伝達関数の決定

高周波補正の計算方法を示したドキュメントである。

※高野の修論の`2.3.解析手法`も参照すること（p.19）。

- コスペクトルとは、2つの変動量の相互相関に関して、周波数別の寄与を表す関数である（http://occco.nies.go.jp/yougo/cospectrum.html）。
- クローズドパス型分析計で測定された微量ガスフラックスのコスペクトルは、高周波数側で減衰が生じる。
  - 顕熱フラックスのコスペクトルを基準とみなし、それに一致するように補正する。

前提

このドキュメントは`.py`ファイルが実行できる環境が構築されていることを前提としています。Jupyter Notebook（`.ipynb`）で実行したい場合は、`.py`ファイルのコードをそのままコピー&ペーストし、`modules`配下のクラスをインポートすれば動かすことができます。

手順

1. FFTによるスペクトル計算
2. 高品質のデータを抽出
3. 伝達関数の決定
4. フラックスの計算

1. FFTによるスペクトル計算

時系列データをFFT（高速フーリエ変換）で処理することにより、各周波数における信号の性質を分析することができます。

1. `data`配下の`template`をコピーして、ディレクトリ名を任意のものに変更する（`2024.08.01`など）。
2. `csv`ディレクトリ内に、解析対象のCSVファイルを配置する。
3. Flux Calculatorを立ち上げる。
4. `Input Dir`で解析対象のCSVファイルが入ったディレクトリを選択する。
5. `Load Setting`で各サイトの設定ファイルを選択する。
6. `Corrections`タブの`High Frequency Loss`を2回クリックする。`Empirical Transfer Functions`のチェックをすべて外す（伝達関数による補正を適用しないようにする）。
7. `FFT`をクリックして、出力先を指定する（`fft`ディレクトリに出力）。

※一次トレンド除去を適用して、再度同様の操作を行うこと。

- `Filtering` の `detrend (1st order)` にチェックを入れる。
- 出力先の指定の際、`fft-detrend` という別フォルダを作成しそこに出力する。

以降の2. 高品質のデータを抽出と3. 伝達関数の決定については、一次トレンド除去を適用したFFTデータと、適用していないFFTデータそれぞれについて行う。

2. 高品質のデータを抽出

共通の下準備

1. 伝達関数を決定する期間ごとにFFTデータをまとめて入れたフォルダを用意する。
2. Flg データを用意する（Monthlyシートの `Flg` シートをCSV形式で出力する）。
   - 相対湿度（RH）と、高品質かどうかを示すフラグ（Flg）。
   - 品質管理、つまり `calc` シートにおいて1つでもフラグが立っている（=1）とき、低品質のデータであるため、伝達関数の決定に用いない。
   - Flgの2行目から最終行までの日時と、FFTファイルの最初から最後の日時は合わせること。

Flux Calculatorを利用する方法

この方法は、ファイルのインデックスがそろっていない場合などで正しく仕分けされない場合があります。基本的に後述の `1-reorganize_files.py` を利用する方法を推奨しています。

1. Flux Calculatorを立ち上げる。
2. `Option` → `Sort Spectral data by RH`
3. `Input Dir` で①のFFTデータをまとめたフォルダを選択する。
4. `Input File` で②の `Flg` データを選択する。
5. `Output Dir` で出力先を指定する（ `sort` のようなの名前の空のフォルダを用意しておく）。
6. `Sort` をクリックすると、高品質のデータと低品質のデータが分別されたフォルダが作成される（高品質のデータはさらに相対湿度ごとに分別される）。
7. 相対湿度ごとに分別された高品質のデータを、全てまとめたフォルダを作成する（ `good_data_all` フォルダを用意し、`RH10` ～ `RH100` のデータをすべてコピーする）。

`1-reorganize_files.py` を利用する方法

1. `1-reorganize_files.py` に必要な情報を入力する（ `base_path` 、`input_dir_name` 、`output_dir_name` 、`flag_file_name` を適宜変更する）。
2. `1-reorganize_files.py` を実行する。`output_dir_name` で指定したディレクトリに仕分けされたファイルが `good_data_all` にコピーされる（使用しないデータは `bad_data` ディレクトリに保存される）。

※下記のように、`FftFileReorganizer` のコンストラクタに渡す `sort_by_rh` にTrueを指定した場合、`good_data_all` に加えてRH10～RH100のディレクトリが作成され、相対湿度（RH）によってFlgが0のデータを分別する。H2Oフラックスの計算には `sort_by_rh=True` を指定すること。

```
# メイン処理
try:
    input_dir_path = os.path.join(base_path, input_dir_name)
    output_dir_path = os.path.join(base_path, output_dir_name)
    flag_file_path = os.path.join(base_path, flag_file_name)

    # インスタンスを作成
    reoirganizer = FftFileReorganizer(
        input_dir_path, output_dir_path, flag_file_path, sort_by_rh=True
    )
    reoirganizer.reorganize()
except KeyboardInterrupt:
    # キーボード割り込みが発生した場合、処理を中止する
    print("KeyboardInterrupt occurred. Abort processing.")
```

3. 伝達関数の決定

1. Flux Calculatorを立ち上げる。
2. `Option` → `Determine closed path TF`
3. `Input Dir` で高品質のFFTデータをまとめたフォルダを選択する（H2Oフラックス以外は `good_data_all`、H2Oフラックスについては `RH10` ～ `RH100` の各フォルダについて行う）。
4. `Output File` で出力先とファイル名を指定する（ `TF_CO.2020.ALL_detrend.csv` など）。
5. `use only negative CO2 flux` のチェックを外す。
6. `Cutoff frequency` を設定する。Ultraにおいては `0.01~1` の範囲とした。
7. `Determine` をクリック→ファイルが出力される。
8. 出力されたファイルをPythonで読み込み、伝達関数を決定する。
    - `2-calculate_transfer_function.py` を開く。
    - パスやファイル名など必要な情報を入力して、実行する。
    - 伝達関数の係数 `a` が算出されるので、メモしておく。
    - Ultraの場合は、伝達関数の係数を記録するCSVファイルを用意している（ `tf-a/TF_Ultra_a.csv` ）ので、求めた係数 `a` を記録して保存する。CH4の場合を例に、以下にカラムの意味を示す。
        - `Date` には算出した日付を `yyyy/MM/dd` の形式で記録する。
        - `a_CH4` はトレンド除去なし、`a_CH4-detrend` は（一次）トレンド除去ありの係数。
        - `a_CH4-used` には、最終的にFlux Calculatorの設定ファイルに入力する値を記録する。

4. フラックスの計算

1. Flux Calculator を立ち上げる
2. `Input Dir`、`Output File`、`Load Setting` についてはいつもどおり選択する。
3. `Corrections` タブの `High Frequency Loss` を2回クリック→ `Empirical Transfer Functions` の `wch4 op`、`high frequency loss for closed path` などの各種該当するフラックスにチェックを入れ、決定された伝達関数の係数を入れる。
4. `Update` をクリックする（※エンターキーを押すと係数の変更が適用されないので注意）。

5. `Exec` で計算を実行する。

5. 全期間の伝達関数を比較

1. `3-plot_all_tf_curves.py` を開き、`TF_Ultra_a.csv` までの絶対パスを `tf_csv_path` に入力する。

```
# メイン処理
try:
    tf_csv_path: str = (
        "C:\\Users\\nakao\\workspace\\sac\\ultra\\transfer_function\\tf-a\\TF_Ultra_a.csv"
    )
    plot_all_tf_curves(tf_csv_path)
except KeyboardInterrupt:
    # キーボード割り込みが発生した場合、処理を中止する
    print("KeyboardInterrupt occurred. Abort processing.")
```

2. `.py` ファイルを実行すると、すべての `a` をもとにした近似曲線と、その平均が示された図が作図される。CH4とC2H6について、それぞれ一枚ずつ出力される。

備考

MarkdownからPDFに変換する方法

MarkdownファイルのPDF化にはVisual Studio Codeの拡張機能である「Markdown PDF」を使用している。スタイルはsindresorhus/github-markdown-cssの `v5.6.1` を適用している。

VS Codeのインストール方法と拡張機能の設定は各自で調べてください。以下にカスタムスタイルの設定を記述します。

1. ローカルに `.css` をダウンロードし、任意のディレクトリに配置する（ `C:/workspace/styles` など）。

2. `.css` ファイルで `.markdown-body` を検索し、すべてを `body` に置換する（これによって `<body>` タグ配下にスタイルが適用される）。

3. VS Codeで `Ctrl + Shift + P` を実行してコマンドパレットを開き、`settings.json` を検索する。開いたら、以下の設定を追加する。

```
{
    "markdown-pdf.includeDefaultStyles": false,
    // GitHubのスタイルを適用
    "markdown-pdf.styles": [
        "C:/workspace/styles/github-markdown-5.6.1-custom.css"
    ],
}
```