# Python Script for Rotating Coil Post-Processing

RotatingCoilAnalysis-ElyttV2.py

Calls:

Postprocessing\_Functions-RCA:

-ContinuousRotatingCoilAnalysis() ***Documented***

-RotatingCoilAnalysisTurn() ***Documented***

-.ReadRoxie() ***Documented***

-interpolate\_Roxie\_MM() ***Documented***

- GetSensitivities() ***Documented***

- ReadFromCplus() ***Documented***

-Parameters()

Analysis\_Functions-RCA.py

-interpolate() ***Documented***

-simulate\_coil() ***Documented***

-shiftit() ***Documented***

-ReadClipboard() ***Documented***

-Differences()

## Post Processing of WMM data

**RotatingCoilAnalysis-ElyttV2.py**

-Takes manually the location of the folder with the data In each folder there are sub folders corresponding to each position along the magnet and to each current (+-5A)

-From the name of the folder defines: Length of Coil, Whether it is MCBXFA or MCBXFB

-Based on above defines lists with the longitudinal positions (paso)

-Reads the sensitivities from C

**-GetSensitivities()**

--Returns a list with two arrays:

--Sensitivities absolute signal

--Sensitivities compensated signal

-From the data folder selects: The fluxes data .txt (For all positions along the magnet) and the raw data (only one file)

**For each position along the magnet:**

-**ContinuousRotatingCoilAnalysis()**

--Takes the fluxes data and puts it in a df

--Separates in turns

--**For each turn**

--Drift Correction of each turn

--**RotatingCoilAnalysisTurn()**

---Fourier transform

---Apply Coil Sensitivity

---Angle Correction (Apply rotation)

---Apply Bucking ratios

---Center localization

---Feed Down

---Normalization

**---Returns [list with B1…B15, list wit A1…A15, value X, value Y, Value PhiOut]**

--Builds a df

|  |  |
| --- | --- |
|  | Average (Turn1 ... Turn10) |
| Rref |  |
| X |  |
| Y |  |
| PhiOut |  |
| B1 |  |
| […] |  |
| B15 |  |
| […] |  |
| A15 |  |

**--Returns df**

-Sets the positions “paso” as the column “position” and Concatenates df to Av\_pos or --Av\_neg depending on the signal of the current

-Creates Av which is the average of the results for Positive and Negative current.

-Takes the A´s, a´s, B´s and b´s and separates Av in Av Normal and Av Skew

-Centers the position of the measurements to be from negative to positive

**--ReadRoxie()**

--Takes the Roxie output corresponding to the measured magnet that is in the measurement folder (The Roxie output could be taken from a common folder and chosen according to the specifications given in the measurement folder name).

--From the .output selects the multipole profile data columns.

--Creates a df with all multipole profiles (alles).

--Shifts the position columns for the profile to be centered in 0.

--Normalizes all the profiles (except the main field) to the main field A1 or B1 depending on whether the magnet is Normal or Skew. Then normalizes the Main Field to its maximum.

--Creates a df with the Multipoles in the center of the magnet (MP) already normalized.

**--Returns [Alles,MP]**

-Removes the path along the field in case it was calculated in Roxie.

-With MP Creates Comp\_Roxie\_Meas: A df to compare the values of Roxie in the middle and the measurement in the middle. Then saves it.

-Creates a df in which the Roxie profiles are interpolated to select the values corresponding to the measured points and is merged to the measured points. Then Saves a different file for normal and skew.

--**Interpolate\_Roxie\_MM()**

--Takes the Roxie dataframe in the shape "Alles" and interpolates it to have values in the same positions as the magnetic measurements.

--Creates a df with one columns which is the positions of the magnetic measurements

--Interpolates Roxie df to have step of 1mm.

--Merge MM and Roxie DFs Eliminating from Roxie df the positions that are not in the MM df

**--Returns Merged DF**

## Description of Analysis\_Functions-RCA

**interpolate()**

This Funtion is meant to interpolate Roxieall-type of dataframes. However it can be used with any dataframe

**Returns interpolated dataframe**

**Simulate\_coil()**

Simulates the result obtained by a roatating coil. NEEDS - The Multipole profiles FROM ROXIE INTERPOLATED 1mm. ATENTION - Returns the SUM of all the values corresponding to the RotCoil NOT THE INTEGRAL. It can be chosen to have the coil displaced

Define the positions of the coil according to the coil length and the centering

Apply displacement

Sum the values corresponding to each Rotating coil section

**Returns the integrated dataframe**

**Shiftit()**

Takes a Roxieall df and shifts the spectrum in the longitudinal position

Defines the step of the Roxie File

Defines the period (number of vertical cells up/down each of the df columns) that the data in each column must be shifted

Creates new DataFrame and with new columns anad applies the df.shift() method with the period previously defined

**Returns the original df with extra shifted columns**

**ReadClipboard()**

Creates a DataFrame with the copied data from Excel. The data is certain number of multipole profiles copied without column names.

The copied data must have the shape (n columns):

|Position|Multipole(B1,B2...B15...A15)|

| Values | Values |

Asign a variable to the copied string

Split the string into rows

Creates a list of rows, each rows is a list with the values in that row

Put the rows into a dataframe

Add column names, i.e. the multipoles indicated when calling the function OR generic col-n if no multipoles were indicated

Normalizes, if wanted, a column selected by the user through the console to its value in the middle point (x=0)

**Returns a DataFreame with the copied data.**

**Parameters()**

Reads the measurement parameters exported by FFMM.

Iterates line by line looking for the desired values to return

**Returns: points per turn, mag order, num FDI, refRadius, Analysisoptions**