

FEKO の計算結果を可視化、サポートするツール群

w2db

w2db(x) mW -> dB $10 * \log_{10}(x)$

db2w

db2w(x) dB -> mW `df.db2w()` or `db2w(df)`

TEST

```
df = pd.DataFrame(np.arange(3 * 10).reshape(-1, 3), columns=list('abc'))
print(df.db2w())
```

	a	b	c
0	1.000000	1.258925	1.584893
1	1.995262	2.511886	3.162278
2	3.981072	5.011872	6.309573
3	7.943282	10.000000	12.589254
4	15.848932	19.952623	25.118864
5	31.622777	39.810717	50.118723
6	63.095734	79.432823	100.000000
7	125.892541	158.489319	199.526231
8	251.188643	316.227766	398.107171
9	501.187234	630.957344	794.328235

`np.power(10, x / 10)`

v2db

v2db(x) V -> dB $20 * \log_{10}(x)$

db2v

db2v(x) dB -> V `np.power(10, x / 20)`

a2comp

`a2comp(mag, arg)` $\text{mag} * (\cos(\text{radians}(\text{arg})) + \sin(\text{radians}(\text{arg})) * 1j)$

- `mag`(大きさ) と `arg`(角度) を引数に、複素数表示で返す。
- 引数:
 - `mag`: magnitude
 - `arg`: argument
- 戻り値: 複素数表示

`rsc_total`

```
rsc_total(Etheta, Ephi, source_power) 4 * np.pi * (((np.abs(Etheta))**2 + (np.abs(Ephi))**2)
/ source_power)
```

- 引数:
 - `Etheta`, `Ephi`: 電界強度 (複素数)
 - `source_power`: ソースの電界強度 [V/m]。普通は 1? out ファイルや `frko` のファイル参照
- 戻り値: 単位ソースパワーあたりの `Etheta` と `Ephi` の絶対値を出して 4π 掛けた値

`import_data`

```
import_data(filename: str)
```

- FEKO の.out ファイルを pandas DataFrame 形式にして返す
- 引数:
 - `filename`: ファイル名 (str 型)
- 戻り値:
 - `df`: 列名が `theta phi`, (pandas.DataFrame 型)

`import_data_comp`

```
import_data_comp(filename: str, ram=1)
```

- FEKO の.out ファイルを pandas DataFrame 形式にして返す
- 引数:
 - `filename`: ファイル名 (str 型)
 - `ram`: 電波吸収体反射係数真数。指定しなければ 1(=変倍しない)(float 型)
- 戻り値:
 - `df`: 列名が `'THETA'`, `'PHI'`, `'ET_COMP'`, `'EP_COMP'`, `'RCS_dBsm'`(pandas.DataFrame 型)

sumdf:

```
sumdf(column_name, dataframes: list)
```

- 引数にしたデータフレームの特定のカラムを足し算してデータフレームとして返す。
- 引数:
 - column_name: カラムの名前
 - dataframes: データフレームを入れたリスト
- 戻り値:
 - df.sum(): データフレームの一部だけ取り出したものをひとつのデータフレームにして、各列を足し算した pandas.Series

```
# TEST
```

```
aa = pd.DataFrame([1,2,3])
bb = pd.DataFrame([4,5,6])
sum_rcs(0,[aa,bb])
```

```
# Out:
```

```
0    5  # 1+4
1    7  # 2+5
2    9  # 3+6
```

fine_ticks

```
fine_ticks(tick, deg)
```

- グラフの ticks をイイ感じにする
- 引数:
 - tick: label に使うリスト (リスト型)
 - deg: label を deg ごとに分割する
- 戻り値: tick の最大、最小値、deg から求めたイイ感じの np.array

```
# TEST
```

```
#In : for i in range(10,180,10):
```

```
      print(fine_ticks(np.arange(181),i))
```

```
#Out :
```

```
[  0.   10.   20.   30.   40.   50.   60.   70.   80.   90.  100.  110.
  120.  130.  140.  150.  160.  170.  180.]
[  0.   20.   40.   60.   80.  100.  120.  140.  160.  180.]
```

```

[ 0.  30.  60.  90. 120. 150. 180.]
[ 0.  45.  90. 135. 180.]
[ 0.  60. 120. 180.]
[ 0.  60. 120. 180.]
[ 0.  90. 180.]
[ 0.  90. 180.]
[ 0.  90. 180.]
[ 0. 180.]
[ 0. 180.]
[ 0. 180.]
[ 0. 180.]
[ 0. 180.]
[ 0. 180.]
[ 0. 180.]
[ 0. 180.]
[ 0. 180.]

```

plot_contourf

```

plot_contourf(df, title='', xti=30, yti=1, alpha=.75,
              xlabel='azimuth(deg)', ylabel='elevation(deg)', zlabel='(dBsm)',
              cmap='jet', cmaphigh=20, cmaplow=0, cmaplevel=100, cmapstep=2,
              fn="Times New Roman", fontsize=12, *args, **kwargs)

```

- pivot されたデータフレームを引数に contourf を描く
- 引数:
 - df: pivot されたデータフレーム
 - * x, y, z は df から計算される
 - title: グラフのタイトル
 - xti, yti: tick の区切り (deg ごとに分割する)
 - alpha: ヒートマップの透過率
 - xlabel, ylabel, zlabel: ラベル名
 - cmap: カラーマップ
 - cmaphigh, cmaplow, cmaplevel: カラーマップの最大値、最小値、段階
 - cmapstep: 右側に表示されるカラーマップの区切りをいくつごとにするか
 - fn, fontsize: フォント、フォントサイズ
- 戻り値: なし

rolling_around

```
rolling_around(df, window, mirror=False, min_periods=None, freq=None, center=False,
               win_type=None, on=None, axis=0, *args, **kwargs)
```

- 全周移動平均の作成
- `mirror=False` (デフォルト) のとき、元データを 2 つ重ねて移動平均をとる。> 周回 360deg のときに使う
- `mirror=True` のとき、元データをの鏡像を重ねて移動平均をとる。> 周回 180deg のときに使う
- 移動平均処理後は重ねた分のデータは不要なので、消してインデックスをリセットする。 >
`df.loc[len(df/2):].reset_index()`
- `pd.DataFrame.rolling` のオプションはすべて使える。> 詳細は `pd.DataFrame.rolling?`
- 引数:
 - `df`: データフレーム
 - `columns`: 平均処理をするカラム (リスト形式など)
 - `window`: 平均を行うの区間 (int 型など)
 - `mirror`: `True` で鏡像データの作成を行ってから平均化処理
 - 以下は `pandas` のドキュメント参照
 - * `min_periods`
 - * `freq`
 - * `center`
 - * `win_type`
 - * `on`
 - * `axis`
- 戻り値: 全周移動平均処理を行ったデータフレーム (`pandas.DataFrame` 型)

```
# TEST
```

```
n=3
```

```
df = pd.DataFrame(np.arange(n*10).reshape(-1, n), columns=list('abc'))
```

```
window = 2
```

```
a = df.copy()
```

```
normal_rolling_mean = a.rolling(window).mean()
```

```
print('original\n', df)
```

```
print('normal rolling mean\n', normal_rolling_mean)
```

```
print('around rolling mean mirror \n', df.rolling_around(2, mirror=True))
```

```
print('around rolling mean NOT mirror\n',
```

```
      df.rolling_around(2, mirror=False)) # mirror=False 省略化
```