README

Callum Dewsnap
Victoria Pinnegar

1 Atmosphere GUI

1.1 Installation

This repository makes use of Python3. Modules Required include:

- numpy
- scipy
- miepython
- sys
- tkinter
- matplotlib
- seaborn

The tkinter package is a standard Python interface and should be included in your python installation. All other modules can be easily installed using pip or conda, for example by entering `pip intall numpy` or `conda intall numpy` into the console/kernel (where numpy is replaced with any of the above models). We recommend installing Anaconda and using the Spyder environment to run the Python code.

1.2 Files

1.2.1 `generateAtmosphere.py`

Contains function `RandomBackscatter` with inputs (in order)

- Alt – The maximum altitude of the atmosphere
- bins – The number of height bins.
- mean_of_scale – The mean scaling parameter of the atmosphere.
- std_of_scale – The standard deviation for the scaling parameter of the atmosphere distribution. Models the deviation in the initial value as well as the shift per hour.
- mean_of_std – The mean value of the standard deviation of the atmosphere distribution.
- std_of_std – The standard deviation of the mean standard deviation value defined above. Models the deviation in the initial value as well as the shift per hour.
- mean_of_mean – The mean height value of the initial distribution.
- std_of_mean – The standard deviation of the mean height value. Models the deviation in the initial value as well as the shift per hour.

`RandomBackscatter` generates a model atmosphere by creating a distribution of $P/T = nk$ which randomly varies over time. Here, P is the pressure, T is temperature, n is the number density, and k is the Boltzmann constant. The initial distribution is a Gaussian which varies every hour for 24 hours. This variation is calculated by shifting each of the mean, standard deviation, and scaling factor by a random amount determined through another normal distribution. The mean height of the distribution is limited to be higher than 800 metres and the standard deviation of the height is constrained to be greater than 150 metres. Each minute between the randomly determined hours are calculated by performing a two-dimensional linear interpolation.

The primary use of this function is to be called in `GooeyAtmosphere.py`.

1.2.2 `LidarModel.py`

`MieBackscatter` utilizes the concepts of Mie scattering to determine backscatter for the remote sensing instruments. Check LidarRadar_Presentation on slide 7 for Mie Equation. The Q_back is the back-scattering efficiency, derived from Maxwell's equations and solved using a Bernoulli. Time bins is a constant for 1 minute bins over 24 hours. The max height is variable. The random atmosphere comes from `RandomBackscatter`. The refractive index is coded for each possible scenario. For $\frac{P}{T}$ under 200, the refractive index used is that of air, (approx. equal to 1.000273). Depending on the calling parameter of Cloud vs. Aerosol, the refractive index changes. Aerosol: $n_{lidar} = 1.00044776$ $n_{radar} = 1.084210372$. Cloud: $n_{lidar} = 1.33$ $n_{radar} = 5 + 2.5j$. Wavelength varies based off of Radar(8.6 m) and Lidar (532 nm).

`Lidar` and `Radar` are their individual equations as explained in LidarRadar_Presentation.

`RadBackscatter` works in the same way as `MieBackscatter` except it utilizes Rayleigh scattering as discussed in the power point for all variation in $\frac{P}{T}$. The refractive index utilizes the same variation as `MieBackscatter`.

`ReflectivitytoBackscatter` and `BackscattertoReflectivity` convert the backscatter to the Reflectivity and back. The reflectivity is traditionally used for Radar studies and is in the units of dBz. VEGA-BetaMol(time bins, height bins, random atmosphere, wavelength)

1.2.3 `GooeyAtmosphere.py`

`GooeyAtmosphere.py` is the code containing the GUI. This code is a script and *not* a function, i.e., no input is required and the GUI can be started by simply running the script. Usage tips are given at the end of this section.

Once the GUI opens, the user is presented with 3 sets of options. These options are: Lidar parameters, radar parameters, and atmosphere parameters. Below these three parameters is a button labelled "Plot" and three empty plots. The plot calculates the backscatter for both the lidar and radar, as well as the corresponding colour ratio and plots the results in the three figures. The parameters in this calculation are taken from the values in the three above-mentioned parameter selections. Selecting one of these options allows the user to vary the parameters used in the calculation. Changes to parameter values are saved while alternating between tabs.

Currently, there are no lidar parameters which can be varied. In the radar tab, the user can toggle between modelling Rayleigh and Mie scattering for the radar backscattering calculation.

The first atmosphere parameter which can be changed is the cloud type. The user can alternate between water and aerosol (CH_4) clouds. This choice accounts for a change in the random atmosphere scaling, the index of refraction, and the particle radius distribution. The user can alter the maximum height of the atmosphere. The initial distribution and time variation of the random atmosphere can be altered.

The bottom parameter given in the Atmosphere Properties tab is the "Create New Atmosphere" toggle. When this is toggled on, a new random atmosphere is created using the values selected above. Otherwise, the previously generated atmosphere is used, regardless of any new atmosphere parameters entered. Note that if "Create New Atmosphere" is toggled on, reusing the same atmosphere parameters will results in a different atmosphere due to the randomness of the variation in time.

Usage tips: When the GUI is in the process of plotting, the calculation can take some time. While calculating, `GooeyAtmosphere.py` may say that it is not responding. It is responding and will complete; do not close the GUI even if it prompts you to do so. The console/kernel should update the user on the progress of the calculation. If you believe the program *actually is* crashing, check the kernel first. If the user enters values which are not valid (e.g. leads to divide-by-zero or entered a string for a number), the calculation will be aborted and the console/kernel will announce an error occurred. If errors occur continuously, try restarting the GUI.

1.3 Work Distribution

`generateAtmosphere.py` primarily written by Callum with help from Victoria.
`LidarModel.py` primarily written by Victoria with help from Callum.
`GooeyAtmosphere.py` primarily written by Callum.
LidarRadar_Presentation.pptx slides 1 through 9 were made by Victoria, slides 10 through 14 were made by Callum.
The README sections were written by the person who did the majority of the work for the respective section.