2025 知能情報基礎演習
4-2 無線通信における降雨減衰特性のモデリング

提出者氏名: 知念 拓弥
提出者学籍番号: 245745J
担当教員名: 宮里 智樹

提出日: 2025 年 12 月 22 日
提出期限日: 2025 年 12 月 23 日
実験日: 2025 年 12 月 20 日

目次

1　1 週目

1.1　課題 1

mattermost から各データをダウンロードし、中身を確認した。
RxData のファイルツリーは以下のようになっていた。

```
RxData
├── 200906
│   ├── 20090601
│   │   ├── 192.168.100.11_csv.log
│   │   └── 192.168.100.9_csv.log
│   ├── 20090602
│   │   ├── 192.168.100.11_csv.log
│   │   └── 192.168.100.9_csv.log
│   ├── 20090603
│   │   ├── 192.168.100.11_csv.log
│   │   └── 192.168.100.9_csv.log
...
    ├── 20091229
    │   ├── 192.168.100.11_csv.log
    │   └── 192.168.100.9_csv.log
    ├── 20091230
    │   ├── 192.168.100.11_csv.log
    │   └── 192.168.100.9_csv.log
    └── 20091231
        ├── 192.168.100.11_csv.log
        └── 192.168.100.9_csv.log
```

RainData のファイルツリーは以下のようになっていた。

```
RainData
├─── 20090601
│    └─── 20090601_rain.csv
├─── 20090602
│    └─── 20090602_rain.csv
├─── 20090603
│    └─── 20090603_rain.csv
...
├─── 20091229
│    └─── 20091229_rain.csv
├─── 20091230
│    └─── 20091230_rain.csv
└─── 20091231
     └─── 20091231_rain.csv
```

1.2 課題 2

2 秒毎に記録されている RainData の測定値と、1 秒毎に記録されている RxData の測定値をいずれも 10 秒おきのデータに変換した。

前処理済みのデータは元のディレクトリ構成を保ったまま、RainData_Processed, RxData_Processed というディレクトリに保存した。

言語は Python を使用し、データの欠損値は直前のデータで補完した。

Listing1 に RxData の前処理プログラム、Listing2

Listing 1: RxData の前処理プログラム

```
1  import pandas as pd
2  import os
3  import glob
4  import numpy as np
5  import datetime
6  import tqdm
7
8  INPUT_PATH = os.path.join("RxData", "*", "*", "*_csv.log")
9  data_path_list = glob.glob(INPUT_PATH)
10 cols = ["time", "1803_RX_LEVEL"]
11
12 def check_type(x):
13     try:
14         x = float(x)
15     except (ValueError, TypeError):
```

```
        try:
            x = datetime.datetime.strptime(x, '%H:%M:%S')
        except (ValueError, TypeError):
            x = x
    return type(x)

for path in tqdm.tqdm(data_path_list):
    date = path.split("/")[2]
    date = f"{date[:4]}-{date[4:6]}-{date[6:]}"
    try:
        df = pd.read_csv(path, header=None, skiprows=2, usecols=[0,1], names=cols)
    except UnicodeDecodeError:
        encodings = ['utf-8', 'cp932', 'latin-1']
        for enc in encodings:
            try:
                df = pd.read_csv(path, header=None, skiprows=2, usecols=[0,1], names
                    =cols, encoding=enc)
                break
            except UnicodeDecodeError:
                continue

    df = df.mask(df.map(check_type) == str, np.nan)
    df = df.dropna()

    df["time"] = date + "␣" + df["time"].astype(str)
    df["time"] = pd.to_datetime(df["time"], format='%Y-%m-%d␣%H:%M:%S')
    df["1803_RX_LEVEL"] = pd.to_numeric(df["1803_RX_LEVEL"])
    df = df.set_index("time")

    date_range = pd.date_range(start=f"{date}␣00:00:00", end=f"{date}␣23:59:59",
        freq="s")
    df_new = pd.DataFrame(index=date_range, columns=df.columns)
    df_new = df_new.infer_objects(copy=False)

    df_new.loc[df.index, :] = df
    df_new = df_new.ffill()
    df_new = df_new.bfill()

    date_range = pd.date_range(start=f"{date}␣00:00:00", end=f"{date}␣23:59:59",
        freq="10s")
    df_new = df_new.loc[date_range, :]

    path = path.split("/")
    output_path = os.path.join("RxData_Processed", path[1], path[2])
    os.makedirs(output_path, exist_ok=True)
    df_new.to_csv(output_path + f"/{path[3]}", index_label="datetime")
```

Listing 2: RainData の前処理プログラム

```
import pandas as pd
import os
import glob
import datetime
```

```
import tqdm

INPUT_PATH = os.path.join("RainData", "*", "*.csv")
data_path_list = glob.glob(INPUT_PATH)
cols = ["time", "rain"]

def check_type(x):
    try:
        x = float(x)
    except (ValueError, TypeError):
        try:
            x = datetime.datetime.strptime(x, '%H:%M:%S')
        except (ValueError, TypeError):
            x = x
    return type(x)

for path in tqdm.tqdm(data_path_list):
    date = path.split("/")[1]
    date = f"{date[:4]}-{date[4:6]}-{date[6:]}"
    try:
        df = pd.read_csv(path, header=None, skiprows=2, names=cols)
    except UnicodeDecodeError:
        encodings = ['utf-8', 'cp932', 'latin-1']
        for enc in encodings:
            try:
                df = pd.read_csv(path, header=None, skiprows=2, names=cols, encoding
                    =enc)
                break
            except UnicodeDecodeError:
                continue

    df["time"] = df["time"].apply(lambda x: x.replace("/","-"))
    # df = df["time"].astype(str).str.replace("/", "-", regex=False)
    df["time"] = pd.to_datetime(df["time"], errors='coerce')
    df["rain"] = pd.to_numeric(df["rain"], errors='coerce')

    df = df.dropna(subset=["rain"])
    df = df.dropna(subset=["time"])
    df = df.set_index("time")

    date_range = pd.date_range(start=f"{date}␣00:00:00", end=f"{date}␣23:59:59",
        freq="10s")
    df_new = pd.DataFrame(index=date_range, columns=df.columns)
    df_new = df_new.infer_objects(copy=False)

    df_new.loc[df.index, :] = df
    df_new = df_new.ffill().bfill().fillna(0)

    path = path.split("/")
    output_path = os.path.join("RainData_Processed", path[1])
    os.makedirs(output_path, exist_ok=True)
    df_new.to_csv(output_path + f"/{path[2]}", index_label="datetime")
```

2　2 週目

2.1　手順 1

RainData_Processed のデータを 1 分間降雨強度に変換した。
計算式は 1 分間降雨強度を y、雨粒のカウントを x としたとき、以下の式を用いた。

$$y = x \times 0.0083333 \times 60$$

計算に使用したプログラムを Listing3 に示す。

Listing 3: 1 分間降雨強度への変換プログラム

```
import pandas as pd
import os
import glob
import tqdm

INPUT_PATH = os.path.join("RainData_Processed", "*", "*.csv")
data_path_list = glob.glob(INPUT_PATH)
cols = ["time", "rain"]

for path in tqdm.tqdm(data_path_list):
    df = pd.read_csv(path, header=None, skiprows=2, names=cols)

    df["time"] = pd.to_datetime(df["time"])
    df["rain"] = pd.to_numeric(df["rain"])

    df["rain"] = df["rain"].apply(lambda x: x * 0.0083333 * 60)

    df = df.set_index("time")
    path = path.split("/")
    output_path = os.path.join("RainData_Processed_fix", path[1])
    os.makedirs(output_path, exist_ok=True)
    df.to_csv(output_path + f"/{path[2]}", index_label="datetime")
```

2.2　手順 2

RxData_Processed に含まれる 18GHz のデータを物理量に変換する計算を行った。
計算式は実際の物理量を P、受信電界元の値を R としたとき、以下の式を用いた。

$$P = \begin{cases} R \div 2 - 121 & (R \geq 0), \\ (R + 256) \div 2 - 121 & (R < 0). \end{cases}$$

計算に使用したプログラムを Listing4 に示す。

Listing 4: 受信電界値を物理量に対応させるプログラム

```
import pandas as pd
import os
import glob
import tqdm

INPUT_PATH = os.path.join("RxData_Processed", "*", "*", "192.168.100.9_csv.log")
data_path_list = glob.glob(INPUT_PATH)
cols = ["time", "1803_RX_LEVEL"]

def fix_data(x):
    if x < 0:
        x += 256
        x = (x / 2) - 256
    else:
        x = (x / 2) - 256
    return x

for path in tqdm.tqdm(data_path_list):
    df = pd.read_csv(path, header=None, skiprows=2, names=cols)

    df["time"] = pd.to_datetime(df["time"])
    df["1803_RX_LEVEL"] = pd.to_numeric(df["1803_RX_LEVEL"])

    df["1803_RX_LEVEL"] = df["1803_RX_LEVEL"].apply(fix_data)

    df = df.set_index("time")

    path = path.split("/")
    output_path = os.path.join("RxData_Processed_fix", path[1], path[2])
    os.makedirs(output_path, exist_ok=True)
    df.to_csv(output_path + f"/{path[3]}", index_label="datetime")
```

2.3 手順 3

頻度分布をを求めるに当たって各データの最大値・最小値を調べ、基準の数値を設定した。

1 時間降雨強度の最大値は 143.499426[mm/h]、小数第一位で切り上げて 144[mm/h]、最小値は 0.0[mm/h] であった。刻み幅は 3[mm/h] とした。

18GHz の受信電界強度の最大値は-173.0[dB]、最小値は-229.5[dB]、小数第一位で切り上げて-230[dB] とした。刻み幅は-3[dB] とした。

26GHz の受信電界強度の最大値は-166.5[dB] 小数第一位で切り上げて-167[dB] とし、最小値は-220.0[dB] であった。刻み幅は-3[dB] とした。

2.4 手順 4

前処理、単位変換が済んだ RainData から、3[mm/h] 刻みの 1 時間降雨強度の頻度分布を求めた。

同様に、RxData から 18GHz および 26GHz の受信電界強度の頻度分布をそれぞれ-3[dB] 刻みで求めた。

2.5 手順 5

頻度分布を足し合わせ、累積分布を作成した。

RainData の 1 時間降雨強度の累積分布を Listing5、18GHz の受信電界強度の累積分布を Listing6、26GHz の受信電界強度の累積分布を Listing7 に示す。

頻度分布をあらかじめ pandas の DataFrame に格納し、cumsum 関数を用いて累積和を計算することで累積分布を求めた。

累積和賀計算された df[”count”] の最終行の値を総サンプル数として、各行の累積和を総サンプル数で割ることで累積分布の割合を求めた。

Listing 5: RainData の累積時間分布作成プログラム

```
import pandas as pd
import os
import glob
import tqdm
import matplotlib.pyplot as plt

INPUT_PATH = os.path.join("RainData_Processed_fix", "*", "*.csv")
data_path_list = glob.glob(INPUT_PATH)
cols = ["time", "rain"]
freq = 3

df_list = []

for path in tqdm.tqdm(data_path_list):
    df = pd.read_csv(path, header=None, skiprows=2, names=cols)
    df[cols[0]] = pd.to_datetime(df[cols[0]])
    df[cols[1]] = pd.to_numeric(df[cols[1]])
    df_list.append(df)

df1 = pd.concat(df_list)
print(f"最大値:␣{df1['rain'].max()}")
print(f"最小値:␣{df1['rain'].min()}")

level = []
count = []
max_rain = int(df1[cols[1]].max()) + 1
for i in range(0, max_rain, freq):
    level.append(i)
```

```
    count.append(((df1[cols[1]] >= i) & (df1[cols[1]] < i + freq)).sum())
df = pd.DataFrame({"rx_level": level[::-1], "count": count[::-1]})
df["count"] = df["count"].cumsum()
total = df["count"].iloc[-1]
df["ratio"] = df["count"] / total * 100
df["ratio"] = df["ratio"].replace(0, 1e-6)

plt.rcParams["font.family"] = "Hiragino␣Sans"
fig, ax = plt.subplots()
ax.plot(df["ratio"], df["rx_level"], label="降雨強度(mm/h)", marker=".")
ax.set_xscale("log")
ax.set_xlabel("累積時間率(%)")
ax.set_ylabel("降雨強度(mm/h)")
ax.set_title("降雨強度累積時間分布(琉大観測:␣2009/06,␣2009/10␣〜␣2009/12)")
ax.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, loc: "{:,}".format(float(x
    ))))
plt.legend()
plt.savefig("rain_dist.png")
# plt.show()
```

Listing 6: 18GHz の RxData の累積時間分布作成プログラム

```
import pandas as pd
import os
import glob
import tqdm
import matplotlib.pyplot as plt

INPUT_PATH = os.path.join("RxData_Processed_fix", "*", "*", "*.log")
data_path_list = glob.glob(INPUT_PATH)
cols = ["time", "RX_LEVEL_18"]
freq = -3

df_list = []

for path in tqdm.tqdm(data_path_list):
    df = pd.read_csv(path, header=None, skiprows=2, names=cols)
    df[cols[0]] = pd.to_datetime(df[cols[0]])
    df[cols[1]] = pd.to_numeric(df[cols[1]])
    df_list.append(df)

df1 = pd.concat(df_list)
print(f"最大値:␣{df1['RX_LEVEL_18'].max()}")
print(f"最小値:␣{df1['RX_LEVEL_18'].min()}")

level = []
count = []
sum_sec = []
min_level = int(df1[cols[1]].min()) - 1
for i in range(0, min_level, freq):
    level.append(i)
    count.append(((df1[cols[1]] <= i) & (df1[cols[1]] > i + freq)).sum())
df = pd.DataFrame({"rx_level": level[::-1], "count": count[::-1]})
```

```
df["count"] = df["count"].cumsum()
total = df["count"].iloc[-1]
df["ratio"] = df["count"] / total * 100
df["ratio"] = df["ratio"].replace(0, 1e-6)

plt.rcParams["font.family"] = "Hiragino␣Sans"
fig, ax = plt.subplots()
ax.plot(df["ratio"], df["rx_level"], label="受信強度(dB)", marker=".")
ax.set_xscale("log")
ax.set_xlabel("累積時間率(%)")
ax.set_ylabel("受信強度(dB)")
ax.set_title("受信強度累積時間分布␣18GHz(琉大観測:␣2009/06,␣2009/10␣〜␣2009/12)")
ax.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, loc: "{:,}".format(float(x
    ))))
plt.legend()
plt.savefig("rx_dist_18.png")
# plt.show()
```

Listing 7: 26GHz の RxData の累積時間分布作成プログラム

```
import pandas as pd
import os
import glob
import tqdm
import matplotlib.pyplot as plt

INPUT_PATH = os.path.join("RxData_Processed", "*", "*", "192.168.100.11_csv.log")
data_path_list = glob.glob(INPUT_PATH)
cols = ["time", "1803_RX_LEVEL"]
freq = -3

df_list = []

for path in tqdm.tqdm(data_path_list):
    df = pd.read_csv(path, header=None, skiprows=2, names=cols)
    df[cols[0]] = pd.to_datetime(df[cols[0]])
    df[cols[1]] = pd.to_numeric(df[cols[1]])
    df_list.append(df)

df1 = pd.concat(df_list)
print(f"最大値:␣{df1['1803_RX_LEVEL'].max()}")
print(f"最小値:␣{df1['1803_RX_LEVEL'].min()}")

level = []
count = []
min_level = int(df1[cols[1]].min()) - 1
for i in range(0, min_level, freq):
    level.append(i)
    count.append(((df1[cols[1]] <= i) & (df1[cols[1]] > i + freq)).sum())
df = pd.DataFrame({"rx_level": level[::-1], "count": count[::-1]})
df["count"] = df["count"].cumsum()
total = df["count"].iloc[-1]
df["ratio"] = df["count"] / total * 100
```

```
34 df["ratio"] = df["ratio"].replace(0, 1e-6)
35
36 plt.rcParams["font.family"] = "Hiragino␣Sans"
37 fig, ax = plt.subplots()
38 ax.plot(df["ratio"], df["rx_level"], label="受信強度(dB)", marker=".")
39 ax.set_xscale("log")
40 ax.set_xlabel("累積時間率(%)")
41 ax.set_ylabel("受信強度(dB)")
42 ax.set_title("受信強度累積時間分布␣26GHz(琉大観測:␣2009/06,␣2009/10␣〜␣2009/12)")
43 ax.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, loc: "{:,}".format(float(x
       ))))
44 plt.legend()
45 plt.savefig("rx_dist_26.png")
46 # plt.show()
```

2.6 手順 6

matplotlib を用いて、グラフをプロットした。累積分布を求めるプログラムはグラフプロットのプログラムと一緒に記述した。

2.7 手順 7

横軸を時間からパーセント表記に変更した。各データの累積時間を総時間で割り、100 をかけることで割合を算出した。

2.8 手順 8

グラフの横軸を対数軸とし、片対数グラフとした。

降雨強度累積時間分布を図 1、18GHz の受信電界強度の累積時間分布を図 2、26GHz の受信電界強度の累積時間分布を図 3 に示す。

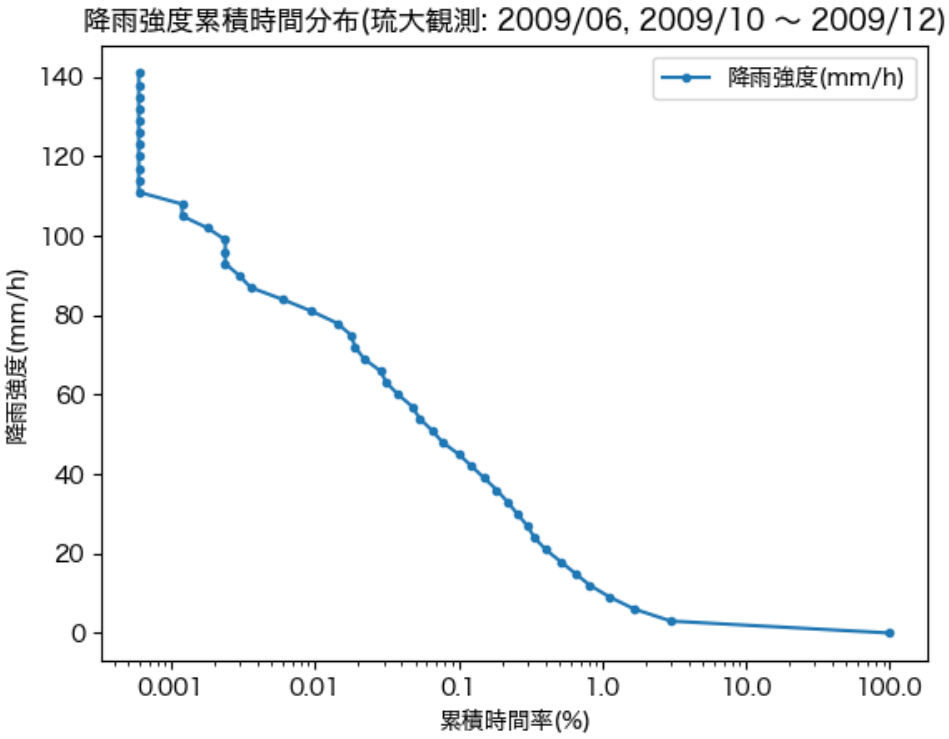

図 1: 1 時間降雨強度の累積時間分布

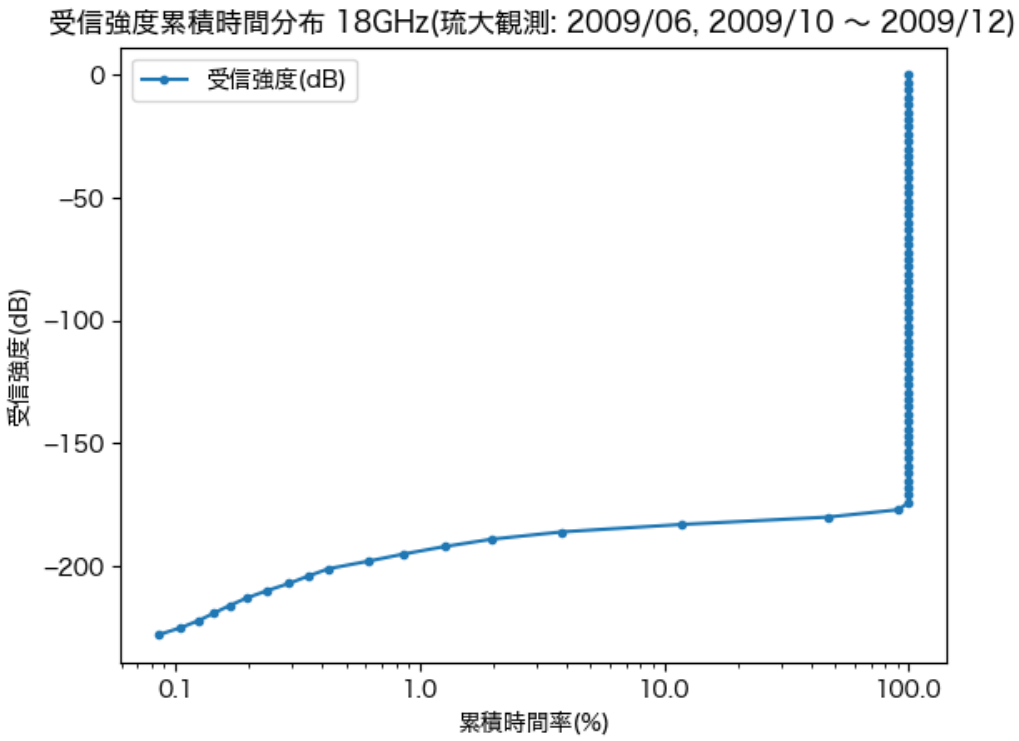

図 2: 18GHz 受信電界強度の累積時間分布

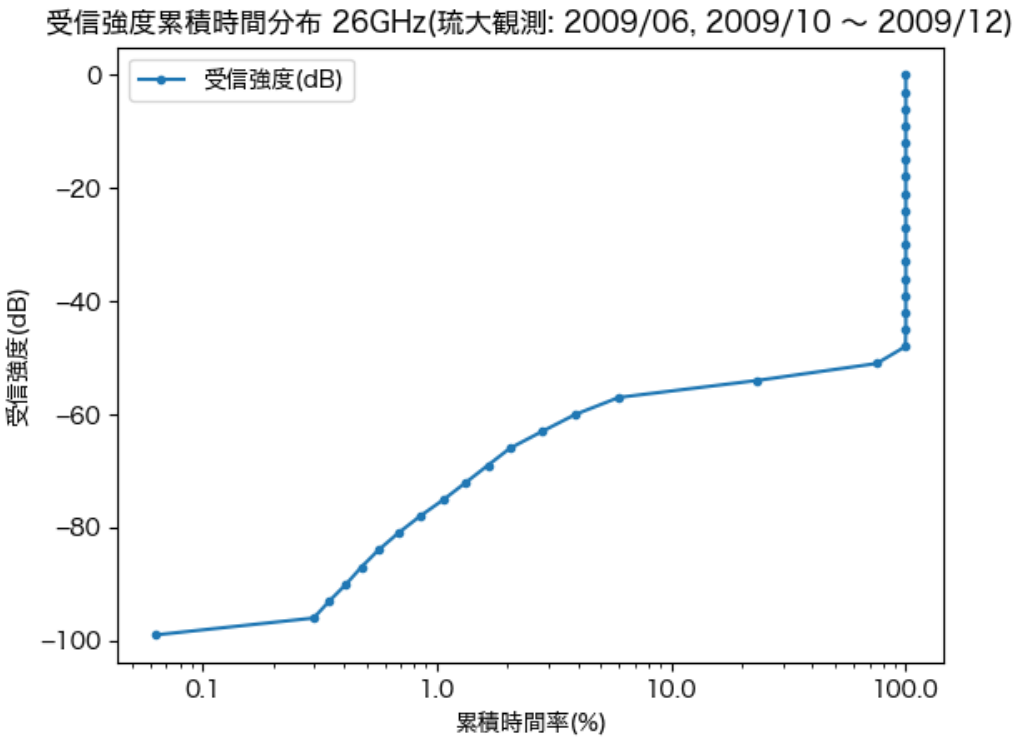

図 3: 26GHz 受信電界強度の累積時間分布