eistoolbox

for

MathWorks® MATLAB

A toolbox for batch fitting of Electrochemical Impedance Spectroscopy data to equivalent circuit models

User Guide

Juan J. Montero-Rodríguez

Version 0.1-54

Contents

Chapter 1

Introduction

eistoolbox is a toolbox for MATLAB® used for batch fitting Electrochemical Impedance Spectroscopy (EIS) data to equivalent circuits.

Currently it is **alpha software**, and it will evolve over time.

1.1 Impedance Spectroscopy

Electrochemical Impedance Spectroscopy (EIS) measures the complex impedance of a sample as a function of the frequency. The experimental results are stored in two possible formats: polar coordinates (magnitude and phase) or rectangular coordinates (real and imaginary).

$$Z(f) = R + jX \tag{1.1}$$

where R is the resistance and X is the reactance of the sample.

The real part of the impedance is proportional to the resistivity, and the imaginary part is proportional to the permittivity. Both parameters can be calculated directly from the measurements, considering the exact geometry of the electrodes and measurement setup. For parallel plate electrodes, the following equations apply:

$$R = \frac{\rho L}{A} \tag{1.2}$$

$$C = \frac{\epsilon A}{D} \tag{1.3}$$

The capacitive reactance is given by

$$X_C = \frac{1}{2\pi f C} \tag{1.4}$$

Substituting (1.3) and (1.4) into (1.1) results in the following equation, which describes the impedance in terms of the resistivity and permittivity of the sample between parallel electrodes:

$$Z(f) = \frac{\rho L}{A} + \frac{1}{2\pi f}\frac{D}{\epsilon A} \tag{1.5}$$

Impedance Spectroscopy is also referred as Dielectric Spectroscopy, because it gives information about the dielectric properties of the measured sample.

1.2 Equivalent Circuits

Experimental data is fitted to equivalent circuit models. The models are designed to describe the interfaces, chemical processes and boundaries of the measured setup.

1.2.1 Circuit Elements

The elements present in the software are described in Table 1.1:

Table 1.1: Equivalent circuit elements and their MATLAB implementation.

Symbol	Element	Equation	MATLAB expression
R1	Resistor	$Z(f) = R$	`z=p*ones(size(f))`
C1	Capacitor	$Z(f) = 1/j2\pi fC$	`z=1./(1i*2*pi*f*p)`
L1	Inductor	$Z(f) = j2\pi fL$	`z=1i*2*pi*f*p`
E2	Constant Phase Element	$Z(f) = 1/p_1(j2\pi f)^{p_2}$	`z=1./(p(1)*(1i*2*pi*f).^p(2))`

The Warburg element can be obtained with a Constant-Phase Element by setting $p_2 = 0.5$

1.2.2 Circuit String Syntax

Circuits can be built using series and parallel combinations of the elements in Table 1.1, using the series and parallel operators s() and p(). These operators can contain any number of elements, separated by commas.

The number next to the element letter is the number of free parameters for this element. For a capacitor (C1) the only free parameter is the capacitance. For the constant-phase element (E2) the free parameters are p1 and p2.

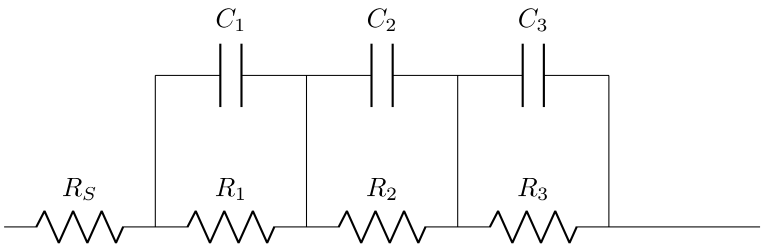
Common mistake: Do not write the circuit elements as s(R1,R2,R3,C4...). The elements cannot be written as labels. Instead, write s(R1,R1,R1,C1...).

1.2.3 Circuit Files (.ckt)

The circuit string, initial parameters and boundary conditions can be stored and loaded from a circuit file with the .ckt extension. This file is read by MATLAB line-to-line.

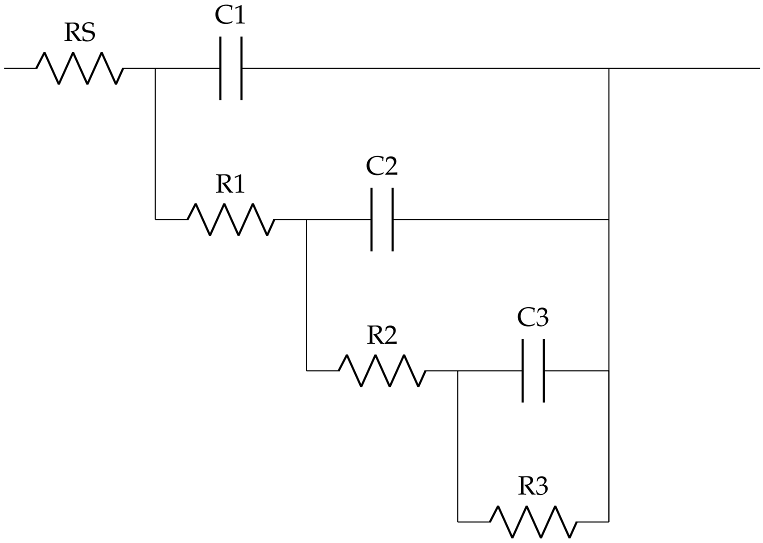
Check the folder 'examples_circuits' for more examples.

Example 1: Voigt model in the form R+R//C+R//C+R//C



Recommended circuit file:
s(R1,p(R1,C1),p(R1,C1),p(R1,C1)) % circuit string
[100,100,1e-6,100,1e-6,100,1e-6] % initial parameters
[0,0,0,0,0,0,0] % lower boundary conditions
[inf,inf,inf,inf,inf,inf,inf] % upper boundary conditions

Example 2: Ladder circuit in the form: ((((R//C)+R)//C)+R)//C+R



Recommended circuit file:
s(p(s(p(s(p(R1,C1),R1),C1),R1),C1),R1) % circuit string
[100,1e-6,100,1e-6,100,1e-6,100] % initial parameters... order: R3,C3,R2,C2,R1,C1,RS
[0,0,0,0,0,0,0] % lower boundary conditions
[inf,inf,inf,inf,inf,inf,inf] % upper boundary conditions

Chapter 2

Current capabilities of this software

It can accept any number of input files, both in CSV and Gamry DTA formats.

The CSV files should contain three columns with the impedance data, in the order: FREQ,REAL,IMAG.

The imaginary part can be positive or negative; the absolute value is taken inside the program before plotting.

The fitting algorithm uses the "fminsearch" function, implemented using the Zfit library from Jean-Luc Dellis.

It accepts any type of circuit model, built with serial and parallel elements, in the Zfit circuit string format.

The currently implemented elements are: resistors, capacitors, inductors and constant-phase elements (CPE).

The Warburg element can be implemented by using a CPE and setting the second parameter to 1/2.

2.1 Planned updates

In the future it will accept Levenberg-Marquard, Nelder-Mead, BFGS and Powell algorithms.

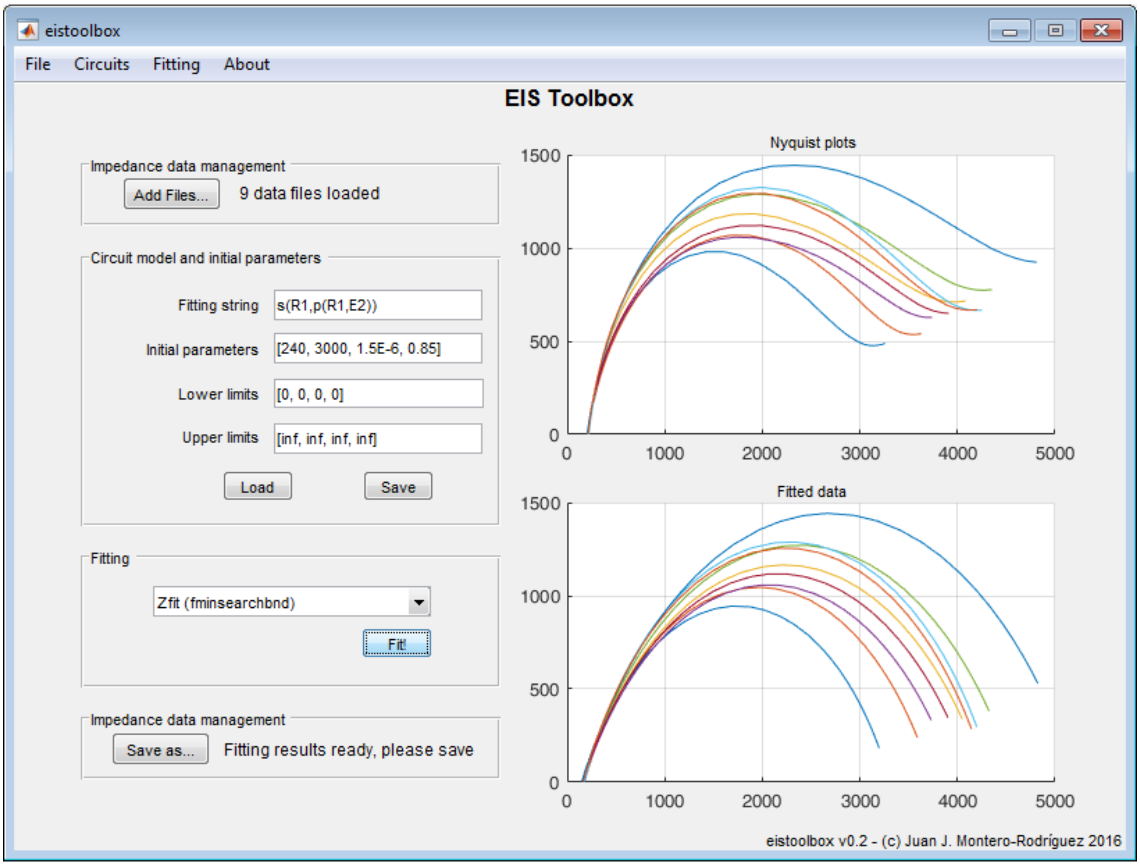
It will also include the error percentages of every fitting parameter, as well as the Pearson coefficient and correlation plot of the fitting results.

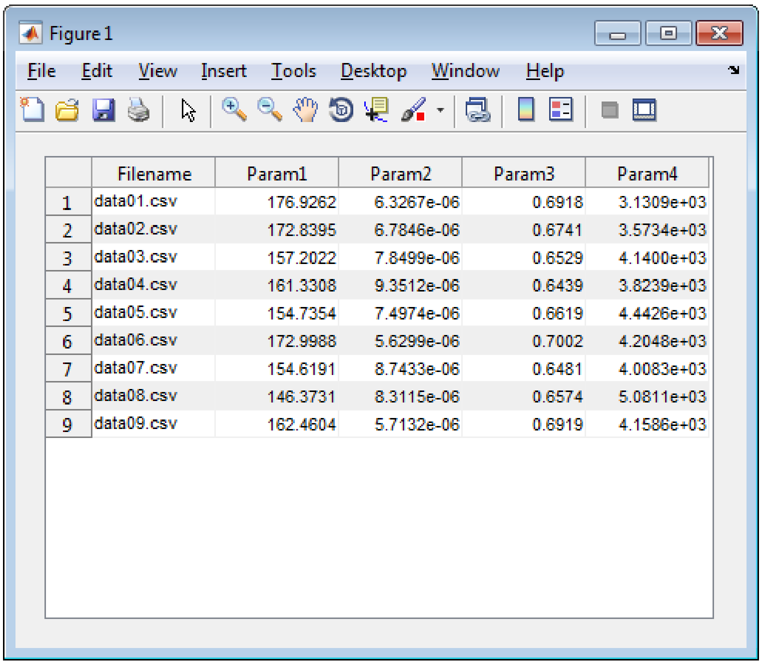
Chapter 3

Using the eistoolbox

3.1 Quick start guide

1. Add files using the "Add file..." button
2. Write the circuit string and fitting parameters (circuit string formatting)
3. Click the "Fit" button
4. Save the results using the "Save..." button





	Filename	Param1	Param2	Param3	Param4
1	data01.csv	176.9262	6.3267e-06	0.6918	3.1309e+03
2	data02.csv	172.8395	6.7846e-06	0.6741	3.5734e+03
3	data03.csv	157.2022	7.8499e-06	0.6529	4.1400e+03
4	data04.csv	161.3308	9.3512e-06	0.6439	3.8239e+03
5	data05.csv	154.7354	7.4974e-06	0.6619	4.4426e+03
6	data06.csv	172.9988	5.6299e-06	0.7002	4.2048e+03
7	data07.csv	154.6191	8.7433e-06	0.6481	4.0083e+03
8	data08.csv	146.3731	8.3115e-06	0.6574	5.0811e+03
9	data09.csv	162.4604	5.7132e-06	0.6919	4.1586e+03

Chapter 4

Fitting Algorithms

The fitting algorithms of this toolbox reduce the following distance function:

```
function dist=distance(param)
    ymod=feval(handlecomputecircuit,param,circuitstring,freq);
    if isequal('fitNP',fitstring)
        dist=sum(sum((ymod-zrzi).^2));
    else
        dist=sum(sum(((ymod-zrzi)./zrzi).^2));
    end
end
```

For **non-proportional fitting** the algorithm minimizes the squared difference between fitted (observed) and measured (expected) data.

$$d = \sum (o - e)^2$$

For **proportional fitting** the algorithm calculates the squared difference between fitted (observed) and measured (expected) data, divided by the measured (expected) data, in a way similar to the Pearson's chi-square test:

$$d = \chi^2 = \sum \frac{(o - e)^2}{e}$$

If the experimental impedance curve matches exactly the simulated curve, the distance function would have a value of zero.

To minimize the distance function, the toolbox adjusts the input parameters and recalculates the simulated data in each iteration. This is done in the second line of the code from above.

4.1 fminsearchbnd

This function was written by John D'Errico and published on MathWorks MATLAB Central under an open-source license. The original file can be downloaded at https://de.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd--fminsearchcon.

The function is based on fminsearch and includes the possibility of using boundary conditions, such as the lower and upper limits for the individual circuit parameters.

4.2 Other algorithms

The following optimization algorithms are not yet implemented in this toolbox. These are some of the most used algorithms for fitting EIS data to equivalent circuit models.

- Levenberg-Marquardt
- Nelder-Mead
- BFGS
- Powell

Chapter 5

Statistics

The program computes the following statistical parameters:

5.1 Linear regressions

Real of fitted vs Real of measured

Imag of fitted vs Imag of measured

MAG of fitted vs MAG of measured

5.2 Chi-square goodness of fit

$$\chi^2 = \sum_{i}^{n} \frac{(Observed_i - Expected_i)^2}{Expected_i}$$

Observed= fitted data

Expected= measured data

5.3 Error estimates for individual parameters

ToDo

Chapter 6

Licenses for included software

6.1 Zfit

The original file was released in 2005 and it is available here:

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

```
Copyright (c) 2005, Jean-Luc Dellis
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

* Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in
the documentation and/or other materials provided with the distribution

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```