# PyEIS: A Python-Based Electrochemical Impedance Spectroscopy analyzer and simulator

## Data Import Syntax

**ex1 = EIS_exp(path, data, cycle, mask)**

- path = 'location'
- data = ['data_file.xxx', …, 'data_file.xxx']
- cycle = [cycle_no, …, cycle_no] (optional)
- mask = [high_freq, low_freq] (optional)

**Data organization:**

- The data is organized in a Pandas dataframe and can be viewed by calling ex1.df
- When multiple cycles are selected, each cycle is organized in separate data frames and can be called by the cycle_no.

**Support:**

- Bio-Logic's ".mpt" file format
- Gamry's ".DTA" file format

**Data:**

- Multiple data files from the same folder can be called into the data called (*i.e.* ex1). The cycle_no in each files is restructured, such that the first file will have cycle_no = 1, and the next file's cycle_no will append to the last occurring cycle_no in file 1.

**Cycle:**

- Specific cycles can be called using "cycle", if none is called, all cycles are imported (default)

**Mask:**

- Mask can be used to limit the frequency of all cycles included in the dataframe. If none are given all data is included. Function is useful after having performed Linear Kramers-Kronig analysis.

**ex1 = EIS_exp(path='test_folder', data=['test_file1.xxx'], cycle=[1,2], mask=['none','none'])**

contains data of cycle 1 and 2

**ex1.df**

contains all experimental data for all cycles called
>>> [f, re, im, Z_mag, Z_phase, times …]

**ex1.df[0]**

outputs data for cycle_no = 1
>>> [f, re, im, Z_mag, Z_phase, times …]

**ex1.df[0].f // ex1.df[0].re // ex1.df[0].im**

Outputs the frequency, real, and imaginary data from cycle_no =1

**ex2 = EIS_exp(path='test_folder', data=['test_file1.xxx', 'test_file2.xxx'])**

Contains all cycles from test_file1 and test_file2

```
[10]: ex1.df[0].head()
```

[10]:

| | f | re | im | Z_mag | Z_phase | times | E_avg | I_avg | Cs/µF | Cp/µF | cycle_number |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 11 | 8325.1924 | 2233.4700 | 149.750820 | 2238.4846 | -3.835855 | 667.807180 | -0.900195 | -0.001823 | 0.127661 | 0.000571 | 1.0 |
| 12 | 6195.4932 | 2233.7661 | 119.028460 | 2236.9351 | -3.050178 | 668.444421 | -0.900275 | -0.001864 | 0.215821 | 0.000611 | 1.0 |
| 13 | 4621.4780 | 2224.8149 | 95.144958 | 2226.8484 | -2.448781 | 669.199281 | -0.900254 | -0.001854 | 0.361954 | 0.000661 | 1.0 |
| 14 | 3456.1052 | 2225.6133 | 94.790283 | 2227.6309 | -2.438790 | 669.953294 | -0.900270 | -0.001864 | 0.485813 | 0.000880 | 1.0 |
| 15 | 2585.8269 | 2237.1299 | 76.495079 | 2238.4373 | -1.958375 | 670.707286 | -0.900204 | -0.001848 | 0.804613 | 0.000940 | 1.0 |

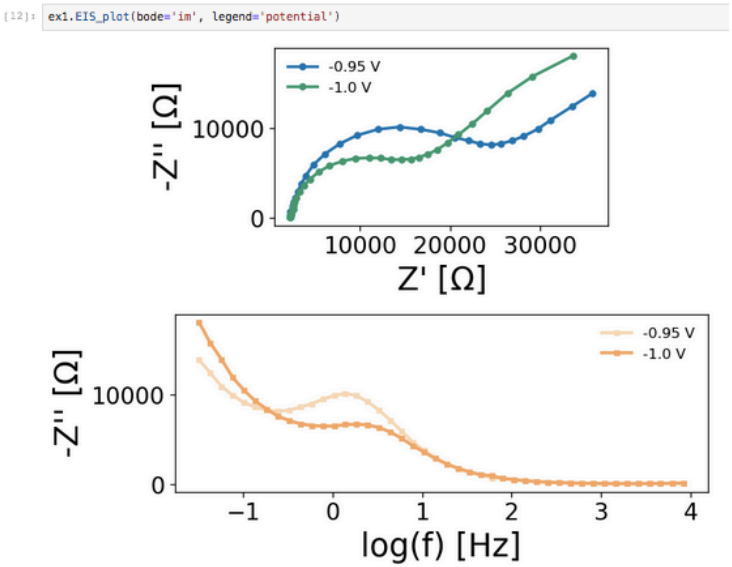
# PyEIS: A Python-Based Electrochemical Impedance Spectroscopy analyzer and simulator

## Plotting data

**ex1.EIS_plot(bode='off', fitting='off', rr='off', nyq_xlim='none', nyq_ylim='none', legend='on', savefig='none')**

Plots the data contained within the dataframe (*i.e.* ex1) in Nyquist and Bode plots. This function is also used to plot experimental- with fitted data through the fitting parameter

- bode
  - 'on' = plots re, im vs. log(freq)
  - 'log' = plots log(re, im) vs. log(freq)
  - 're' = plots re vs. log(freq)
  - 'log_re' = plots log(re) vs. log(freq)
  - 'im' = plots im vs. log(freq)
  - 'log_im' = plots log(im) vs. log(freq)
  - 'off' = no Bode plot is shown
- Fitting
  - 'on' = if fits have been performed, turn on and fits will be shown
  - 'off' = should be turned off, when no fits have been performed (default)
- rr
  relative residuals between the fit and experimental data. This is not to be confused with the relative residuals from the linear Kramer-Kronig analysis.
  - 'on' = opens an additional subplot that illustrates the result of the error analysis
  - 'off'
- nyq_xlim // nyq_ylim
  - 'none' = automatic
  - [x/y_min, x/y_max] = insert number to limit x/y limits in the Nyquist plot
- legend
  - 'on' = cycle no is shown in legend (default)
  - 'off' = no legend
  - 'potential' = potential for each cycle is shown in legend
- savefig
  - The figure can be saved through this call as *.png, *.svg, *.pdf, *.pdf, *.jpg



# PyEIS: A Python-Based Electrochemical Impedance Spectroscopy analyzer and simulator

## Evaluating data quality (Linear Kramers-Kronig analysis)

**ex1.Lin_KK(num_RC='auto', legend='on', plot='residuals', bode='off', nyq_xlim='none', nyq_ylim='none', weight_func='Boukamp', savefig='none')**

Performs linear Kramers-Kronig validity analysis of the experimental data contained within the data called (*i.e.* ex1). The analysis investigates the causality, linearity, stability, and finite. The algorithm uses a weighed complex non-linear least-squares fitting procedure (CNLS) with RC-elements, where R's are fitted to experimental data with log spaced time constants. The relative residuals are illustrated as a function of the measured frequency. The procedure results in a $\mu$-value that ensures an optimal fit through an ideal number of RC-elements by evaluating the sum of resistances greater or smaller than zero. The $\mu$-value is thus defined as $\frac{\sum_{R_k<0}|R_k|}{\sum_{R_k\geq 0}|R_k|}$ [Electrochimica Acta 131 (2014) 20-27] and should lie within 0.75-0.85 for an optimal fit.

- num_RC
  - 'auto' = automatically finds the correct number of RC-elements, which avoids over- or under-fitting. (recommended)
  - 'number/decade' = The number of RC circuits can be hardwired by selected any number (maximum 80 RC circuits). Note number inserted is number of RC-elements/decade
- legend
  - 'on' = cycle no is shown in legend (default)
  - 'off' = no legend
  - 'potential' = potential for each cycle is shown in legend
- bode
  - 'on' = plots re, im vs. log(freq)
  - 'log' = plots log(re, im) vs. log(freq)
  - 're' = plots re vs. log(freq)
  - 'log_re' = plots log(re) vs. log(freq)
  - 'im' = plots im vs. log(freq)
  - 'log_im' = plots log(im) vs. log(freq)
  - 'off' = no Bode plot is shown
- nyq_xlim // nyq_ylim
  - 'none' = automatic
  - [x/y_min, x/y_max] = insert number to limit x/y limits in the Nyquist plot
- weight_func
  - 'Boukamp' = The Weight function applied to the CNLS fitting procedure as suggested per Boukamp B.A. (J. Electrochem. Soc., 142, 6, 1885-1894): [1/(re**2), 1/(im**2)]
- savefig
  - The figure can be saved through this call as *.png, *.svg, *.pdf, *.pdf, *.jpg

- plot
  - 'residuals' = plots the real and imaginary relative residuals from the Lin_KK analysis in separate subplots for each cycle_no (default)
  - 'w_data' = plots the real and imaginary relative residuals from the Lin_KK analysis in a single subplot with the Nyquist and possibly Bode plot

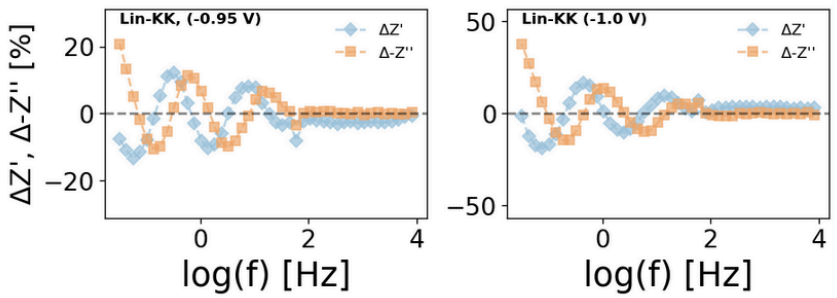
**Example**

ex1.Lin_KK(legend='potential')

Outputs two figures with the relative residuals from an optimized linear Kramers-Kronig analysis

>>> prints cycle_no, number of RC-elements, and $\mu$-value

```
[8]: ex1.Lin_KK(legend='potential')

cycle || No. RC-elements ||   u
[1]            10            0.77
[2]            9             0.82
```



# PyEIS: A Python-Based Electrochemical Impedance Spectroscopy analyzer and simulator

## Equivalent Circuit Fitting

**ex1.EIS_fit(circuit, params, weight_func='modulus', nan_policy='raise')**

The purpose of analyzing EIS data is to understand the nature of electrode processes. The field utilize circuit elements that are combined to generate an equivalent circuit model. This is a sensitive technique that requires other measurements to have been carried out previously to understand the reasoning for a circuit element to be included the final model.

PyEIS contains a number of build-in circuit models that can be viewed in the equivalent circuit overview and can be used to fit experimental data. This function allows to fit experimental data to equivalent circuit models through weighed complex non-linear least-squares fitting.

The function fits all cycles included in the dataset (*i.e.* ex1) and thus is able to easily batch fit any large dataset.

- circuit
  - The equivalent circuit overview illustrates a "fit string" for each equivalent circuit, which should be entered here as a string
- params
  - The parameters required by the equivalent circuit model for the initial guesses and bounds are called through params
- Weight_func
  The weight function is used in the weighed complex non-linear least-squares fitting procedure
  - 'Modulus' = Weighs fit based on the modulus (recommended)
  - 'proportional' = proportional weight
  - 'unity' = each data point is weighed equally
- nan_policy
  - How to handle Nan or missing values in dataset
  - 'raise' = raise a value error (default)
  - 'propagate' = do nothing
  - 'omit' = drops missing data

**Example**

The data of ex1, as shown in EIS_plot(), can be fitted to a Randles circuit (experimental data from a macro disk electrode). First, the parameter space and bounds of each parameter is defined, hereafter the EIS_fit() function is called, with the 'fit string' representing the Randles circuit.

fit_params = Parameters()

fit_params.add('Rs', value=2500, min=2200, max=3000)
fit_params.add('R', value=25000, min=10000, max=40000)
fit_params.add('Q', value=$10^{-5}$, min=$10^{-7}$, max=$10^{-3}$)
fit_params.add('n', value=0.8, min=0.7, max=1)
fit_params.add('sigma', value=1000, min=10, max=100000)

ex1.EIS_fit(circuit='R-(Q(RW))', params=fit_params, weight_func='modulus')
output a fitting statistics, such as $\chi^2$, number of evaluations, and the result of the fitted parameters

The ex1 dataset contains two spectra and each fitted parameter in the equivalent circuit is saved within the data called (*e.g.* ex1) with the prefix "fit_" included the potential at which the impedance was measured.

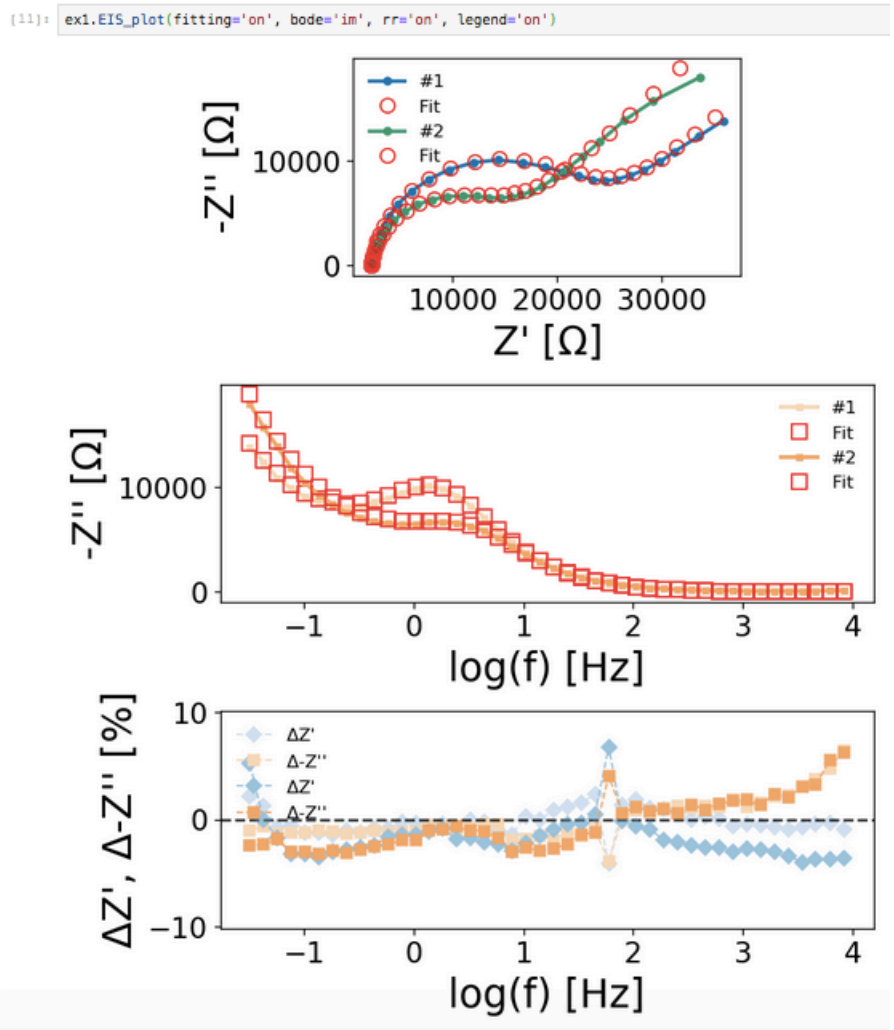
```
[14]: print(ex1.fit_E)
      print(ex1.fit_Rs)
      print(ex1.fit_R)
      print(ex1.fit_Q)
      print(ex1.fit_n)
      print(ex1.fit_sigma)

      [-0.95023994409090917, -1.000260077272727]
      [2232.9765232764707, 2286.9095874454529]
      [20662.992386262384, 12092.898909995018]
      [4.9828223427640279e-06, 4.360382414870465e-06]
      [0.92163795739795384, 0.93949259398255514]
      [5966.9583154590991, 8281.5805076029392]
```

# PyEIS: A Python-Based Electrochemical Impedance Spectroscopy analyzer and simulator

## Plotting fitted data

**ex1.EIS_plot(bode='off', fitting='off', rr='off', nyq_xlim='none', nyq_ylim='none', legend='on', savefig='none')**

- Similar to the example of plotting experimental data, the same function is called, with the exception of having fitting = 'on'.
- Note that any of bode plot options can be called
- In the example to the right, the relative residuals are activated (rr='on') and subplot three depicts the relative residuals between the fitted Randles circuit and the experimental data. This gives the user a great understanding of where the fit performs poor.



# PyEIS: A Python-Based Electrochemical Impedance Spectroscopy analyzer and simulator

## Simulating EIS

**EIS_sim(frange, circuit, bode, nyq_xlim, nyq_ylim, legend, savefig)**

This function simulates and plots EIS spectra using equivalent circuits. The equivalent circuits available in the software to simulate are shown in the equivalent circuit overview from which the simulation function should be used.

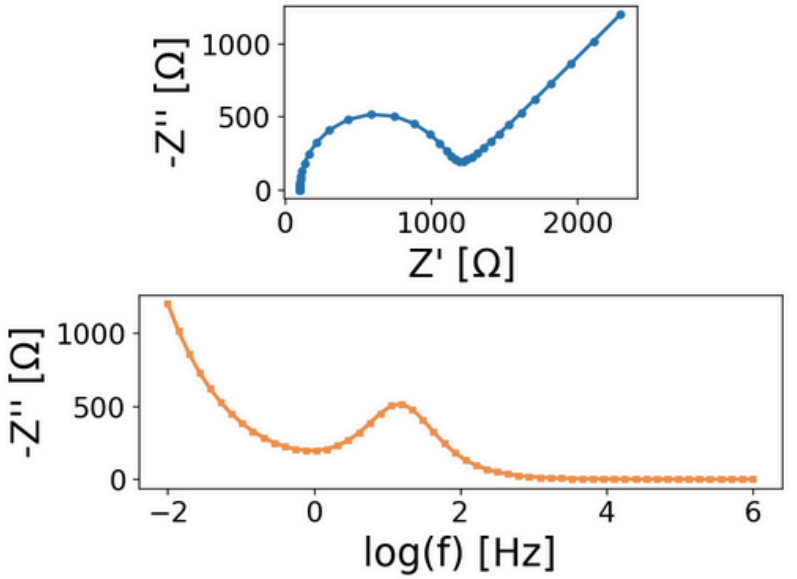
- frange
  - Besides the parameters needed for a given equivalent circuit, the frequency over which the impedance is simulated is required. A frequency generator function is included that generates log-spaced frequencies with a user-defined start, stop, and points/decade.
- circuit
  - This parameter requires the function for a desired equivalent circuit. Unlike the EIS_fit() function where circuit requires a string, circuit in EIS_sim() requires the function for the given equivalent circuits that generates the real and imaginary values. Circuit in EIS_sim() requires a “simulation function” which are given in the equivalent circuit overview.
- nyq_xlim // nyq_ylim
  - ‘none’ = automatic
  - [x/y_min, x/y_max] = insert number to limit x/y limits in the Nyquist plot
- legend
  - ‘on’ = cycle_no is shown in legend
  - ‘off’ = no legend
- savefig
  - The figure can be saved through this call as *.png, *.svg, *.pdf, *.pdf, *.jpg

freq_gen(f_start, f_stop, pts_decade)

This function generates log-spaced frequencies from start to stop and outputs outputs two frequencies

>>> [0] = frequency range [Hz] and [1] = Angular frequency [1/s]

```
[53]: f_range = freq_gen(f_start=10**6, f_stop=0.01, pts_decade=7)
      Randles = cir_Randles_simplified(w=f_range[1], Rs=100, R=1000, n=1, sigma=300, Q=10**-5)
      Randles_example = EIS_sim(frange=f_range[0], circuit=Randles, bode='im', legend='off')
```



# PyEIS: A Python-Based Electrochemical Impedance Spectroscopy analyzer and simulator

## Fitting of simulated EIS data

**EIS_sim_fit(circuit, params, weight_func='modulus', nan_policy='raise', bode='on', nyq_xlim='none', nyq_ylim='none', legend='on', savefig='none')**

This function fits simulated data and is primarily useful for the development and testing of new equivalent circuits functions and their fitting functions. The function is similar to the EIS_fit() function.

- circuit
  - The equivalent circuit overview illustrates a "fit string" for each equivalent circuit. The given "fit string" for an equivalent circuit model should entered here
- params
  - The parameters required by the equivalent circuit model for the initial guesses and bounds are called through params similar to EIS_fit()
- Weight_func
  - The weight function is used in the weighed complex non-linear least-squares fitting procedure
  - 'Modulus' = Weighs fit based on the modulus (recommended)
  - 'proportional' = proportional weight
  - 'unity' = each data point is weighed equally
- nan_policy
  - How to handle Nan or missing values in dataset
  - 'raise' = raise a value error (recommended)
  - 'propagate' = do nothing
  - 'omit' = drops missing data
- nyq_xlim // nyq_ylim
  - 'none' = automatic
  - [x/y_min, x/y_max] = insert number to limit x/y limits in the Nyquist plot
- legend
  - 'on' = cycle no is shown in legend
  - 'off' = no legend
- savefig
  - The figure can be saved through this call as *.png, *.svg, *.pdf,

- bode
  - 'on' = plots re, im vs. log(freq)
  - 'log' = plots log(re, im) vs. log(freq)
  - 're' = plots re vs. log(freq)
  - 'log_re' = plots log(re) vs. log(freq)
  - 'im' = plots im vs. log(freq)
  - 'log_im' = plots log(im) vs. log(freq)
  - 'off' = no Bode plot is shown

```
[10]: params = Parameters()
Rs_guess = 90
params.add('Rs', value=Rs_guess, min=Rs_guess*.01, max=Rs_guess*10)
R_guess = 1250
params.add('R', value=R_guess, min=R_guess*.01, max=R_guess*10)
Q_guess = 10**-6
params.add('Q', value=Q_guess, min=Q_guess*.01, max=Q_guess*1000)
n_guess = 0.8
params.add('n', value=n_guess, min=.7, max=1)
sigma_guess = 2000
params.add('sigma', value=sigma_guess, min=sigma_guess*.01, max=sigma_guess*10)

Randles_example.EIS_sim_fit(params=params, circuit='R-(Q(RW))', bode='on')
```