DOCUMENTATION

This package describes how to generate the spectral polarimetric variables from a 94 GHz cloud radar in moderate rain conditions, including the noise and turbulence of real measurements.

The folder contains one requirement file: "**requirements.txt**" one configuration file: "**test_config_file.txt**" and two python files: "**simulation_method1.py**" and "**functions.py**".

- **requirements.txt**

The requirements file includes all the necessary Python packages and dependencies required to run the code in this package. To ensure smooth execution, these dependencies need to be installed in your environment.

Below is a list of the key dependencies:

numpy: Fundamental package for numerical computations in Python.
matplotlib: Used for plotting and visualizing data.
pytmatrix: A package for T-matrix scattering calculations, particularly useful for radar and polarimetric simulations. Includes modules such as Scatterer, radar, and orientation for advanced scattering and radar modeling. As mentioned in the overview, it can be found here: https://github.com/jleinonen/pytmatrix/wiki
scipy: Provides tools for scientific computing, including interpolation methods.
os: Standard library for interacting with the operating system (e.g., file paths).
random: Standard library for generating random numbers.

The functions module will be described in detail later in the document.
functions: Custom module containing user-defined functions:
parametrization_axis_ratio: Parameterizes the axis ratio of particles.
velocity_diameters_rel: Relates velocity to particle diameters.
fast_plotting: Generates simple and quick plots for x and y values.

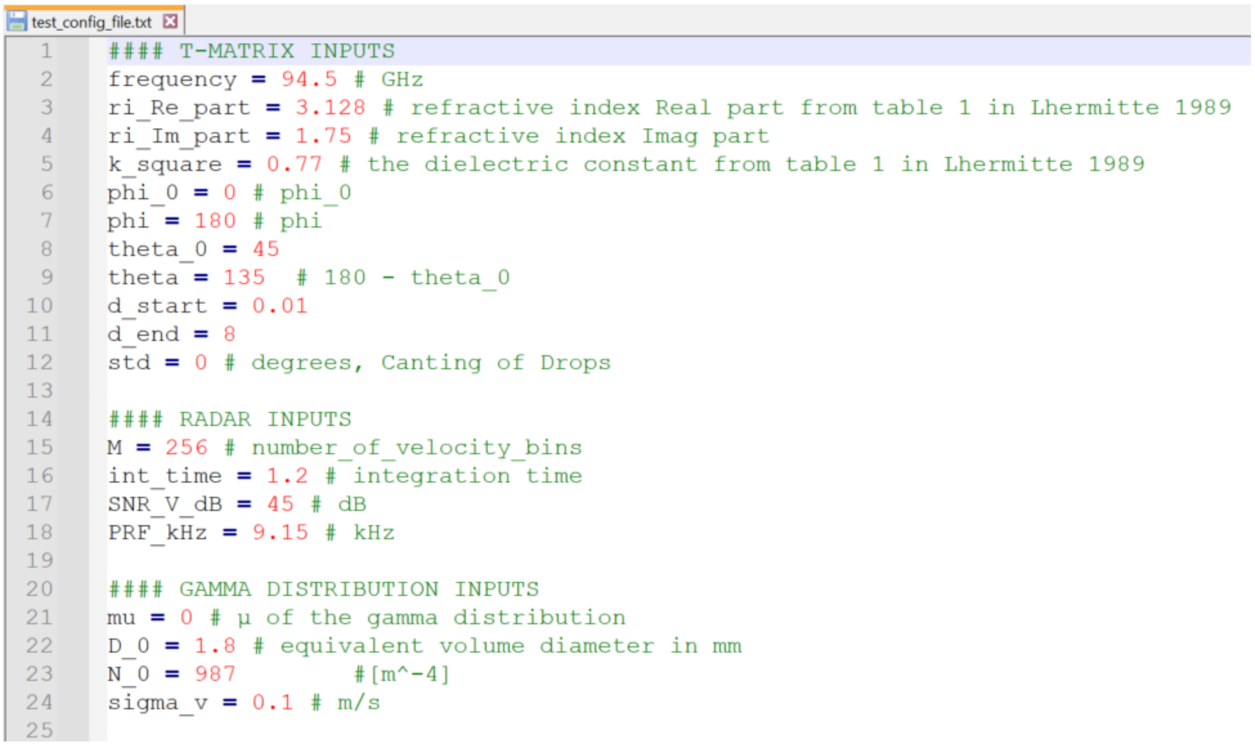
- **test_config_file.txt**

A configuration file is used to store and organize all the parameters and settings required for the code to run. By centralizing these inputs, the code execution is simplified, and users can modify parameters without changing the core code.

In this package, the configuration file contains all the necessary parameters needed for the simulations, such as:
T-matrix parameters.
Radar parameters.
Raindrop Size Distribution Parameters.

This is also useful for easily testing and adapting for different use cases or scenarios.

The configuration file looks like this:



```
#### T-MATRIX INPUTS
frequency = 94.5 # GHz
ri_Re_part = 3.128 # refractive index Real part from table 1 in Lhermitte 1989
ri_Im_part = 1.75 # refractive index Imag part
k_square = 0.77 # the dielectric constant from table 1 in Lhermitte 1989
phi_0 = 0 # phi_0
phi = 180 # phi
theta_0 = 45
theta = 135  # 180 - theta_0
d_start = 0.01
d_end = 8
std = 0 # degrees, Canting of Drops

#### RADAR INPUTS
M = 256 # number_of_velocity_bins
int_time = 1.2 # integration time
SNR_V_dB = 45 # dB
PRF_kHz = 9.15 # kHz

#### GAMMA DISTRIBUTION INPUTS
mu = 0 # μ of the gamma distribution
D_0 = 1.8 # equivalent volume diameter in mm
N_0 = 987        #[m^-4]
sigma_v = 0.1 # m/s

```

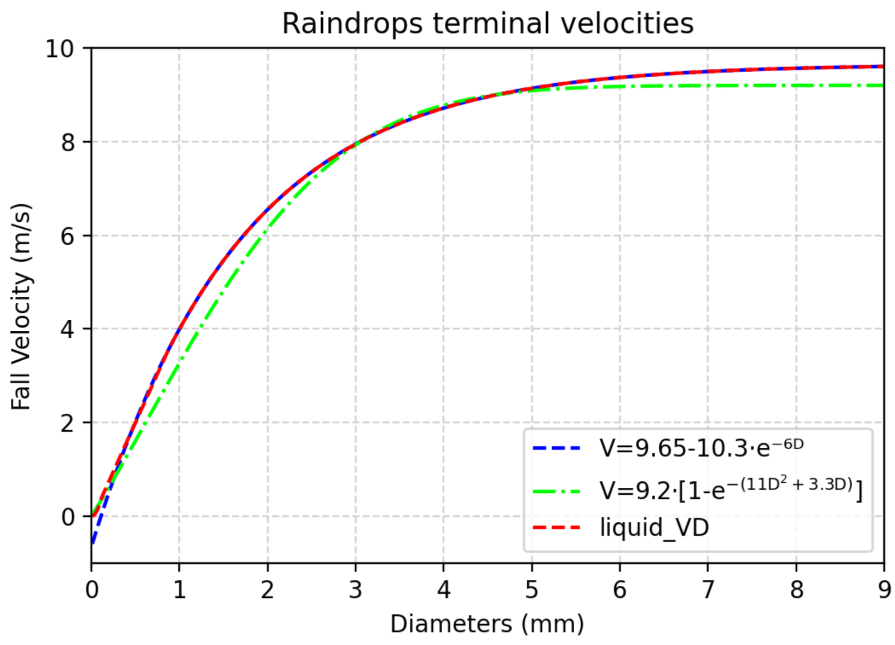
The parameters are related to the T-matrix scattering code (frequency of EM wave, refractive index, geometry settings, diameters array, etc.), the Radar inputs (number of FFT bins, PRF, etc.) and the Gamma Fit that represents the measurements (equivalent volume diameter, mu, etc.). In order to input the parameters to start the simulation process, the path of the configuration file is fed in the main script “simulation_method1.py”. More information on these parameters, can be found in the Theoretical Overview.

- **functions.py**

The “functions.py” script, contains the functions that are necessary for the axis ratio parametrization, the raindrops’ velocity representation and the plotting.

More Specifically, “*parametrization_axis_ratio(diameters)*” uses as input the array of diameters and generates an array of axis ratio according to keenan et. al (2001), Bringi and Chandrasekar (2001), and Beard and Chuang (1987) models. The raindrops are considered as spheroids, acquiring a more oblate shape as they grow. The very small drops are considered as perfect spheres.

The “*velocity_diameters_rel(d_m)*” is a function that uses as input the array of diameters and produces the array of raindrop velocity. Some common velocity-diameter functions are $V(D)=9.65-10.3e^{-6D}$ (blue line), and $V(D)=9.2[1-e^{-(11D\text{^}2\ +\ 3.3D)}]$ (green line). The *velocity_diameters_rel(d_m)* which is used in the simulation is plotted with red line in the figure below.



Finally, the “*fast_plotting*” function, generates quick and straightforward plots for any given x and y values, like the figures of the Example demonstration.

- **simulation_method1.py**

The “simulation_method1.py” is separated in several blocks, indicated with # In[...] and represent different processes.

# In[Read the inputs from the configuration file]

In this block, the parameters are read from the configuration file. Some parameters are calculated from the inputs, for example the wavelength is calculated from wl = 299792458/ (frequency*1e6) and the number of spectra averaging is calculated from the integration time and number of FFT bins K = 2*int(int_time*PRF_kHz*1e3/M). The lines that start with “#” in the configuration file, are ignored.

# In[Calculate the spectrum parameters]

The spectrum parameters are related to the construction of a typical Doppler spectrum as described in eq. (1)

$$S_{VV}(D) = \frac{\lambda^4}{\pi^5 |K|^2} N(D)\sigma_{vv}(D)\frac{dD}{du} \quad (1)$$

λ is the radar wavelength, $|K^2|$ is derived from the dielectric factor of water, N(D) is the drop size distribution, σ_{VV} is the backscattering cross section for V channel and u is the velocity of droplets along the line of sight of the radar beam. The array of the diameters is defined and then the raindrops' velocities are calculated according to the function "*velocity_diameters_rel(d_m)*" described earlier. The backscattering cross section $\sigma_{VV}(D)$ is calculated from the T-matrix. The raindrop size distribution N(D), is generated as a gamma size distribution with characteristics as set in the configuration file. Moreover, in this block, the spectrum is extended into the negative velocity domain with zero values, mimicking the real radar observations. Following the extension, the spectrum is interpolated to ensure equal velocity bins.

# In[Noise Implementation]

Following Yu et al. (2012), the complex voltage signal in the V channel in the velocity domain is calculated from:

$$V_V(u) = \sqrt{S_{VV}(u) \ln g^{[1]}} e^{i\theta^{[1]}} \quad (2)$$

where $g^{[1]}$ and $\theta^{[1]}$ are independent, identically distributed random variables with uniform distribution between 0 and 1 and between $-\pi$ and π, respectively. This process is repeated iteratively to generate independent stochastic realizations of the same spectrum. Similarly, for the H channel in the velocity domain:

$$V_H(u) = \sqrt{sZ_{DR}(u)}\,[s\rho_{hv}(u) V_V^{[1]}(u) + \sqrt{1 - s\rho_{hv}^2(u)} V_V^{[2]}(u)] e^{i\delta_{hv}(u)} \qquad (3)$$

where the spectral variables $s\rho_{hv}$, $s\delta_{hv}$ and sZ_{DR} are calculated for each velocity bin, and hold the prefix s in the notation to differentiate them from the commonly used integral polarimetric variables. $V_V^{[2]}$ is generated according to (2) with the same model spectrum S_{VV}, but with a second independent sequence of random numbers ($u^{[2]}$ and $\theta^{[2]}$). This process is repeated for each velocity bin for the total of the FFT spectral points within the Nyquist interval. The inverse Fourier transform of V_V and V_H represent simulated time series of complex signals for the V and H channels. For the implementation of white noise, an approach similar to Eq. (2) is used:

$$N_V(u) = \sqrt{-\aleph_V \ln g^{[3]}} e^{i\theta^{[3]}} \quad \text{and} \quad N_H(u) = \sqrt{-\aleph_H \ln g^{[4]}} e^{i\theta^{[4]}} \quad (4)$$

where $\aleph_V$ and $\aleph_H$ are the noise level values for the V and H channel corresponding to the prescribed values of signal-to-noise ratios (SNR), and $u^{[3]}$, $\theta^{[3]}$, $u^{[4]}$, $\theta^{[4]}$ are again generated independently. The complex numbers that represent the simulation of the noisy I and Qs in the frequency domain for the V and H channels are calculated from:

$$S_V(u) = V_V(u) + N_V(u) \quad \text{and} \quad S_H(u) = V_H(u) + N_H(u) \quad (5)$$

Figures and more information on these processes is found at Theoretical Overview.

# In[Turbulence Implementation]

The turbulence is implemented by convolving the Doppler spectra with a turbulence term Sair (6), that accounts for the turbulent motions within the atmosphere.

$$S_{air}(u) = \frac{1}{\sqrt{2\pi}\sigma_t} e^{-\frac{u^2}{2\sigma_t^2}} \quad (6)$$

$$S_{VV}^{turb} = \int_{-\infty}^{\infty} S_{VV}(u-\xi) S_{air}(\xi) d\xi \quad (7)$$

where ξ is the convolution variable and Sair accounts for the turbulent motions within the atmosphere. Atmospheric turbulence causes random fluctuations in the velocity of hydrometeors, thus broadening the Doppler spectrum. The convolution is performed for the spectrum in the horizontal and vertical channels as well as for the HV cross spectrum.

# In[Calculate the spectral polarimetric variables]

In this final processing block, the broadened sZ_{DR} is computed as the ratio of $S^{turb}{}_{HH}(v)$ to $S^{turb}{}_{VV}(v)$ whereas the turbulent-broadened parameters $\rho^{turb}{}_{HV}$ and $\delta^{turb}{}_{HV}$ are then calculated respectively as the amplitude and the phase of the variable:

$$\rho_{hv}(u) e^{i\delta_{hv}(u)} = \frac{<S_H(u) S_V^*(u)>}{\sqrt{<|S_H(u)|^2><|S_V(u)|^2>}} \quad (8)$$

# In[Plot the polarimetric variables]

This block is optional. The figures of the spectral polarimetric variables are produced for the ideal situation of no noise and no turbulence, for the case of turbulence existence and for the case of noise and turbulence existence. The user is advised to see the Example demonstration before implementing the core code. More information on the gamma fit of the drop size distribution, and on the second method of turbulence implementation, will be released in the same link after the publication of the corresponding paper.