0.5</td>
</tr>
<tr>
<td>448.001</td>
<td>10.41</td>
<td>1.405</td>
<td>25.64</td>
<td>0.66</td>
<td>5.028</td>
<td>0.67</td>
</tr>
<tr>
<td>470.889</td>
<td>0.3254</td>
<td>3.597</td>
<td>21.34</td>
<td>0.66</td>
<td>4.506</td>
<td>0.65</td>
</tr>
<tr>
<td>474.689</td>
<td>1.26</td>
<td>2.379</td>
<td>23.2</td>
<td>0.65</td>
<td>4.804</td>
<td>0.64</td>
</tr>
<tr>
<td>488.491</td>
<td>0.2529</td>
<td>2.852</td>
<td>25.86</td>
<td>0.69</td>
<td>5.201</td>
<td>0.72</td>
</tr>
<tr>
<td>503.569</td>
<td>0.0372</td>
<td>6.731</td>
<td>16.12</td>
<td>0.61</td>
<td>3.98</td>
<td>0.43</td>
</tr>
<tr>
<td>504.483</td>
<td>0.0124</td>
<td>6.731</td>
<td>16.12</td>
<td>0.61</td>
<td>4.01</td>
<td>0.45</td>
</tr>
<tr>
<td>547.6764</td>
<td>0.9785</td>
<td>0.158</td>
<td>26</td>
<td>0.7</td>
<td>4.5</td>
<td>1</td>
</tr>
<tr>
<td>552.021</td>
<td>0.184</td>
<td>0.158</td>
<td>26</td>
<td>0.7</td>
<td>4.5</td>
<td>1</td>
</tr>
<tr>
<td>556.936</td>
<td>497</td>
<td>0.159</td>
<td>30.86</td>
<td>0.69</td>
<td>4.552</td>
<td>1</td>
</tr>
<tr>
<td>620.701</td>
<td>5.015</td>
<td>2.391</td>
<td>24.38</td>
<td>0.71</td>
<td>4.856</td>
<td>0.68</td>
</tr>
<tr>
<td>645.7661</td>
<td>0.0067</td>
<td>8.633</td>
<td>18</td>
<td>0.6</td>
<td>4</td>
<td>0.5</td>
</tr>
<tr>
<td>658.006</td>
<td>0.2732</td>
<td>7.816</td>
<td>32.1</td>
<td>0.69</td>
<td>4.14</td>
<td>1</td>
</tr>
<tr>
<td>752.033</td>
<td>243.4</td>
<td>0.396</td>
<td>30.86</td>
<td>0.68</td>
<td>4.352</td>
<td>0.84</td>
</tr>
<tr>
<td>841.074</td>
<td>0.0134</td>
<td>8.177</td>
<td>15.9</td>
<td>0.33</td>
<td>5.76</td>
<td>0.45</td>
</tr>
<tr>
<td>859.865</td>
<td>0.1325</td>
<td>8.055</td>
<td>30.6</td>
<td>0.68</td>
<td>4.09</td>
<td>0.84</td>
</tr>
<tr>
<td>899.407</td>
<td>0.0547</td>
<td>7.914</td>
<td>29.85</td>
<td>0.68</td>
<td>4.53</td>
<td>0.9</td>
</tr>
<tr>
<td>902.555</td>
<td>0.0386</td>
<td>8.429</td>
<td>28.65</td>
<td>0.7</td>
<td>5.1</td>
<td>0.95</td>
</tr>
<tr>
<td>906.206</td>
<td>0.1836</td>
<td>5.11</td>
<td>24.08</td>
<td>0.7</td>
<td>4.7</td>
<td>0.53</td>
</tr>
<tr>
<td>916.172</td>
<td>8.4</td>
<td>1.441</td>
<td>26.73</td>
<td>0.7</td>
<td>5.15</td>
<td>0.78</td>
</tr>
<tr>
<td>923.1127</td>
<td>0.0079</td>
<td>10.293</td>
<td>29</td>
<td>0.7</td>
<td>5</td>
<td>0.8</td>
</tr>
<tr>
<td>970.315</td>
<td>9.009</td>
<td>1.919</td>
<td>25.5</td>
<td>0.64</td>
<td>4.94</td>
<td>0.67</td>
</tr>
<tr>
<td>987.927</td>
<td>134.6</td>
<td>0.257</td>
<td>29.85</td>
<td>0.68</td>
<td>4.55</td>
<td>0.9</td>
</tr>
<tr>
<td>1780</td>
<td>17506</td>
<td>0.952</td>
<td>196.3</td>
<td>2</td>
<td>24.15</td>
<td>5</td>
</tr>
</tbody>
</table>
1.5.2 Loss from clouds/fog

Attenuation due to condensed water in clouds or fog requires knowledge of the liquid water density and temperature in the atmosphere, in addition to cloud or fog height [6]. It can be found from

\[ \gamma_C = K \cdot M \]  

\( \gamma_C \) = attenuation due to liquid water droplets (dB/km)

\( K \) = liquid water specific attenuation coefficient

\( M \) = liquid water density (g/m³)

Equation 32 for the liquid water specific attenuation coefficient has a lower range than other equations used here, in that its upper limit is 200 GHz [6]. It is based upon the complex permittivity of water [6].

\[ K = \frac{0.819 \cdot f}{\varepsilon'' \cdot (1 + \eta^2)} \]  

\[ \eta = \frac{2 + \varepsilon'}{\varepsilon''} \]  

\[ \varepsilon' = \frac{\varepsilon_0 - \varepsilon_1}{1 + \left( \frac{f}{f_p} \right)^2} + \frac{\varepsilon_1 - \varepsilon_2}{1 + \left( \frac{f}{f_s} \right)^2} + \varepsilon_2 \]
\[ \varepsilon'' = \frac{f \cdot (\varepsilon_0 - \varepsilon_1)}{f_p \cdot \left[1 + \left(\frac{f}{f_p}\right)^2\right]} + \frac{f \cdot (\varepsilon_1 - \varepsilon_2)}{f_s \cdot \left[1 + \left(\frac{f}{f_s}\right)^2\right]} \]  

Equation (35)

\[ \varepsilon_0 = 77.66 + 103.3 \cdot (\theta - 1) \]  

Equation (36)

\[ \varepsilon_1 = 0.0671 \cdot \varepsilon_0 \]  

Equation (37)

\[ \varepsilon_2 = 3.52 \]  

Equation (38)

\[ f_p = 20.20 - 146 \cdot (\theta - 1) + 316 \cdot (\theta - 1)^2 \]  

Equation (39)

\[ f_s = 39.8 \cdot f_p \]  

Equation (40)

\( f_p \) = principal relaxation frequency (GHz)

\( f_s \) = secondary relaxation frequency (GHz)

\( \theta \) here refers to the normalized temperature of the liquid water droplets, not of the atmospheric gases [6]. However, the application assumes them to be the same. Figure 11 displays calculated data for an example situation.

![Figure 11](image-url)

Fig. 11 – Example specific attenuation due to clouds. Based upon a surface temperature of 15°C and a liquid water density of .33g/m³.

### 1.5.3 Loss from rain

Attenuation due to rain can be expressed by the following equation [7].

\[ \gamma_R = k \cdot R^\alpha \]  

Equation (41)
\( \gamma_R \) = attenuation due to rain

The parameters \( k \) and \( \alpha \) are given by

\[
k = \frac{k_H + k_V + (k_H - k_V) \cdot \cos(\varphi)^2 \cdot \cos(2 \cdot \tau)}{2}
\]

\[
\alpha = \frac{k_H \cdot \alpha_H + k_V \cdot \alpha_V + (k_H \cdot \alpha_H - k_V \cdot \alpha_V) \cdot \cos(\varphi)^2 \cdot \cos(2 \cdot \tau)}{2 \cdot k}
\]

\( \varphi \) = apparent elevation angle (rad)

\( \tau \) = polarization angle (° from horizontal)

\( k_H \) = parameter \( k \) if wave polarization is entirely horizontal

\( k_V \) = parameter \( k \) if wave polarization is entirely vertical

\( \alpha_H \) = parameter \( \alpha \) if polarization is entirely horizontal

\( \alpha_V \) = parameter \( \alpha \) if polarization is entirely vertical

Polarization angle is an input parameter of the script, and apparent elevation angle is discussed in the following section. As for the \( k \) and \( \alpha \) values, for the horizontal case they can be found using

\[
\log_{10} k_H = \sum_{i=1}^{4} \left( a_{ikH} \cdot e^{-\frac{(\log_{10} f - b_{ikH})}{c_{ikH}}^2} \right) + m_{kH} \cdot \log_{10} f + c_{kH}
\]

\[
\alpha_H = \sum_{i=1}^{5} \left( a_{i\alpha H} \cdot e^{-\frac{(\log_{10} f - b_{i\alpha H})}{c_{i\alpha H}}^2} \right) + m_{\alpha H} \cdot \log_{10} f + c_{\alpha H}
\]

The expressions for vertical polarization are identical, save the constants [7].

\[
\log_{10} k_V = \sum_{i=1}^{4} \left( a_{i\alpha V} \cdot e^{-\frac{(\log_{10} f - b_{i\alpha V})}{c_{i\alpha V}}^2} \right) + m_{kV} \cdot \log_{10} f + c_{kV}
\]

\[
\alpha_V = \sum_{i=1}^{5} \left( a_{i\alpha V} \cdot e^{-\frac{(\log_{10} f - b_{i\alpha V})}{c_{i\alpha V}}^2} \right) + m_{\alpha V} \cdot \log_{10} f + c_{\alpha V}
\]

Table 5 - Rain attenuation coefficients [7]

<table>
<thead>
<tr>
<th>( i )</th>
<th>( a_{i\alpha H} )</th>
<th>( b_{i\alpha H} )</th>
<th>( c_{i\alpha H} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-5.33980</td>
<td>-.10008</td>
<td>1.13098</td>
</tr>
<tr>
<td>2</td>
<td>-.35351</td>
<td>1.26970</td>
<td>.45400</td>
</tr>
<tr>
<td>3</td>
<td>-.23789</td>
<td>.86036</td>
<td>.15354</td>
</tr>
<tr>
<td>4</td>
<td>-.94158</td>
<td>.64552</td>
<td>.16817</td>
</tr>
<tr>
<td>( m_{kH} )</td>
<td>( _ )</td>
<td>( c_{\alpha H} )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-.18961</td>
<td></td>
<td>.71147</td>
</tr>
</tbody>
</table>
### Coefficients for horizontal polarization:

<table>
<thead>
<tr>
<th>i</th>
<th>a_{ih}</th>
<th>b_{ih}</th>
<th>c_{ih}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-.14318</td>
<td>1.82442</td>
<td>-.55187</td>
</tr>
<tr>
<td>2</td>
<td>.29591</td>
<td>.77564</td>
<td>.19822</td>
</tr>
<tr>
<td>3</td>
<td>.32177</td>
<td>.63773</td>
<td>.13164</td>
</tr>
<tr>
<td>4</td>
<td>-5.37610</td>
<td>-.96230</td>
<td>1.47828</td>
</tr>
<tr>
<td>5</td>
<td>16.1721</td>
<td>-3.29980</td>
<td>3.43990</td>
</tr>
<tr>
<td>m_{ih}</td>
<td>c_{ih}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.67849</td>
<td>-1.95537</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Coefficients for vertical polarization:

<table>
<thead>
<tr>
<th>i</th>
<th>a_{iv}</th>
<th>b_{iv}</th>
<th>c_{iv}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-3.80595</td>
<td>.56934</td>
<td>.81061</td>
</tr>
<tr>
<td>2</td>
<td>-3.44965</td>
<td>-.22911</td>
<td>.51059</td>
</tr>
<tr>
<td>3</td>
<td>-3.9902</td>
<td>.73042</td>
<td>.11899</td>
</tr>
<tr>
<td>4</td>
<td>.50167</td>
<td>1.07319</td>
<td>.27195</td>
</tr>
<tr>
<td>m_{iv}</td>
<td>c_{iv}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-.16398</td>
<td>.63297</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>i</th>
<th>a_{iv}</th>
<th>b_{iv}</th>
<th>c_{iv}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-.07771</td>
<td>2.33840</td>
<td>-.76284</td>
</tr>
<tr>
<td>2</td>
<td>.56727</td>
<td>.95545</td>
<td>.54039</td>
</tr>
<tr>
<td>3</td>
<td>-.20238</td>
<td>1.14520</td>
<td>.26809</td>
</tr>
<tr>
<td>4</td>
<td>-48.2991</td>
<td>.791669</td>
<td>.116226</td>
</tr>
<tr>
<td>5</td>
<td>48.5833</td>
<td>.791459</td>
<td>.116479</td>
</tr>
<tr>
<td>m_{iv}</td>
<td>c_{iv}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-.053739</td>
<td>.83433</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An example of attenuation caused by rain is Figure 12, and Figure 13 shows the total specific attenuation for the example case.
Fig. 12 – Example rain specific attenuation. Based upon a rain rate of 4mm/hr.
1.6 Path length

The total attenuation of electromagnetic waves propagating through the atmosphere is simply a path integral of the specific attenuation of every region they pass through. However, as waves propagate they are also refracted, and as a consequence their path is not entirely linear. Therefore the apparent elevation angle, the angle the Poynting vector of the wave makes with the horizontal, must be accounted for not only at the transmitting point, but at every point along the path [4]. The integral as a function of altitude is shown below [4].

\[
A = \int_{h_1}^{h_2} \frac{\gamma}{\sin(\varphi)} dh
\]

\(A\) = total attenuation (dB)
\(\gamma\) = specific attenuation (dB/km)
\(h_1\) = height of lower point
\(h_2\) = height of higher point

\[
\sin(\varphi) = \sqrt{1 - \cos(\varphi)^2}
\]

\[
\cos(\varphi) = \frac{(R_E + h_1) \cdot n(h_1)}{(R_E + h) \cdot n(h)} \cdot \cos(\varphi_1)
\]
\[ R_E = \text{average radius of the Earth (6371km)} \]
\[ n = \text{refractive index at specified height} \]

If the apparent elevation angle is known at the higher point rather than the lower, the following formula may be used to convert [4].

\[
\varphi_1 = \arccos \left( \frac{\left(R_E + h_2\right) \cdot n(h_2) \cdot \cos(\varphi_2)}{\left(R_E + h_1\right) \cdot n(h_1)} \right) \tag{51}
\]

The refractive index used in the above equations may be found from recommendation P.453-14 [1].

\[
n = 1 + N \cdot 10^{-6} \tag{52}
\]

\[ N = \text{refractivity (N-units)} \]

\[
N = 77.6 \cdot \frac{P_d}{T} + 72 \cdot \frac{P_{wv}}{T} + 3.75 \cdot 10^5 \cdot \frac{P_{wv}}{T^2} \tag{53}
\]

An important note here is that it addresses the refractive index only of gas, and does not account for aerosols or precipitation. The accuracy will therefore be degraded if weather is input into the application. Further investigation is needed to remedy this issue.

From [4], the integral may be evaluated numerically using

\[
A = \sum_{i=1}^{L} \frac{a_i \cdot \gamma_i}{\sin(\varphi_i)} \tag{54}
\]

\[ L = \text{number of atmospheric layers considered} \]
\[ a_i = \text{path length through the } i^{\text{th}} \text{ layer (km)} \]

Figure 14 shows the attenuation to the surface for the example atmosphere as a function of height.
Fig. 14 – Attenuation to ground in the example case. Calculated assuming the beam is transmitted at nadir, or with a grazing angle of -90°.

1.7 Dispersion

The atmosphere also introduces phase shift into EM waves that travel through it [4]. In this calculation, only dispersion due to atmospheric gases is given. Any effect from condensed water or other weather is not accounted for.

The gas-induced specific dispersion for an example atmosphere is shown in Figure 15. While attenuation depends upon the imaginary component of complex refractivity, phase dispersion depends on the real component [4]. It can be calculated in a similar way, by summing the dispersion due to dry air and water vapor [4] as shown here.

\[ \beta_G = \beta_O + \beta_W \] (55)

\( \beta_G \) = dispersion due to gas (°/km)
\( \beta_O \) = dispersion due to dry air (°/km)
\( \beta_W \) = dispersion due to water vapor (°/km)

**Dry air dispersion**

The dry air dispersion, shown in Equation 56, is calculated very similarly to the attenuation but uses the real part of certain quantities rather than the imaginary part. The real part of the dry air continuum is based only upon pressure-induced nitrogen absorption, and does not include an effect from the non-resonant oxygen spectrum [4,5].

\[ \beta_O = -1.2008 \cdot f \cdot N_O' \] (56)
Atmospheric Modeling for Electromagnetic Propagation

\( N'_o = \text{real part of complex refractivity of dry air} \)

\[ N'_o = \sum S_{iO} \cdot F'_{iO} + N'_D \] (57)

\( F'_{iO} = \text{real part of } i^{th} \text{ oxygen line shape factor} \)

\( N'_D = \text{real part of dry air continuum} \)

\[ F'_{iO} = \frac{f}{f_{iO}} \left[ \frac{(f_{iO} - f) + \delta \cdot \Delta f_{O}}{(f_{iO} - f)^2 + \Delta f_{O}^2} + \frac{(f_{iO} + f) - \delta \cdot \Delta f_{O}}{(f_{iO} + f)^2 + \Delta f_{O}^2} \right] \] (58)

\[ N'_D = \frac{-6.14 \cdot 10^{-5} \cdot P_d \cdot \theta^2 \cdot f^2}{f^2 + d^2} \] (59)

**Water vapor dispersion**

Vapor dispersion is found similarly to dry air dispersion, but of course uses different spectroscopic data and the vapor continuum is not found separately [4]. It is given by

\( \beta_w = -1.2008 \cdot f \cdot N'_w \) (60)

\( N'_w = \text{real part of complex refractivity of water vapor} \)

\[ N'_w = \sum S_{iW} \cdot F'_{iW} \] (61)

\( F'_{iW} = \text{real part of } i^{th} \text{ vapor line shape factor} \)

\[ F'_{iW} = \frac{f}{f_{iW}} \left[ \frac{(f_{iW} - f) + \delta \cdot \Delta f_{W}}{(f_{iW} - f)^2 + \Delta f_{W}^2} + \frac{(f_{iW} + f) - \delta \cdot \Delta f_{W}}{(f_{iW} + f)^2 + \Delta f_{W}^2} \right] \] (62)
Fig. 15 – Example phase dispersion, calculated from a surface temperature of 15°C, pressure of 1013mb, and humidity of 50%. Only the effects of gases are included.
CONCLUSION

The software discussed here is capable of modeling the Earth’s atmosphere and calculating RF attenuation and phase dispersion within it. Supported weather types are clear air, clouds, rain, and fog, although dispersion cannot be calculated correctly through aerosols. Effects of the ionosphere are also neglected, and these limitations are noted and displayed should they occur. Appendix 1 contains MATLAB source code, and Appendix 2 explains an alternate version that bypasses the user interface.

REFERENCES


APPENDIX 1

atmos_app_lite.mlapp

classdef atmos_app_lite < matlab.apps.AppBase

% Properties that correspond to app components
properties (Access = public)
    UIFigure
    GroundTemperatureEditFieldLabel
    GroundTemperatureEditField
    GroundPressureEditFieldLabel
    GroundPressureEditField
    RelativeHumidityEditFieldLabel
    RelativeHumidityEditField
    CloudsCheckBox
    AltitudeEditFieldLabel
    AltitudeEditField
    HeightEditField
    HeightEditFieldLabel
    WaterConcentrationEditFieldLabel
    WaterConcentrationEditField
    FogCheckBox
    HeightEditField_2Label
    HeightEditField_2
    WaterConcentrationEditField_2Label
    WaterConcentrationEditField_2
    RainCheckBox
    RateEditFieldLabel
    RateEditField
    AltitudeEditField_2Label
    AltitudeEditField_2
    GrazingAngleEditFieldLabel
    GrazingAngleEditField
    RunButton
    SaveButton
    LoadButton
    IntervalEditFieldLabel
    IntervalEditField
    KLabel
    mbLabel
    Label
    mLabel
    mLabel_2
    gm3Label
    mLabel_3
    gm3Label_2
    kmLabel
    Label_3
% Callbacks that handle component events
methods (Access = private)

% Value changed function: CloudsCheckBox
function CloudsCheckBoxValueChanged(app, event)
    value = app.CloudsCheckBox.Value;
end

% Button pushed function: RunButton
function RunButtonPushed(app, event)
[filepath,~,~] = fileparts(mfilename('fullpath'));
cd(filepath);
loss_calculator_lite(app);
end

% Button pushed function: SaveButton
function SaveButtonPushed(app, event)
    [savefile,savepath] = uiputfile;
    savefile = savefile(1:end-7);
    params = struct('angle',app.GrazingAngleEditField.Value,'altitude',app.AltitudeEditField_2.Value,
    'fogH2Oconc',app.WaterConcentrationEditField_2.Value,'fogtop',app.HeightEditField_2.Value,
    'cloudH2Oconc',app.WaterConcentrationEditField.Value,
    'cloudht',app.HeightEditField.Value,'cloudalt',app.AltitudeEditField.Value,
    'freq',app.FrequencyEditField.Value,'interval',app.IntervalEditField.Value,
    'freqStep',app.StepEditField.Value,
    'freqStart',app.StartEditField.Value,'freqStop',app.StopEditField.Value,
    'notes',app.NotesTextArea.Value,'saveFile',app.SaveFileCheckBox.Value);
    save(strcat(savepath,savefile,'.mat'),"params");
end

% Button pushed function: LoadButton
function LoadButtonPushed(app, event)
    [loadfile,loadpath] = uigetfile;
    load(strcat(loadpath,loadfile));
    app.GrazingAngleEditField.Value = params.angle;
    app.AltitudeEditField_2.Value = params.altitude;
    app.PolarizationEditField.Value = params.pol;
    app.GroundTemperatureEditField.Value = params.Tgnd;
app.GroundPressureEditField.Value = params.Pgnd;
app.RelativeHumidityEditField.Value = params.RH;
app.RateEditField.Value = params.rain_rt;
app.WaterConcentrationEditField_2.Value = params.fogH2Oconc;
app.HeightEditField_2.Value = params.fogtop;
app.WaterConcentrationEditField.Value = params.cloudH2Oconc;
app.HeightEditField.Value = params.cloudht;
app.AltitudeEditField.Value = params.cloudalt;
app.TargetAltitudeEditField.Value = params.targetht;
app.FrequencyEditField.Value = params.freq;
app.IntervalEditField.Value = params.interval;
app.AttenuationEditField.Value = params.atten;
app.DispersionEditField.Value = params.disp;
app.IncidenceAngleEditField.Value = params.resAngle;
app.FrequencySweepCheckBox.Value = params.sweepCheck;
app.StartEditField.Value = params.freqStart;
app.StepEditField.Value = params.freqStep;
app.StopEditField.Value = params.freqStop;
app.NotesTextArea.Value = params.notes;
app.SaveFileCheckBox.Value = params.saveFile;

end
end

% Component initialization
% methods (Access = private)

% Create UIFigure and components
function createComponents(app)

% Create UIFigure and hide until all components are created
app.UIFigure = uifigure('Visible', 'off');
app.UIFigure.Position = [100 100 743 575];
app.UIFigure.Name = 'UI Figure';

% Create GroundTemperatureEditFieldLabel
app.GroundTemperatureEditFieldLabel = uilabel(app.UIFigure);
app.GroundTemperatureEditFieldLabel.HorizontalAlignment = 'right';
app.GroundTemperatureEditFieldLabel.Position = [31 497 116 22];
app.GroundTemperatureEditFieldLabel.Text = 'Ground Temperature';

% Create GroundTemperatureEditField
app.GroundTemperatureEditField = uieditfield(app.UIFigure, 'numeric');
app.GroundTemperatureEditField.Position = [162 497 100 22];
app.GroundTemperatureEditField.Value = 288;
% Create GroundPressureEditFieldLabel
app.GroundPressureEditFieldLabel = uilabel(app.UIFigure);
app.GroundPressureEditFieldLabel.HorizontalAlignment = 'right';
app.GroundPressureEditFieldLabel.Position = [50 476 97 22];
app.GroundPressureEditFieldLabel.Text = 'Ground Pressure';

% Create GroundPressureEditField
app.GroundPressureEditField = uieditfield(app.UIFigure, 'numeric');
app.GroundPressureEditField.Position = [162 476 100 22];
app.GroundPressureEditField.Value = 1013;

% Create RelativeHumidityEditFieldLabel
app.RelativeHumidityEditFieldLabel = uilabel(app.UIFigure);
app.RelativeHumidityEditFieldLabel.HorizontalAlignment = 'right';
app.RelativeHumidityEditFieldLabel.Position = [48 455 99 22];
app.RelativeHumidityEditFieldLabel.Text = 'Relative Humidity';

% Create RelativeHumidityEditField
app.RelativeHumidityEditField = uieditfield(app.UIFigure, 'numeric');
app.RelativeHumidityEditField.Position = [162 455 100 22];
app.RelativeHumidityEditField.Value = 50;

% Create CloudsCheckBox
app.CloudsCheckBox = uicheckbox(app.UIFigure);
app.CloudsCheckBox.ValueChangedFcn = createCallbackFcn(app, @CloudsCheckBoxValueChanged, true);
app.CloudsCheckBox.Text = 'Clouds';
app.CloudsCheckBox.Position = [57 397 59 22];

% Create AltitudeEditFieldLabel
app.AltitudeEditFieldLabel = uilabel(app.UIFigure);
app.AltitudeEditFieldLabel.HorizontalAlignment = 'right';
app.AltitudeEditFieldLabel.Position = [101 376 46 22];
app.AltitudeEditFieldLabel.Text = 'Altitude';

% Create AltitudeEditField
app.AltitudeEditField = uieditfield(app.UIFigure, 'numeric');
app.AltitudeEditField.Position = [162 376 100 22];

% Create HeightEditFieldLabel
app.HeightEditFieldLabel = uilabel(app.UIFigure);
% Create HeightEditField
app.HeightEditField = uieditfield(app.UIFigure, 'numeric');
app.HeightEditField.Position = [162 355 100 22];

% Create WaterConcentrationEditFieldLabel
app.WaterConcentrationEditFieldLabel = uilabel(app.UIFigure);
app.WaterConcentrationEditFieldLabel.HorizontalAlignment = 'right';
app.WaterConcentrationEditFieldLabel.Position = [32 334 115 22];
app.WaterConcentrationEditFieldLabel.Text = 'Water Concentration';

% Create WaterConcentrationEditField
app.WaterConcentrationEditField = uieditfield(app.UIFigure, 'numeric');
app.WaterConcentrationEditField.Position = [162 334 100 22];

% Create FogCheckBox
app.FogCheckBox = uicheckbox(app.UIFigure);
app.FogCheckBox.Text = 'Fog';
app.FogCheckBox.Position = [57 289 43 22];

% Create HeightEditField_2Label
app.HeightEditField_2Label = uilabel(app.UIFigure);
app.HeightEditField_2Label.HorizontalAlignment = 'right';
app.HeightEditField_2Label.Position = [107 268 40 22];
app.HeightEditField_2Label.Text = 'Height';

% Create HeightEditField_2
app.HeightEditField_2 = uieditfield(app.UIFigure, 'numeric');
app.HeightEditField_2.Position = [162 268 100 22];

% Create WaterConcentrationEditField_2Label
app.WaterConcentrationEditField_2Label = uilabel(app.UIFigure);
app.WaterConcentrationEditField_2Label.HorizontalAlignment = 'right';
app.WaterConcentrationEditField_2Label.Position = [32 247 115 22];
app.WaterConcentrationEditField_2Label.Text = 'Water Concentration';

% Create WaterConcentrationEditField_2
app.WaterConcentrationEditField_2 = uieditfield(app.UIFigure, 'numeric');
app.WaterConcentrationEditField_2.Position = [162 247 100 22];
% Create RainCheckBox
app.RainCheckBox = uicontrol(app.UIFigure);
app.RainCheckBox.Text = 'Rain';
app.RainCheckBox.Position = [57 206 47 22];

% Create RateEditFieldLabel
app.RateEditFieldLabel = uicontrol(app.UIFigure);
app.RateEditFieldLabel.HorizontalAlignment = 'right';
app.RateEditFieldLabel.Position = [116 185 31 22];
app.RateEditFieldLabel.Text = 'Rate';

% Create RateEditField
app.RateEditField = uicontrol(app.UIFigure, 'numeric');
app.RateEditField.Position = [162 185 100 22];

% Create AltitudeEditField_2Label
app.AltitudeEditField_2Label = uicontrol(app.UIFigure);
app.AltitudeEditField_2Label.HorizontalAlignment = 'right';
app.AltitudeEditField_2Label.Position = [397 318 46 22];
app.AltitudeEditField_2Label.Text = 'Altitude';

% Create AltitudeEditField_2
app.AltitudeEditField_2 = uicontrol(app.UIFigure, 'numeric');
app.AltitudeEditField_2.Position = [458 318 100 22];
app.AltitudeEditField_2.Value = 6;

% Create GrazingAngleEditFieldLabel
app.GrazingAngleEditFieldLabel = uicontrol(app.UIFigure);
app.GrazingAngleEditFieldLabel.HorizontalAlignment = 'right';
app.GrazingAngleEditFieldLabel.Position = [361 297 82 22];
app.GrazingAngleEditFieldLabel.Text = 'Grazing Angle';

% Create GrazingAngleEditField
app.GrazingAngleEditField = uicontrol(app.UIFigure, 'numeric');
app.GrazingAngleEditField.Position = [458 297 100 22];
app.GrazingAngleEditField.Value = -90;

% Create RunButton
app.RunButton = uicontrol(app.UIFigure, 'push');
app.RunButton.ButtonPushedFcn = createCallbackFcn(app, @.RunButtonPushed, true);
app.RunButton.Position = [601 36 100 22];
app.RunButton.Text = 'Run';

% Create SaveButton
app.SaveButton = uibutton(app.UIFigure, 'push');
app.SaveButton.ButtonPushedFcn = createCallbackFcn(app, @SaveButtonPushed, true);
app.SaveButton.Position = [478 57 100 22];
app.SaveButton.Text = 'Save';

% Create LoadButton
app.LoadButton = uibutton(app.UIFigure, 'push');
app.LoadButton.ButtonPushedFcn = createCallbackFcn(app, @LoadButtonPushed, true);
app.LoadButton.Position = [478 36 100 22];
app.LoadButton.Text = 'Load';

% Create IntervalEditFieldLabel
app.IntervalEditFieldLabel = uilabel(app.UIFigure);
app.IntervalEditFieldLabel.HorizontalAlignment = 'right';
app.IntervalEditFieldLabel.Position = [539 90 45 22];
app.IntervalEditFieldLabel.Text = 'Interval';

% Create IntervalEditField
app.IntervalEditField = uieditfield(app.UIFigure, 'numeric');
app.IntervalEditField.Position = [599 90 100 22];
app.IntervalEditField.Value = 100;

% Create KLabel
app.KLabel = uilabel(app.UIFigure);
app.KLabel.Position = [267 497 25 22];
app.KLabel.Text = 'K';

% Create mbLabel
app.mbLabel = uilabel(app.UIFigure);
app.mbLabel.Position = [267 476 25 22];
app.mbLabel.Text = 'mb';

% Create Label
app.Label = uilabel(app.UIFigure);
app.Label.Position = [267 455 25 22];
app.Label.Text = '%';
% Create mLabel
app.mLabel = uilabel(app.UIFigure);
app.mLabel.Position = [267 376 25 22];
app.mLabel.Text = 'm';

% Create mLabel_2
app.mLabel_2 = uilabel(app.UIFigure);
app.mLabel_2.Position = [267 355 25 22];
app.mLabel_2.Text = 'm';

% Create gm3Label
app.gm3Label = uilabel(app.UIFigure);
app.gm3Label.Position = [267 334 38 22];
app.gm3Label.Text = 'g/m^3';

% Create mLabel_3
app.mLabel_3 = uilabel(app.UIFigure);
app.mLabel_3.Position = [267 268 25 22];
app.mLabel_3.Text = 'm';

% Create gm3Label_2
app.gm3Label_2 = uilabel(app.UIFigure);
app.gm3Label_2.Position = [267 247 38 22];
app.gm3Label_2.Text = 'g/m^3';

% Create kmLabel
app.kmLabel = uilabel(app.UIFigure);
app.kmLabel.Position = [563 318 25 22];
app.kmLabel.Text = 'km';

% Create Label_3
app.Label_3 = uilabel(app.UIFigure);
app.Label_3.Position = [563 297 25 22];
app.Label_3.Text = '°';

% Create mmhrLabel
app.mmhrLabel = uilabel(app.UIFigure);
app.mmhrLabel.Position = [266 185 39 22];
app.mmhrLabel.Text = 'mm/hr';

% Create mLabel_6
app.mLabel_6 = uilabel(app.UIFigure);
app.mLabel_6.Position = [700 90 25 22];
app.mLabel_6.Text = 'm';

% Create FrequencyEditFieldLabel
app.FrequencyEditFieldLabel = uilabel(app.UIFigure);
app.FrequencyEditFieldLabel.HorizontalAlignment = 'right';
app.FrequencyEditFieldLabel.Position = [419 497 62 22];
app.FrequencyEditFieldLabel.Text = 'Frequency';

% Create FrequencyEditField
app.FrequencyEditField = uieditfield(app.UIFigure, 'numeric');
app.FrequencyEditField.Position = [496 497 100 22];
app.FrequencyEditField.Value = 20;

% Create GHzLabel
app.GHzLabel = uilabel(app.UIFigure);
app.GHzLabel.Position = [602 497 29 22];
app.GHzLabel.Text = 'GHz';

% Create AttenuationEditFieldLabel
app.AttenuationEditFieldLabel = uilabel(app.UIFigure);
app.AttenuationEditFieldLabel.HorizontalAlignment = 'right';
app.AttenuationEditFieldLabel.Position = [466 206 66 22];
app.AttenuationEditFieldLabel.Text = 'Attenuation';

% Create AttenuationEditField
app.AttenuationEditField = uieditfield(app.UIFigure, 'numeric');
app.AttenuationEditField.Position = [547 206 100 22];

% Create DispersionEditFieldLabel
app.DispersionEditFieldLabel = uilabel(app.UIFigure);
app.DispersionEditFieldLabel.HorizontalAlignment = 'right';
app.DispersionEditFieldLabel.Position = [470 185 62 22];
app.DispersionEditFieldLabel.Text = 'Dispersion';

% Create DispersionEditField
app.DispersionEditField = uieditfield(app.UIFigure, 'numeric');
app.DispersionEditField.Position = [547 185 100 22];

% Create dBLabel
app.dBLabel = uilabel(app.UIFigure);
app.dBLabel.Position = [654 206 25 22];
app.dBLabel.Text = 'dB';

% Create Label_4
app.Label_4 = uilabel(app.UIFigure);
app.Label_4.Position = [655 185 25 22];
app.Label_4.Text = '°';

% Create PolarizationEditFieldLabel
app.PolarizationEditFieldLabel = uilabel(app.UIFigure);
app.PolarizationEditFieldLabel.HorizontalAlignment = 'right';
app.PolarizationEditFieldLabel.Position = [413 476 68 22];
app.PolarizationEditFieldLabel.Text = 'Polarization';

% Create PolarizationEditField
app.PolarizationEditField = uieditfield(app.UIFigure, 'numeric');
app.PolarizationEditField.Position = [496 476 100 22];

% Create Label_2
app.Label_2 = uilabel(app.UIFigure);
app.Label_2.Position = [602 476 25 22];
app.Label_2.Text = '°';

% Create IncidenceAngleEditFieldLabel
app.IncidenceAngleEditFieldLabel = uilabel(app.UIFigure);
app.IncidenceAngleEditFieldLabel.HorizontalAlignment = 'right';
app.IncidenceAngleEditFieldLabel.Position = [441 164 91 22];
app.IncidenceAngleEditFieldLabel.Text = 'Incidence Angle';

% Create IncidenceAngleEditField
app.IncidenceAngleEditField = uieditfield(app.UIFigure, 'numeric');
app.IncidenceAngleEditField.Position = [547 164 100 22];

% Create Label_5
app.Label_5 = uilabel(app.UIFigure);
app.Label_5.Position = [655 164 25 22];
app.Label_5.Text = '°';

% Create ResultsLabel
app.ResultsLabel = uilabel(app.UIFigure);
app.ResultsLabel.FontWeight = 'bold';
app.ResultsLabel.Position = [470 227 49 22];
app.ResultsLabel.Text = 'Results';
% Create StartEditFieldLabel
app.StartEditFieldLabel = uilabel(app.UIFigure);
app.StartEditFieldLabel.HorizontalAlignment = 'right';
app.StartEditFieldLabel.Position = [412 410 31 22];
app.StartEditFieldLabel.Text = 'Start';

% Create StartEditField
app.StartEditField = uieditfield(app.UIFigure, 'numeric');
app.StartEditField.Position = [458 410 100 22];

% Create StepEditFieldLabel
app.StepEditFieldLabel = uilabel(app.UIFigure);
app.StepEditFieldLabel.HorizontalAlignment = 'right';
app.StepEditFieldLabel.Position = [413 389 30 22];
app.StepEditFieldLabel.Text = 'Step';

% Create StepEditField
app.StepEditField = uieditfield(app.UIFigure, 'numeric');
app.StepEditField.Position = [458 389 100 22];

% Create StopEditFieldLabel
app.StopEditFieldLabel = uilabel(app.UIFigure);
app.StopEditFieldLabel.HorizontalAlignment = 'right';
app.StopEditFieldLabel.Position = [413 368 30 22];
app.StopEditFieldLabel.Text = 'Stop';

% Create StopEditField
app.StopEditField = uieditfield(app.UIFigure, 'numeric');
app.StopEditField.Position = [458 368 100 22];

% Create FrequencySweepCheckBox
app.FrequencySweepCheckBox = uicheckbox(app.UIFigure);
app.FrequencySweepCheckBox.Text = 'Frequency Sweep';
app.FrequencySweepCheckBox.Position = [386 439 119 22];

% Create GHzLabel_2
app.GHzLabel_2 = uilabel(app.UIFigure);
app.GHzLabel_2.Position = [562 410 29 22];
app.GHzLabel_2.Text = 'GHz';
% Create GHzLabel_3
app.GHzLabel_3 = uilabel(app.UIFigure);
app.GHzLabel_3.Position = [562 389 29 22];
app.GHzLabel_3.Text = 'GHz';

% Create GHzLabel_4
app.GHzLabel_4 = uilabel(app.UIFigure);
app.GHzLabel_4.Position = [563 368 29 22];
app.GHzLabel_4.Text = 'GHz';

% Create NotesTextAreaLabel
app.NotesTextAreaLabel = uilabel(app.UIFigure);
app.NotesTextAreaLabel.HorizontalAlignment = 'right';
app.NotesTextAreaLabel.Position = [33 114 37 22];
app.NotesTextAreaLabel.Text = 'Notes';

% Create NotesTextArea
app.NotesTextArea = uitextarea(app.UIFigure);
app.NotesTextArea.Position = [85 78 269 60];

% Create kmLabel_2
app.kmLabel_2 = uilabel(app.UIFigure);
app.kmLabel_2.Position = [564 276 25 22];
app.kmLabel_2.Text = 'km';

% Create TargetAltitudeEditFieldLabel
app.TargetAltitudeEditFieldLabel = uilabel(app.UIFigure);
app.TargetAltitudeEditFieldLabel.HorizontalAlignment = 'right';
app.TargetAltitudeEditFieldLabel.Position = [361 276 82 22];
app.TargetAltitudeEditFieldLabel.Text = 'Target Altitude';

% Create TargetAltitudeEditField
app.TargetAltitudeEditField = uieditfield(app.UIFigure, 'numeric');
app.TargetAltitudeEditField.Position = [458 276 100 22];

% Create SaveFileCheckBox
app.SaveFileCheckBox = uicheckbox(app.UIFigure);
app.SaveFileCheckBox.Text = 'Save File';
app.SaveFileCheckBox.Position = [616 227 72 22];

% Show the figure after all components are created
app.UIFigure.Visible = 'on';
% App creation and deletion methods (Access = public)

% Construct app
function app = atmos_app_lite

  % Create UIFigure and components
  createComponents(app)

  % Register the app with App Designer
  registerApp(app, app.UIFigure)

  if nargout == 0
    clear app
  end

% Code that executes before app deletion
function delete(app)

  % Delete UIFigure when app is deleted
  delete(app.UIFigure)

end
end
end

function loss_calculator_lite(uihandle)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Based upon ITU recommendations and Mathcad code by H. Bruce Wallace %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Create output file, if desired
saveFile = uihandle.SaveFileCheckBox.Value;
if saveFile
    [outfile,outpath] = uiputfile('', 'Create an output file');
    dotI = strfind(outfile, '.');
    if ~isempty(dotI)
        outfile(:,dotI(end):end) = [];
    end
    outfile = [outfile '.xlsx'];
end

%Input values from GUI
interval = uihandle.IntervalEditField.Value;
angle_down = uihandle.GrazingAngleEditField.Value;
altitude = uihandle.AltitudeEditField_2.Value.*1000;
polar = uihandle.PolarizationEditField.Value;
target_height = uihandle.TargetAltitudeEditField.Value.*1000;
Tamb = uihandle.GroundTemperatureEditField.Value;
PPP = uihandle.GroundPressureEditField.Value;
RH = uihandle.RelativeHumidityEditField.Value;
Rn_Rt = uihandle.RateEditField.Value;
w_conc_fog = uihandle.WaterConcentrationEditField_2.Value;
Fog_Top = uihandle.HeightEditField_2.Value;
w_conc_cloud = uihandle.WaterConcentrationEditField.Value;
Cloud_Top = uihandle.AltitudeEditField.Value+uihandle.HeightEditField.Value;
Cloud_Bot = uihandle.AltitudeEditField.Value;
Freq = uihandle.FrequencyEditField.Value.*1e9;
if uihandle.FrequencySweepCheckBox.Value
    freqStart = uihandle.StartEditField.Value.*1e9;
    freqStep = uihandle.StepEditField.Value.*1e9;
    freqStop = uihandle.StopEditField.Value.*1e9;
    FreqVec = [freqStart:freqStep:freqStop];
else
    FreqVec = Freq;
end
FreqVec = FreqVec';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Calculate ground water vapor density based on relative humidity %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TambC = Tamb-273;

%ITU-R P.453-14 (2019)

if TambC>0
    consta = 6.1121;
    constb = 18.678;
    constc = 257.14;
    constd = 234.5;

    EF = 1+1e-4*(7.2+PPP*(.032+5.9e-6*TambC^2));
else
    consta = 6.1115;
    constb = 23.036;
    constc = 279.82;
    constd = 333.7;

    EF = 1+1e-4*(2.2+PPP*(.0383+6.4e-6*TambC^2));
end

es = EF*consta*exp((constb-TambC/constd)*TambC/(TambC+constc));

el = RH/100*es; %water vapor partial pressure
v1 = (el*216.7)/Tamb; %water vapor concentration

%Set up array of heights and corresponding arrays for temperatures and pressures
jamax = ceil(altitude./interval);
ja = 0:jamax;

%Assume negligible loss above 100km
%ITU reference atmosphere recommendations do not give conditions above this altitude

if altitude>86000
    Tatm = zeros(1,86000/interval);
    Patm = zeros(1,86000/interval);
    WVDensity1 = zeros(1,86000/interval);
else
    Tatm = zeros(1,jamax+1);
    Patm = zeros(1,jamax+1);
    WVDensity1 = zeros(1,jamax+1);
end

%Populate atmospheric conditions based on ground conditions
for ii=1:size(Tatm,2)
    Tatm(ii) = Temp(ja(ii).*interval,Tamb);
    Patm(ii) = Pres(ja(ii).*interval,PPP,Tamb);
    WVDensity1(ii) = Denswv4(ja(ii).*interval,v1,PPP,Tamb);
end

%ITU-R P.676-12 (2019)
%<1000 GHz

%Populate gas loss and dispersion
Loss_oxy = Atten_oxy(FreqVec,Tatm,Patm,WVDensity1);
Loss_wv = Atten_wv(FreqVec,Tatm,Patm,WVDensity1);
Dispersion = Disp_oxy(FreqVec,Tatm,Patm,WVDensity1)+Disp_wv(FreqVec,Tatm,Patm,WVDensity1);

if (size(Loss_oxy,2)<jamax)
    Loss_oxy = [Loss_oxy zeros(size(Loss_oxy,1),(jamax+1-size(Loss_oxy,2)))];
end

if (size(Loss_wv,2)<jamax)
    Loss_wv = [Loss_wv zeros(size(Loss_wv,1),(jamax+1-size(Loss_wv,2)))];
end

Loss_clear = Loss_oxy+Loss_wv;

if (size(Dispersion,2)<jamax)
    Dispersion = [Dispersion zeros(size(Dispersion,1),(jamax+1-size(Dispersion,2)))];
end

%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calculate refractive indices based on gases

%ITU-R P.676-12 (2019)
Pwv = WVDensity1.*Tatm./216.7;
Pd = Patm-Pwv;

%%%%%%%%%%%%%%%%%%%%%%%%

%ITU-R P.453-14 (2019)
N = 77.6*Pd./Tatm+72.*Pwv./Tatm+3.75e5.*Pwv./Tatm.^2;
n = N.*1e-6+1;

%%%%%%%%%%%%%%%%%%%%%%%%

if (size(n,2)<jamax)
    n = [n ones(1,(jamax+1-size(n,2)))];
end

%%%%%%%%%%%%%%%%%%%%%%%
%ITU-R P.676-12 (2019)
%<1000 GHz

%Calculate ascending apparent elevation angle based on descending angle
Re = 6371000;  %average Earth radius (m)
surf_height = Re;  %add local elevation for more accuracy
n_surf = n(1);
angle_up = acosd((ja(1,end).*interval+surf_height).*n(1,end)./(surf_height*n_surf)*cosd(angle_dow n));

cos_phi = surf_height.*n_surf.*/((ja.*interval+surf_height).*n).*cosd(angle_up);
sin_phi = sqrt(1-cos_phi.^2);
phi = acosd(cos_phi);

%%%%%%%%%%%%%%%%%%%%%%%
%Set up arrays for loss in weather
rrr = round(Cloud_Bot./interval);
fff = round(Fog_Top./interval);
cccb = round(Cloud_Bot./interval);
ccct = round(Cloud_Top./interval);
Loss_rain = zeros(size(FreqVec,1),jamax+1);
Loss_fog = zeros(size(FreqVec,1),jamax+1);
Loss_cloud = zeros(size(FreqVec,1),jamax+1);
Appendix 1

```matlab
Loss_vs_ht_all = zeros(size(FreqVec,1),jamax+1);
Disp_vs_ht = zeros(size(FreqVec,1),jamax+1);
picture = ones(size(FreqVec,1),size(ja,2),3); %stores color data for visualization

%Populate specific attenuation due to weather
for ii=1:size(ja,2)
    if ((uihandle.RainCheckBox.Value)&&(ii<=rrr))
        Loss_rain(:,ii) = Atten_rn(FreqVec,Rn_Rt,polar,phi(1,ii));
        picture(:,ii,:) = picture(:,ii,:) - permute([.2 .2 .1],[1 3 2]);
    end
    if ((uihandle.FogCheckBox.Value)&&(ii<=fff))
        Loss_fog(:,ii) = Atten_Fog(FreqVec,Tatm(1,ii),w_conc_fog);
        picture(:,ii,:) = picture(:,ii,:) - permute([.07 .1 .1],[1 3 2]);
    end
    if ((uihandle.CloudsCheckBox.Value)&&(ii>cccb)&&(ii<=ccct))
        Loss_cloud(:,ii) = Atten_Fog(FreqVec,Tatm(1,ii),w_conc_cloud);
        picture(:,ii,:) = picture(:,ii,:) - permute([.25 .25 .25],[1 3 2]);
    end
end
Loss_all = Loss_clear+Loss_rain+Loss_cloud+Loss_fog;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ITU-R P.676-12 (2019)
%<1000 GHz
%Calculate attenuation and dispersion using numerical integration

target_index = floor(target_height./interval)+1;
for ii=target_index:size(ja,2)
    if (ii>target_index)
        Loss_vs_ht_all(:,ii) = Loss_vs_ht_all(:,ii-1)+interval./1000.*Loss_all(:,ii-1)./sin_phi(ii-1);
        Disp_vs_ht(:,ii) = Disp_vs_ht(:,ii-1)+interval./1000.*Dispersion(:,ii-1)./sin_phi(ii-1);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Write data blocks to Excel file
if saveFile
    writecell({'Specific Attenuation (dB/km)'},[outpath outfile],'Sheet',1,'Range','A1');
    writecell({'Altitude (m)/Frequency (GHz)'},[outpath outfile],'Sheet',1,'Range','A2');
    writematrix(FreqVec'./1e9,[outpath outfile],'Sheet',1,'Range','B2');
    writematrix(ja'.*interval,[outpath outfile],'Sheet',1,'Range','A3');
    writematrix(Loss_all',[outpath outfile],'Sheet',1,'Range','B3');
    writematrix(FreqVec'./1e9,[outpath outfile],'Sheet',2,'Range','B2');
    writecell({'Total Attenuation (dB)'},[outpath outfile],'Sheet',2,'Range','A1');
    writecell({'Altitude (m)/Frequency (GHz)'},[outpath outfile],'Sheet',2,'Range','A2');
    writematrix(FreqVec'./1e9,[outpath outfile],'Sheet',2,'Range','B2');
end
```
writematrix(ja' .* interval, [outpath outfile], 'Sheet', 2, 'Range', 'A3');
writematrix(Loss_vs_ht_all', [outpath outfile], 'Sheet', 2, 'Range', 'B3');

writematrix(Loss_vs_ht_all', [outpath outfile], 'Sheet', 2, 'Range', 'B3');
writematrix(ja' .* interval, [outpath outfile], 'Sheet', 3, 'Range', 'A3');

writecell({'Specific Dispersion (deg/km)'}, [outpath outfile], 'Sheet', 3, 'Range', 'A1');
writecell({'Altitude (m)/Frequency (GHz)'}, [outpath outfile], 'Sheet', 3, 'Range', 'A2');
writematrix(FreqVec'./1e9, [outpath outfile], 'Sheet', 3, 'Range', 'B2');
writematrix(ja' .* interval, [outpath outfile], 'Sheet', 3, 'Range', 'A3');

writematrix(Dispersion', [outpath outfile], 'Sheet', 3, 'Range', 'B3');
writecell({'Total Dispersion (deg)'}, [outpath outfile], 'Sheet', 4, 'Range', 'A1');
writecell({'Altitude (m)/Frequency (GHz)'}, [outpath outfile], 'Sheet', 4, 'Range', 'A2');
writematrix(FreqVec'./1e9, [outpath outfile], 'Sheet', 4, 'Range', 'B2');
writematrix(ja' .* interval, [outpath outfile], 'Sheet', 4, 'Range', 'A3');
writecell({'Total Dispersion (deg)'}, [outpath outfile], 'Sheet', 4, 'Range', 'B3');

end

% Create note list
note = '';
if ((altitude>86000)&&(Freq<12e9))
  note = ['Ionospheric effects not calculated' newline];
end

if ((uihandle.CloudsCheckBox.Value)||(uihandle.FogCheckBox.Value)||(uihandle.RainCheckBox .Value))
  note = [note 'Bending/dispersion incomplete' newline];
end

% Find specified frequency in vector
freqIndex = find(FreqVec==Freq);
if size(freqIndex,1)==0

  % Output loss and dispersion to GUI
  uihandle.AttenuationEditField.Value = Loss_vs_ht_all(freqIndex,end);
  uihandle.IncidenceAngleEditField.Value = angle_up;
  uihandle.DispersionEditField.Value = Disp_vs_ht(freqIndex,end);

  fig = figure;
  ax = axes('YAxisLocation', 'right', 'YColor', [0 .447 .741], 'XScale', 'log');
  line(ja' .* interval./1000,loss_all(freqIndex,:));
  ylabel('Specific Attenuation (dB/km)');
  xlabel('Altitude (km)');
  pbaspect([1 1 1]);
  hold on;
  otherax = axes('Position', ax.Position, 'YAxisLocation', 'left', 'YColor', [1 0 0], 'XScale', 'log', 'Color', 'none', 'XColor', 'none');
  line(ja' .* interval./1000,Dispersion(freqIndex,:), 'Color', [1 0 0]);
  ylabel('Specific Dispersion (char(176) /km)');
  pbaspect([1 1 1]);
  hold off;
  fig.Position = fig.Position+fig.Position.*[0 0 0 .3]+[0 -150 0 0];

  % Get MATLAB automatic x- and y-limits of figure
fig = figure;
loglog(ja.*interval./1000,Loss_vs_ht_all(freqIndex,:));
axs = gca;
XLim = axs.XLim;
YLim = axs.YLim;
rectangle('Position',[ja(1).*interval./1000 YLim(1) interval./1000 (YLim(2)-
YLim(1)).*1.25],'LineStyle','none','FaceColor',picture(1,1,:));
XLim = axs.XLim;
YLim = axs.YLim;
clear fig;

%Reset figure with same settings
axs = gca;
axs.XScale = 'log';
axs.YScale = 'log';
axs.Layer = 'top';

%Draw weather in background
for ii=1:jamax
    rectangle('Position',[ja(ii).*interval./1000 YLim(1) interval./1000 (YLim(2)-
YLim(1))],'LineStyle','none','FaceColor',picture(1,ii,:));
end
hold on;

%Plot data
loglog(ja.*interval./1000,Loss_vs_ht_all(freqIndex,:));
ylabel('Attenuation to Ground (dB)');
xlabel('Altitude (km)');
hold off;

else

%Specified frequency is not in sweep, so was not calculated
uihandle.AttenuationEditField.Value = 0;
uihandle.IncidenceAngleEditField.Value = angle_up;
uihandle.DispersionEditField.Value = 0;

note = [note 'Specified frequency not in sweep- file output only' newline];
end

%Output notes to GUI
uihandle.NotesTextArea(FontColor = [1 0 0];
uihandle.NotesTextArea.Value = note;
end
Temp.m

function temp = Temp(hZ,T_gZ)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Based upon ITU recommendations and Mathcad code by H. Bruce Wallace %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ITU-R P.835-6 (2017)
%convert to km
hZ = hZ./1000;

%Use correct altitude regime

%convert to h' from h
hZprime = 6356.766.*hZ./(6356.766+hZ);

H = [0;11;20;32;47;51;71;84.852]; %altitude division start heights
L = [-6.5;0;1;2.8;0;-2.8;-2.0;0]; %change in temp. for each division

%other equations for above 84.852 km'
T = -1.*ones(1,8);
T(1) = T_gZ;

%Find temperature at each division boundary
for jj=1:7
    T(jj+1) = T(jj)+L(jj).*(H(jj+1)-H(jj));
end

if (hZ<86)
    z = T(1);
    ii = 1;
    %Find closest boundary and calculate temperature at height
    while (hZprime>H(ii))
        z = T(ii)+L(ii).*(hZprime-H(ii));
        ii = ii+1;
    end
    temp = z;
else
    if (hZ<91)
        temp = T(end);
    else
        temp = 76.3232+T(end)-76.3232*sqrt(1-(hZ-91)/19.9429)^2);
    end
end
Pres.m

function pressure = Pres(hZ,P_gZ,T_gZ)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Based upon ITU recommendations and Mathcad code by H. Bruce Wallace %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ITU-R P.835-6 (2017)

hZ = hZ./1000;

%convert to h' from h

hZ = 6356.766.*hZ./(6356.766+hZ);

H = [0;11;20;32;47;51;71;84.852];  %altitude division start heights
L = [-6.5;0;1.2.8;0;-2.8;-2.0;0];  %change in pressure for each division

P = -1.*ones(1,8);
T = -1.*ones(1,8);

P(1) = P_gZ;
T(1) = T_gZ;

w = 34.1632;

%Find boundary temperatures first

Hnotprime = 6356.766.*H./(6356.766-H);

for jj=1:7
    T(jj) = Temp(Hnotprime(jj).*1000,T_gZ);
end

%Find boundary pressures

for jj=1:7
    if (L(jj)==0)  %floating point comparison
        P(jj+1) = P(jj).*exp(-w.*(H(jj+1)-H(jj))./T(jj));
    else
        P(jj+1) = P(jj).*(T(jj)./T(jj+1)).^(w./L(jj));
    end
end

ii = 1;

z = P(1);

%Find closest boundary and calculate pressure at height

while (hZ>H(ii))
    if (L(ii)==0)  %floating point comparison
        z = exp(-w.*(hZ-H(ii))./T(ii));
    else
        z = (T(ii)./(T(ii)+L(ii).*((hZ-H(ii)))).^(w./L(ii));
    end

    z = z.*P(ii);
    ii = ii+1;
end
if (ii>size(H,1))
    break;
end

pressure = z;
end
% ITU-R P.835-6 (2017)

const = 216.7;

% Find preliminary vapor density

vh = v_gZ.*exp(-hZ./2000);

% Find mixing ratio

y = (vh.*TZ)./(const);

w = y./PZ;

% Check if mixing ratio should be constant change density if necessary

if (w<2e-6)
    x = const./TZ.*PZ.*2e-6;
else
    x = vh;
end

vapor = x;
end
function o2atten = Atten_oxy(FZ,TZ,PZ,VZ)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Based upon ITU recommendations and Mathcad code by H. Bruce Wallace %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ITU-R P.676-12 (2019)
%<1000 GHz
load('oxZ.mat');

oxZ = permute(oxZ,[3 2 1]);
theta = 300./TZ;
e1 = VZ.*TZ./216.7;
% vapor partial pressure %add direct e1?
p1 = PZ-e1;
% dry air partial pressure
FE = FZ./1e9;
d = 5.6e-4.*(p1+e1).*theta.^0.8;
% Debye spectrum width parameter
DFreq = 6.14e-5./(d.*(1+(FE./d).^2));
% corresponds to non-resonant Debye spectrum
InterN2D = 1.4e-12.*p1.*theta.^1.5./(1+1.9e-5.*FE.^1.5);
% corresponds to pressure-induced nitrogen attenuation
N2D = FE.*p1.*theta.^2.*(DFreq+InterN2D);
% dry air continuum
S = oxZ(1,2,:).*p1.*theta.^0.3.*exp(oxZ(1,3,:).*(1-theta)).*1e-7;
% oxygen line strength
G = oxZ(1,4,:).*p1.*theta.^0.8-oxZ(1,5,:)]+1.1.*el.*theta).*1e-4;
% oxygen line width
G = sqrt(G.^2+2.25e-6);
% correction for Zeeman splitting
delta = (oxZ(1,6,:)+oxZ(1,7,:).*theta).*p1.*theta.^0.8.*1e-4;
% correction factor for interference in oxygen lines
F2oxy = FE./oxZ(1,1,:).*((G-(oxZ(1,1,:)-FE).*delta)./(oxZ(1,1,:)-1)+G.^2)+(G-oxZ(1,1,:)+FE).*delta)/((oxZ(1,1,:)+FE).^2+G.^2));
% line-shape factor
o2atten = .182.*FE.*(sum(S.*F2oxy,3)+N2D);
% attenuation
end
The document contains a MATLAB function named `h2oatten` for calculating atmospheric attenuation due to water vapor. The function is based on ITU recommendations and Mathcad code by H. Bruce Wallace. The code snippet is as follows:

```matlab
function h2oatten = Atten_wv(FZ, TZ, PZ, VZ)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Based upon ITU recommendations and Mathcad code by H. Bruce Wallace
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ITU-R P.676-12 (2019)
%<1000 GHz

load('wvZ.mat');

wvZ = permute(wvZ,[3 2 1]);

theta = 300./TZ;

el = VZ.*TZ./216.7;  % vapor partial pressure % add direct el?

FE = FZ./1e9;

pl = PZ - el;  % dry air partial pressure

G = wvZ(1,4,:).*(pl.*theta.^wvZ(1,5,:)+wvZ(1,6,:).*el.*theta.^wvZ(1,7,:)).*1e-4;

S = wvZ(1,2,:).*el.*theta.^3.5.*exp(wvZ(1,3,:).*(1-theta)).*1;  % vapor line strength

G = wvZ(1,4,:).*(pl.*theta.^wvZ(1,5,:)+wvZ(1,6,:).*el.*theta.^wvZ(1,7,:)).*1e-4;

G = .535.*G+sqrt(.217.*G.^2+2.1316e-12.*wvZ(1,1,:).^2./theta);  % correction for Doppler broadening

d = 0;  % N/A for vapor

F2wv = FE./wvZ(1,1,:).*((G-(wvZ(1,1,:)-FE).*d)./((wvZ(1,1,:)-FE).^2+G.^2)+(G-(wvZ(1,1,:)+FE).*d)./((wvZ(1,1,:)+FE).^2+G.^2));  % line-shape factor

h2oatten = .182.*FE.*sum(S.*F2wv,3);  % attenuation

end
```
function o2disp = Disp_oxy(FZ,TZ,PZ,VZ)

% ITU-R P.676-12 (2019)
%<1000 GHz
load('oxZ.mat');

oxZ = permute(oxZ,[3 2 1]);
const = 216.7;
theta = 300./TZ;
e1 = VZ.*TZ./const;
pl = PZ-e1;
FE = FZ./1e9;
d = 5.6e-4.*(pl+e1).*theta.^8;
N1D = (-6.14e-5.*pl.*theta.^2.*FE.^2)/(FE.^2+d.^2);

S = oxZ(1,2,:).*pl.*theta.^3.*exp(oxZ(1,3,:).*(1-theta)).*1e-7;
G = oxZ(1,4,:).*pl.*theta.^8.*G.^2+2.25e-6;

G = sqrt(G.^2+2.25e-6);

%Zeeman splitting
delta = (oxZ(1,6,:)+oxZ(1,7,:).*theta).*pl.*theta.^8.*1e-4;
F1oxy = FE./oxZ(1,1,:).*(oxZ(1,1,:)-FE+delta.*G)./((oxZ(1,1,:)-FE).^2+G.^2)+...
(oxZ(1,1,:)+FE-delta.*G)./((oxZ(1,1,:)+FE).^2+G.^2));

o2disp = -1.2008.*FE.*(sum(S.*F1oxy,3)+N1D);

end
Disp_wv.m

function wvdisp = Disp_wv(FZ,TZ,PZ,VZ)

%ITU-R P.676-12 (2019)
%<1000 GHz
load('wvZ.mat');
wvZ = permute(wvZ,[3 2 1]);
const = 216.7;
theta = 300./TZ;
e1 = VZ.*TZ./const;
p1 = PZ-e1;
FE = FZ./1e9;
S = wvZ(1,2,:).*e1.*theta.^3.5.*exp(wvZ(1,3,:).*(1-theta)).*1;
G = wvZ(1,4,:).*(p1.*theta.*wvZ(1,5,:)+wvZ(1,6,:).*e1.*theta.*wvZ(1,7,:)).*1e-4;
G = .535.*G+sqrt(.217.*G.^2+2.1316e-12.*wvZ(1,1,:).^2./theta);
delta = 0;
F1wv = FE./wvZ(1,1,:).*((wvZ(1,1,:)-FE+delta.*G)./((wvZ(1,1,:)-FE).^2+G.^2)+...
(wvZ(1,1,:)+FE-delta.*G)./((wvZ(1,1,:)+FE).^2+G.^2));
wvdisp = -1.2008.*FE.*sum(S.*F1wv,3);
end
function rainatten = Atten_rn(FZ,Rn_RtZ,PolZ,phi)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Based upon ITU recommendations and Mathcad code by H. Bruce Wallace %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ITU-R P.838-3 (2005)
%1-1000 GHz
%coefficient tables

ak = [-5.3398 -3.8059; -.35351 -3.4496; -.23789 -.3990; .94158 .50167];
bk = [-.10009 .5693; 1.26970 -.2291; .86036 .7304; .64552 1.073];
ck = [.13092 .81061; .45400 .5105; .15354 .1289; .16817 .2719];
aalpha = [-.14318 -.0777; .9591 .5672; .32177 -.2023; -.5376 .4829; 16.1721];
balpha = [1.8244 2.3384; .7786 .9555; 1.1452 -.9623; .791459];
calpha = [-.5518 -.7628; .1982 .5403; .1316 .2680; .4782 .1162; .34390 .1164];
m2k = [-.18961; -.1639];
c2k = [.71147; .6329];
m2alpha = [.67849; -.05379];
c2alpha = [-1.953; .8343];
F = FZ./1e9;
for ii=1:size(F,1)
L10krH(ii,1) = sum(ak(:,1).*exp(-((log10(F(ii,1))-bk(:,1))./ck(:,1)).^2))+m2k(1).*log10(F(ii,1))+c2k(1);
L10krV(ii,1) = sum(ak(:,2).*exp(-((log10(F(ii,1))-bk(:,2))./ck(:,2)).^2))+m2k(2).*log10(F(ii,1))+c2k(2);
krH(ii,1) = 10.^L10krH(ii,1);
krV(ii,1) = 10.^L10krV(ii,1);
alpharH(ii,1) = sum(aalpha(:,1).*exp(-((log10(F(ii,1))-balpha(:,1))./calpha(:,1)).^2))+m2alpha(1).*log10(F(ii,1))+c2alpha(1);
alpharV(ii,1) = sum(aalpha(:,2).*exp(-((log10(F(ii,1))-balpha(:,2))./calpha(:,2)).^2))+m2alpha(2).*log10(F(ii,1))+c2alpha(2);
end
kr = (krH+krV+(krH-krV).*cosd(phi)^2.*cosd(2*PolZ))./2;
alphar = (krH.*alpharH+krV.*alpharV+(krH.*alpharH-krV.*alpharV).*cosd(phi)^2.*cosd(2*PolZ))./(2*kr);

rainatten = kr.*Rn_RtZ.^alphar; %attenuation
end
Appendix 1

Atten_Fog.m

function fogatten = Atten_Fog(FZ,TZ,waterZ)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Based upon ITU recommendations and Mathcad code by H. Bruce Wallace %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ITU-R P.840-8 (2019)
%<200 GHz

theta = 300./TZ;
F = FZ./1e9;
e0 = 77.66+103.3.*(theta-1);
e1 = .0671.*e0;
e2 = 3.52;
fp = 20.20-146.*(theta-1)+316.*(theta-1).^2;    %principle relaxation frequency
fs = 39.8.*fp;                                    %secondary relaxation frequency
E1 = (e0-e1)./(1+(F./fp).^2)+(e1-e2)./(1+(F./fs).^2)+e2;   %complex dielectric permittivity of water (real?)
E2 = ((e0-e1).*(F./fp))./(1+(F./fp).^2)+((e1-e2).*(F./fs))./(1+(F./fs).^2);    %complex dielectric permittivity of water (imag?)
eta = (E1+2)./E2;
K1 = (.819.*F)./(E2.*(1+eta.^2));    %specific attenuation coefficient
fogatten = K1.*waterZ;
end
APPENDIX 2

An alternative version of this software has been created to bypass the MATLAB application code and work directly from a script. Fields within the user interface are replaced with variables set using literal constants. Rather than launching the application, the user will instead open the below script and make desired changes before running. Dependent functions remain the same and are documented in Appendix 1.

loss_calculator_noGUI.m

% Toggle for saving to Excel file
saveFile = 0;

% Create output file
if saveFile
    [outfile,outpath] = uiputfile('', 'Create an output file');
    dotI = strfind(outfile, '.');
    if ~isempty(dotI)
        outfile(:,dotI(end):end) = [];
    end
    outfile = [outfile '.xlsx'];
end

% Input values
interval = 100; % m
angle_down = -90; % degrees
altitude = 5000; % m
polar = 0; % degrees
target_height = 0; % m
Tamb = 288; % K
PPP = 1013; % mb
RH = 50; % percent

% Determines if weather is included in calculation
cloudsOn = 0;
rainOn = 0;
fogOn = 0;
Rn_Rt = 4; % mm/hr
w_conc_fog = .5; % g/m^3
Fog_Top = 200; % m
w_conc_cloud = .33; % g/m^3
cloudAlt = 500; % m
cloudHeight = 1000; % m
Cloud_Top = cloudAlt+cloudHeight;
Cloud_Bot = cloudAlt;

Freq = 50.*1e9; % Hz % Frequency selected for plots
FreqVec = [10; 50; 100].*1e9; % Hz

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calculate ground water vapor density based on relative humidity

TambC = Tamb-273;

%ITU-R P.453-14 (2019)

if TambC>0
  consta = 6.1121;
  constb = 18.678;
  constc = 257.14;
  constd = 234.5;
  
  EF = 1+1e-4*(7.2+PPP*(.032+5.9e-6*TambC^2));
else
  consta = 6.1115;
  constb = 23.036;
  constc = 279.82;
  constd = 333.7;
  
  EF = 1+1e-4*(2.2+PPP*(.0383+6.4e-6*TambC^2));
end

es = EF*consta*exp((constb-TambC/constd)*TambC/(TambC+constc));

el = RH/100*es; %water vapor partial pressure
v1 = el*216.7/Tamb; %water vapor concentration

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Set up array of heights and corresponding arrays for temperatures and %pressures

jamax = ceil(altitude./interval);
ja = 0:jamax;

%Assume negligible loss above 100km %ITU reference atmosphere recommendations do not give conditions above this altitude

if altitude>86000
  Tatm = zeros(1,86000/interval);
  Patm = zeros(1,86000/interval);
  WVDensity1 = zeros(1,86000/interval);
else
  Tatm = zeros(1,jamax+1);
  Patm = zeros(1,jamax+1);
  WVDensity1 = zeros(1,jamax+1);
end

%Populate atmospheric conditions based on ground conditions
for ii=1:size(Tatm,2)
  Tatm(ii) = Temp(ja(ii).*interval,Tamb);
  Patm(ii) = Pres(ja(ii).*interval,PPP,Tamb);
  WVDensity1(ii) = Denswv4(ja(ii).*interval,v1,PPP,Tamb);
end

%ITU-R P.676-12 (2019)
%<1000 GHz

%Populate gas loss and dispersion
Loss_oxy = Atten_oxy(FreqVec,Tatm,Patm,WVDensity1);
Loss\_wv = \text{Atten\_wv}(\text{FreqVec}, \text{Tatm}, \text{Patm}, \text{WVDensity1});
\text{Dispersion} =
\text{Disp\_oxy}(\text{FreqVec}, \text{Tatm}, \text{Patm}, \text{WVDensity1}) + \text{Disp\_wv}(\text{FreqVec}, \text{Tatm}, \text{Patm}, \text{WVDensity1});

\text{if} \ (\text{size(Loss\_oxy, 2)} < \text{jamax})
\quad \text{Loss\_oxy} = \left[\text{Loss\_oxy} \ \text{zeros}(\text{size(Loss\_oxy, 1)}, (\text{jamax} + 1 - \text{size(Loss\_oxy, 2))))\right];
\text{end}

\text{if} \ (\text{size(Loss\_wv, 2)} < \text{jamax})
\quad \text{Loss\_wv} = \left[\text{Loss\_wv} \ \text{zeros}(\text{size(Loss\_wv, 1)}, (\text{jamax} + 1 - \text{size(Loss\_wv, 2))))\right];
\text{end}

\text{Loss\_clear} = \text{Loss\_oxy} + \text{Loss\_wv};
\text{if} \ (\text{size(Dispersion, 2)} < \text{jamax})
\quad \text{Dispersion} = \left[\text{Dispersion} \ \text{zeros}(\text{size(Dispersion, 1)}, (\text{jamax} + 1 - \text{size(Dispersion, 2))))\right];
\text{end}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calculate refractive indices based on gases

\%ITU-R P.676-12 (2019)
P\_wv = \text{WVDensity1} \cdot \text{Tatm} / 216.7;
P\_d = \text{Patm} - P\_wv;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ITU-R P.453-14 (2019)
N = 77.6 \cdot P\_d / \text{Tatm} + 72. \cdot P\_wv / \text{Tatm} + 3.75 \cdot 10^5 \cdot P\_wv / \text{Tatm}^2;
n = N \cdot 10^{-6} + 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\text{if} \ (\text{size(n, 2)} < \text{jamax})
\quad n = \left[n \ \text{ones}(1, (\text{jamax} + 1 - \text{size(n, 2))))\right];
\text{end}

%ITU-R P.676-12 (2019)
\%<1000 GHz
%Calculate ascending apparent elevation angle based on descending angle
\Re = 6371000; \quad \%average Earth radius (m)
\text{surf\_height} = \Re; \quad \%add local elevation for more accuracy
n\_surf = n(1);
\text{angle\_up} =
\text{acosd}((\text{ja}(1, \text{end}) \cdot \text{interval} + \text{surf\_height}) \cdot n(1, \text{end}) / (\text{surf\_height} \cdot n\_surf) \cdot \text{cosd(\text{angle\_down})});
\cos\_phi = \text{surf\_height} \cdot n\_surf / ((\text{ja} \cdot \text{interval} + \text{surf\_height}) \cdot n) \cdot \text{cosd(\text{angle\_up})};
\sin\_phi = \sqrt{1 - \cos\_phi^2});
\phi = \text{acosd(\cos\_phi)}^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%Set up arrays for loss in weather
\text{rrr} = \text{round(Cloud\_Bot./\text{interval})};
\text{fff} = \text{round(Fog\_Top./\text{interval})};
\text{cccb} = \text{round(Cloud\_Bot./\text{interval})};
\text{ccct} = \text{round(Cloud\_Top./\text{interval})};
\text{Loss\_rain} = \text{zeros(size(FreqVec, 1), \text{jamax} + 1)};
\text{Loss\_fog} = \text{zeros(size(FreqVec, 1), \text{jamax} + 1)};
Appendix 2

Loss_cloud = zeros(size(FreqVec,1),jamax+1);
Loss_vs_ht_all = zeros(size(FreqVec,1),jamax+1);
Disp_vs_ht = zeros(size(FreqVec,1),jamax+1);
picture = ones(size(FreqVec,1),size(ja,2),3);  %stores color data for visualization

%Populate specific attenuation due to weather
for ii=1:size(ja,2)
    if ((rainOn)&&(ii<=rrr))
        Loss_rain(:,ii) = Atten_rn(FreqVec,Rn_Rt,polar,phi(1,ii));
picture(:,ii,:) = picture(:,ii,:)-permute([.2 .2 .1],[1 3 2]);
    end
    if ((fogOn)&&(ii<=fff))
        Loss_fog(:,ii) = Atten_Fog(FreqVec,Tatm(1,ii),w_conc_fog);
picture(:,ii,:) = picture(:,ii,:)-permute([.07 .1 .1],[1 3 2]);
    end
    if ((cloudsOn)&&(ii>cccb)&&(ii<=ccct))
        Loss_cloud(:,ii) = Atten_Fog(FreqVec,Tatm(1,ii),w_conc_cloud);
picture(:,ii,:) = picture(:,ii,:)-permute([.25 .25 .25],[1 3 2]);
    end
end
Loss_all = Loss_clear+Loss_rain+Loss_cloud+Loss_fog;

%ITU-R P.676-12 (2019)
%<1000 GHz
%Calculate attenuation and dispersion using numerical integration

Loss_vs_ht_all = zeros(size(FreqVec,1),jamax+1);
Disp_vs_ht = zeros(size(FreqVec,1),jamax+1);
picture = ones(size(FreqVec,1),size(ja,2),3);  %stores color data for visualization

%Populate specific attenuation due to weather
for ii=1:size(ja,2)
    if ((rainOn)&&(ii<=rrr))
        Loss_rain(:,ii) = Atten_rn(FreqVec,Rn_Rt,polar,phi(1,ii));
picture(:,ii,:) = picture(:,ii,:)-permute([.2 .2 .1],[1 3 2]);
    end
    if ((fogOn)&&(ii<=fff))
        Loss_fog(:,ii) = Atten_Fog(FreqVec,Tatm(1,ii),w_conc_fog);
picture(:,ii,:) = picture(:,ii,:)-permute([.07 .1 .1],[1 3 2]);
    end
    if ((cloudsOn)&&(ii>cccb)&&(ii<=ccct))
        Loss_cloud(:,ii) = Atten_Fog(FreqVec,Tatm(1,ii),w_conc_cloud);
picture(:,ii,:) = picture(:,ii,:)-permute([.25 .25 .25],[1 3 2]);
    end
end
Loss_all = Loss_clear+Loss_rain+Loss_cloud+Loss_fog;

%Write data blocks to Excel file
if saveFile
    writecell({'Specific Attenuation (dB/km)'},[outpath outfile],’Sheet’,1,’Range’,’A1’);
    writecell({’Frequency (GHz)/Altitude (m)’},[outpath outfile],’Sheet’,1,’Range’,’A2’);
    writematrix(FreqVec’,/le9,[outpath outfile],’Sheet’,1,’Range’,’B2’);
    writematrix(ja’.*interval,[outpath outfile],’Sheet’,1,’Range’,’A3’);
    writematrix(Loss_all’,[outpath outfile],’Sheet’,1,’Range’,’B3’);
    writecell({’Total Attenuation (dB)’},[outpath outfile],’Sheet’,2,’Range’,’A1’);
    writecell({’Frequency (GHz)/Altitude (m)’},[outpath outfile],’Sheet’,2,’Range’,’A2’);
%Create note list

note = ''; 
if ((altitude>86000)&&(Freq<12e9))  
  note = ['Ionospheric effects not calculated' newline]; 
end 
if ((cloudsOn)||(fogOn)||(rainOn)) 
  note = [note 'Bending/dispersion incomplete' newline]; 
end 

%Find specified frequency in vector, if it exists

freqIndex = find(FreqVec==Freq); 
if size(freqIndex,1)==0 
  %Plot attenuation and dispersion for selected frequency
  fig = figure; 
  ax = axes('YAxisLocation','right','YColor',[0 .447 .741],'XScale','log'); 
  line(ja.*interval./1000,Loss_all(freqIndex,:)); 
  ylabel('Specific Attenuation (dB/km)'); 
  xlabel('Altitude (km)'); 
  pbaspect([1 1 1]); 
  hold on; 
  otherax = axes('Position',ax.Position,'YAxisLocation','left','YColor',[1 0 0],'XScale','log','Color','none','XColor','none'); 
  line(ja.*interval./1000,Dispersion(freqIndex,:),'Color',[1 0 0]); 
  ylabel(['Specific Dispersion (' char(176) '/km)']); 
  pbaspect([1 1 1]); 
  hold off; 
  fig.Position = fig.Position+fig.Position.*[0 0 0 .3]+[0 -150 0 0]; 
  %Get automatic figure size
  fig = figure; 
  loglog(ja.*interval./1000,Loss_vs_ht_all(freqIndex,:)); 
  axs = gca; 
  XLim = axs.XLim;
YLim = axs.YLim;
rectangle('Position', [ja(l).*interval./1000 YLim(l) interval./1000 (YLim(2)-
YLim(l)).*1.25], 'LineStyle', 'none', 'FaceColor', picture(l,1,:));
XLim = axs.XLim;
YLim = axs.YLim;
clear fig;

%Reset figure with same settings

axs = gca;
axs.XScale = 'log';
axs.YScale = 'log';
axs.Layer = 'top';

%Draw weather in background
for ii=1:jamax
    rectangle('Position', [ja(ii).*interval./1000 YLim(1) interval./1000 (YLim(2)-
    YLim(l))], 'LineStyle', 'none', 'FaceColor', picture(l,ii,:));
end
hold on;

%Plot data
loglog(ja.*interval./1000, Loss_vs_ht_all(freqIndex,:));
ylabel('Attenuation to Ground (dB)');
xlabel('Altitude (km)');
hold off;

else

%Specified frequency is not in sweep, so was not calculated

note = [note 'Specified frequency not in sweep- file output only' newline];
end