Journal of Colloid and Interface Science 419 (2014) 102–106

Contents lists available at ScienceDirect

Journal of Colloid and Interface Science

www.elsevier.com/locate/jcis

A program for the fitting of Debye, Cole–Cole, Cole–Davidson, and Havriliak–Negami dispersions to dielectric data

Constantino Grosse

Departamento de Física, Universidad Nacional de Tucumán, Avenida Independencia 1800, 4000 San Miguel de Tucumán, Argentina
Consejo Nacional de Investigaciones Científicas y Técnicas, Avenida Rivadavia 1917, 1033 Buenos Aires, Argentina

ARTICLE INFO

Article history:
Received 25 October 2013
Accepted 12 December 2013
Available online 23 December 2013

Keywords:
Levenberg–Marquardt algorithm
Debye
Cole–Cole
Cole–Davidson
Havriliak–Negami
Non-linear parameter fitting
Dielectric dispersion

ABSTRACT

The description and interpretation of dielectric spectroscopy data usually require the use of analytical functions, which include unknown parameters that must be determined iteratively by means of a fitting procedure. This is not a trivial task and much effort has been spent to find the best way to accomplish it.

While the theoretical approach based on the Levenberg–Marquardt algorithm is well known, no freely available program specifically adapted to the dielectric spectroscopy problem exists to the best of our knowledge. Moreover, even the more general commercial packages usually fail on the following aspects: (1) allow to keep temporarily fixed some of the parameters, (2) allow to freely specify the uncertainty values for each data point, (3) check that parameter values fall within prescribed bounds during the fitting process, and (4) allow to fit either the real, or the imaginary, or simultaneously both parts of the complex permittivity.

A program that satisfies all these requirements and allows fitting any superposition of the Debye, Cole–Cole, Cole–Davidson, and Havriliak–Negami dispersions plus a conductivity term to measured dielectric spectroscopy data is presented. It is available on request from the author.

1. Introduction

A recurring problem present in all dielectric spectroscopy laboratories is the necessity to describe measured data using analytical functions. These expressions, arising from theoretical or empirical models, include unknown parameters that must be determined by means of a fitting procedure. If the model expression depends non-linearly on at least one of the unknown parameters, an iterative procedure becomes necessary to determine the best parameter set. This is not a trivial task and much effort has been spent to find the best way to accomplish it.

One of the best widely known solutions is presented in the Ref. [1], which discusses in detail the Levenberg–Marquardt algorithm that iteratively finds the parameter values that best fit the data, starting with guess values of these parameters provided by the user. It also includes the source code that makes it possible to implement a program that makes the calculations.

While this constitutes, in principle, an excellent way to solve the above mentioned problem, a series of difficulties arise in practice:

(1) The code is written in the form of a calling program (xmrqmin) and a series of subroutines (mrqmin, mrqcof, gaussj, covsrt, fgauss, gasdev, and ran1). These subroutines are used by many other programs in the book so that they have a general character and behave as black boxes. The whole system works, but it is not easy for a non-professional programmer to fully understand the code. Unfortunately, this becomes necessary since the code needs to be modified in order to make the program perform a function not envisioned by the authors.
(2) The calling program accomplishes two tasks: it first generates an artificial data set using an analytical expression together with parameter values input by the user and random "noise" used to simulate real data. It then determines the parameter values that best fit this data. Therefore, in order to use the code it is first necessary to strip from the calling program all the code used to generate the artificial data and then add the code required to read real data from a file. It is also necessary to add a subroutine corresponding to the model expression chosen by the user to describe the experimental data.

While these modifications required to perform the fitting process are not especially difficult, they are usually not sufficient. The main reasons for this are the following:

(3) Fixed parameters. When the model function includes many parameters, the fitting process becomes a complex task: the final parameters strongly depend on the initial guess

E-mail address: cgrosse@herrera.unt.edu.ar

0021-9797/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcis.2013.12.031

values, the iteration often diverges, or it converges to a set that is clearly wrong. This leads to the necessity to include in the program the possibility to perform an initial fitting with some of the parameters kept at fixed values. This requirement is then progressively released in successive program runs. Fortunately, this capability is included in the original code.

(4) Parameter uncertainties. The obtained results also depend quite strongly on the uncertainties of each measured data point. While the program uses uncertainty values, it assumes that these uncertainties are proportional to each data value with a proportionality constant that is the same for all data points. While this assumption might be adequate in some cases, a practical program should allow the free input of the data uncertainties (using information provided by the manufacturer of the measuring instrument, for example).

(5) Parameter bounds. In many cases, the parameters included in theoretical or empirical models have natural bounds: must be positive, or smaller than one, for example. It is then necessary to include these bounds in the fitting process in order to avoid unacceptable solutions. This capability is not included in the original code and should be implemented.

(6) Complex data. Experimental impedance spectroscopy data usually includes both the real and the imaginary parts of the measured magnitude. It is therefore essential that the fitting program is able to determine the goodness of fit from the distances on the complex plane between the measured and the calculated points rather than just their real or their imaginary parts. Again, this capability is absent in the original code.

In this work we present a program: DielParamFit.exe, that allows to fit any superposition of the Debye, Cole–Cole, Cole–Davidson, and Havriliak–Negami dispersions plus a conductivity term to measured dielectric spectroscopy data. The program is based on the theory, not the code, presented in Ref. [1], and includes all the above-mentioned extensions. A detailed account of the program implementation is presented as Supplementary material. The executable program is available on request from the author at cgrosse@herrera.unt.edu.ar.

2. Theory

We consider a data set made of N measured data points (x_i, y_i) with $1 \leqslant i \leqslant N$, where x and y are the independent and dependent variables, respectively. We want to represent these data by means of an analytical expression $y(x)$ that depends on M parameters a_k with $1 \leqslant k \leqslant M$. The problem is to determine the set of a_k parameter values that leads to the best agreement between the measured and the calculated data points. More precisely, we wish to determine the parameter set that minimizes the function:

$$\chi^2 = \sum_{i=1}^{N} \left[\frac{y_i - y(x_i)}{\sigma_i} \right]^2 \tag{1}$$

Note that this expression, called Chi-squared, also depends on the data point uncertainties σ_i.

We so set to zero the derivatives of the above expression with respect to the parameters a_k, which leads to M equations:

$$\sum_{i=1}^{N} \left[\frac{y_i - y(x_i)}{\sigma_i^2} \frac{\partial y(x_i)}{\partial a_k} \right] = 0 \tag{2}$$

In the case that all the model parameters are linear:

$$y(x) = \sum_{k=1}^{M} \left[a_k \frac{\partial y(x)}{\partial a_k} \right] \tag{3}$$

Eq. (2) transforms into the final expression

$$\sum_{j=1}^{M} (\alpha_{kj} a_j) = \beta_k \tag{4}$$

where

$$\beta_k = \sum_{i=1}^{N} \left[\frac{y_i}{\sigma_i^2} \frac{\partial y(x_i)}{\partial a_k} \right] \tag{5}$$

$$\alpha_{kj} = \sum_{i=1}^{N} \left[\frac{1}{\sigma_i^2} \frac{\partial y(x_i)}{\partial a_k} \frac{\partial y(x_i)}{\partial a_j} \right] \tag{6}$$

Eq. (4) constitutes a set of M linear equations that can be solved for the parameters a_k using any of the standard methods.

However, if at least some of the parameters are non-linear, Eq. (3) and following expressions no longer hold, so that a different approach must be used. Imagine a plot of χ^2 in the M parameter space. Since it is impossible, in practice, to systematically explore all the resulting "surface" looking for the lowest minimum, we have to start at an initial guess point and follow a path that descends to the closest minimum. Note that the outcome of this approach depends on the coordinates of the initial guess point. A bad guess could easily lead to a local minimum rather than the lowest minimum. Also note that the whole "surface" depends on the values of the uncertainties σ_i as also do the coordinates of the local minimum.

The problem is, therefore, to find a systematic algorithm that finds the way from the initial guess point to the local minimum. Any intermediate stage of the process is characterized by the position vector $\vec{a}$ in the parameter space, which has the components $a_1 \ldots a_M$. We seek to determine the next point $\vec{a} + \delta\vec{a}$ that is closer to the local minimum. The simplest approach is the steepest descent method that consists in following the direction of minus the gradient of χ^2:

$$\delta a_k = -\text{constant} \frac{\partial \chi^2}{\partial a_k} \tag{7}$$

The constant in this expression should be sufficiently small so that the downhill direction does not change too much over a single step. This method is good for points far from the minimum but becomes extremely slow close to the minimum where the gradient tends to zero.

Close to the minimum we can write down the second order expansion of χ^2:

$$\chi^2(\vec{a} + \delta\vec{a}) = \chi^2(\vec{a}) - 2\sum_{k=1}^{M} (\beta_k \delta a_k) + \sum_{k=1}^{M}\sum_{j=1}^{M} (\alpha_{kj} \delta a_k \delta a_j) + \ldots \tag{8}$$

where

$$\beta_k = -\frac{1}{2} \frac{\partial \chi^2}{\partial a_k} \tag{9}$$

$$\alpha_{kj} = \frac{1}{2} \frac{\partial^2 \chi^2}{\partial a_k \partial a_j} \tag{10}$$

Eq. (8) makes it possible to calculate the gradient of χ^2 at $\vec{a} + \delta\vec{a}$:

$$\left. \frac{\partial \chi^2}{\partial a_k} \right|_{\vec{a}+\delta\vec{a}} = -2\beta_k + 2\sum_{j=1}^{M} (\alpha_{kj} \delta a_j) \tag{11}$$

At the minimum the gradient should vanish so that the last parameter change $\delta\vec{a}$ required to attain the minimum is determined by: