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:

$$\sum_{j=1}^{M}(\alpha_{kj}\delta a_j) = \beta_k \tag{12}$$

2.1. Levenberg–Marquardt algorithm

This algorithm first proposes a way to satisfy the requirement that the constant appearing in Eq. (7) is sufficiently small so that the downhill direction does not change too much over a single step. Consider an expansion of χ^2 with respect to each parameter:

$$\chi^2(a_k+\delta a_k) = \chi^2(a_k) + \frac{\partial\chi^2}{\partial a_k}\delta a_k + \frac{1}{2}\frac{\partial^2\chi^2}{\partial a_k^2}\delta a_k^2 + \dots \tag{13}$$

Since the second order term determines the deviation from a linear behavior, the constant should be inversely proportional to it:

$$\delta a_k = -\frac{1}{\lambda\frac{\partial^2\chi^2}{\partial a_k^2}}\frac{\partial\chi^2}{\partial a_k} \tag{14}$$

where λ is a dimensionless constant (large if the step should be small). Using Eqs. (9) and (10), this expression reduces to

$$\lambda\alpha_{kk}\delta a_k = \beta_k \tag{15}$$

Eqs. (12) and (15) are then regarded as limiting cases of the following expression:

$$\sum_{j=1}^{M}(\alpha'_{kj}\delta a_j) = \beta_k \tag{16}$$

where

$$\alpha'_{kk} \equiv \alpha_{kk}(1+\lambda); \quad \alpha'_{kj} \equiv \alpha_{kj} \text{ for } k \neq j \tag{17}$$

For small values of λ, Eq. (16) reduces to Eq. (12) corresponding to points close to the minimum. On the contrary, for large λ values, the α'_{kk} terms become dominant and Eq. (16) reduces to Eq. (15) corresponding to points far away from the minimum.

The basic implementation of the Levenberg–Marquardt algorithm includes the following steps:

(1) Pick an initial guess for each of the unknown parameters a_k.
(2) Pick a small initial value for $\lambda = 0.001$.
(3) Compute: α_{kj}, β_k, and $\chi^2(\vec{a})$.
(4) Alter the α_{kj} matrix: $\alpha'_{kk} \equiv \alpha_{kk}(1+\lambda)$; $\alpha'_{kj} \equiv \alpha_{kj}$ for $k \neq j$.
(5) Solve for $\delta\vec{a}$ the linear equations $\sum_{j=1}^{M}\alpha'_{kj}\delta a_j = \beta_k$.
(6) Compute $\chi^2(\vec{a}+\delta\vec{a})$.
(7) If $\chi^2(\vec{a}+\delta\vec{a}) < \chi^2(\vec{a})$ accept the new $\vec{a}$, decrease λ by 10, and go back to (3).
(8) If $\chi^2(\vec{a}+\delta\vec{a}) \geqslant \chi^2(\vec{a})$ reject the new $\vec{a}$, increase λ by 10, and go back to (4).

At each iteration step the following magnitudes, Eqs. (9) and (10), must be evaluated:

$$\beta_k = \sum_{i=1}^{N}\left[\frac{y_i - y(x_i)}{\sigma_i^2}\frac{\partial y(x_i)}{\partial a_k}\right] \tag{18}$$

and

$$\alpha_{kj} = \sum_{i=1}^{N}\left[\frac{1}{\sigma_i^2}\frac{\partial y(x_i)}{\partial a_k}\frac{\partial y(x_i)}{\partial a_j}\right] \tag{19}$$

Note that Eq. (19) does not include the last addend appearing in the full expression for the second derivative of χ^2:

$$\frac{1}{2}\frac{\partial^2\chi^2}{\partial a_k\partial a_j} = \sum_{i=1}^{N}\frac{1}{\sigma_i^2}\left\{\frac{\partial y(x_i)}{\partial a_k}\frac{\partial y(x_i)}{\partial a_j} - [y_i - y(x_i)]\frac{\partial^2 y(x_i)}{\partial a_k\partial a_j}\right\} \tag{20}$$

The main justification for this simplification is that the second order derivative is multiplied by $[y_i - y(x_i)]$, which should be close to zero for random errors.

Finally, the code should include some criterion to finish the calculation and should also display the obtained results. These should include estimated values of the uncertainties of the obtained parameters, which are determined by the diagonal elements of the inverse α_{kj} matrix:

$$\sigma^2(a_k) = [\alpha]_{kk}^{-1} \tag{21}$$

The actual implementation of the program presented in this work includes a series of additional steps, which are described in Supplementary material.

2.2. Dielectric spectrum

In the present implementation of the program, the dielectric spectrum is represented by the following terms:

$$\varepsilon^* = -i\frac{\sigma_0}{\varepsilon_0\omega} + \varepsilon_\infty + \frac{A_d}{1+i\omega\tau_d} + \frac{A_{cc}}{1+(i\omega\tau_{cc})^{1-\alpha_{cc}}} + \frac{A_{cd}}{(1+i\omega\tau_{cd})^{\beta_{cd}}} + \frac{A_{hn}}{\left[1+(i\omega\tau_{hn})^{\alpha_{hn}}\right]^{\beta_{hn}}} \tag{22}$$

In this expression, $\varepsilon^* = \varepsilon' - i\varepsilon''$ is the relative permittivity (the asterisk denotes a complex magnitude); ω is the angular frequency; σ_0 (S/m) is the limiting low frequency conductivity; ε_∞ is the limiting high frequency permittivity; A_d, A_{cc}, A_{cd}, and A_{hn}, are the Debye [2], Cole–Cole [3], Cole–Davidson [5], and Havriliak–Negami [6,7] dispersion amplitudes, while τ_d, τ_{cc}, τ_{cd}, and τ_{hn}, are the corresponding relaxation times (s). Note that the symbols used for the exponents appearing in Eq. (22) have no universal acceptance: in [4], for example, the exponent β_{cd} is replaced by $1-\alpha_{cd}$ while in [8], the exponent α_{hn} is replaced by $1-\alpha_{hn}$. This should pose no problem, however, for the use of the proposed program.

Eq. (22) includes 14 parameters: σ_0, ε_∞, 4 dispersion amplitudes, 4 relaxation times, and 4 exponents. Each addend in this equation can be selected or rejected. Furthermore, each parameter of the selected addends can be chosen to be adjustable or to remain fixed. This last feature makes it possible to make transformations of the different terms: setting $\beta_{cd} = 1$ and making it fixed transforms the Cole–Davidson dispersion into an additional Debye dispersion, for example.

3. Discussion

In order to evaluate the proposed program we shall compare it with the original code presented in [1], considering a dielectric spectrum characterized by two Cole–Cole dispersions with parameter values given in Table 1. Note that since Eq. (22) only includes a single Cole–Cole term, the second one is represented by a Havriliak–Negami dispersion with $\beta_{hn} = 1$ (and kept fixed during the calculation). This corresponds to a Cole–Cole dispersion with $A_{cc2} = 10$, $\tau_{cc2} = 10^{-9}$ s, and $\alpha_{cc2} = 0.1$.

The spectrum extends from 1 kHz to 1 GHz with 10 data points per decade that are equally spaced in a log scale.

We first consider that the data points are calculated using the above parameters without any random errors, white diamonds in Figs. 1 and 2. Under these conditions both implementations are able to fit exactly the model spectra to the data points with the following differences:

Table 1
Dielectric spectrum terms and parameter values used for the calculation of the sample data.

Conductivity	$\sigma_0 = 10^{-4}$ S/m			
Epsilon infinity	$\varepsilon_\infty = 5$			
Cole–Cole	$A_{cc} = 100$	$\tau_{cc} = 10^{-6}$ s	$\alpha_{cc} = 0.3$	
Havriliak–Negami	$A_{hn} = 10$	$\tau_{hn} = 10^{-9}$ s	$\alpha_{hn} = 0.9$	$\beta_{hn} = 1$ kept fixed

(1) The original code requires two calculations, one for the real part of the data (providing all the parameters except σ_0) and another for the imaginary part (providing all the parameters except ε_∞).
(2) The presented program requires just one calculation using both the real and the imaginary parts of the data and providing all the parameters at once. However, it can also be used in the same way as the original code.
(3) The ranges of the allowed guess values that lead to the expected results are essentially the same for both programs except for the parameters given in Table 2.

For both programs, the calculations were performed using only the real part of the data. The guess values of all the parameters, except the one whose range was being determined, were set initially to their "exact" values (but allowed to change during the calculation).

As can be seen, the original code requires a much higher precision in the determination of the guess values than the proposed program. In the case of the relaxation times, the differences attain several orders of magnitude. This is a very important advantage of the proposed program since, in practice, it is very difficult to provide precise initial guesses of all the parameters.

The observed differences occur because of the following features introduced in the proposed program:

(1) At each iteration step when new parameter values are determined, these values are only accepted when they fall within the allowed parameter bounds. For example, all the dispersion amplitudes must be positive.
(2) Rather than using the relaxation times as adjustable parameters, the program uses internally their logarithm. This way of doing greatly improves the calculation stability since it automatically avoids negative relaxation time values during the fitting procedure (see Supplementary material for more details).

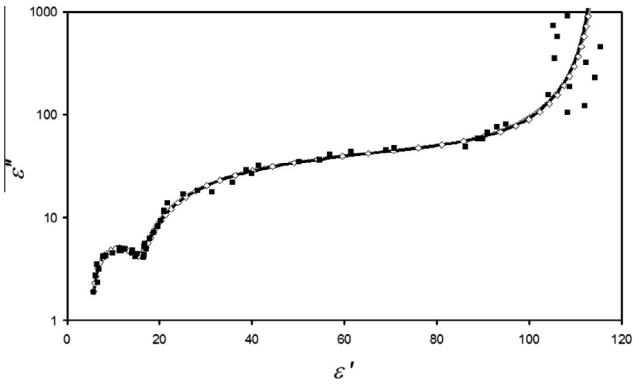

Fig. 1. Cole–Cole plot with logarithmic scale for the ordinate, necessary due to the large low frequency ε'' values arising from the conductivity term, Eq. (22). Data points obtained with parameters in Table 1 without random errors (white diamonds) and with 5% relative uncertainties (black squares). Spectra drawn using parameters from Table 3: original code (gray line) and proposed program (black line). Both lines overlap almost exactly.

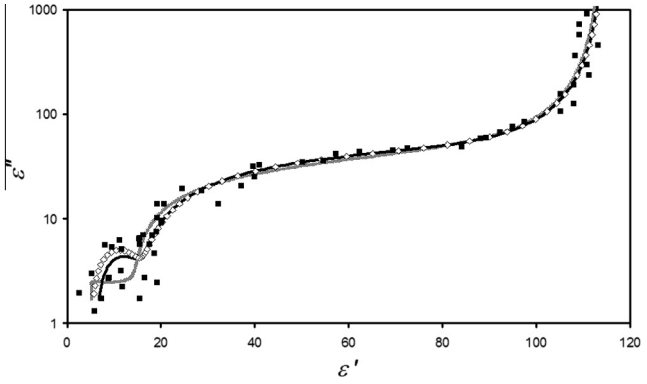

Fig. 2. Cole–Cole plot with logarithmic scale for the ordinate, necessary due to the large low frequency ε'' values arising from the conductivity term, Eq. (22). Data points obtained with parameters in Table 1 without random errors (white diamonds) and with absolute uncertainties of 2.5 dielectric units (black squares). Spectra drawn using parameters from Table 4: original code (gray line) and proposed program (black line).

Table 2
Initial guess value ranges that lead to the expected results given in Table 1.

	Original code	Presented program
τ_{cc} (s)	$3 \times 10^{-7} \leqslant guess \leqslant 3 \times 10^{-6}$	$5 \times 10^{-8} \leqslant guess \leqslant 3 \times 10^{-1}$
α_{cc}	$0 \leqslant guess \leqslant 5 \times 10^{-1}$	$0 \leqslant guess \leqslant 7 \times 10^{-1}$
τ_{hn} (s)	$2 \times 10^{-10} \leqslant guess \leqslant 9 \times 10^{-9}$	$4 \times 10^{-14} \leqslant guess \leqslant 1 \times 10^{-6}$
α_{hn}	$4 \times 10^{-2} \leqslant guess \leqslant 1$	$2 \times 10^{-4} \leqslant guess \leqslant 1$

We next consider the influence of random errors of the sample data on the performance of the considered programs in the following simple cases:

(1) Relative uncertainties of 5%.
(2) Absolute uncertainties of 2.5 dielectric units.

In both cases, the data were generated using the procedure described in Ref. [1]. In order to assure the best possible fit to the data, the programs were run using the "exact" parameter values, Table 1, as initial guesses.

The results obtained using data with *Relative* uncertainties, black squares in Fig. 1, are given in Table 3. As expected, the two programs lead to very similar parameter values: the gray and black lines in Fig. 1 almost overlap. This happens because both programs implement relative uncertainty values that match the data uncertainties. However, the original code has the disadvantage of leading to two values for each parameter (except for σ_0 and ε_∞) since real and imaginary parts of the data have to be processed individually. Note that the parameter values provided by the proposed program, Table 3, always lie in between these two values.

Table 3
Parameter values obtained using data with relative 5% uncertainties.

Part of data	Original code		Presented program
	Real	Imaginary	Real and imaginary
σ_0 (S/m)	NA	0.999×10^{-4}	0.996×10^{-4}
ε_∞	5.51	NA	4.99
A_{cc}	101	98.3	99.7
τ_{cc} (s)	1.03×10^{-6}	0.967×10^{-6}	1.00×10^{-6}
α_{cc}	0.314	0.295	0.302
A_{hn}	8.75	10.2	9.92
τ_{hn} (s)	0.963×10^{-9}	1.02×10^{-9}	1.00×10^{-9}
α_{hn}	0.986	0.872	0.884

Table 4
Parameter values obtained using data with absolute 2.5 dielectric unit uncertainties.

Part of data	Original code		Presented program
	Real	Imaginary	Real and imaginary
σ_0 (S/m)	NA	0.977×10^{-4}	1.00×10^{-4}
ε_∞	4.95	NA	6.03
A_{cc}	101	112	99.8
τ_{cc} (s)	0.952×10^{-6}	1.35×10^{-6}	0.992×10^{-6}
α_{cc}	0.296	0.382	0.296
A_{hn}	8.29	3.96	8.99
τ_{hn} (s)	1.28×10^{-9}	0.398×10^{-9}	1.09×10^{-9}
α_{hn}	1.17	0.918	0.846

The results obtained using data with *Absolute* uncertainties, black squares in Fig. 2, are given in Table 4. The two programs lead now to quite different parameter values since the original code can only use relative uncertainties, while the proposed program uses in this case absolute uncertainties that match the data uncertainties. The differences are particularly important for the high frequency dispersion since the relative data scatter is most important for low permittivity values, Fig. 2. Note the huge difference between the two τ_{hn} values obtained using the original code, Table 4. Also note in this table the α_{hn} value greater than unity, which is possible using the original code since it does not perform any bounds checking.

These results illustrate the importance of including into the fitting procedure the uncertainties of the data values. Actually, relative and absolute uncertainties are just the two simplest situations. In real data, the uncertainties are usually much more complex and can depend on the frequency and on the impedance of the measurement cell. For instance, the modulus of the impedance could have a relative uncertainty while the uncertainty of the phase angle has an absolute value. In order to obtain the best fit parameters, the user should use the information provided by the instrument manufacturer, calculate the ε' and ε'' uncertainties, include these results in the data file, and select in the proposed program the "use uncertainties from file" option.

4. Conclusion

A program is presented that allows to describe dielectric spectroscopy data by means of a superposition of the most frequently used dispersion functions: Debye, Cole–Cole, Cole–Davidson, and Havriliak–Negami, plus a conductivity term. The program is based on the theory given in [1], that has been adapted to the dielectric dispersion problem including data input from file, output of results to file, and the coding of the above mentioned dispersion functions.

It incorporates, furthermore, a series of extensions with respect to the implementation presented in that reference, the most important being the possibility to include data uncertainty values, the use of parameter bounds during the fitting process, and the possibility to fit simultaneously the real and the imaginary parts of the data. Other improvements, as well as a full description of the program can be found in Supplementary material.

We hope that this work will contribute to save unnecessary repetitive work and, hopefully, improve the quality of fitted dielectric dispersion parameters.

Acknowledgments

The author wishes to acknowledge financial support for this work provided by CIUNT (Project 26/E419) of Argentina.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.jcis.2013.12.031.

References

[1] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in Fortran 77. The Art of Scientific Computing, second ed., Cambridge University Press, New York, 1995.
[2] P. Debye, Polar Molecules, Dover Publications, New York, 1929.
[3] K.S. Cole, R.H. Cole, J. Chem. Phys. 9 (1941) 341–351.
[4] D.W. Davidson, R.H. Cole, J. Chem. Phys. 18 (1950) 1417.
[5] D.W. Davidson, R.H. Cole, J. Chem. Phys. 19 (1951) 1484–1490.
[6] S. Havriliak, S. Negami, J. Polym. Sci. C14 (1966) 99.
[7] S. Havriliak, S. Negami, Polymer 8 (1967) 161–210.
[8] C.J.F. Böttcher, P. Bordewijk, Theory of Electric Polarization. Dielectrics in Time-Dependent Fields, vol. II, Elsevier, New York, 1978.