**distance\_label\_generator.m**

Call the **distance\_label** function and adjust all the input data for it:

* ***name***: Folder where the data is saved.
* ***N***: Number of receiving antennas in the system.
* ***antenna\_type***: Type of receiving antennas in the system.
* ***PtdBm***: Transmitting antenna power.
* ***frequency***: Transmission frequency.
* ***p\_matrix***: Array of that contains all the distances for which the received powers in each antenna will be calculated.
  + : Maximum distance. Integer number.
  + : Minimum distance. Integer number.
  + : How many iterations is wanted for each distance.

**distance\_label.m**

File containing the **distance\_label** function. The **distance\_label** function calculates the powers received at the ***N-***th antenna for all possible angles that the transmitter and the Nth receiving antenna can form for all distances contained in the p\_matrix matrix. Figure 1 is a flow chart of the **distance\_label** function.

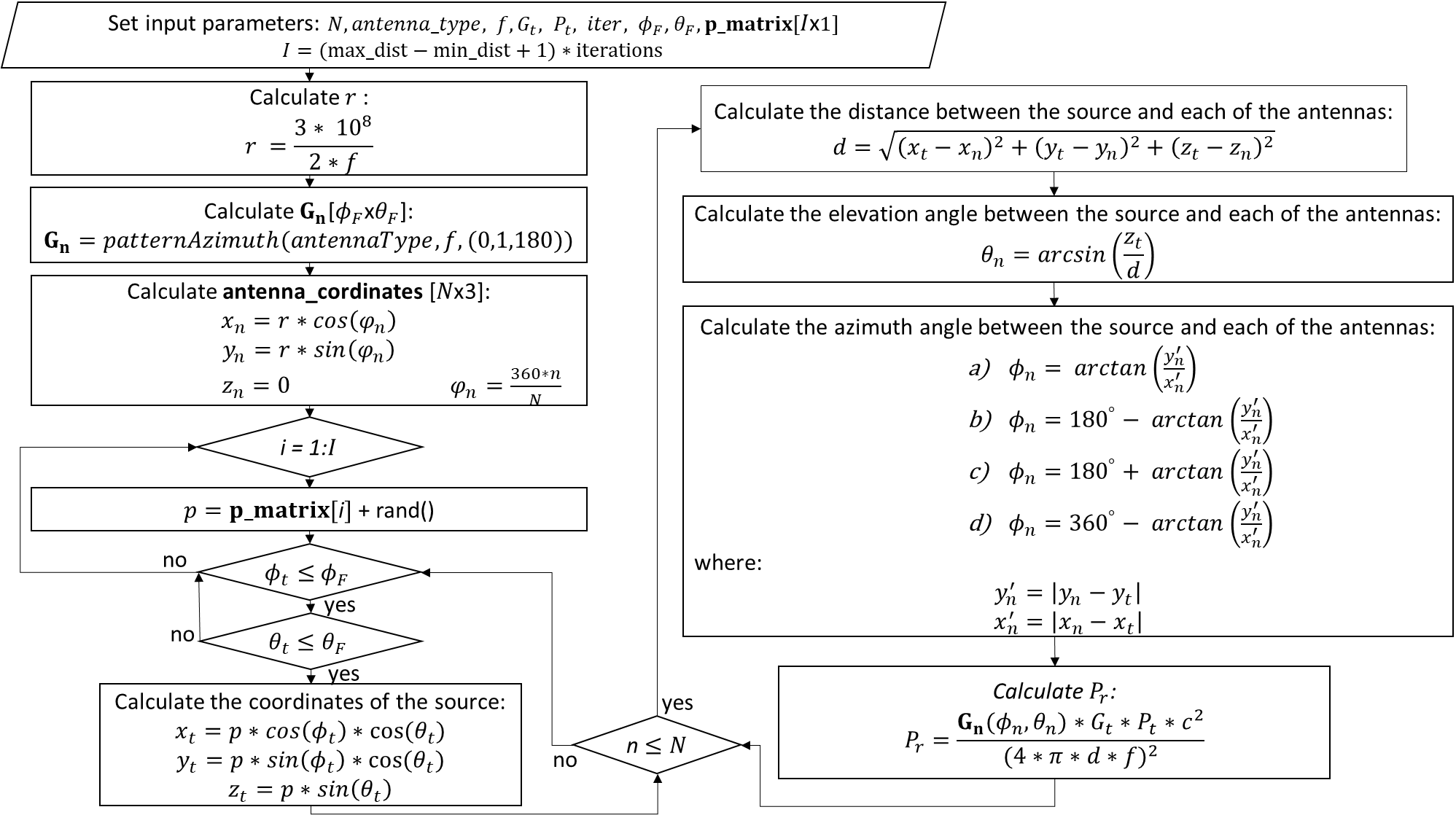


Figure 1 Flow chart of the **distance\_label** function.

**main.py**

Figure 2 shows a brief summary of the **main.py** file in flowchart form.

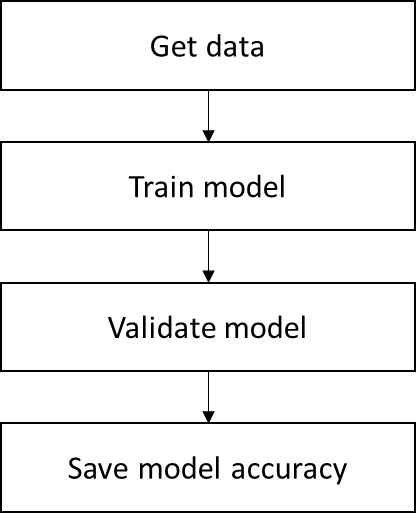
****

Figure 2 Flow chart of the **main.py**.

Get data

First, the data is obtained to train and validate the *Machine Learning* (ML) models with the use of the **HandleData** class contained in the **get\_data** file. When the **HandleData** class is instantiated, the path of the files where the data is found (***folder***) is passed as a parameter. In addition, the class is instantiated with certain parameters to choose which files you want to train and validate the ML models. These parameters are:

* ***phase\_files***: Point from which file you want to take the data, giving jumps equal to the specified number.
* ***until***: Up to which file you want to select (up to what distance you want to select).
* ***samples\_by\_distance***: How many files of the same distance do you want to take. How many files are to be taken from the file pointed to by the ***phase\_files*** pointer.

For example, Table 1 shows a hypothetical database with 21 files for distances from 10 to 16 meters, and for each distance there are three files. In the first column are the names of the files and in the second the distances with which the powers that are stored in said files were calculated. Assuming that you want to train a model with distances of 10 to 14 meters, instead of 16 meters, you have to set the parameter . If you do not want to take distances 11 and 13 meters, set the parameter , and if for the rest of the distances you want two files, set the parameter .

|  |  |
| --- | --- |
| Iter\_1.m | p = 10 + rand |
| Iter\_2.m | p = 10 + rand |
| Iter\_3.m | p = 10 + rand |
| Iter\_4.m | p = 11 + rand |
| Iter\_5.m | p = 11 + rand |
| Iter\_6.m | p = 11 + rand |
| Iter\_7.m | p = 12 + rand |
| Iter\_8.m | p = 12 + rand |
| Iter\_9.m | p = 12 + rand |
| Iter\_10.m | p = 13 + rand |
| Iter\_11.m | p = 13 + rand |
| Iter\_12.m | p = 13 + rand |
| Iter\_13.m | p = 14 + rand |
| Iter\_14.m | p = 14 + rand |
| Iter\_15.m | p = 14 + rand |
| Iter\_16.m | p = 15 + rand |
| Iter\_17.m | p = 15 + rand |
| Iter\_18.m | p = 15 + rand |
| Iter\_19.m | p = 16 + rand |
| Iter\_20.m | p = 16 + rand |
| Iter\_21.m | p = 16 + rand |

Train and validate model

Then with the **HandleModel** class from the **model.py** file the model is trained and validated.

Save model accuracy

And then with the **logs** class in the **save\_logs.py** file, the accuracy of the model is saved in the files.