2023-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: 城間 颯
提出者学籍番号: 225719A
担当教員名: 宮里 智樹

提出日: 2023 年 12 月 13 日
提出期限日: 2023 年 12 月 19 日
実験日: 2023 年 11 月 5 日, 12 日

目次

1　1 週目

1.1　課題 1

mattermost からデータをダウンロードし、中身を確認した。

RainData のフォルダ構成を以下に示す。

```
RainData
├─── 20090601
│    └─── 20090601_rain.csv
├─── 20090602
│    └─── 20090602_rain.csv
├─── 20090603
│    └─── 20090603_rain.csv
├─── 20090604
│    └─── 20090604_rain.csv
...
├─── 20091230
│    └─── 20091230_rain.csv
└─── 20091231
└─── 20091231_rain.csv
```

RxData のフォルダ構成を以下に示す。

```
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
│    ...
│
├─── 200910
│    ├─── 20091001
│    │   ├─── 192.168.100.11_csv.log
│    │   └─── 192.168.100.9_csv.log
│    ...
│
├─── 200911
│    ├─── 20091101
│    │   ├─── 192.168.100.11_csv.log
│    │   └─── 192.168.100.9_csv.log
│    ...
│
└─── 200912
     ├─── 20091201
     │   ├─── 192.168.100.11_csv.log
     │   └─── 192.168.100.9_csv.log
     ...
```

1.2 課題 2

2 秒毎に記録されている RainData の測定値と、1 分毎に記録されている RxData の測定値を、10 秒毎のデータに作りかえた。

前処理済みのデータを、元データとフォルダ構成は同じにして、RxData_fix と、RainData_fix

という名前のフォルダに保存した。

RainData の前処理を行うスクリプトを、listings1 に示す。

RxData の前処理を行うスクリプトを、listings2 に示す。

スクリプトは、Python を用いて作成した。

データの欠損値は、直前のデータで置き換えた。直前のデータがない場合は、直後のデータで置き換えた。

Listing 1　RainData の前処理用スクリプト

```
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("week2","RainData","*","*.csv")
9  data_path_list = glob.glob(INPUT_PATH)
10 col_names = ["time", "rain"]
11
12 def typecheck(x):
13     '''数値または日付型に変換できるかできないかを判定する関数
14
15     :param x:
16     :return: 変換後のの型x
17     '''
18     try:
19         x = float(x)
20     except:
21         try:
22             x = datetime.datetime.strptime(x, "%Y-%m-%d␣%H:%M")
23         except:
24             x = x
25     return type(x)
26
27 for path in tqdm.tqdm(data_path_list):
28     date = path.split("/")[2]
29     date = f"{date[:4]}-{date[4:6]}-{date[6:]}␣"
30     try:
31         df = pd.read_csv(path, header=None, skiprows=2, names=col_names)
32     except:
33         df = pd.read_csv(path, header=None, skiprows=2, names=col_names,
               encoding="shift-jis")
34
35     df["time"] = df["time"].apply(lambda x: x.replace("/","-"))
36     df = df.mask(df.applymap(typecheck)==str, np.nan) # 型チェック後、になるものは
           strnp.で置き換えnan
37     df = df.dropna()
38
39     df["time"] = pd.to_datetime(df["time"])
40     df["rain"] = pd.to_numeric(df["rain"])
41
```

```
    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.loc[df.index, :] = df
    df_new = df_new.fillna(method='ffill')
    df_new = df_new.fillna(method='bfill')

    path = path.split("/")
    output_path = os.path.join(path[0], "RainData_fix", path[2])
    os.makedirs(output_path, exist_ok=True)
    df_new.to_csv(output_path + f"/{path[3]}", index_label="datetime")
```

Listing 2　RxData の前処理用スクリプト

```
import pandas as pd
import os
import glob
import numpy as np
import datetime
import tqdm

INPUT_PATH = os.path.join("week2","RxData","*","*","*.log")
data_path_list = glob.glob(INPUT_PATH)
col_names = ["time", "1803_RX_LEVEL"]

def typecheck(x):
    '''数値または日付型に変換できるかできないかを判定する関数

    :param x:
    :return: 変換後のの型x
    '''
    try:
        x = float(x)
    except:
        try:
            x = datetime.datetime.strptime(x, "%H:%M:%S")
        except:
            x = x
    return type(x)

for path in tqdm.tqdm(data_path_list):
    date = path.split("/")[3]
    date = f"{date[:4]}-{date[4:6]}-{date[6:]}␣"
    try:
        df = pd.read_csv(path, header=None, skiprows=2, usecols=[0, 1], names=
            col_names)
    except:
        df = pd.read_csv(path, header=None, skiprows=2, usecols=[0, 1], names=
            col_names, encoding="shift-jis")

```

```
    df = df.mask(df.applymap(typecheck)==str, np.nan) # 型チェック後、になるものは
        strnp.で置き換えnan
    df = df.dropna()

    if df["time"][0] == "00:00:00":
        is_zero_origin = True

    df["time"] = date + df["time"]
    df["time"] = pd.to_datetime(df["time"])
    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.loc[df.index, :] = df
    df_new = df_new.fillna(method='ffill')
    df_new = df_new.fillna(method='bfill')

    date_range = pd.date_range(start=f"{date}00:00:00", end=f"{date}23:59:59",
        freq="10s") # 秒ごとの時間のリスト
        10
    df_new = df_new.loc[date_range, :]

    path = path.split("/")
    output_path = os.path.join(path[0], "RxData_fix", path[2], path[3])
    os.makedirs(output_path, exist_ok=True)
    df_new.to_csv(output_path + f"/{path[4]}", index_label="datetime")
```

2　2 週目

2.1　手順 1

1 分間降雨強度を 1 時間 (60 分) 降雨強度に変換した。

変換後のデータは、元データとフォルダ構成は同じにして、RainData_unit_conv という名前のフォルダに保存した。

計算式は以下を用いた。雨粒の量を c とする。

$$S[mm/h] = c \times 0.0083333 \times 60$$

雨粒量から、1 時間 (60 分) 降雨強度に変換するスクリプトを listings3 に示す。

Listing 3　RainData の単位変換スクリプト

```
import pandas as pd
import os
import glob
import numpy as np
```

```
import datetime
import tqdm

INPUT_PATH = os.path.join("week2","RainData_fix","*","*.csv")
data_path_list = glob.glob(INPUT_PATH)
col_names = ["time", "rain"]

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

    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(path[0], "RainData_unit_conv", path[2])
    os.makedirs(output_path, exist_ok=True)
    df.to_csv(output_path + f"/{path[3]}", index_label="datetime")
```

2.2 手順 2

RxData の、18GHz の場合の、受信強度の生データを、物理量 $[dB]$ に変換した。

変換後のデータは、元データとフォルダ構成は同じにして、RxData_unit_conv という名前のフォルダに保存した。

計算式は以下を用いた。受信電界元の値を E とする。

$$P[dB] = \begin{cases} E \div 2 - 121 & \text{if } E \geq 0 \\ (E + 256) \div 2 - 121 & \text{if } E < 0 \end{cases}$$

RxData を単位変換するスクリプトを listings4 に示す。

Listing 4　RxData の単位変換スクリプト

```
import pandas as pd
import os
import glob
import numpy as np
import datetime
import tqdm

INPUT_PATH = os.path.join("week2","RxData_fix","*","*","192.168.100.9_csv.log")
data_path_list = glob.glob(INPUT_PATH)
col_names = ["time", "1803_RX_LEVEL"]

def unit_conv(x):
    if x < 0:
        x += 256
        x = (x / 2) - 256
```

```
16      else:
17          x = (x / 2) - 256
18      return x
19
20  for path in tqdm.tqdm(data_path_list):
21      df = pd.read_csv(path, header=None, skiprows=2, names=col_names)
22
23      df["time"] = pd.to_datetime(df["time"])
24      df["1803_RX_LEVEL"] = pd.to_numeric(df["1803_RX_LEVEL"])
25
26      df["1803_RX_LEVEL"] = df["1803_RX_LEVEL"].apply(unit_conv)
27
28      df = df.set_index("time")
29
30      path = path.split("/")
31      output_path = os.path.join(path[0], "RxData_unit_conv", path[2], path[3])
32      os.makedirs(output_path, exist_ok=True)
33      df.to_csv(output_path + f"/{path[4]}", index_label="datetime")
```

2.3 手順 3

頻度分布を求めるにあたって、基準となる数値を決めた。

1 時間降雨強度の最大値は、143.499426[mm/h]。最小値は、0.0[mm/h] であった。

したがって、RainData の、1 時間降雨強度の出現頻度を求める最大値は、144[mm/h]。最小値は、0[mm/h] とした。また、刻み間隔は 3[mm/h] とした。

18GHz の場合の受信強度の最大値は、-173.0[dB]。最小値は、-229.5[dB] であった。

したがって、RxData の、18GHz の場合の、受信強度の出現頻度を求める最大値は、0[dB]。最小値は、-230[dB] とした。また、刻み間隔は-3[dB] とした。

26GHz の場合の受信強度の最大値は、-1.0[dB]。最小値は、-99.0[dB] であった。

したがって、RxData の、26GHz の場合の、受信強度の出現頻度を求める最大値は、0[dB]。最小値は、-100[dB] とした。また、刻み間隔は-3[dB] とした。

2.4 手順 4

前処理と単位変換済みの RainData から、3[mm/h] の刻み間隔ごとに、当てはまるデータが何個あったのかを数え、頻度分布を作成した。

同様に、18GHz の場合と、26GHz の場合の、RxData でも、-3[dB] の刻み間隔ごとに、当てはまるデータが何個あったのかを数え、頻度分布を作成した。

2.5 手順 5

頻度分布を足し合わせて累積分布を作成した。

RainData の頻度分布を求めるプログラムを、listings5 に示す。

RxData の、18GHz の場合の、頻度分布を求めるプログラムを、listings6 に示す。

RxData の、26GHz の場合の、頻度分布を求めるプログラムを、listings7 に示す。

頻度分布を、pandas の DataFrame として、変数に保存していたため、pandas の cumsum() を用いて、累積和を計算した。

また、RainData の $0[mm/h] \sim 3[mm/h]$ および、RxData の $0[dB] \sim -3[dB]$ の区間のデータは、4 ヶ月の総時間数、1051000[10 秒] に置き換えた。

Listing 5　RainData のグラフ作成スクリプト

```
import pandas as pd
import os
import glob
import numpy as np
import datetime
import tqdm
import matplotlib.pyplot as plt
import japanize_matplotlib

INPUT_PATH = os.path.join("week2","RainData_unit_conv","*","*.csv")
data_path_list = glob.glob(INPUT_PATH)
col_names = ["time", "rain"]
freq = 3

df_list = []

for path in tqdm.tqdm(data_path_list):
    df = pd.read_csv(path, header=None, skiprows=2, names=col_names)
    df["time"] = pd.to_datetime(df["time"])
    df["rain"] = pd.to_numeric(df["rain"])

    df_list.append(df)

df = pd.concat(df_list)

print(f"最大値{df['rain'].max()}")
print(f"最小値{df['rain'].min()}")

rain = []
count = []
sum_sec = []
max_rain = int(df["rain"].max()) + 1
for i in range(0, max_rain, freq):
    rain.append(i)
    count.append(((df["rain"] >= i) & (df["rain"] < i+freq)).sum())
df = pd.DataFrame({"rain": rain[::-1], "count": count[::-1]})
df["count"] = df["count"].cumsum()

df.at[df.index[-1], "count"] = 1051000
df["ratio"] = df["count"] / 10510

print(df["ratio"].min())

```

```
fig, ax = plt.subplots()
ax.plot(df["ratio"], df["rain"], marker='.', label="降雨強度[mm/h]")
ax.set_xscale('log')
ax.set_title("降雨強度累積時間分布琉大観測(:␣2009/06,␣2009/10␣〜␣2009/12)")
ax.set_xlabel("累積時間率(%)")
ax.set_ylabel("降雨強度(mm/h)")
ax.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, loc: "{:,}".format(
    float(x))))
plt.legend()
plt.savefig("week2/rain_dist.png")
#plt.show()
```

Listing 6　RxData(18GHz) のグラフ作成スクリプト

```
import pandas as pd
import os
import glob
import numpy as np
import datetime
import tqdm
import matplotlib.pyplot as plt
import japanize_matplotlib

INPUT_PATH = os.path.join("week2","RxData_unit_conv","*","*","*.log")
data_path_list = glob.glob(INPUT_PATH)
col_names = ["time", "1803_RX_LEVEL"]
min_level = -230
freq = -3

df_list = []

for path in tqdm.tqdm(data_path_list):
    df = pd.read_csv(path, header=None, skiprows=2, names=col_names)
    df["time"] = pd.to_datetime(df["time"])
    df["1803_RX_LEVEL"] = pd.to_numeric(df["1803_RX_LEVEL"])

    df_list.append(df)

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

level = []
count = []
sum_sec = []
min_level = int(df["1803_RX_LEVEL"].min()) - 1
for i in range(0, min_level, freq):
    level.append(i)
    count.append(((df["1803_RX_LEVEL"] <= i) & (df["1803_RX_LEVEL"] > i+freq)).
        sum())
df = pd.DataFrame({"rx_level": level[::-1], "count": count[::-1]})
df["count"] = df["count"].cumsum()
df.at[df.index[-1], "count"] = 1051000
```

```
df["ratio"] = df["count"] / 10510

#df["rx_level"] *= -1

fig, ax = plt.subplots()
ax.plot(df["ratio"], df["rx_level"], marker='.', label="受信強度[dB]")
ax.set_xscale('log')
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.set_xlabel("累積時間率(%)")
ax.set_ylabel("降雨強度(mm/h)")
ax.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, loc: "{:,}".format(
    float(x))))
plt.legend()
plt.savefig("week2/rx_dist_18.png")
#plt.show()
```

Listing 7　RxData(26GHz) のグラフ作成スクリプト

```
import pandas as pd
import os
import glob
import numpy as np
import datetime
import tqdm
import matplotlib.pyplot as plt
import japanize_matplotlib

INPUT_PATH = os.path.join("week2","RxData_fix","*","*","192.168.100.11_csv.log"
    )
data_path_list = glob.glob(INPUT_PATH)
col_names = ["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=col_names)
    df["time"] = pd.to_datetime(df["time"])
    df["1803_RX_LEVEL"] = pd.to_numeric(df["1803_RX_LEVEL"])

    df_list.append(df)

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

level = []
count = []
sum_sec = []
min_level = int(df["1803_RX_LEVEL"].min()) -1
for i in range(0, min_level, freq):
    level.append(i)
    count.append(((df["1803_RX_LEVEL"] <= i) & (df["1803_RX_LEVEL"] > i+freq)).
```

```
        sum())
df = pd.DataFrame({"rx_level": level[::-1], "count": count[::-1]})
df["count"] = df["count"].cumsum()
df.at[df.index[-1], "count"] = 1051000
df["ratio"] = df["count"] / 10510

#df["rx_level"] *= -1

fig, ax = plt.subplots()
ax.plot(df["ratio"], df["rx_level"], marker='.', label="受信強度[dB]")
ax.set_xscale('log')
ax.set_title("受信強度累積時間分布␣26GHz琉大観測(:␣2009/06,␣2009/10␣〜␣2009/12)")
ax.set_xlabel("累積時間率(%)")
ax.set_ylabel("降雨強度(mm/h)")
ax.xaxis.set_major_formatter(plt.FuncFormatter(lambda x, loc: "{:,}".format(
    float(x))))
plt.legend()
plt.savefig("week2/rx_dist_26.png")
#plt.show()
```

2.5.1 手順 6

matplotlib を用いて、累積分布のグラフを作成した。

累積分布を求めるプログラムは、頻度分布を求めるプログラムと一緒に記述した。

2.6 手順 7

横軸を時間から、パーセンテージに直した。各データの累積時間 (単位: 10 秒) を、4 ヶ月の総時間数、1051000[10 秒] で割り、結果に 100 をかけて、パーセンテージを出した。

計算式を以下に示す。各データの累積時間 (単位: 10 秒) を t とする。

$$r[\%] = \frac{t}{1051000} \times 100$$

2.7 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 に示す。

18GHz の場合の、受信強度累積時間分布を図 2 に示す。

26GHz の場合の、受信強度累積時間分布を図 3 に示す。

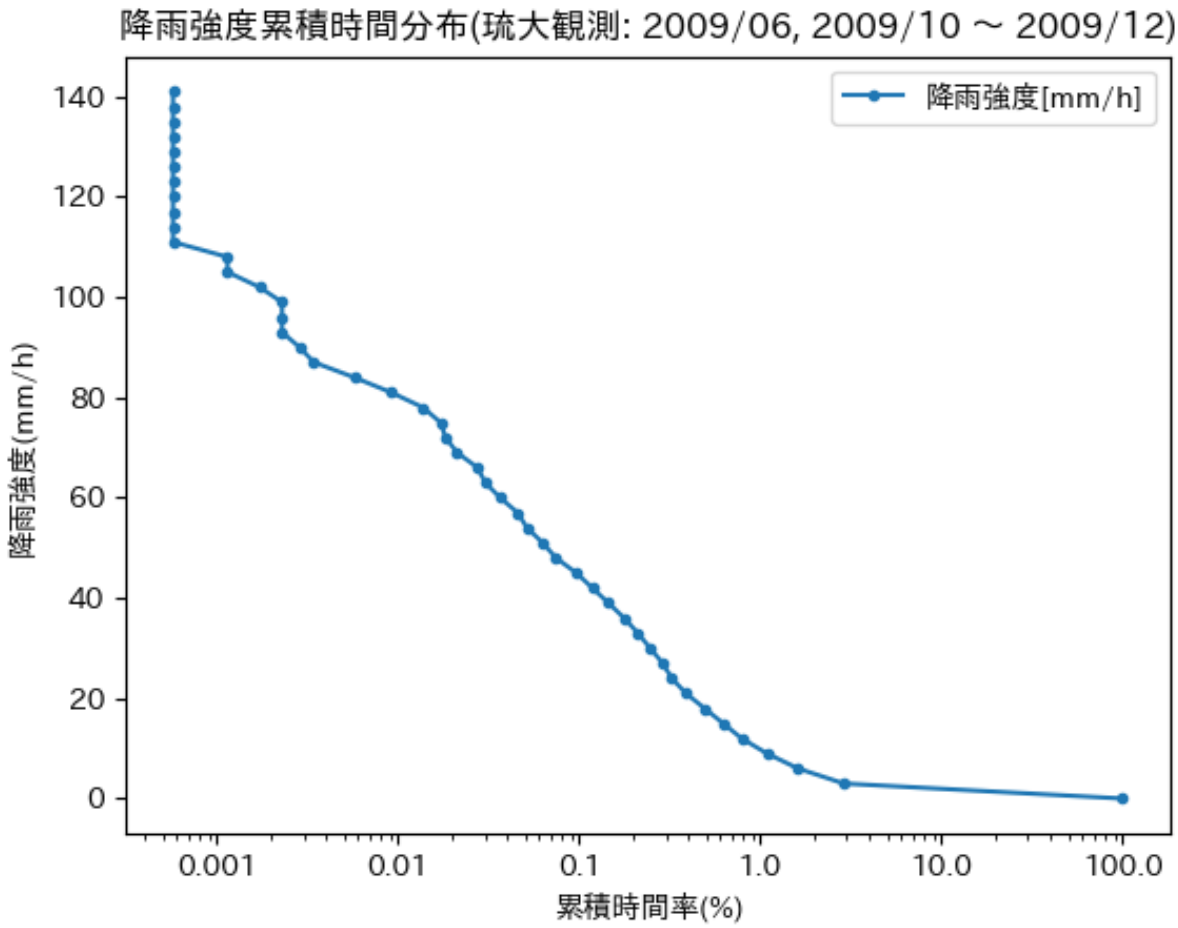

図 1　降雨強度累積時間分布

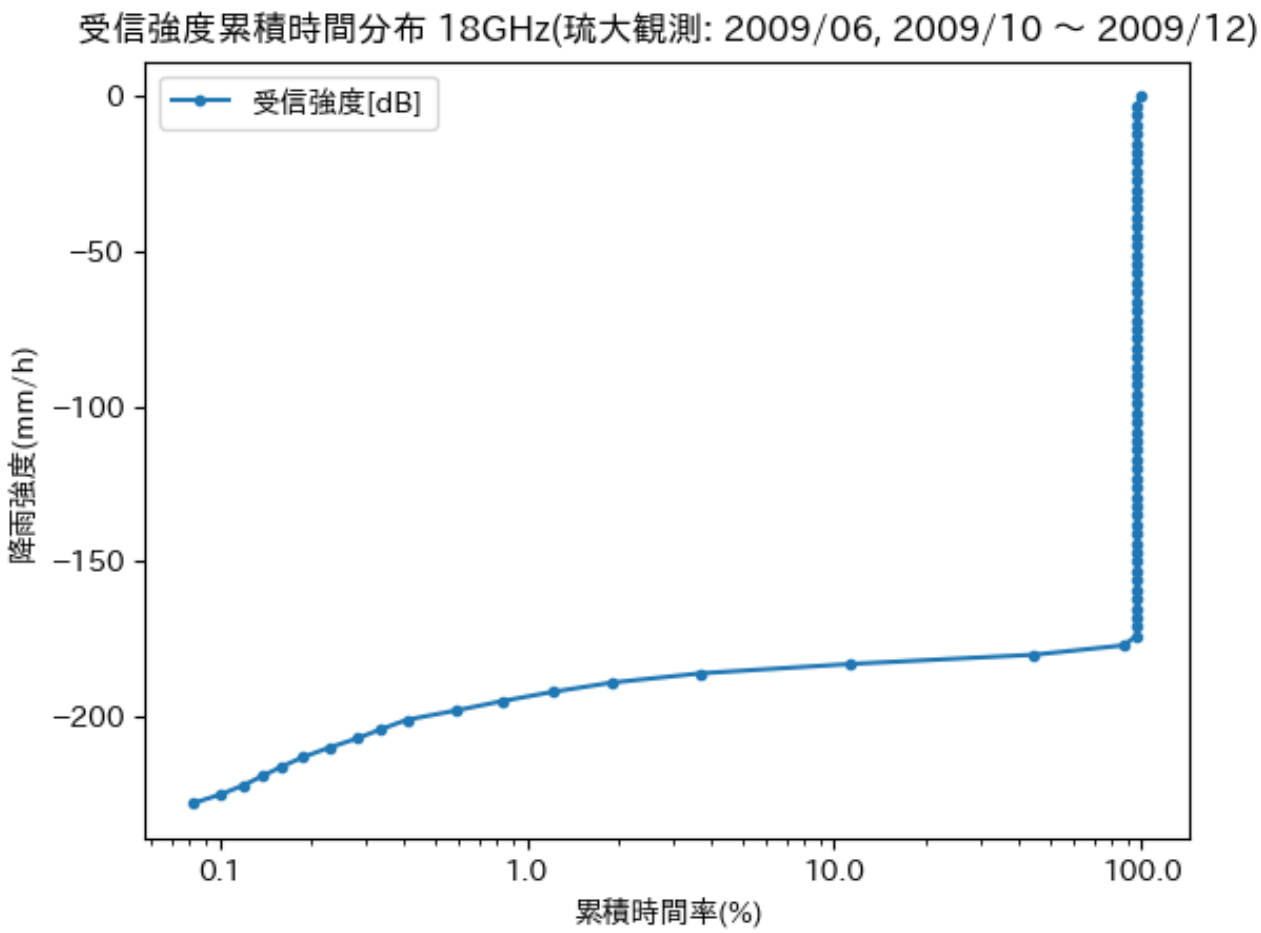

図 2　受信強度累積時間分布 (18GHz)

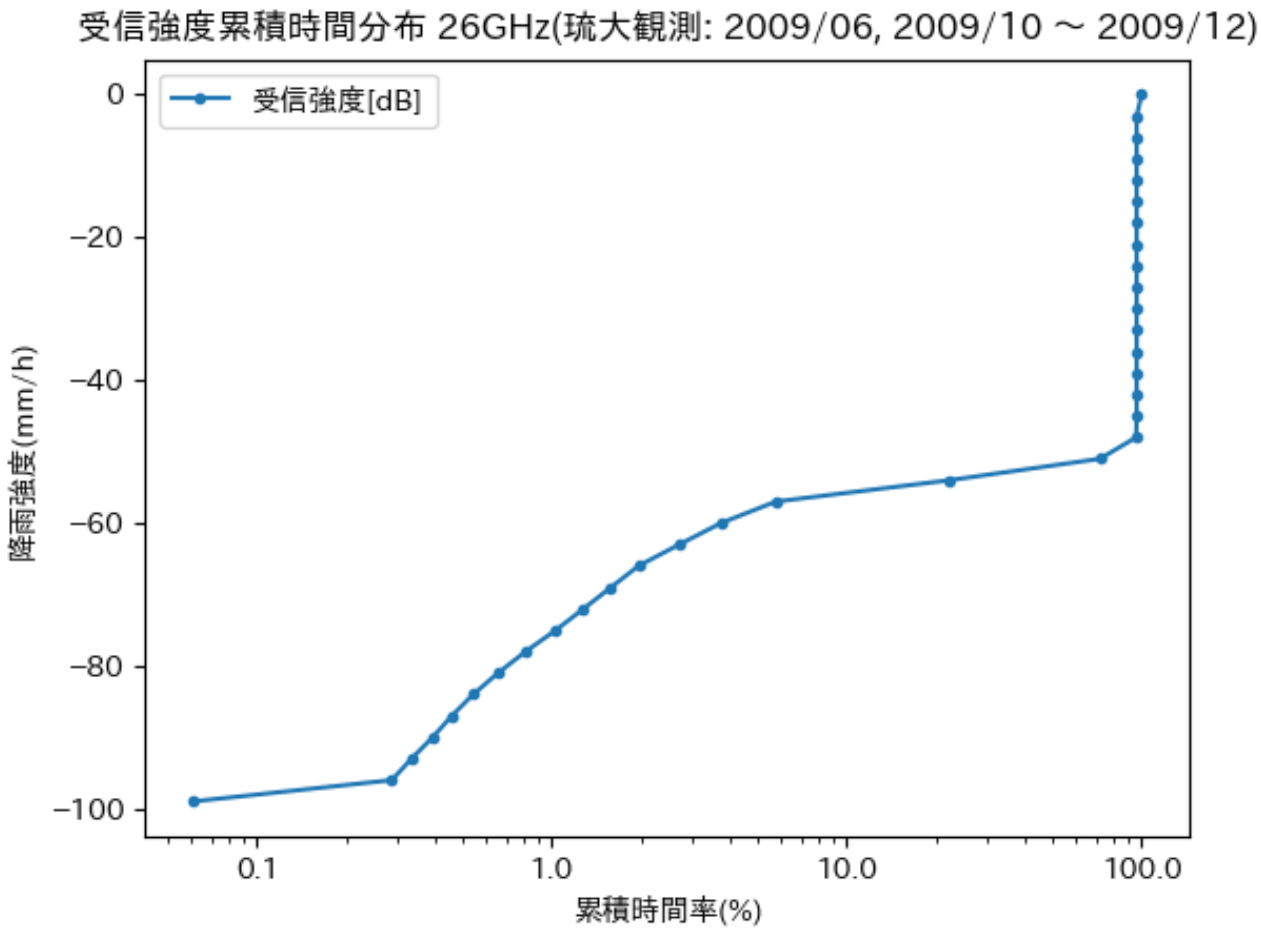

図 3　受信強度累積時間分布 (26GHz)

参考文献・引用文献

参考文献

[1] 宮里 智樹 . “2023-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 . 2023 知能情報基礎演習 4 授業資料 .