Impedance Analyzer Documentation

Release v0.9.0-beta

Matt Murbach

May 11, 2020

Contents:

Impedance Analyzer is an open-source, web-based analysis platform aimed at making physics-based models as easy to use as equivalent circuits for quantitative analysis of EIS experimental data.

The tool is currently hosted here.

It should be noted that the ImpedanceAnalyzer is currently a beta release. Improved documentation, tests, features, and an improved dataset are upcoming in the v1.0.0 release.

CHAPTER 1

Contribute to Impedance Analyzer

1.1 Local Setup

Recommended minimum environment:

- Python
- git
- conda

The following assumes you have all of the above tools installed already and are using the git Bash shell.

1.1.1 Windows

1. Clone the project:

```
> git clone https://github.com/mdmurbach/ImpedanceAnalyzer.git
> cd ImpedanceAnalyzer
```

2. Create and initialize the virtual environment for the project:

```
> conda env create -n impedance-analyzer-env python=3.4
> conda install scipy=0.19.1
> pip install -r requirements.txt
```

3. Use start.bat to activate the environment and start the application

```
> ./start.bat
```

4. If a browser window doesn't open. Navigate to http://localhost:5000/

1.2 Flask Application Structure

ImpedanceAnalyzer's structure is a Flask application with the structure shown below.

```
\ImpedanceAnalyzer
    \.ebextensions      <-- setup files for executing code on EC2 instances
    \.elasticbeanstalk  <-- config files for setting up Elastic Beanstalk environment
    \application        <-- main module
        \static         <-- folder for static (data, images, js, css, etc.) files
        \templates      <-- contains html templates for pages
        __init.py__     <-- makes this folder a module
        fitPhysics.py   <-- python functions for fitting physics-based models
        ECfit           <-- module for fitting equivalent circuits
        views.py        <-- responsible for routing requests to different pages
    \docs               <-- contains files associated with this documentation
    application.py      <-- .py file for starting Flask app
    config.py           <-- config file for Flask app
    requirements.txt    <-- list of python packages used to setup environment
```

1.2.1 Flask API

The views module contains the routing structure for the flask application

application.views.**fitCircuit**()
fits equivalent circuit

Parameters

circuit [str] string defining circuit to fit

data [str] string of data of comma separated values of freq, real, imag

p0 [str] comma deliminated string of initial parameter guesses

Returns

names [list of strings] list of parameter names

values [list of strings] list of parameter values

errors [list of strings] list of parameter errors

ecFit :

application.views.**fitPhysics**()
fits physics model

Parameters

request.values["data"] [string] comma-separated data string

Returns

fit [list] list of tuples containing the (f, Zr, Zi) of the best fit P2D model

names [list] list of parameter names

units [list]

values=values,

errors=errors,

results=p2d_residuals,

simulations=p2d_simulations,

fit_points :

application.views.**getExampleData**()
Gets the data from the selected example

Parameters

filename [str] filename for selecting example

Returns

data [jsonified data]

application.views.**getUploadData**()
Gets uploaded data

application.views.**index**()
Impedance Analyzer Main Page

application.views.**to_array**(*input*)
parse strings of data from ajax requests to return

1.3 Functions for Model Fitting

At the heart of ImpedanceAnalyzer is the ability to fit models to data:

1.3.1 Physics-based Models

Provides functions for fitting physics-based models

application.fitPhysics.**fit_P2D_by_capacity**(*data_string*, *target_capacity*)
Fit physics-based model by matching the capacity and then sliding along real axes to determine contact resistance

Parameters

data [list of tuples] (frequency, real impedance, imaginary impedance) of the experimental data to be fit

Returns

fit_points [list of tuples] (frequency, real impedance, imaginary impedance) of points used in the fitting of the physics-based model

best_fit [list of tuples] (frequency, real impedance, imaginary impedance) of the best fitting model

full_results [pd.DataFrame] DataFrame of top fits sorted by their residual

application.fitPhysics.**interpolate_points**(*data*, *exp_freq*)
Interpolates experimental data to the simulated frequencies

Parameters

data [pd.DataFrame]

application.fitPhysics.**prepare_data**(*data*)
Prepares the experimental data for fitting

Parameters

data [list of tuples] experimental impedance spectrum given as list of (frequency, real impedance, imaginary impedance)

Returns

data_df [pd.DataFrame] sorted DataFrame with f, real, imag, mag, and phase columns and

1.3.2 Equivalent Circuit Models

Functions for fitting an equivalent circuit analog to data

Loosely based off of a matlab routine, Zfit, from Jean-Luc Dellis.

https://www.mathworks.com/matlabcentral/fileexchange/19460-zfit

`application.ECfit.fitEC.`**`buildCircuit`**(*circuit_string*, *parameters*, *frequencies*)
transforms a circuit_string, parameters, and frequencies into a string that can be evaluated

Parameters

circuit_string [str]

parameters [list of floats]

frequencies [list of floats]

Returns

eval_string [str] Python expression for calculating the resulting fit

`application.ECfit.fitEC.`**`computeCircuit`**(*circuit_string*, *parameters*, *frequencies*)
evaluates a circuit string for a given set of parameters and frequencies

Parameters

circuit_string [string]

parameters [list of floats]

frequencies [list of floats]

Returns

array of floats

`application.ECfit.fitEC.`**`equivalent_circuit`**(*data*, *circuit_string*, *initial_guess*)
Main function for fitting an equivalent circuit to data

Parameters

data [list of tuples] list of (frequency, real impedance, imaginary impedance)

circuit_string [string] string defining the equivalent circuit to be fit

initial_guess [list of floats] initial guesses for the fit parameters

Returns

fit [list of tuples] list of (frequency, real impedance, imaginary impedance)

p_values [list of floats] best fit parameters for specified equivalent circuit

p_errors [list of floats] error estimates for fit parameters

Notes

Need to do a better job of handling errors in fitting. Currently, an error of -1 is returned.

application.ECfit.fitEC.**residuals**(*param*, *Z*, *f*, *circuit_string*)
Calculates the residuals between a given circuit_string/parameters (fit) and *Z*/*f* (data). Minimized by scipy.leastsq()

Parameters

param [array of floats] parameters for evaluating the circuit

Z [array of complex numbers] impedance data being fit

f [array of floats] frequencies to evaluate

circuit_string [str] string defining the circuit

Returns

residual [ndarray] returns array of size 2*len(f) with both real and imaginary residuals

application.ECfit.fitEC.**valid**(*circuit_string*, *param*)
checks validity of parameters

Parameters

circuit_string [string] string defining the circuit

param [list] list of parameter values

Returns

valid [boolean]

Notes

All parameters are considered valid if they are greater than zero – except for E2 (the exponent of CPE) which also must be less than one.

Circuit elements can be added to the circuit_elements.py

application.ECfit.circuit_elements.**A**(*p*, *f*)
defines a semi-infinite Warburg element

application.ECfit.circuit_elements.**C**(*p*, *f*)
defines a capacitor

$$Z = \frac{1}{C \times j2\pi f}$$

application.ECfit.circuit_elements.**E**(*p*, *f*)
defines a constant phase element

Notes

$$Z = \frac{1}{Q \times (j2\pi f)^{\alpha}}$$

where Q = p[0] and α = p[1].

application.ECfit.circuit_elements.**G**(*p*, *f*)
defines a Gerischer Element

Notes

$$Z = \frac{1}{Y \times \sqrt{K + j2\pi f}}$$

`application.ECfit.circuit_elements.`**`R`**(p, f)
defines a resistor

Notes

$$Z = R$$

`application.ECfit.circuit_elements.`**`W`**(p, f)
defines a blocked boundary Finite-length Warburg Element

Notes

$$Z = \frac{R}{\sqrt{T \times j2\pi f}} \coth \sqrt{T \times j2\pi f}$$

where R = p[0] (Ohms) and T = p[1] (sec) = $\frac{L^2}{D}$

`application.ECfit.circuit_elements.`**`p`**$(parallel)$
adds elements in parallel

Notes

$$Z = \frac{1}{\frac{1}{Z_1} + \frac{1}{Z_2} + ... + \frac{1}{Z_n}}$$

`application.ECfit.circuit_elements.`**`s`**$(series)$
sums elements in series

Notes

$$Z = Z_1 + Z_2 + ... + Z_n$$

1.4 Documentation

This project is documented using Sphinx. To rebuild the documentation:

```
> cd docs
> ./make.bat html
```

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index