Naval Research Laboratory
Washington, DC 20375-5320

NRL/5307/MR--2021/1

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

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
05-02-2021

2. REPORT TYPE
NRL Memorandum Report

3. DATES COVERED *(From - To)*
Sept. 2019 – June 2020

4. TITLE AND SUBTITLE
Computational Model for Atmospheric Propagation Losses in the Radiofrequency Through Millimeter Wave Range

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
062114N

6. AUTHOR(S)
Judith V. Hutson* and Christopher T. Rodenbeck

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
6A44

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER
NRL/5307/MR--2021/1

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

10. SPONSOR / MONITOR'S ACRONYM(S)
NRL

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

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

13. SUPPLEMENTARY NOTES
*Formerly, Advanced Concepts Group, Radar Division

14. 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.

15. SUBJECT TERMS
Radar ISR Propagation modeling

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON / 19b. TELEPHONE NUMBER (include area code)
Unclassified Unlimited	Unclassified Unlimited	Unclassified Unlimited	Unclassified Unlimited	73	Christopher Rodenbeck / (202) 404-1596

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

This page intentionally left blank.

CONTENTS

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.

Manuscript approved February 5, 2021.

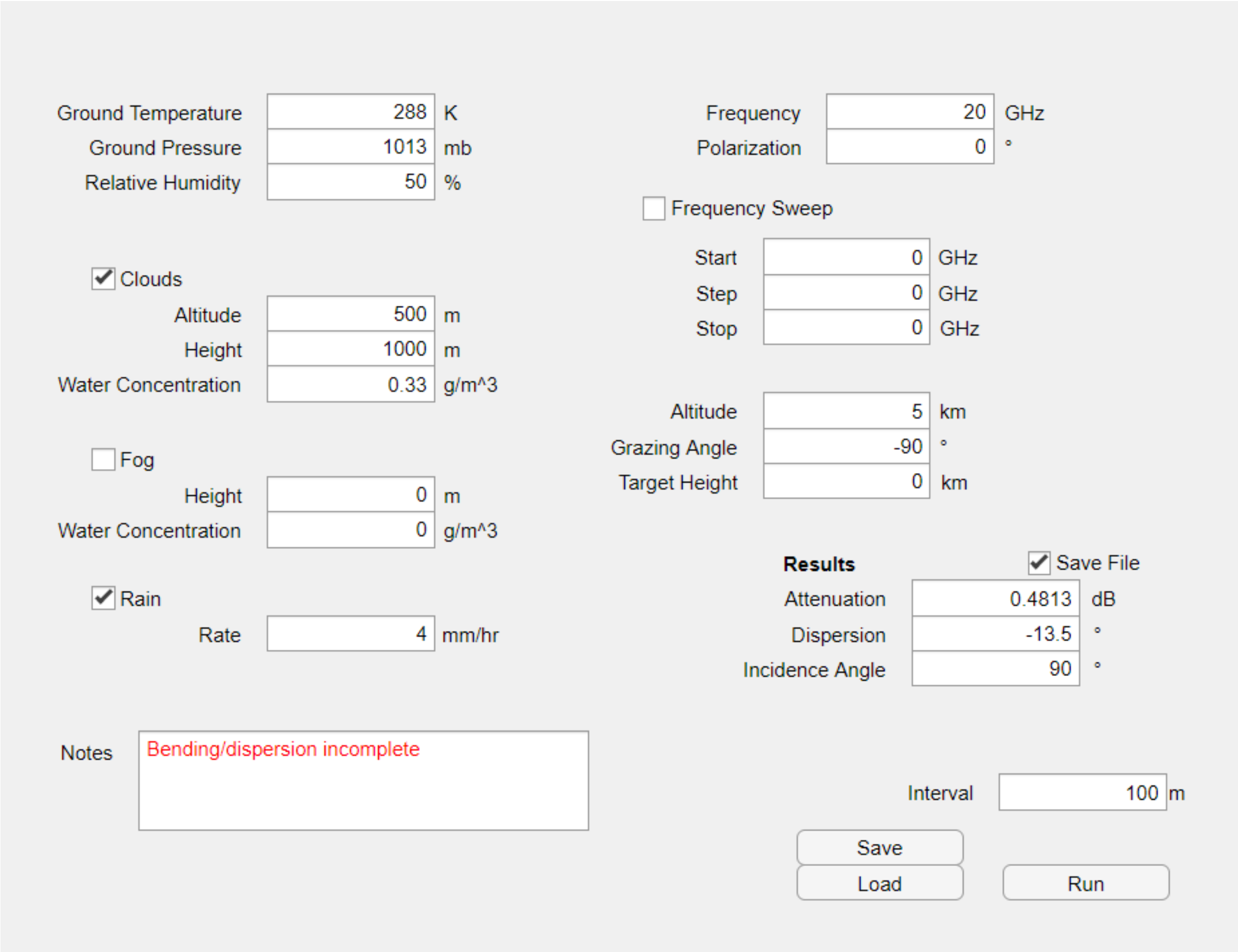

Fig. 1 – User interface with example settings

Table 1 - GUI variable description

Ground Temperature	Temperature at surface	K
Ground Pressure	Pressure at surface	mb
Relative Humidity	Relative humidity at surface	%
Clouds: Altitude	Altitude of bottom of cloud	m
Clouds: Height	Extension of cloud above bottom altitude	m
Clouds: Water Concentration	Liquid water density in cloud	g/m^3
Fog: Height	Extension of fog above surface	m
Fog: Water Concentration	Liquid water density in fog	g/m^3
Rain: Rate	Rain rate	mm/hr
Frequency	Transmitted frequency	GHz
Polarization	Transmitted polarization; degrees above horizontal	°
Frequency Sweep: Start	Lowest frequency in sweep	GHz
Frequency Sweep: Step	Frequency sweep step value	GHz
Frequency Sweep: Stop	Highest frequency in sweep	GHz
Altitude	Altitude of transmitter	km
Grazing Angle	Angle of transmission; degrees below horizontal (negative)	°

Target Altitude	Position of the target above surface	km
Interval	Resolution of calculation	m
Results: Attenuation	Attenuation of transmission to surface	dB
Results: Dispersion	Phase dispersion of transmission to surface	°
Results: Incidence Angle	Incidence angle of transmission upon surface; degrees above horizontal	°
Notes	Program text output	

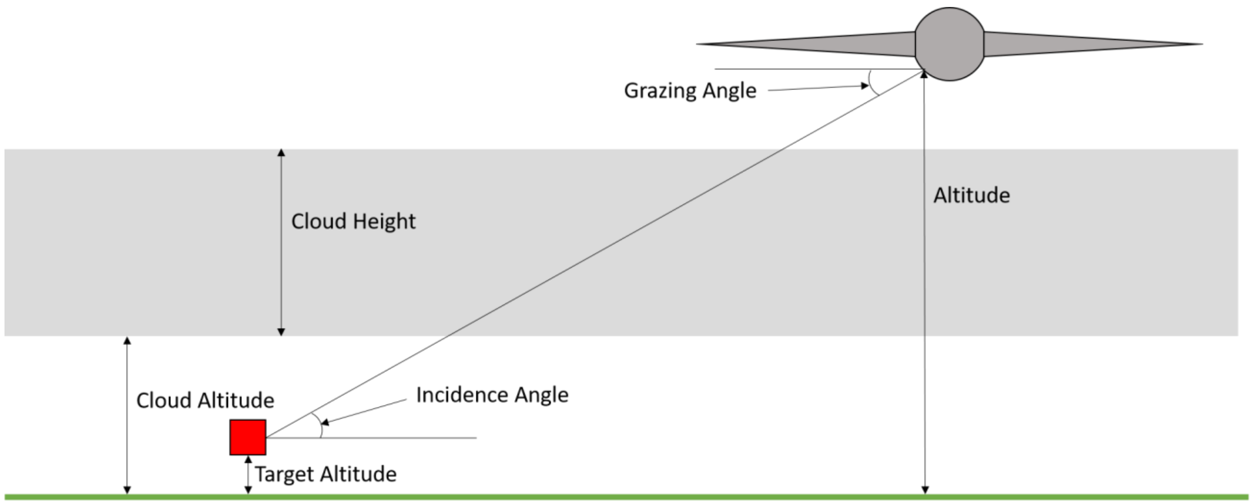

Fig. 2 – Clarification on grazing and incidence angles, as well as the difference between cloud altitude and cloud height

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.

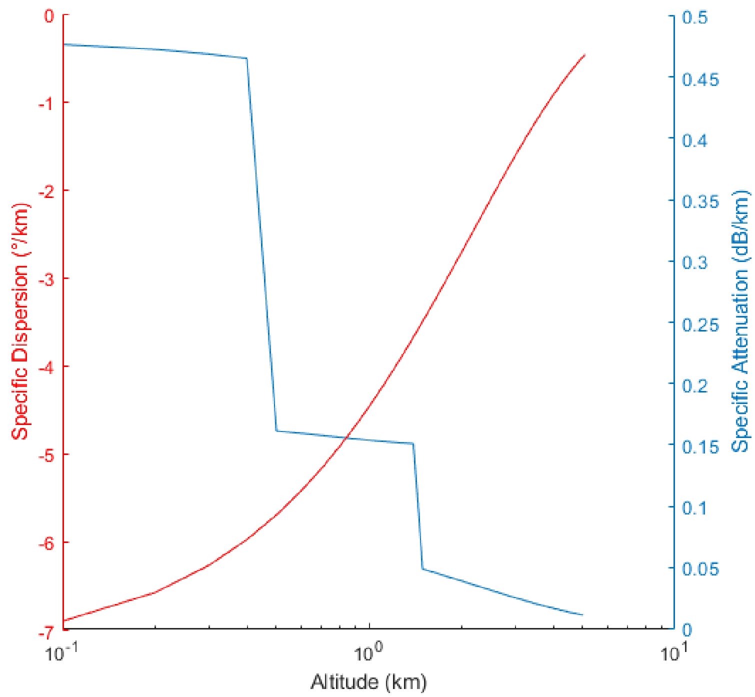

Fig. 3 – Program output figure showing specific attenuation and dispersion as functions of altitude

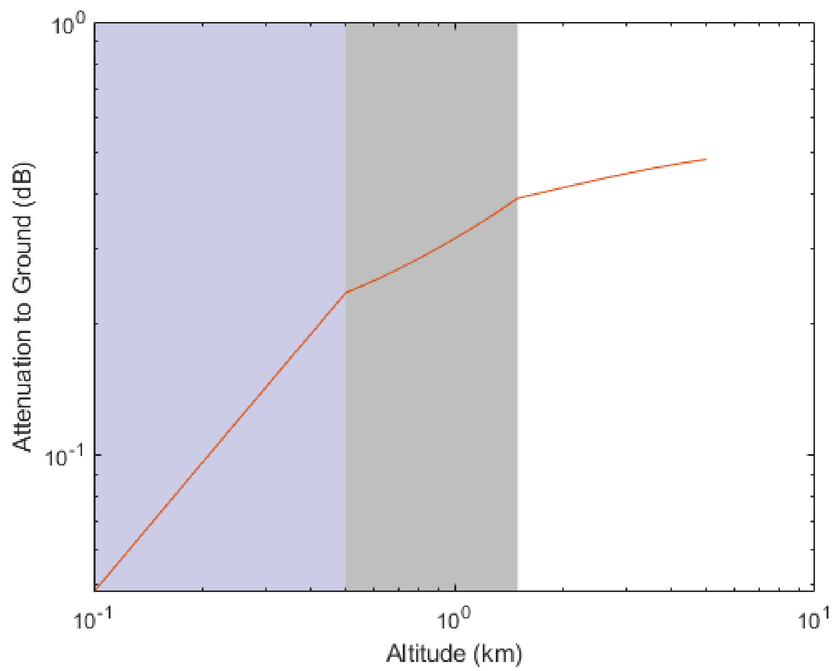

Fig. 4 – Program output figure showing total path loss vs. altitude and a representation of weather. Here the gray represents a cloud layer and the blue indicates rain.

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.

	A	B	C
1	Specific Attenuation (dB/km)		
2	Altitu	20	
3	0	0.480833856	
4	100	0.476673817	
5	200	0.472699818	
6	300	0.468903509	
7	400	0.465276922	
8	500	0.161265536	
9	600	0.159405323	
10	700	0.157734201	
11	800	0.156246816	
12	900	0.154938131	
13	1000	0.153803412	
14	1100	0.152838211	

Fig. 5 – Example of saved data file for one frequency at 20 GHz

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 ρ is given by

$$\rho = \frac{216.7 \cdot P_{wv}}{T} \tag{1}$$

ρ = water vapor density (g/m^3)

T = temperature (K)

P_{wv} = vapor pressure (mb)

$$P_{wv} = \frac{RH}{100} \cdot P_{wvs} \tag{2}$$

RH = relative humidity (%)

P_{wvs} = saturation vapor pressure (mb)

$$P_{wvs} = EF \cdot a \cdot e^{\frac{\left(b - \frac{t}{d}\right)t}{t+c}} \tag{3}$$

t = temperature (°C)

$$EF = \begin{cases} 1 + 10^{-4} \cdot [7.2 + P \cdot (.0320 + 5.9 \cdot 10^{-6} \cdot t^2)] & t > 0 \\ 1 + 10^{-4} \cdot [2.2 + P \cdot (.0383 + 6.4 \cdot 10^{-6} \cdot t^2)] & t < 0 \end{cases} \tag{4}$$

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].

$$a = \begin{cases} 6.1121 & t > 0 \\ 6.1115 & t < 0 \end{cases} \tag{5}$$

$$b = \begin{cases} 18.678 & t > 0 \\ 23.036 & t < 0 \end{cases} \tag{6}$$

$$c = \begin{cases} 257.14 & t > 0 \\ 279.82 & t < 0 \end{cases} \tag{7}$$

$$d = \begin{cases} 234.5 & t > 0 \\ 333.7 & t < 0 \end{cases} \tag{8}$$

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

$$h' = \frac{6356.766 \cdot h}{6356.766 + h} \tag{9}$$

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 2 - Sub-thermosphere temperature gradients [2]

Lower Boundary (km')	Upper Boundary (km')	Temperature Change (K/km')
0	11	-6.5
11	20	0
20	32	1
32	47	2.8
47	51	0
51	71	-2.8

71	84.852	-2

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

$$T = T_b + m \cdot (h' - h'_b) \tag{10}$$

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.

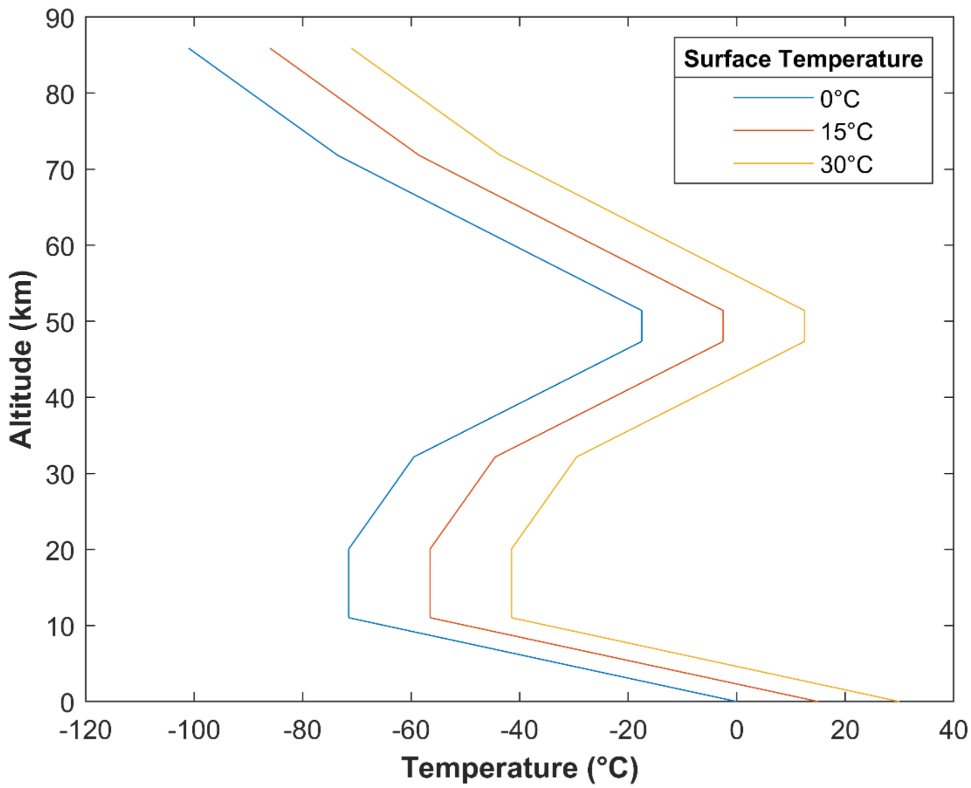

Fig. 6 – Temperature profile with different surface values

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

$$P = \begin{cases} P_b \cdot \left(\frac{T_b}{T}\right)^{\frac{34.1632}{m}} & m \neq 0 \\ P_b \cdot e^{\frac{-34.1632 \cdot (h' - h'_b)}{T}} & m = 0 \end{cases} \tag{11}$$

P = pressure (mb)

P_b = pressure of highest exceeded boundary (mb)

Sample pressure profiles are shown in Figure 7.

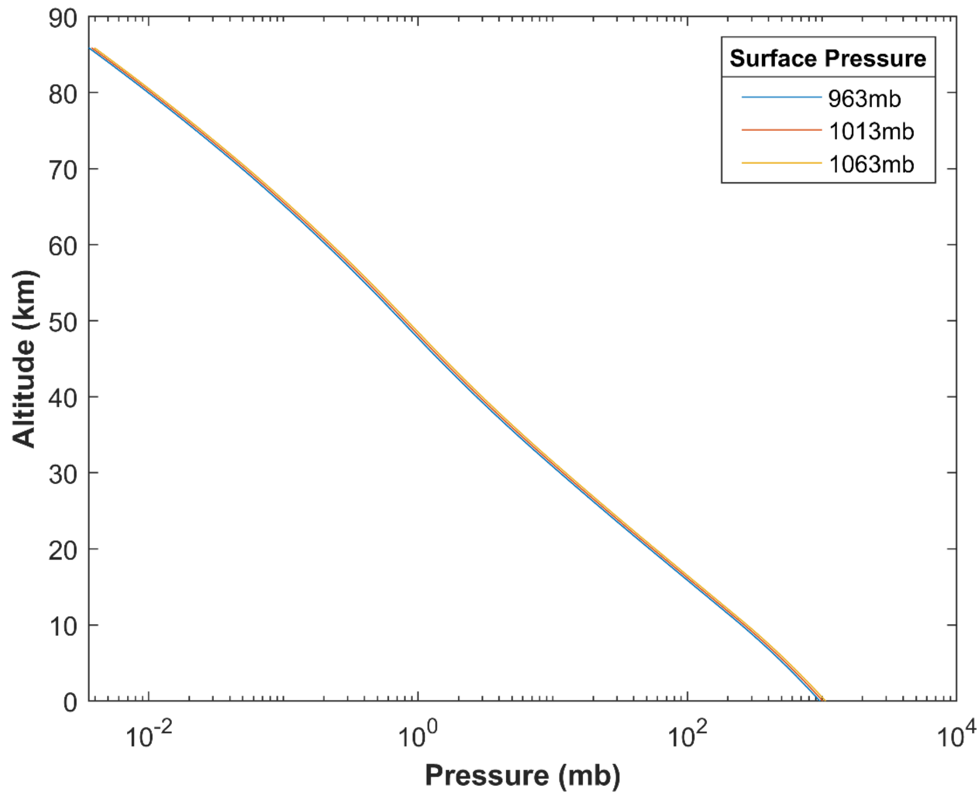

Fig. 7 – Pressure profile for different surface conditions. Pressure depends less on atmospheric layers than temperature does.

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 $2\cdot10^{-6}$, 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

$$\rho = \rho_0 \cdot e^{\frac{-h}{2}} \tag{12}$$

ρ = vapor density (g/m^3)

ρ_0 = vapor density at surface (g/m^3)

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

$$P_{wv} = \frac{\rho \cdot T}{216.7} \tag{13}$$

Example vapor density profiles at varying values of surface humidity are plotted in Figure 8.

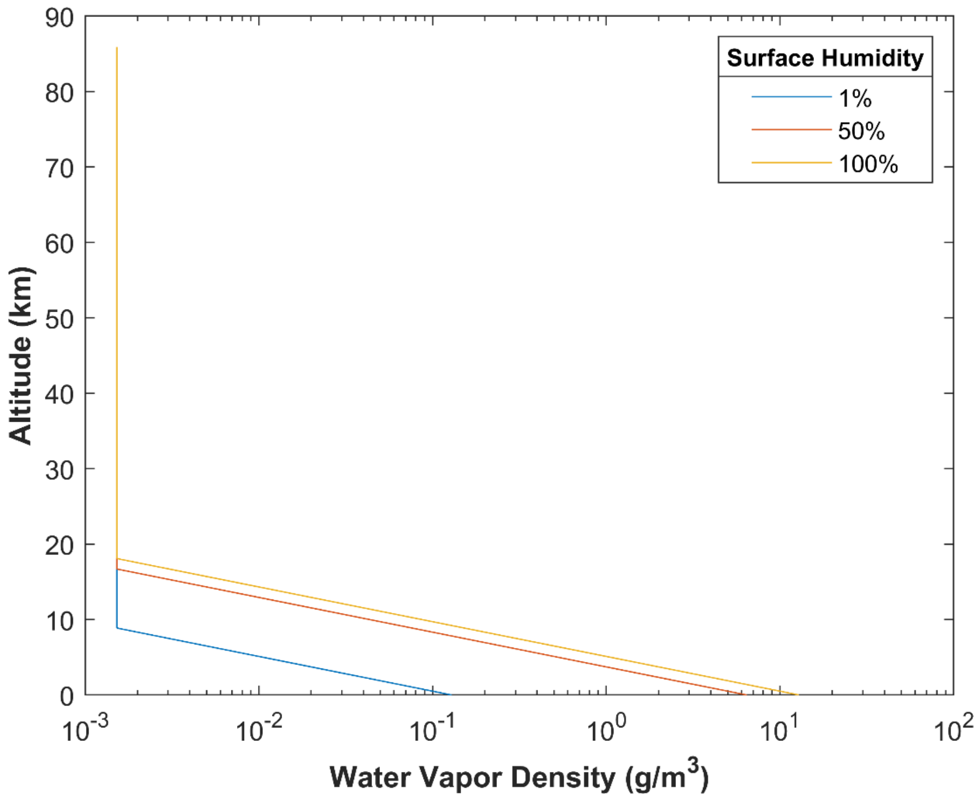

Fig. 8 – Vapor density profile for different surface conditions. The point at which the mixing ratio reaches $2{\cdot}10^{-6}$ and remains constant is clearly seen in all three cases.

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].

$$\gamma_G = \gamma_O + \gamma_W \tag{14}$$

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.

$$\gamma_O = .1820 \cdot f \cdot N_O'' \tag{15}$$

γ_O = attenuation due to dry air (dB/km)

f = frequency (GHz)

N_O'' = 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].

$$N_O'' = \sum_i S_{iO} \cdot F_{iO} + N_D'' \tag{16}$$

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

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

N_D'' = imaginary part of dry air continuum

$$S_{iO} = a_1 \cdot 10^{-7} \cdot P_d \cdot \theta^3 \cdot e^{a_2 \cdot (1-\theta)} \tag{17}$$

a_j = value of j^{th} column of spectroscopic data table for oxygen

P_d = partial pressure of dry air (mb)

θ = normalized temperature

$$P_d = P - P_{wv} \tag{18}$$

$$\theta = \frac{300}{T} \tag{19}$$

$$F_{iO} = \frac{f}{f_{iO}} \cdot \left[\frac{\Delta f_O - \delta \cdot (f_{iO} - f)}{(f_{iO} - f)^2 + \Delta f_O{}^2} + \frac{\Delta f_O - \delta \cdot (f_{iO} + f)}{(f_{iO} + f)^2 + \Delta f_O{}^2}\right] \tag{20}$$

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

Δf_O = oxygen line width (GHz)

δ = correction factor for oxygen line interference effects

$$\Delta f_{Oint} = a_3 \cdot 10^{-4} \cdot \left(P_d \cdot \theta^{(.8-a_4)} + 1.1 \cdot P_{wv} \cdot \theta\right) \tag{21}$$

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

$$\Delta f_O = \sqrt{\Delta f_{Oint}{}^2 + 2.25 \cdot 10^{-6}} \tag{22}$$

$$\delta = (a_5 + a_6 \cdot \theta) \cdot 10^{-4} \cdot P \cdot \theta^{.8} \tag{23}$$

$$N_D'' = f \cdot P_D \cdot \theta^2 \cdot \left[\frac{6.14 \cdot 10^{-5}}{d \cdot \left[1 + \left(\frac{f}{d}\right)^2\right]} + \frac{1.4 \cdot 10^{-12} \cdot P_d \cdot \theta^{1.5}}{1 + 1.9 \cdot 10^{-5} \cdot f^{1.5}}\right] \tag{24}$$

d = width parameter for non-resonant oxygen spectrum

Table 3 - Oxygen spectroscopic data [4]

f_{iO}	a_1	a_2	a_3	a_4	a_5	a_6
50.474	0.975	9.651	6.69	0	2.566	6.85
50.988	2.529	8.653	7.17	0	2.246	6.8
51.503	6.193	7.709	7.64	0	1.947	6.729
52.021	14.32	6.819	8.11	0	1.667	6.64
52.542	31.24	5.983	8.58	0	1.388	6.526
53.067	64.29	5.201	9.06	0	1.349	6.206
53.596	124.6	4.474	9.55	0	2.227	5.085
54.13	227.3	3.8	9.96	0	3.17	3.75
54.671	389.7	3.182	10.37	0	3.558	2.654
55.221	627.1	2.618	10.89	0	2.56	2.952
55.784	945.3	2.109	11.34	0	-1.172	6.135
56.265	543.4	0.014	17.03	0	3.525	-0.978
56.363	1331.8	1.654	11.89	0	-2.378	6.547
56.968	1746.6	1.255	12.23	0	-3.525	6.451
57.612	2120.1	0.91	12.62	0	-5.416	6.056
58.324	2363.7	0.621	12.95	0	-1.932	0.436
58.447	1442.1	0.083	14.91	0	6.768	-1.273
59.164	2379.9	0.387	13.53	0	-6.561	2.309
59.591	2090.7	0.207	14.08	0	6.957	-0.776
60.306	2103.4	0.207	14.15	0	-6.395	0.699
60.435	2438	0.386	13.39	0	6.342	-2.825
61.151	2479.5	0.621	12.92	0	1.014	-0.584
61.8	2275.9	0.91	12.63	0	5.014	-6.619
62.411	1915.4	1.255	12.17	0	3.029	-6.759
62.486	1503	0.083	15.13	0	-4.499	0.844
62.998	1490.2	1.654	11.74	0	1.856	-6.675
63.569	1078	2.108	11.34	0	0.658	-6.139
64.128	728.7	2.617	10.88	0	-3.036	-2.895
64.679	461.3	3.181	10.38	0	-3.968	-2.59
65.224	274	3.8	9.96	0	-3.528	-3.68
65.765	153	4.473	9.55	0	-2.548	-5.002
66.302	80.4	5.2	9.06	0	-1.66	-6.091

66.837	39.8	5.982	8.58	0	-1.68	-6.393
67.37	18.56	6.818	8.11	0	-1.956	-6.475
67.901	8.172	7.708	7.64	0	-2.216	-6.545
68.431	3.397	8.652	7.17	0	-2.492	-6.6
68.96	1.334	9.65	6.69	0	-2.773	-6.65
118.75	940.3	0.01	16.64	0	-0.439	0.079
368.498	67.4	0.048	16.4	0	0	0
424.763	637.7	0.044	16.4	0	0	0
487.249	237.4	0.049	16	0	0	0
715.393	98.1	0.145	16	0	0	0
773.84	572.3	0.141	16.2	0	0	0
834.145	183.1	0.145	16.2	0	0	0

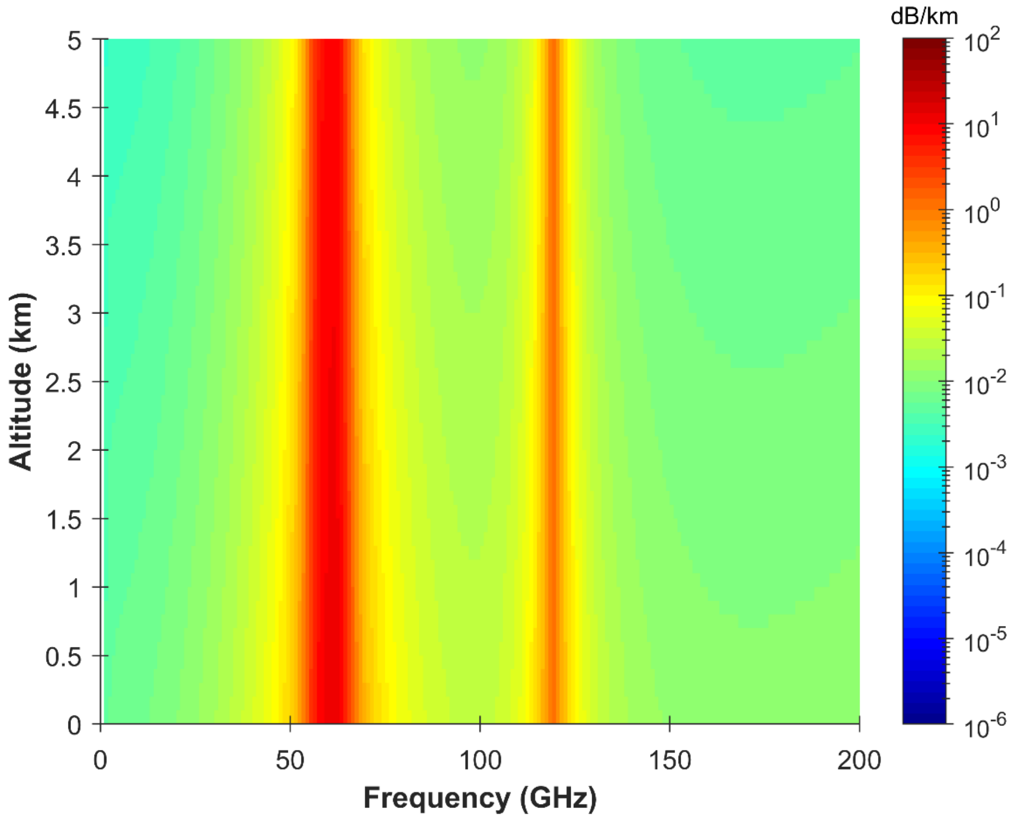

Fig. 9 – Example dry air specific attenuation. Based upon a surface temperature of 15°C and a surface pressure of 1013mb.

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.

$$\gamma_W = .1820 \cdot f \cdot N_W'' \tag{25}$$

γ_W = attenuation due to water vapor (dB/km)

N_W'' = imaginary part of complex refractivity of water vapor

$$N_W'' = \sum_i S_{iW} \cdot F_{iW} \tag{26}$$

S_{iW} = strength of ith vapor line

F_{iW} = ith vapor line shape factor

$$S_{iW} = b_1 \cdot 10^{-1} \cdot P_{wv} \cdot \theta^{3.5} \cdot e^{b_2 \cdot (1-\theta)} \tag{27}$$

b_j = value of jth column of spectroscopic data table for water vapor

$$F_{iW} = \frac{f}{f_{iW}} \cdot \left[\frac{\Delta f_W}{(f_{iW} - f)^2 + \Delta f_W{}^2} + \frac{\Delta f_W}{(f_{iW} + f)^2 + \Delta f_W{}^2}\right] \tag{28}$$

f_{iW} = center frequency of ith vapor line (GHz)

Δf_W = vapor line width (GHz)

$$\Delta f_{Wint} = b_3 \cdot 10^{-4} \cdot \left(P_d \cdot \theta^{b_4} + b_5 \cdot P_{wv} \cdot \theta^{b_6}\right) \tag{29}$$

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

$$\Delta f_W = .535 \cdot \Delta f_{Wint} + \sqrt{.217 \cdot \Delta f_{Wint}{}^2 + \frac{2.1316 \cdot 10^{-12} \cdot f_{iW}{}^2}{\theta}} \tag{30}$$

Table 4 - Water vapor spectroscopic data [4]

f_{iW}	b_1	b_2	b_3	b_4	b_5	b_6
22.235	0.1079	2.144	26.38	0.76	5.087	1
67.814	0.0011	8.732	28.58	0.69	4.93	0.82
119.996	0.0007	8.353	29.48	0.7	4.78	0.79
183.31	2.273	0.668	29.06	0.77	5.022	0.85
321.226	0.047	6.179	24.04	0.67	4.398	0.54
325.153	1.514	1.541	28.23	0.64	4.893	0.74
336.187	0.001	9.825	26.93	0.69	4.74	0.61
380.197	11.67	1.048	28.11	0.54	5.063	0.89
390.135	0.0045	7.347	21.52	0.63	4.81	0.55
437.347	0.0632	5.048	18.45	0.6	4.23	0.48

439.151	0.9098	3.595	20.07	0.63	4.483	0.52
443.018	0.192	5.048	15.55	0.6	5.083	0.5
448.001	10.41	1.405	25.64	0.66	5.028	0.67
470.889	0.3254	3.597	21.34	0.66	4.506	0.65
474.689	1.26	2.379	23.2	0.65	4.804	0.64
488.491	0.2529	2.852	25.86	0.69	5.201	0.72
503.569	0.0372	6.731	16.12	0.61	3.98	0.43
504.483	0.0124	6.731	16.12	0.61	4.01	0.45
547.6764	0.9785	0.158	26	0.7	4.5	1
552.021	0.184	0.158	26	0.7	4.5	1
556.936	497	0.159	30.86	0.69	4.552	1
620.701	5.015	2.391	24.38	0.71	4.856	0.68
645.7661	0.0067	8.633	18	0.6	4	0.5
658.006	0.2732	7.816	32.1	0.69	4.14	1
752.033	243.4	0.396	30.86	0.68	4.352	0.84
841.074	0.0134	8.177	15.9	0.33	5.76	0.45
859.865	0.1325	8.055	30.6	0.68	4.09	0.84
899.407	0.0547	7.914	29.85	0.68	4.53	0.9
902.555	0.0386	8.429	28.65	0.7	5.1	0.95
906.206	0.1836	5.11	24.08	0.7	4.7	0.53
916.172	8.4	1.441	26.73	0.7	5.15	0.78
923.1127	0.0079	10.293	29	0.7	5	0.8
970.315	9.009	1.919	25.5	0.64	4.94	0.67
987.927	134.6	0.257	29.85	0.68	4.55	0.9
1780	17506	0.952	196.3	2	24.15	5

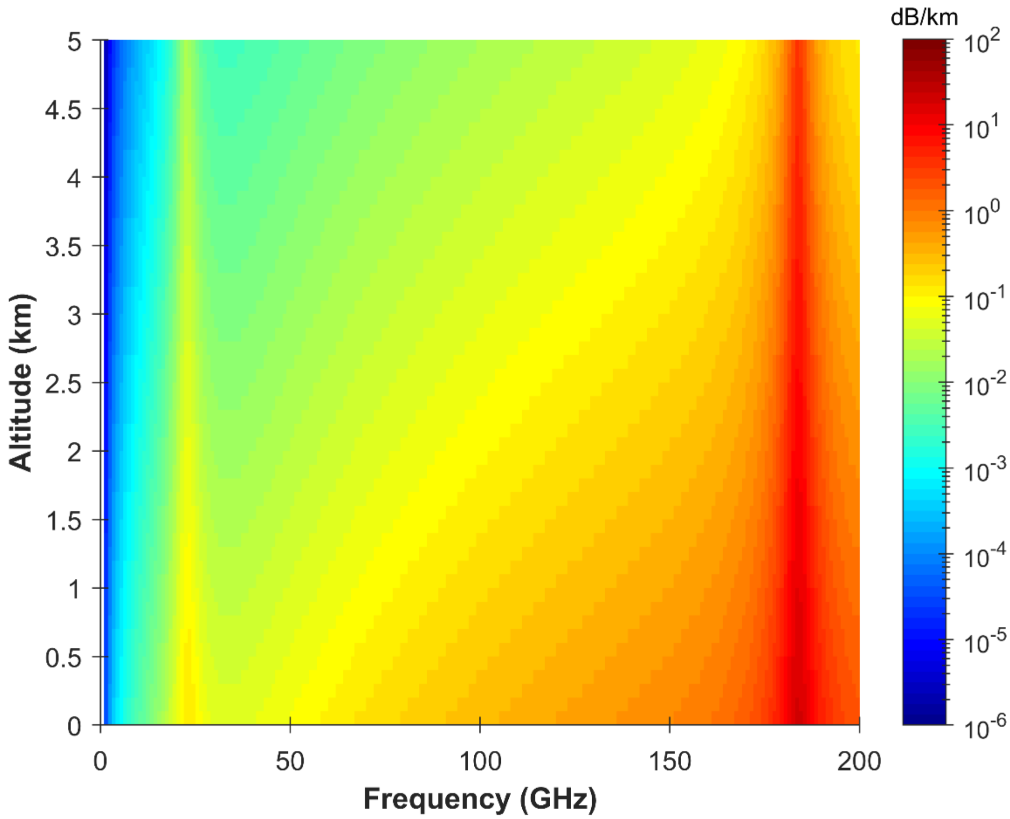

Fig. 10 – Example water vapor specific attenuation. Based upon a surface temperature of 15°C, pressure of 1013mb, and relative humidity of 50%.

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

$$\gamma_C = K \cdot M \tag{31}$$

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

K = liquid water specific attenuation coefficient

M = liquid water density (g/m^3)

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].

$$K = \frac{.819 \cdot f}{\varepsilon'' \cdot (1 + \eta^2)} \tag{32}$$

$$\eta = \frac{2 + \varepsilon'}{\varepsilon''} \tag{33}$$

$$\varepsilon' = \frac{\varepsilon_0 - \varepsilon_1}{1 + \left(\frac{f}{f_p}\right)^2} + \frac{\varepsilon_1 - \varepsilon_2}{1 + \left(\frac{f}{f_s}\right)^2} + \varepsilon_2 \tag{34}$$

$$\varepsilon'' = \frac{f \cdot (\varepsilon_0 - \varepsilon_1)}{f_p \cdot \left[1 + \left(\frac{f}{f_p}\right)^2\right]} + \frac{f \cdot (\varepsilon_1 - \varepsilon_2)}{f_s \cdot \left[1 + \left(\frac{f}{f_s}\right)^2\right]} \tag{35}$$

$$\varepsilon_0 = 77.66 + 103.3 \cdot (\theta - 1) \tag{36}$$

$$\varepsilon_1 = .0671 \cdot \varepsilon_0 \tag{37}$$

$$\varepsilon_2 = 3.52 \tag{38}$$

$$f_p = 20.20 - 146 \cdot (\theta - 1) + 316 \cdot (\theta - 1)^2 \tag{39}$$

$$f_s = 39.8 \cdot f_p \tag{40}$$

f_p = principal relaxation frequency (GHz)

f_s = secondary relaxation frequency (GHz)

θ 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.

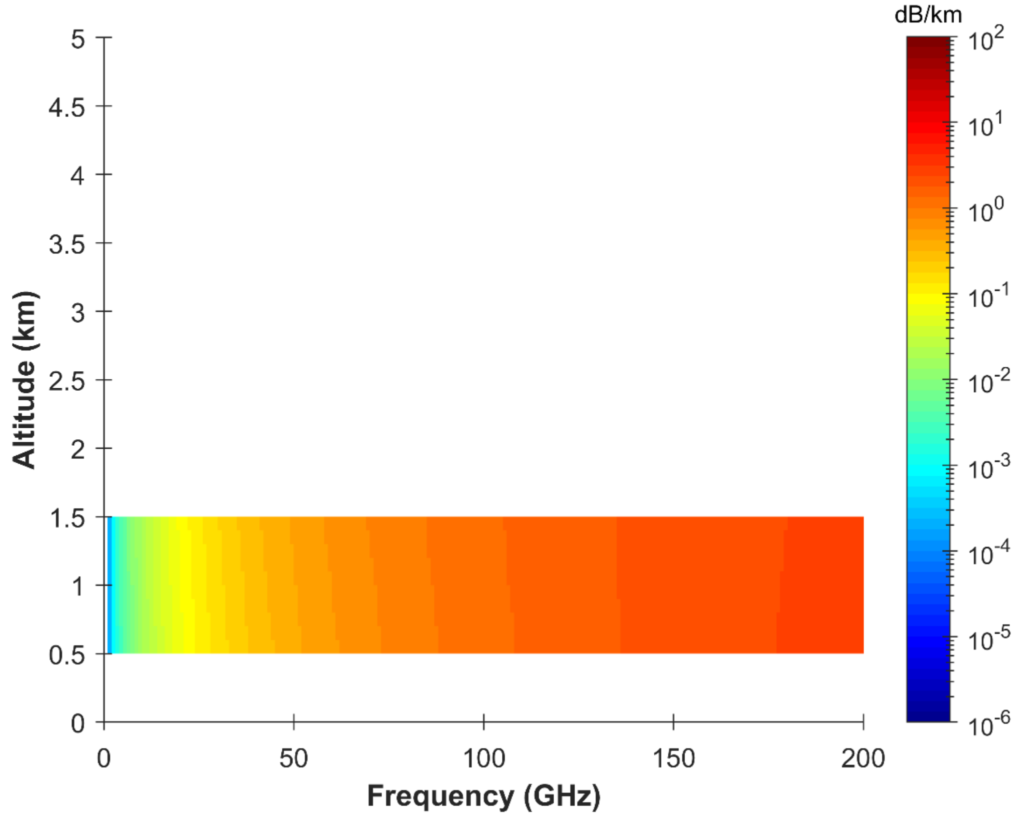

Fig. 11 – Example specific attenuation due to clouds. Based upon a surface temperature of 15°C and a liquid water density of .33g/m³.

1.5.3 Loss from rain

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

$$\gamma_R = k \cdot R^{\alpha} \tag{41}$$

γ_R = attenuation due to rain

The parameters k and α are given by

$$k = \frac{k_H + k_V + (k_H - k_V) \cdot \cos(\varphi)^2 \cdot \cos(2 \cdot \tau)}{2} \tag{42}$$

$$\alpha = \frac{k_H \cdot \alpha_H + k_V \cdot \alpha_V + (k_H \cdot \alpha_H - k_V \cdot \alpha_V) \cdot \cos(\varphi)^2 \cdot \cos(2 \cdot \tau)}{2 \cdot k} \tag{43}$$

φ = apparent elevation angle (rad)

τ = polarization angle (° from horizontal)

k_H = parameter k if wave polarization is entirely horizontal

k_V = parameter k if wave polarization is entirely vertical

α_H = parameter α if polarization is entirely horizontal

α_V = parameter α 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 α values, for the horizontal case they can be found using

$$log_{10}k_H = \sum_{i=1}^{4}\left(a_{ikH} \cdot e^{-\left(\frac{log_{10}f - b_{ikH}}{c_{ikH}}\right)^2}\right) + m_{kH} \cdot log_{10}f + c_{kH} \tag{44}$$

$$\alpha_H = \sum_{i=1}^{5}\left(a_{i\alpha H} \cdot e^{-\left(\frac{log_{10}f - b_{i\alpha H}}{c_{i\alpha H}}\right)^2}\right) + m_{\alpha H} \cdot log_{10}f + c_{\alpha H} \tag{45}$$

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

$$log_{10}k_V = \sum_{i=1}^{4}\left(a_{ikV} \cdot e^{-\left(\frac{log_{10}f - b_{ikV}}{c_{ikV}}\right)^2}\right) + m_{kV} \cdot log_{10}f + c_{kV} \tag{46}$$

$$\alpha_V = \sum_{i=1}^{5}\left(a_{i\alpha V} \cdot e^{-\left(\frac{log_{10}f - b_{i\alpha V}}{c_{i\alpha V}}\right)^2}\right) + m_{\alpha V} \cdot log_{10}f + c_{\alpha V} \tag{47}$$

Table 5 - Rain attenuation coefficients [7]

Coefficients for horizontal polarization:

i	a_{ikH}	b_{ikH}	c_{ikH}
1	-5.33980	-.10008	1.13098
2	-.35351	1.26970	.45400
3	-.23789	.86036	.15354
4	-.94158	.64552	.16817
m_{kH}		c_{kH}	
-.18961		.71147	

i	$a_{i\alpha H}$	$b_{i\alpha H}$	$c_{i\alpha H}$
1	-.14318	1.82442	-.55187
2	.29591	.77564	.19822
3	.32177	.63773	.13164
4	-5.37610	-.96230	1.47828
5	16.1721	-3.29980	3.43990
$m_{\alpha H}$		$c_{\alpha H}$	
.67849		-1.95537	

Coefficients for vertical polarization:

i	a_{ikV}	b_{ikV}	c_{ikV}
1	-3.80595	.56934	.81061
2	-3.44965	-.22911	.51059
3	-.39902	.73042	.11899
4	.50167	1.07319	.27195
m_{kV}		c_{kV}	
-.16398		.63297	

i	$a_{i\alpha V}$	$b_{i\alpha V}$	$c_{i\alpha V}$
1	-.07771	2.33840	-.76284
2	.56727	.95545	.54039
3	-.20238	1.14520	.26809
4	-48.2991	.791669	.116226
5	48.5833	.791459	.116479
$m_{\alpha V}$		$c_{\alpha V}$	
-.053739		.83433	

An example of attenuation caused by rain is Figure 12, and Figure 13 shows the total specific attenuation for the example case.

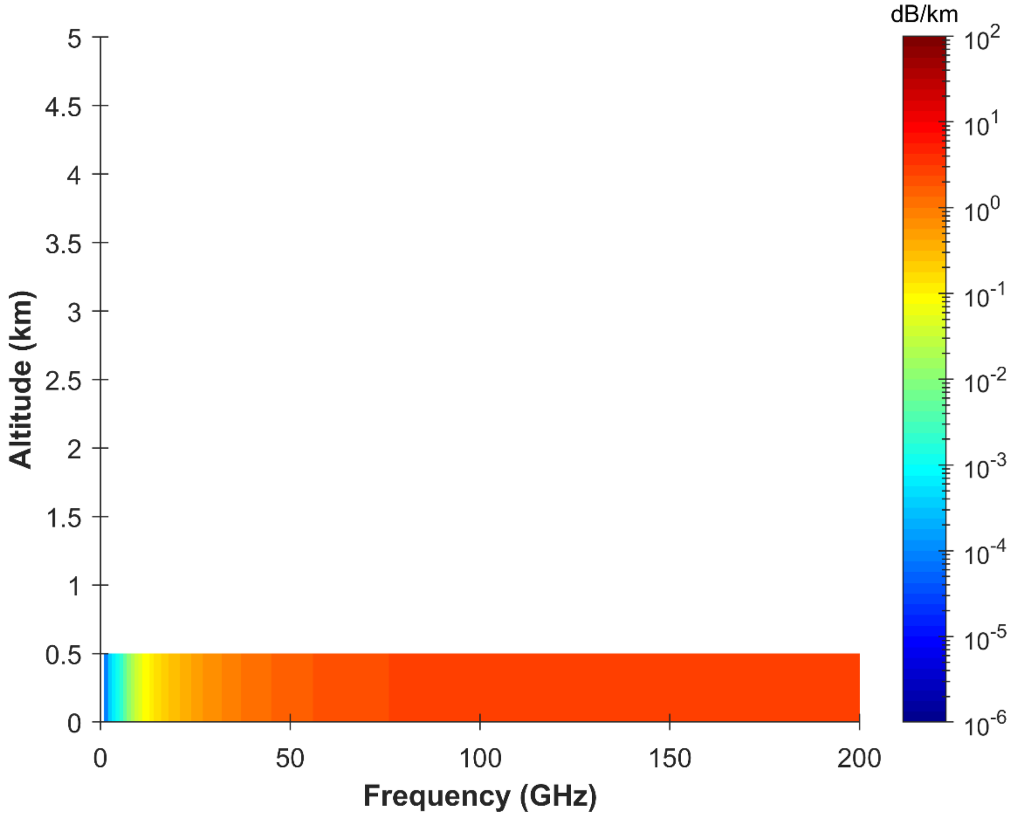

Fig. 12 – Example rain specific attenuation. Based upon a rain rate of 4mm/hr.

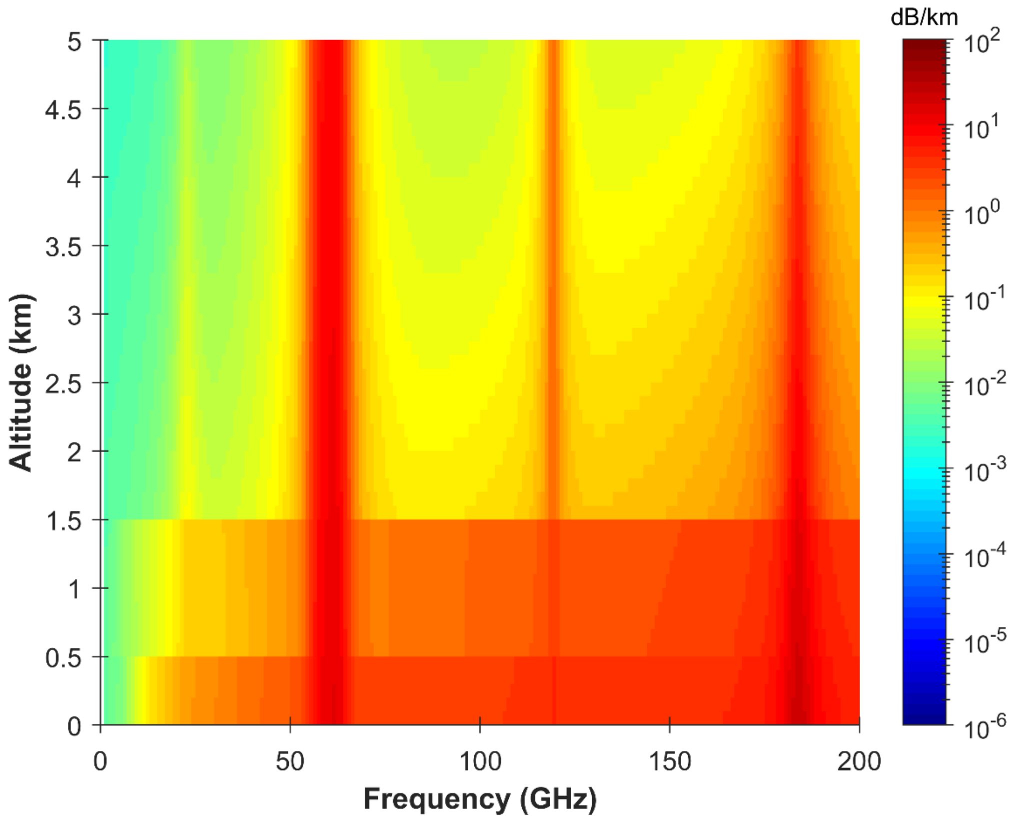

Fig. 13 – Specific attenuation due to all factors shown above for the example case. As attenuation is expressed in dB, this is found via addition.

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].

$$A = \int_{h_1}^{h_2} \frac{\gamma}{\sin(\varphi)} dh \tag{48}$$

A = total attenuation (dB)

γ = specific attenuation (dB/km)

h_1 = height of lower point

h_2 = height of higher point

$$\sin(\varphi) = \sqrt{1 - \cos(\varphi)^2} \tag{49}$$

$$\cos(\varphi) = \frac{(R_E + h_1) \cdot n(h_1)}{(R_E + h) \cdot n(h)} \cdot \cos(\varphi_1) \tag{50}$$

R_E = average radius of the Earth (6371km)

n = 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].

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

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 = 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 = number of atmospheric layers considered

a_i = path length through the i^{th} layer (km)

Figure 14 shows the attenuation to the surface for the example atmosphere as a function of height.

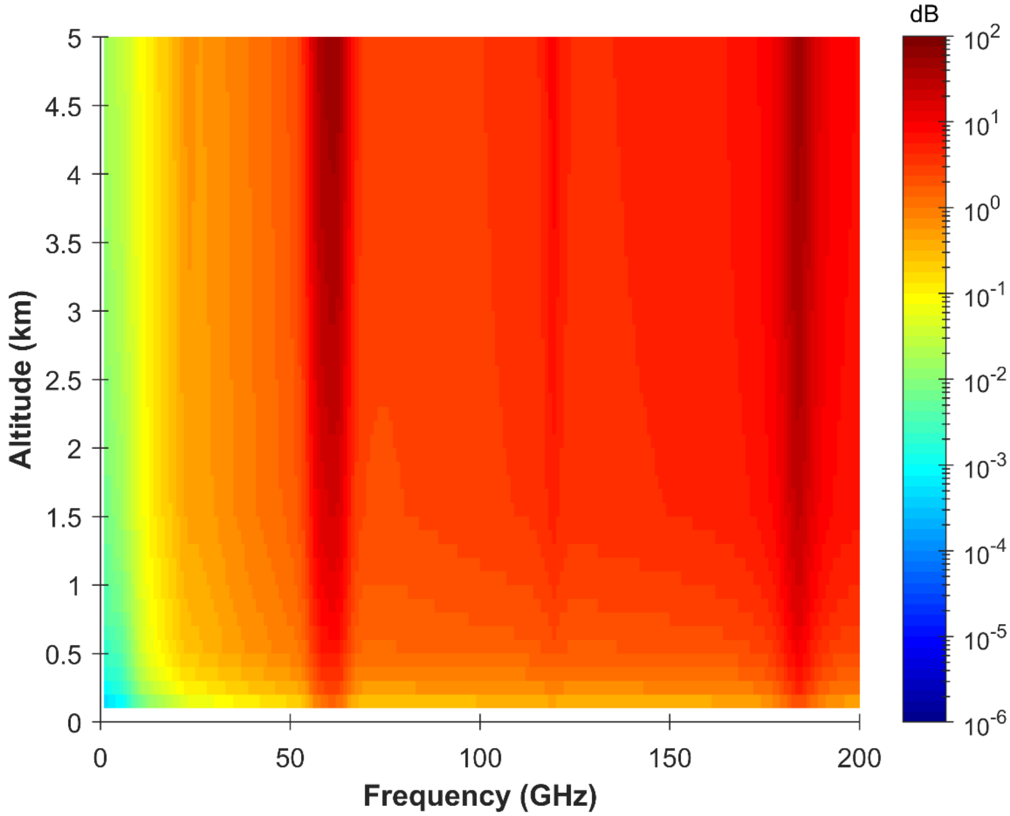

Fig. 14 – Attenuation to ground in the example case. Calculated assuming the beam is transmitted at nadir, or with a grazing angle of -90°.

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.

$$\beta_G = \beta_O + \beta_W \tag{55}$$

β_G = dispersion due to gas (°/km)

β_O = dispersion due to dry air (°/km)

β_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].

$$\beta_O = -1.2008 \cdot f \cdot N_O' \tag{56}$$

N_O' = real part of complex refractivity of dry air

$$N_O' = \sum_i S_{iO} \cdot F_{iO}' + N_D' \tag{57}$$

F_{iO}' = real part of i[th] oxygen line shape factor

N_D' = real part of dry air continuum

$$F_{iO}' = \frac{f}{f_{iO}} \cdot \left[\frac{(f_{iO} - f) + \delta \cdot \Delta f_O}{(f_{iO} - f)^2 + {\Delta f_O}^2} + \frac{(f_{iO} + f) - \delta \cdot \Delta f_O}{(f_{iO} + f)^2 + {\Delta f_O}^2} \right] \tag{58}$$

$$N_D' = \frac{-6.14 \cdot 10^{-5} \cdot P_d \cdot \theta^2 \cdot f^2}{f^2 + d^2} \tag{59}$$

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

$$\beta_W = -1.2008 \cdot f \cdot N_W' \tag{60}$$

N_W' = real part of complex refractivity of water vapor

$$N_W' = \sum_i S_{iW} \cdot F_{iW}' \tag{61}$$

F_{iW}' = real part of i[th] vapor line shape factor

$$F_{iW}' = \frac{f}{f_{iW}} \cdot \left[\frac{(f_{iW} - f) + \delta \cdot \Delta f_W}{(f_{iW} - f)^2 + {\Delta f_W}^2} + \frac{(f_{iW} + f) - \delta \cdot \Delta f_W}{(f_{iW} + f)^2 + {\Delta f_W}^2} \right] \tag{62}$$

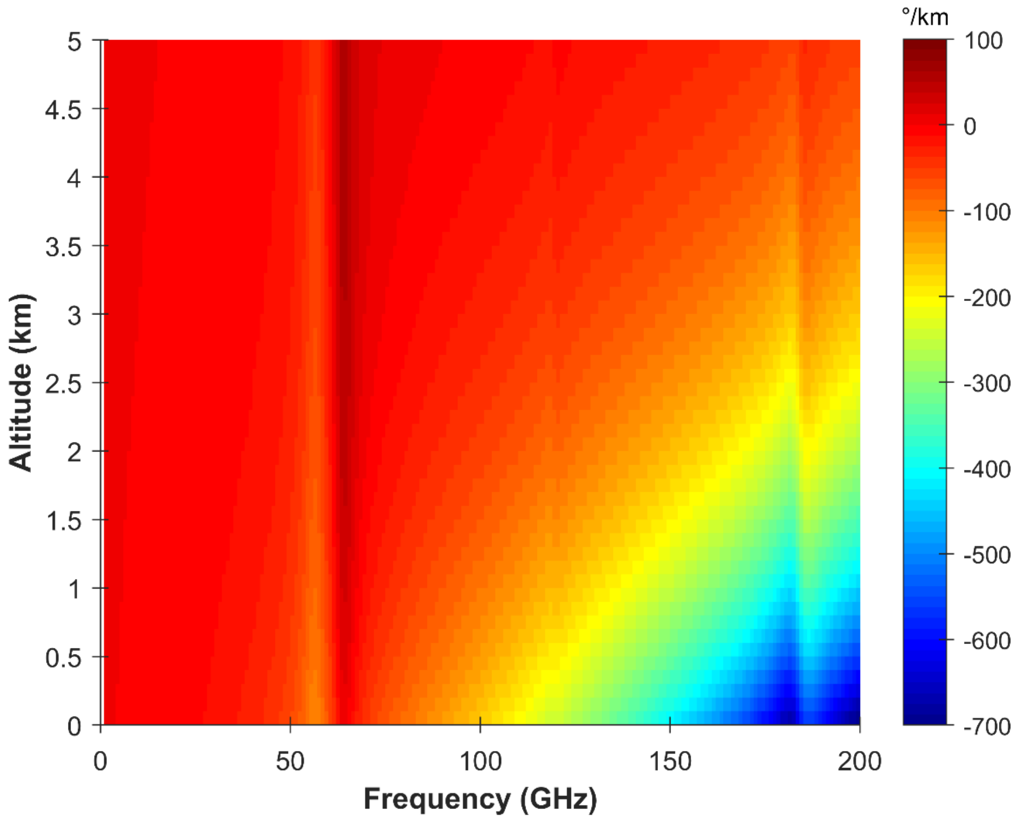

Fig. 15 – Example phase dispersion, calculated from a surface temperature of 15°C, pressure of 1013mb, and humidity of 50%. Only the effects of gases are included.

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

[1] International Telecommunication Union, *Recommendation ITU-R P.453-14: The radio refractive index: its formula and refractivity data*, Geneva, Aug. 25, 2019. Accessed on: Dec. 2, 2019. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.453-14-201908-I!!PDF-E.pdf

[2] International Telecommunication Union, *Recommendation ITU-R P.835-6: Reference standard atmospheres*, Geneva, Dec. 4, 2017. Accessed on: Dec. 2, 2019. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.835-6-201712-I!!PDF-E.pdf

[3] R. A. Minzner et al, "Defining Constants, Equations, and Abbreviated Tables of the 1975 U.S. Standard Atmosphere," NASA, Washington, D.C., NASA TR R-459, May 1976, pp. 13-14. Accessed on: Dec. 2, 2019. [Online]. Available: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19760017709.pdf

[4] International Telecommunication Union, *Recommendation ITU-R P.676-12: Attenuation by atmospheric gases and related effects*, Geneva, Aug. 14, 2019. Accessed on: Dec. 2, 2019. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.676-12-201908-I!!PDF-E.pdf

[5] H. J. Liebe, "An updated model for millimeter wave propagation in moist air," *Radio Science*, vol. 20, no. 5, pp. 1069-1089, Sept./Oct. 1985. Accessed on: Apr. 7, 2020. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/epdf/10.1029/RS020i005p01069

[6] International Telecommunication Union, *Recommendation ITU-R P.840-8: Attenuation due to clouds and fog*, Geneva, Aug. 14, 2019. Accessed on: Dec. 2, 2019. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.840-8-201908-I!!PDF-E.pdf

[7] International Telecommunication Union, *Recommendation ITU-R P.838-3: Specific attenuation model for rain for use in prediction methods*, Mar. 8, 2005. Accessed on: Dec. 2, 2019. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.838-3-200503-I!!PDF-E.pdf

APPENDIX 1

atmos_app_lite.mlapp

```
classdef atmos_app_lite < matlab.apps.AppBase


    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        GroundTemperatureEditFieldLabel  matlab.ui.control.Label
        GroundTemperatureEditField      matlab.ui.control.NumericEditField
        GroundPressureEditFieldLabel    matlab.ui.control.Label
        GroundPressureEditField         matlab.ui.control.NumericEditField
        RelativeHumidityEditFieldLabel  matlab.ui.control.Label
        RelativeHumidityEditField       matlab.ui.control.NumericEditField
        CloudsCheckBox                  matlab.ui.control.CheckBox
        AltitudeEditFieldLabel          matlab.ui.control.Label
        AltitudeEditField               matlab.ui.control.NumericEditField
        HeightEditFieldLabel            matlab.ui.control.Label
        HeightEditField                 matlab.ui.control.NumericEditField
        WaterConcentrationEditFieldLabel  matlab.ui.control.Label
        WaterConcentrationEditField     matlab.ui.control.NumericEditField
        FogCheckBox                     matlab.ui.control.CheckBox
        HeightEditField_2Label          matlab.ui.control.Label
        HeightEditField_2               matlab.ui.control.NumericEditField
        WaterConcentrationEditField_2Label  matlab.ui.control.Label
        WaterConcentrationEditField_2   matlab.ui.control.NumericEditField
        RainCheckBox                    matlab.ui.control.CheckBox
        RateEditFieldLabel              matlab.ui.control.Label
        RateEditField                   matlab.ui.control.NumericEditField
        AltitudeEditField_2Label        matlab.ui.control.Label
        AltitudeEditField_2             matlab.ui.control.NumericEditField
        GrazingAngleEditFieldLabel      matlab.ui.control.Label
        GrazingAngleEditField           matlab.ui.control.NumericEditField
        RunButton                       matlab.ui.control.Button
        SaveButton                      matlab.ui.control.Button
        LoadButton                      matlab.ui.control.Button
        IntervalEditFieldLabel          matlab.ui.control.Label
        IntervalEditField               matlab.ui.control.NumericEditField
        KLabel                          matlab.ui.control.Label
        mbLabel                         matlab.ui.control.Label
        Label                           matlab.ui.control.Label
        mLabel                          matlab.ui.control.Label
        mLabel_2                        matlab.ui.control.Label
        gm3Label                        matlab.ui.control.Label
        mLabel_3                        matlab.ui.control.Label
        gm3Label_2                      matlab.ui.control.Label
        kmLabel                         matlab.ui.control.Label
        Label_3                         matlab.ui.control.Label
```

```
        mmhrLabel                      matlab.ui.control.Label
        mLabel_6                       matlab.ui.control.Label
        FrequencyEditFieldLabel        matlab.ui.control.Label
        FrequencyEditField             matlab.ui.control.NumericEditField
        GHzLabel                       matlab.ui.control.Label
        AttenuationEditFieldLabel      matlab.ui.control.Label
        AttenuationEditField           matlab.ui.control.NumericEditField
        DispersionEditFieldLabel       matlab.ui.control.Label
        DispersionEditField            matlab.ui.control.NumericEditField
        dBLabel                        matlab.ui.control.Label
        Label_4                        matlab.ui.control.Label
        PolarizationEditFieldLabel     matlab.ui.control.Label
        PolarizationEditField          matlab.ui.control.NumericEditField
        Label_2                        matlab.ui.control.Label
        IncidenceAngleEditFieldLabel   matlab.ui.control.Label
        IncidenceAngleEditField        matlab.ui.control.NumericEditField
        Label_5                        matlab.ui.control.Label
        ResultsLabel                   matlab.ui.control.Label
        StartEditFieldLabel            matlab.ui.control.Label
        StartEditField                 matlab.ui.control.NumericEditField
        StepEditFieldLabel             matlab.ui.control.Label
        StepEditField                  matlab.ui.control.NumericEditField
        StopEditFieldLabel             matlab.ui.control.Label
        StopEditField                  matlab.ui.control.NumericEditField
        FrequencySweepCheckBox         matlab.ui.control.CheckBox
        GHzLabel_2                     matlab.ui.control.Label
        GHzLabel_3                     matlab.ui.control.Label
        GHzLabel_4                     matlab.ui.control.Label
        NotesTextAreaLabel             matlab.ui.control.Label
        NotesTextArea                  matlab.ui.control.TextArea
        kmLabel_2                      matlab.ui.control.Label
        TargetAltitudeEditFieldLabel   matlab.ui.control.Label
        TargetAltitudeEditField        matlab.ui.control.NumericEditField
        SaveFileCheckBox               matlab.ui.control.CheckBox
    end


    % 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.Valu
e,...

'targetht',app.TargetAltitudeEditField.Value,'pol',app.PolarizationEditField.Value,...
            'Tgnd',app.GroundTemperatureEditField.Value,...

'Pgnd',app.GroundPressureEditField.Value,'RH',app.RelativeHumidityEditField.Value,'rai
n_rt',app.RateEditField.Value,...

'fogH2Oconc',app.WaterConcentrationEditField_2.Value,'fogtop',app.HeightEditField_2.Va
lue,...
            'cloudH2Oconc',app.WaterConcentrationEditField.Value,...

'cloudht',app.HeightEditField.Value,'cloudalt',app.AltitudeEditField.Value,...

'freq',app.FrequencyEditField.Value,'interval',app.IntervalEditField.Value,...

'atten',app.AttenuationEditField.Value,'disp',app.DispersionEditField.Value,'resAngle'
,app.IncidenceAngleEditField.Value,...

'sweepCheck',app.FrequencySweepCheckBox.Value,'freqStart',app.StartEditField.Value,'fr
eqStep',app.StepEditField.Value,...

'freqStop',app.StopEditField.Value,'notes',app.NotesTextArea.Value,'saveFile',app.Save
FileCheckBox.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);
```

```
app.HeightEditFieldLabel.HorizontalAlignment = 'right';
app.HeightEditFieldLabel.Position = [107 355 40 22];
app.HeightEditFieldLabel.Text = 'Height';

% 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 = uicheckbox(app.UIFigure);
            app.RainCheckBox.Text = 'Rain';
            app.RainCheckBox.Position = [57 206 47 22];

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

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

            % Create AltitudeEditField_2Label
            app.AltitudeEditField_2Label = uilabel(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 = uieditfield(app.UIFigure, 'numeric');
            app.AltitudeEditField_2.Position = [458 318 100 22];
            app.AltitudeEditField_2.Value = 6;

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

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

            % Create RunButton
            app.RunButton = uibutton(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';
```

```
        end
    end

    % 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
        end

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

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

loss_calculator_lite.m

```
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));

e1 = RH/100*es;         %water vapor partial pressure
v1 = e1*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);
```

```
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');

    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');
```

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

    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');
    writematrix(Disp_vs_ht',[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

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;
```

```
    if (ii>size(H,1))
        break;
    end
end

pressure = z;

end
```

Denswv4.m

```
function vapor = Denswv4(hZ,v_gZ,PZ,TZ)

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

%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
```

Atten_oxy.m

```
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.^.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.^3.*exp(oxZ(1,3,:).*(1-theta)).*1e-7;     %oxygen line
strength
G = oxZ(1,4,:).*(p1.*theta.^(.8-oxZ(1,5,:))+1.1.*e1.*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.^.8.*1e-4;        %correction factor
for interference in oxygen lines
F2oxy = FE./oxZ(1,1,:).*((G-(oxZ(1,1,:)-FE).*delta)./((oxZ(1,1,:)-FE).^2+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
```

Atten_wv.m

```
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 e1?

FE = FZ./1e9;

pl = PZ-el;              %dry air partial pressure

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;
%vapor line width
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
```

Disp_oxy.m

```
function o2disp = Disp_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]);

const = 216.7;

theta = 300./TZ;

e1 = VZ.*TZ./const;

p1 = PZ-e1;

FE = FZ./1e9;

d = 5.6e-4.*(p1+e1).*theta.^.8;

N1D = (-6.14e-5.*p1.*theta.^2.*FE.^2)./(FE.^2+d.^2);

S = oxZ(1,2,:).*p1.*theta.^3.*exp(oxZ(1,3,:).*(1-theta)).*1e-7;
G = oxZ(1,4,:).*(p1.*theta.^(.8-oxZ(1,5,:))+1.1.*e1.*theta).*1e-4;
G = sqrt(G.^2+2.25e-6);     %Zeeman splitting
delta = (oxZ(1,6,:)+oxZ(1,7,:).*theta).*p1.*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)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% 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]);

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
```

Atten_rn.m

```
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 -3.80595; -.35351 -3.44965; -.23789 -.39902; -.94158 .50167];
bk = [-.10008 .56934; 1.26970 -.22911; .86036 .73042; .64552 1.07319];
ck = [1.13098 .81061; .45400 .51059; .15354 .11899; .16817 .27195];
aalpha = [-.14318 -.07771; .29591 .56727; .32177 -.20238; -5.37610 -48.2991; 16.1721
48.5833];
balpha = [1.82442 2.33840; .77564 .95545; .63773 1.14520; -.96230 .791669; -3.29980
.791459];
calpha = [-.55187 -.76284; .19822 .54039; .13164 .26809; 1.47828 .116226; 3.43990
.116479];
m2k = [-.18961; -.16398];
c2k = [.71147; .63297];
m2alpha = [.67849; -0.053739];
c2alpha = [-1.95537; .83433];

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
```

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));

e1 = RH/100*es;          %water vapor partial pressure
v1 = e1*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);

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

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({'Frequency (GHz)/Altitude (m)'},[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');

    writecell({'Total Attenuation (dB)'},[outpath outfile],'Sheet',2,'Range','A1');
    writecell({'Frequency (GHz)/Altitude (m)'},[outpath
outfile],'Sheet',2,'Range','A2');
```

```
    writematrix(FreqVec'./1e9,[outpath outfile],'Sheet',2,'Range','B2');
    writematrix(ja'.*interval,[outpath outfile],'Sheet',2,'Range','A3');
    writematrix(Loss_vs_ht_all',[outpath outfile],'Sheet',2,'Range','B3');


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

end

%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(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

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

end
```