User manual for DEC*i*M: Program for the Determination of Equivalent Circuit Models

Hendrik P. Rodenburg and Peter Ngene
Utrecht University

May 15, 2024

Table of contents

1 Introduction

1.1 Equivalent circuit modelling

The analysis of (electrochemical) impedance spectra usually involves fitting a model to the impedance data. In most cases, such a model takes the form of an equivalent circuit model, *i.e.* an electrical circuit whose impedance is similar to that of the sample. Equivalent circuit models may contain many different circuit elements, including capacitors, inductors, resistors, and more complex elements such as Warburg elements and constant phase elements, which can be arranged in many different ways. The circuit elements typically represent physical processes, and the values of the elements' parameters inform on the properties of the measured system. For instance, a resistor can represent the resistance of an electrolyte. If the dimensions of the electrolyte are known and its resistance can be determined, then its conductivity can be calculated. Similarly, a capacitor can represent a double layer capacitance, which can be useful for determining the electrochemically active surface area of an electrode. However, extracting the value of a parameter such as a resistance or capacitance usually requires the entire impedance

spectrum to be modeled, which can be challenging for more complex systems. In this manual, a program for the Determination of Equivalent Circuit Models (DEC*i*M) is presented, which can be used to model impedance spectra.

1.2 DEC*i*M's purpose

DEC*i*M is designed to allow both expert and non-expert users to adequately model impedance spectra. The program offers manual and automatic fitting, three different methods for creating equivalent circuit models, data validation, and different options for visualization. For non-expert users, there is a simple workflow consisting of drawing a circuit or typing a circuit string followed by manual fitting and a quick 'simple refinement'. For expert users, there is Z-HIT data validation, and there is the option to define a custom model (which can be a transmission line model or even a non-circuit model). Additionally, one may separately refine parameters in frequency ranges of choice with weighting schemes of choice. During and after data processing, there are seven different types of plots available for visualization, with the option of including multiple data sets for comparison. Finally, the data and model can be saved to a result file which is human-readable and can be loaded by DEC*i*M.

2 Installation

2.1 Prerequisites

DEC*i*M is written in Python[1] and requires Python 3.10 or a more recent version of the Python interpreter. In addition, the following packages are required:

- NumPy[2]
- SciPy[3]
- Matplotlib[4]
- Optax[5]
- tkinter (Python Standard Library)
- functools (Python Standard Library)
- copy (Python Standard Library)
- webbrowser (Python Standard Library)

2.2 Installing DEC*i*M

Installing DEC*i*M simply involves downloading DEC*i*M from GitHub.[6] All necessary files for DEC*i*M to start are in the 'src' folder. These files are:

- DECiM.py
- ecm_circuits.py
- ecm_custom_models.py
- ecm_datastructure.py

- ecm_file_io.py
- ecm_fit.py
- ecm_helpers.py
- ecm_history.py
- ecm_manual.py
- ecm_plot.py
- ecm_user_input.py
- ecm_zhit.py

In addition, a file containing circuit presets, 'ecm_presets.decim_circuits' should be present, as well as a file containing information about data file structure, 'ecm_datafiles.decim_specification'.

2.3 Starting DEC*i*M

After the installation has been completed, DEC*i*M can be started by navigating to the `DECiM/src` folder on the command line and then (provided you can start Python 3 as `python`) typing `python DEC*i*M.py`, followed by pressing RETURN/ENTER.

3 Equivalent circuit model selection

By default, DEC*i*M chooses (RQ) as the circuit, but it is recommended to define a new equivalent circuit model before data are loaded into DEC*i*M. There are multiple ways to do this: typing a circuit string, choosing a circuit preset, drawing a circuit or defining a custom model. Once an equivalent circuit model has been chosen, data can be loaded and the model parameters' values can be determined.

3.1 Typing a circuit

A circuit can be described in text as a circuit string following a notation similar to that of Boukamp.[7] In DEC*i*M, the symbols R, L, C, Q, O, S, G, and H are used to refer to the resistor, inductor, capacitor, constant phase element, open Warburg, short Warburg, Gerischer, and Havriliak-Negami elements, respectively. They can be connected in series by placing them between curly brackets '{}' or in parallel by placing them between parentheses '()'. Two examples are given in Figure 1.

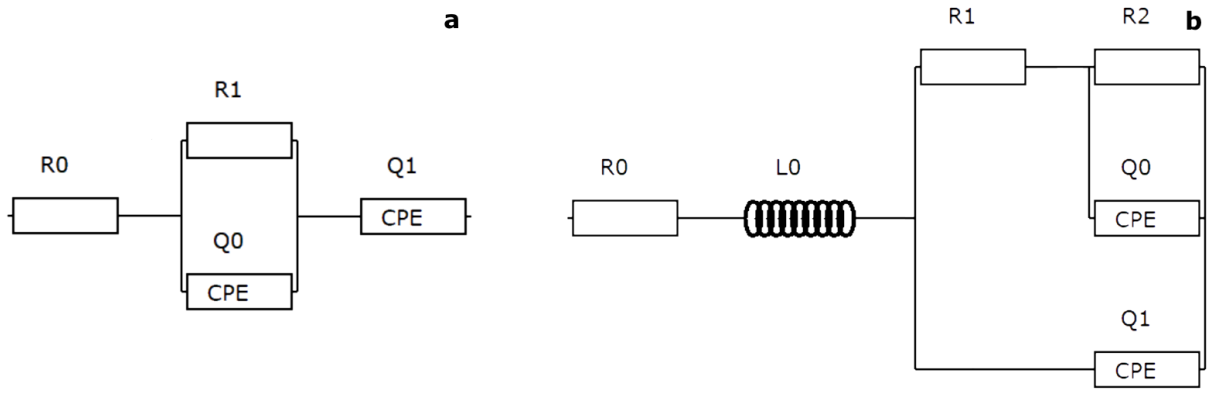

Figure 1: Two example circuits. a) $\{R_0(R_1Q_0)Q_1\}$; b) $\{R_0L_0(Q_1\{R_1(R_2Q_0)\})\}$.

The circuit typing input box can be accessed via the 'Circuit' menu as 'Type circuit'. A new window will be opened in which the circuit string can be typed. When the window is closed, the text is read and validated. If the text is a valid circuit string, then the circuit is updated.

3.2 Circuit presets

Circuit strings can also be typed in the circuit_presets.decim_circuits text file. One circuit string can be entered on each line. On program startup, DEC*i*M reads this file and places every valid circuit string in the 'Circuit' menu, from where the circuit can be selected.

3.3 Drawing a circuit

It is possible to draw a circuit diagram in DEC*i*M by clicking the circuit button below the slider or by going to the 'Circuit' menu and clicking 'Draw circuit'. A new window will open. On the left, there are some controls, and on the right, there is a drawing canvas. In the top left, the connection mode (series or parallel) is indicated. Drawing in series mode will autmatically connect elements drawn to the left or right of each other on the canvas, whereas drawing in parallel mode will automatically connect elements drawn above or below each other. This way, the elements are grouped into so-called *units*: groups of elements connected in parallel or series. When elements that were drawn in different modes need to be connected, their units must be merged by activating merge mode. This will create a new unit into which the smaller units are merged. The connection mode determines how the smaller units are connected.

To draw a new element or to merge units, you must click on the drawing canvas. If merge mode is enabled and there are units that can be merged, the highlighted units will be merged. If merge mode is disabled, clicking on the canvas will place the element on the canvas at the mouse position (first element) or at the closest available '+' position (all other elements). To illustrate the drawing process, an example in which the RL(RQ) circuit is drawn, is given below in Figures 2–11.

Figure 2: First, switch to series mode by clicking 'Series/parallel mode', then select a resistor from the element dropdown.

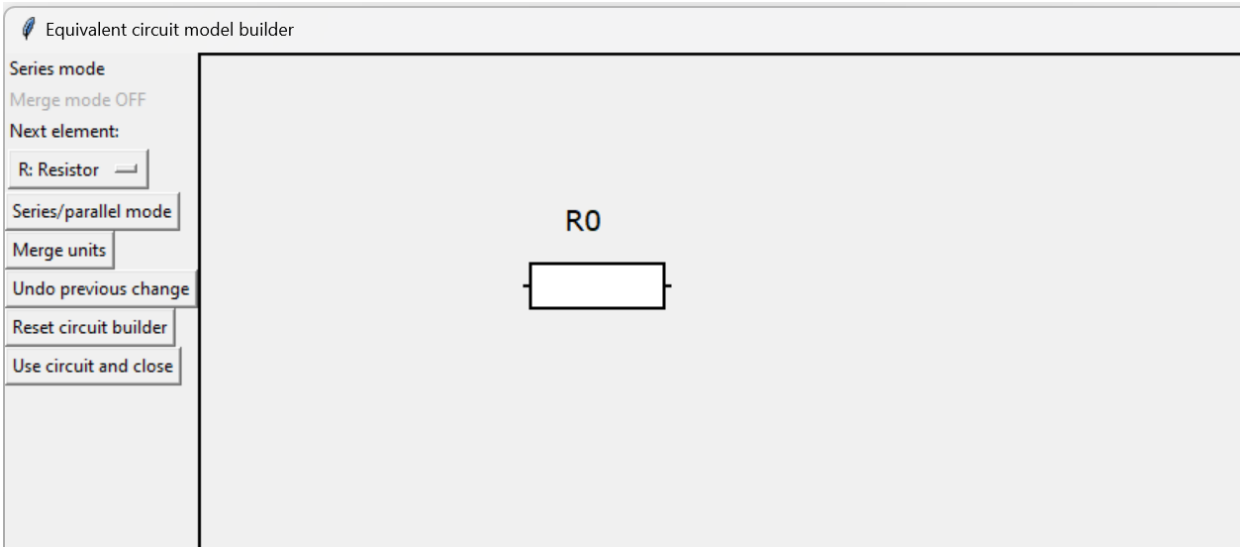

Figure 3: Draw the resistor by clicking on the canvas.

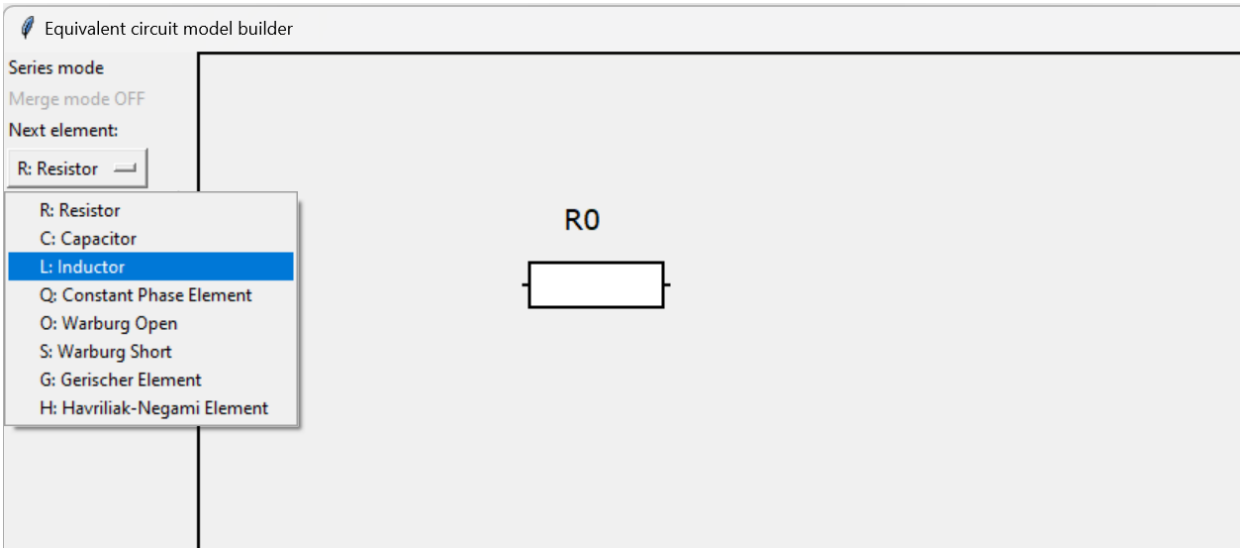

Figure 4: Select an inductor and draw it on the canvas by clicking to the right of the resistor.

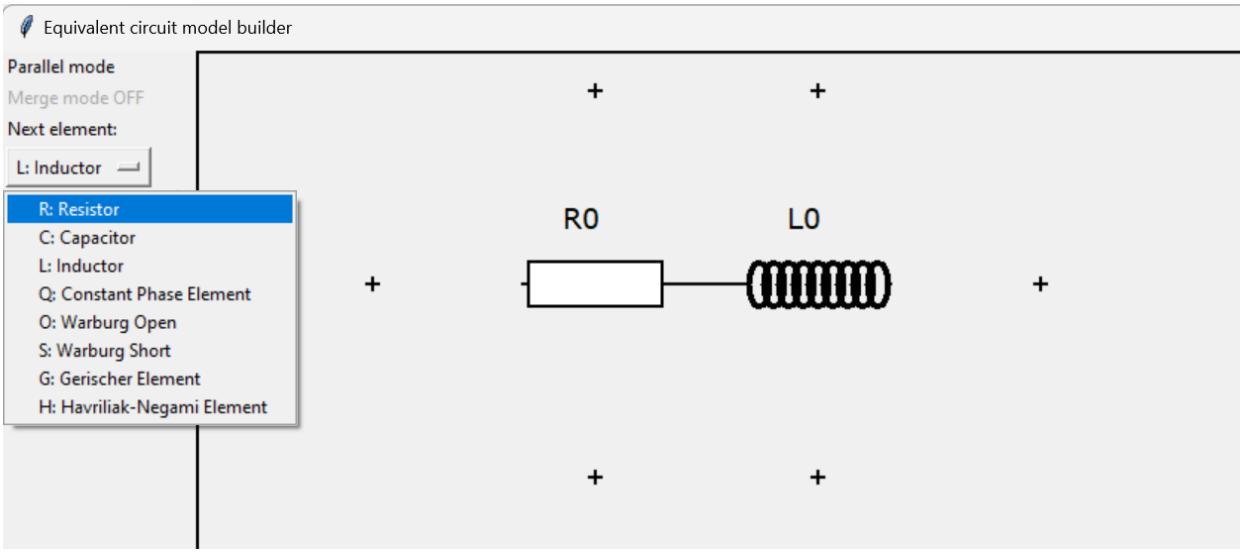

Figure 5: Switch to parallel mode and select a new resistor.

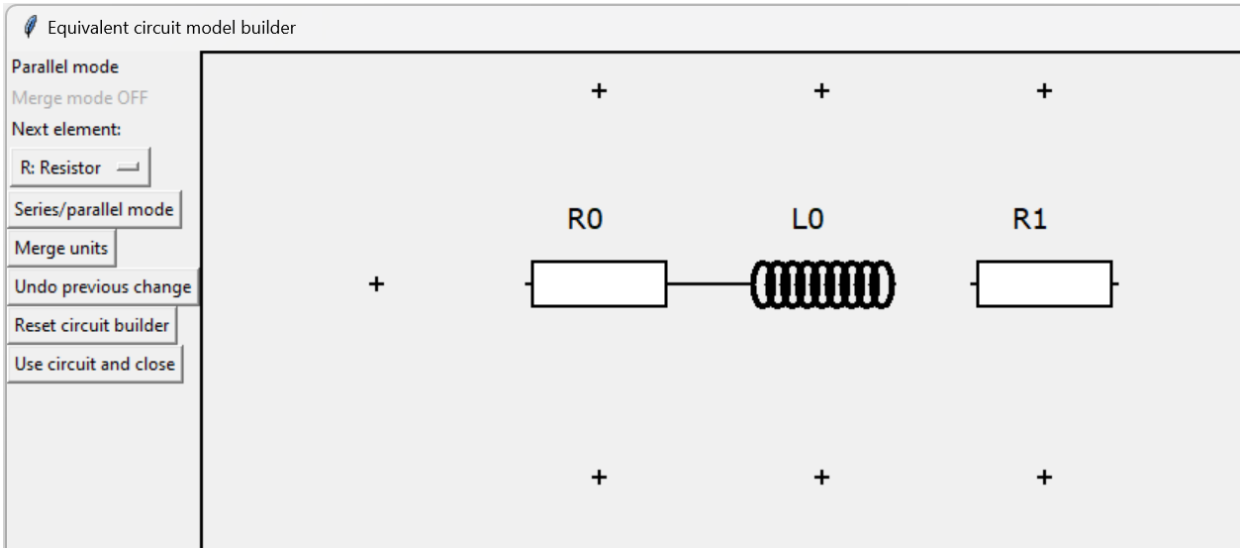

Figure 6: Draw the resistor by clicking to the right of the inductor. The resistor is in a new parallel unit and is not connected to the other elements, which are in a series unit.

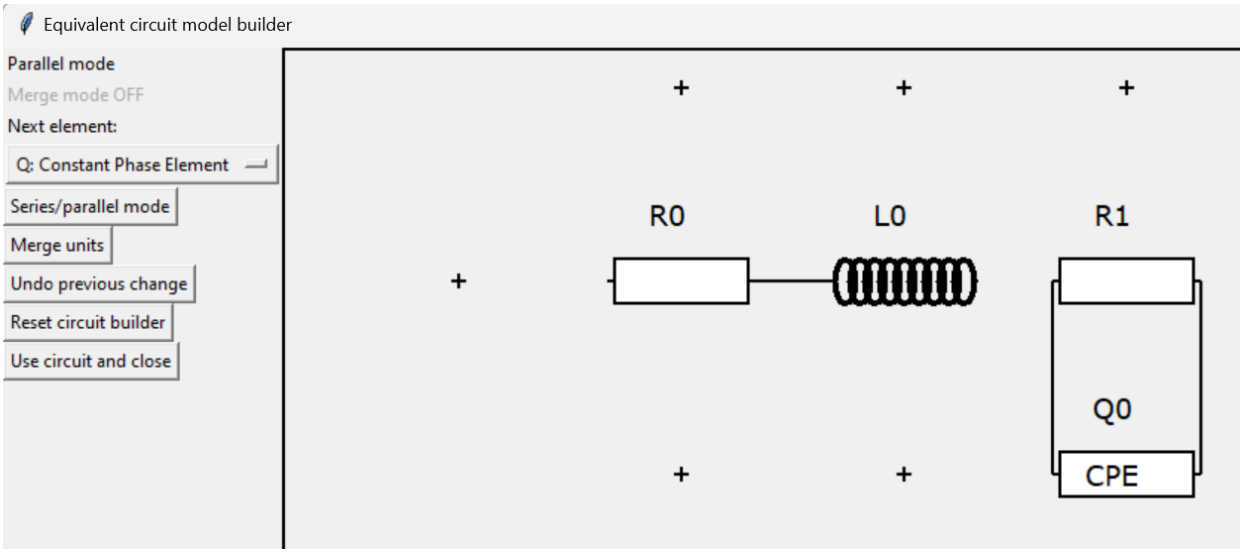

Figure 7: Select a constant phase element from the element dropdown and draw it below the new resistor. It will be automatically added to the same parallel unit, and the connections will be drawn.

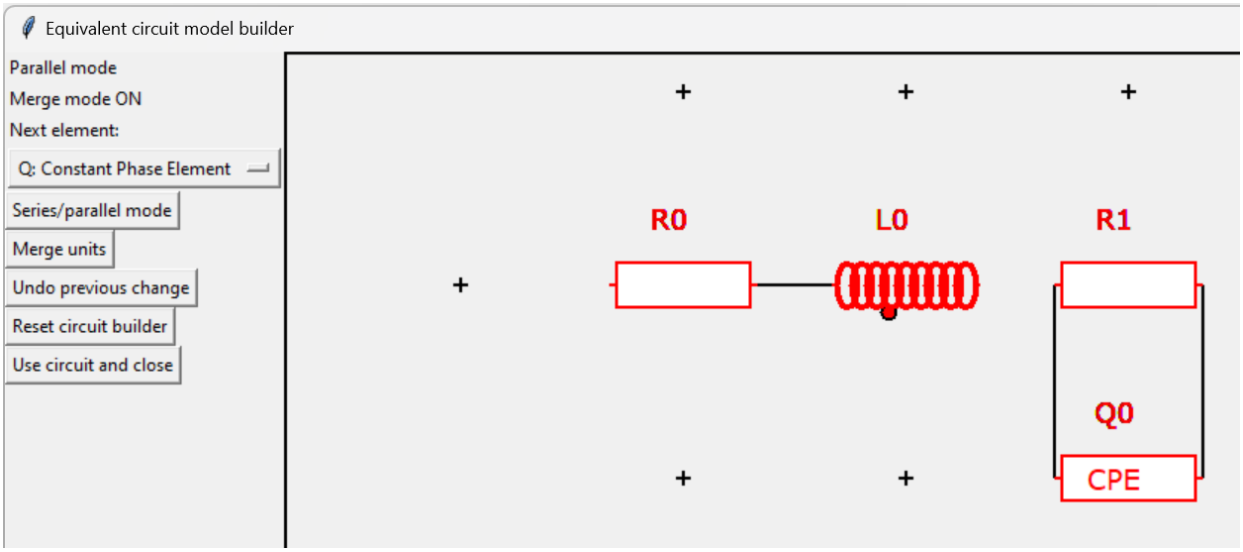

Figure 8: Enable merge mode by clicking 'Merge units' and move the mouse to the red dot. Click on or near it to try to merge the units.

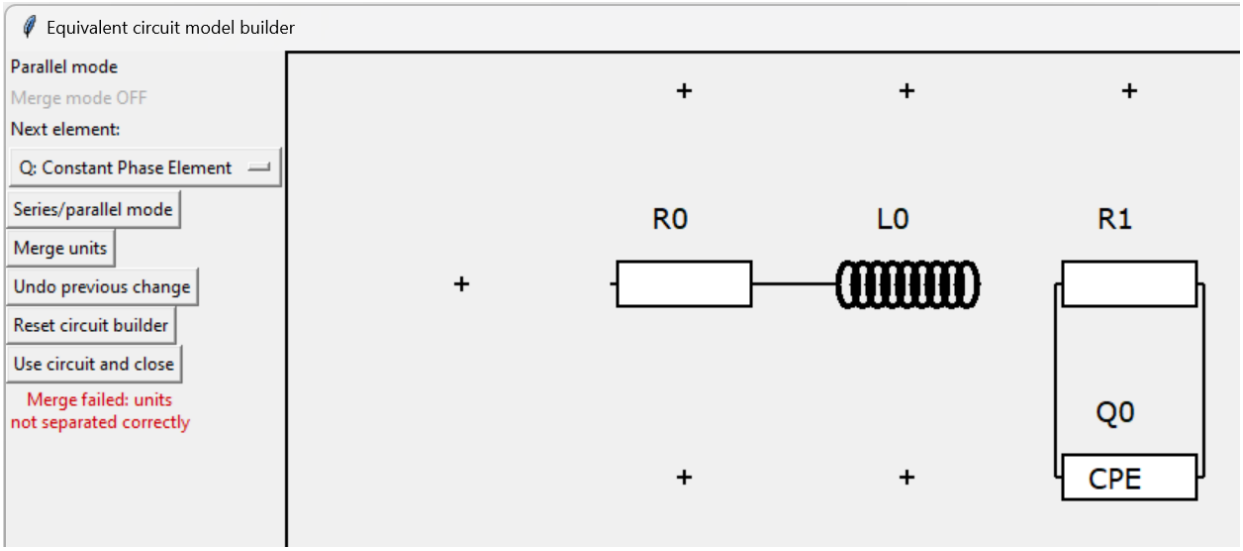

Figure 9: The units are not merged because they are only horizontally separated. To merge two units in parallel, they must be vertically separated. Similarly, merging two units in series requires them to be horizontally separated.

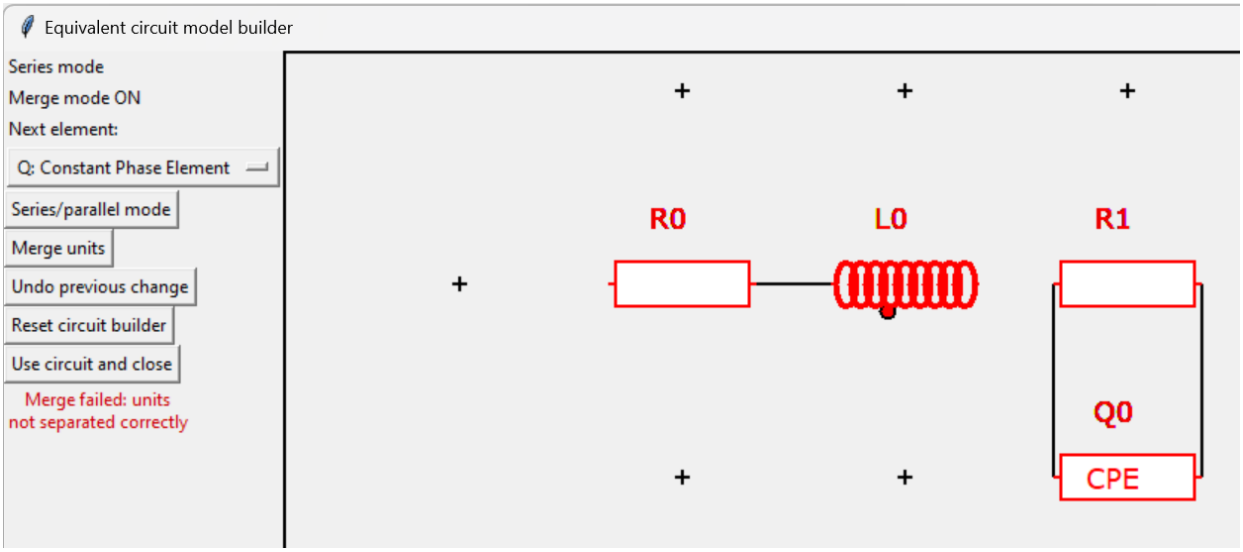

Figure 10: Try again in series mode.

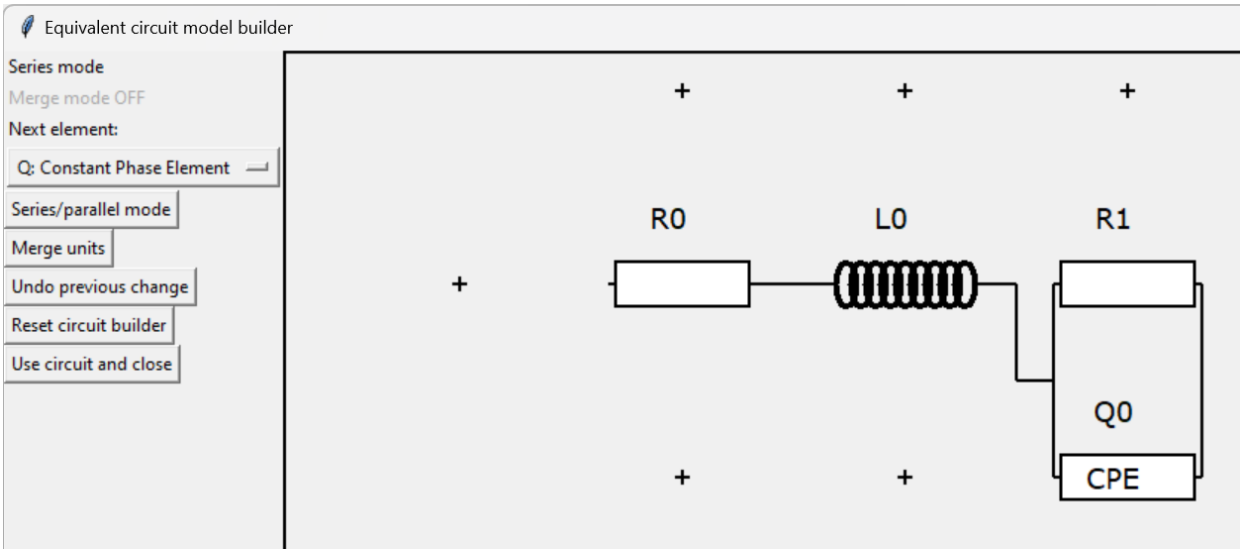

Figure 11: The two units are now merged and the circuit is complete.

With the circuit made, click 'Use circuit and close' to select it as the model for the analysis.

3.4 Custom models

Custom models can be defined in 'ecm_custom_models.py'. Custom models are functions that return the impedance based on the fit parameters, but do not necessarily involve any circuit diagram. This is useful for different physical models of the impedance, or for transmission line

models, which cannot be defined using the circuit typing or drawing interfaces. A custom model can be created as follows:

1. Create a function *Z(fp, freq)*, where *fp* is a list of fit parameters and freq a NumPy array of frequencies. The function should return a NumPy array of complex impedances.
2. Add the function to the custom_model_diagrams dictionary as *'name': ('diagram', Z)*, where 'diagram' is a circuit string that contains as many parameters as the model uses.

Once the model has been defined, it can be accessed from the 'Circuit' menu, where the name entered into the custom_model_diagrams dictionary will be displayed.

Note that, if you want to use optimizers provided by Optax, you should use JAX.NumPy[5] instead of NumPy. This will break the automatic initial guess and simple refinement options. It may be useful to define your custom model twice: once with NumPy and once with JAX.NumPy.

4 Loading and saving files with impedance data

4.1 Data files

Data files can be loaded from the 'File' menu in the menu bar at the top of the program window. DEC*i*M expects data files to be text files consisting of columns separated by whitespace characters (tabs, spaces) or commas. The first line can be a header specifying the quantities and units; DEC*i*M will skip any lines that do not consist solely of numbers, separators and line breaks. By default, DEC*i*M expects the first value to be a linear frequency (f, **not** $\omega = 2\pi f$) in units of Hz, the second value to be the real impedance component $\mathrm{Re}[Z]$ in units of Ω and the third value to be the imaginary impedance component $\mathrm{Im}[Z]$ in units of Ω. Alternative data file layouts can be defined using the 'Specify data file layout…' option in the 'File' menu, which will open the window shown in Figure 12. Doing so will modify the settings file 'ecm_datafiles.decim_specification'.

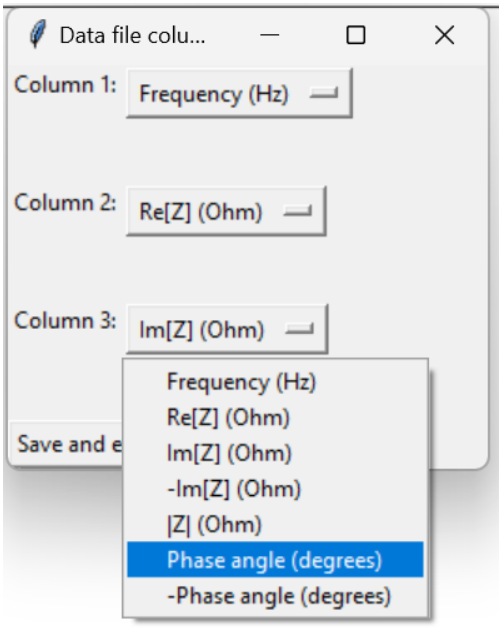

Figure 12: The data file specification window.

Data files can only be loaded by DEC*i*M, not saved. Even if the data are tranformed with Z-HIT, it is not possible to save a new data file. Only a result file (.recm2) may be saved.

4.2 Result files

Result files (.recm2 file extension) may be saved and loaded by DEC*i*M, via the 'File' menu. They contain more information than data files: besides data, also limited statistical information, model parameters and points to plot the model are provided. The structure of a result file is as follows:

1. '>CIRCUIT DEFINITION' header.
2. Text describing the circuit diagram (circuit string or custom model information).
3. '>MODEL PARAMETERS' header.
4. List of model parameters' names, values and indices; the latter are important for fitting.
5. '>STATISTICAL DATA' header.
6. List of number of parameters, number of frequencies, degrees of freedom, observation-to-parameter ratio and the proprotionally weighted sum of the squares S_v (a measure of the goodness of fit).
7. '>IMPEDANCE DATA' header.
8. 'Frequency (Hz), Re(Z) / Ohm, Im(Z) / Ohm' header.
9. Impedance data as described by the above header.
10. '>IMPEDANCE FIT' header.
11. 'Frequency (Hz), Re(Z) / Ohm, Im(Z) / Ohm' header.
12. 500 points of impedance that describe the model, formatted as described by the above header.

Result files are designed to be human-readable and to be a useful step towards preparing publication-quality figures. It is not necessary to evaluate the impedance using the model description and parameter values after the result file is generated. The model can simply be plotted from the data points below the '>IMPEDANCE FIT' header.

5 Data validation

5.1 Z-HIT transform

For data validation, DEC*i*M provides the Z-HIT transform[8] in the 'Calculate menu'. Clicking 'Perform Z-HIT transform' will open a new window with complex plane and Bode plots in the upper half and two entry boxes and two tick boxes, three buttons, and then two more entry boxes in the lower half. With the upper entry boxes, the desired frequency range for data validation can be selected, and edge artifacts generated by the filter can be cut with the tick-boxes. The buttons can be used to start the calculation and to close with or without accepting the result. The lower two entry boxes can be used to tune the filter that is used to smoothen the phase.

The algorithm for the Z-HIT transform is:

1. Smoothen the phase data with a Savitzky-Golay filter.[9] Via the lower entry boxes, you can set the window length and polynomial order used by the filter. By default, these are set to 50 and 3, respectively. For description of the filter's implementation, see the SciPy[3] documentation for the *scipy.signal.savgol_filter*.
2. Calculate $\ln|Z|$ from the phase spline.
3. Determine the constant of integration in the Z-HIT transform.
4. Display the result.

If you only want to inspect the Z-HIT transform, you should close the window with the 'Reject and close button'. If you wish to use the transformed data instead of the measured data, click 'Accept and close'.

6 Manual parameter adjustment

The user interface of the main DEC*i*M window contains controls for manual parameter adjustment below the plots. These controls allow the selection of parameters, modification of parameters via a slider and directly setting a parameter's value by typing it. The controls used for this are shown in Figure 13.

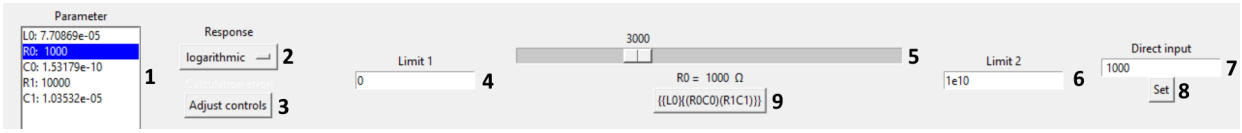

Figure 13: The controls for manually changing parameters' values. 1) Parameter list box. 2) Slider response dropdown. 3) Slider controls adjustment button. 4) First slider limit (can be upper or lower limit). 5) Slider. 6). Second slider limit. 7) Direct input box. 8) Parameter value set button. 9) Circuit selection button.

6.1 Parameter list box

The parameter list box is contains a list of all parameters and their values in the current equivalent circuit model. Clicking a parameter in this list makes it adjustable by the slider and 'Direct input' box. If the equivalent circuit model is changed, the parameter dropdown is automatically updated.

6.2 Slider controls

After a parameter has been chosen, it can be adjusted by the slider. It is, however, important to set the lower and upper limits of the parameter first. For example, resistors typically take values between 0 Ω and 10^{10} Ω, while capacitors normally take values between 1 F and 10^{-12} F. These limits can be set in the 'Limit 1' and 'Limit 2' text boxes. However, note that for values between 0 and 1 on a log-scaled axis, 'Limit 1' should be 0, not 1; 'Limit 2' should be the lower limit (*e.g.* 10^{-12}). After the limits have been set correctly, the slider response can be set with the menu under 'Response'. Usually, a logarithmic response is appropriate, but in the case of exponents (CPE n, Havriliak-Negami b & g), a linear response is often better.

To save time in setting up the slider limits and response, the 'Adjust controls' button can be used to automatically set up the slider controls. The limits and slider response will be set to typical values for the selected parameter. For any parameter which has units of Ω (R, O, S, G, and H), 'Limit 1' will be 0, 'Limit 2' will be 10^{10} and the response will be logarithmic. For C, Q, and L, and the second parameter of G and H (m and t, respectively), 'Limit 1' will be 0, 'Limit 2' will be 10^{-12} and the response will be logarithmic. For the second parameter of O and S (k and l, respectively), 'Limit 1' will be 0, 'Limit 2' will be 10^{6} $s^{-1/2}$ and the response will be logarithmic. Finally, for a CPE exponent n and the Havriliak-Negami exponents b and g, 'Limit 1' will be 0, 'Limit 2' will be 1 and the response will be linear.

6.3 Slider and Set button

With the slider controls ready, the value of the selected parameter can be changed through the slider in three different ways:

1. Dragging the slider with the left mouse button pressed. The slider will move quickly.

2. Holding the left mouse button while the cursor is to the left or right of the button inside the slider. The slider will move slowly.
3. Clicking at any point on the slider bar with the right mouse button. The slider will immediately move to the indicated position.

The value of the parameter will be automatically updated and displayed below the slider. The plots will also be updated.

As an alternative to the slider, the parameter value may also be updated with the 'Direct input' text box. To do this, first enter the desired value of the parameter and then press the 'Set' button. The parameter's value will immediately be changed to the value you entered and the plots will be updated.

6.4 Guidelines for manual fitting

Manual fitting can be difficult and time-consuming, especially for more complex spectra. Fortunately, there is a systematic approach that works for many spectra. It works as follows:

1. Set all capacitors C and CPEs Q to 1 F, CPE and Havriliak-Negami exponents n, b, and g to 1, inductors L to 10^{-12} H, and resistors R to 1 Ω. Set Warburg, Gerischer, and Havriliak-Negami elements O, S, G, and H to 1 Ω as well and don't touch the parameter influencing the time constant (k, l, m, and t).
2. Estimate the resistors' values from the ends of the semicircles in the complex plane and the heights of the $|Z|$ plateaus in the Bode plot.
3. Decrease the C and Q values of the capacitors and CPEs that are connected in parallel with the resistors until semicircles start to appear. Make sure the phase and amplitude are fitted well.
4. For CPEs, adjust the exponent n until the fit is as close as possible.
5. Increase L until the fit no longer improves.
6. For Warburg and Gerischer elements, estimate O, S, or G from the real part of the electrode response. Then carefully adjust k, l, or m until the electrode response is fitted well.
7. Fine-tune all parameters.

Note that the approach for Havriliak-Negami elements is different; there, H should be tuned first, followed by t, and finally b and g.

7 Automatic model refinement

In many cases, it is useful to automatically refine the parameters obtained by manual fitting. DEC*i*M includes two options for this: the simple and advanced refinement. These can be found in the 'Calculate' menu in the menu bar.

7.1 Automatic initial guess

In some cases, particularly for relatively simple models, it is possible to generate a reasonably accurate initial guess with the 'Automatic initial guess' option from the 'Calculate' menu. This option will start multiple refinements from different starting positions and pick the best solution. It can save much time spent on manual fitting. While the calculation is running, the text 'Guessing…' will be displayed in the complex plane plot. Do not click anywhere inside the window while this text is displayed to avoid DEC*i*M becoming unresponsive.

7.2 Simple refinement

The simple refinement refines all parameters without applying any weights. It is mainly useful for simple circuits. Only the frequency range can be controlled; this is done with the 'Set simple refinement frequency range' option in the 'Calculate' menu. After clicking 'Refine solution (simple)', the refinement will immediately begin, and the text 'Refining…' will be displayed in the complex plane plot. Do not click anywhere inside the window while the refinement is running; this could cause DEC*i*M to become unresponsive. However, the refinement is usually fast, as its length is fixed to 10000 iterations. After the refinement is completed, the parameters and plots are automatically updated.

7.3 Advanced refinement

The advanced refinement can be accessed via the 'Advanced refinement…' option in the 'Calculate' menu. Clicking this option will launch a new window with three plots and three columns of controls. The plots are the residuals, the complex plane plot and the Bode plot of the impedance spectrum. The controls are shown in Figure 14.

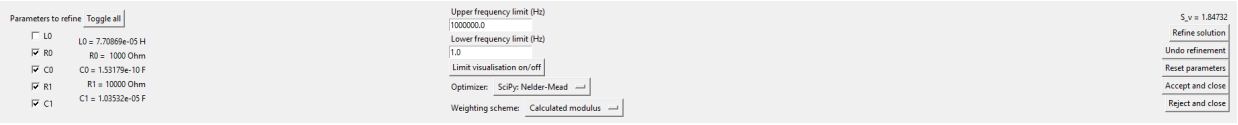

Figure 14: The refinement controls.

It is possible to select and deselect parameters to be refined (with the `SciPy: Nelder-Mead` optimizer only), to set the frequency range, to choose a weighting scheme, and to choose an optimizer. The frequency limits can be shown in the residuals and Bode plots with the 'Limit visualisation on/off' button. The available weighting schemes are 'Unit' (no weights), 'Observed modulus' (modulus weighting based on measured data), 'Calculated modulus' (modulus weighting based on model), 'Observed proportional' (proportional weighting based on measured data), and 'Calculated proportional' (proportional weighting based on model). Unit weighting (the default) is fast, but not as accurate as modulus or proportional weighting. Proportional weighting is slow and can run into problems when the imaginary impedance component is close to 0. Therefore,

modulus weighting is recommended, with modulus weighting based on the model ('Calculated modulus') typically being the best performing option.

Once all desired parameters have been selected, the limits have been set and a weighting scheme has been chosen, the solution can be refined with the 'Refine solution' button. If the result is not as desired, then the parameters can be reset with 'Undo refinement' to go back to the previous refinement result, or 'Reset parameters' to go back to the parameters as they were when the window was launched. If no satisfactory result can be obtained at all or you wish to cancel the refinement, then 'Reject and close' will close the window and discard the refinement result. If, however, an acceptable result is obtained, 'Accept and close' will update the model parameters in the main part of DEC*i*M and update the plots there.

7.4 Undoing refinements

In case a simple or advanced refinement leads to an undesirable result, the refinement can be undone with the 'Undo refinement' button in the 'Calculate' menu. It is possible to go back more than once; after every refinement, the parameters are saved and can be recovered. This is not done for slider movements, as these are more easily undone by hand.

8 Plotting options

By default, DEC*i*M displays two plots: a complex plane plot and a Bode plot that shows both the amplitude $|Z|(\omega)$ and the phase $\phi(\omega)$. These can be customized in various different ways.

8.1 Plotting multiple data sets

Multiple data sets can be plotted via the 'History' menu. If one data set is loaded, then saved with 'Save current dataset and sample dimensions to history' and then a new data set is loaded, the old data set can be plotted alongside the new one with 'Plot or remove non-interactive dataset (max. 3)'. This can be done with up to three additional data sets. Additionally, you can switch between different data sets for analysis with 'Select other dataset'; this is equivalent to loading a result file, but then from RAM.

8.2 Marking specific frequencies

Frequencies that are integer powers of ten (*e.g.* 0.1 Hz, 10 Hz, 100 kHz, etc.) can be highlighted in the complex plane plot with the 'Mark frequencies that are integer powers of 10' option under 'Plot'. The points belonging to the frequencies in question (or those with the best matching frequencies for frequencies that are not present exactly) will be circled and the frequencies will be indicated in text beside them. Clicking the marking option again will turn all of this off again.

8.3 Switching between different plot types

The axes of both plots can be changed in a few different ways. The options are as follows:

Left-hand side (LHS):

- Complex plane impedance: Z'' vs. Z'
- Complex plane admittance: Y'' vs. Y'

Right-hand side (RHS):

- Bode amplitude/phase: |Z| vs. f and ϕ vs. f
- Y' vs. f and Y'' vs. f
- Z' vs. f and Z'' vs. f
- Conductivity vs. frequency: σ' vs. f and σ'' vs. f
- Permittivity vs. frequency: ϵ' vs. f and ϵ'' vs. f

For the conductivity or permittivity plots, the sample dimensions should be entered into the dialog that opens with 'Plot'>'Set sample geometry'.

8.4 Other plotting options

There are several other options in the 'Plot' menu. These are:

- 'Toggle data visibility': turn measured data on/off in both plots.
- 'Toggle model visibility': turn model curve(s) on/off in both plots.
- 'Reset view': reset the view limits of both plots.
- 'Toggle RHS primary log scale': change the scaling of the left axis in the RHS plot from linear to logarithmic or vice versa.
- 'Toggle RHS secondary log scale': change the scaling of the right axis in the RHS plot from linear to logarithmic or vice versa.
- 'Toggle RHS primary visibility': turn the left axis in the RHS plot on or off.
- 'Toggle RHS secondary visibility': turn the right axis in the RHS plot on or off.

There is also a toolbar below the plots (which also features in the Z-HIT and refinement windows). This allows zooming, panning, adjusting the shape of the plots and the saving of image files.

9 Other functions

9.1 Apex frequencies

The 'Get apex frequencies' option in the 'Calculate' menu will open a new window in which the frequencies at which maxima in the complex plane plot are found. The frequencies given here are angular frequencies $\omega = 2\pi f$, not linear frequencies f (which are displayed everywhere else in DEC*i*M).

9.2 Help menu

The 'Help' menu has only two options:

- 'Show instructions', which will display brief instructions for how to use the program.
- 'Open manual', which will open the PDF manual in a web browser.

10 Examples

10.1 Example 1: RbOH·xH$_2$O

In this example, an impedance spectrum of RbOH·xH$_2$O is analyzed with Z-HIT and an equivalent circuit model is fitted to the data.

1. Start DEC*i*M with `python DECiM.py` (or the equivalent on your system for starting a Python 3 script) in the `src` folder.
2. Go to `File>Load data...` and navigate to the `examples` folder (`path_to_decim/DECiM/examples`). Go to `RbOH_xH2O`. Change the `Text files` option to `All files` and open the `CH3_cool_40C_2.csv` file. The data should now appear in the main DEC*i*M window, looking like Figure 15.

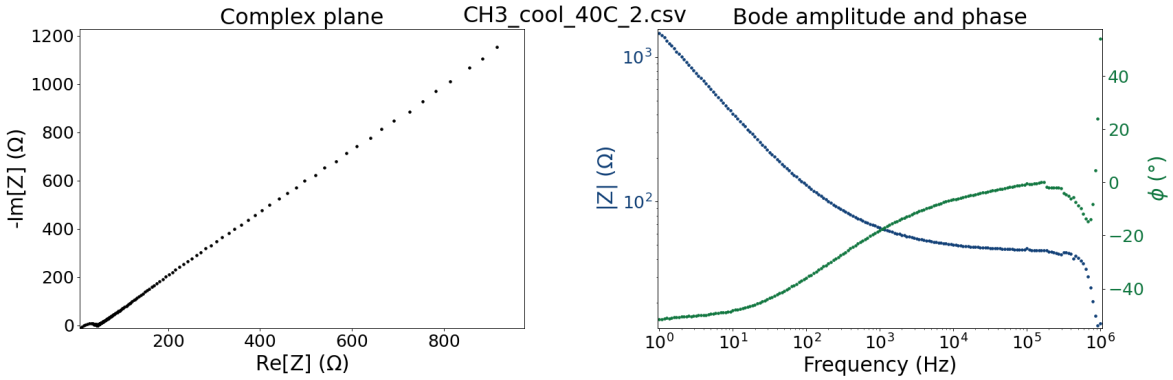

Figure 15: Impedance spectrum of RbOH·xH$_2$O.

3. Go to `Calculate>Perform Z-HIT tranform`. A new window will open.
4. Click the `Calculate Z-HIT impedance` button. The phase will now be smoothed and the amplitude calculated. You may notice that the high-frequency phase data are not well-estimated. Adjust the window the length to 10 and try again. The result should look like Figure 16.

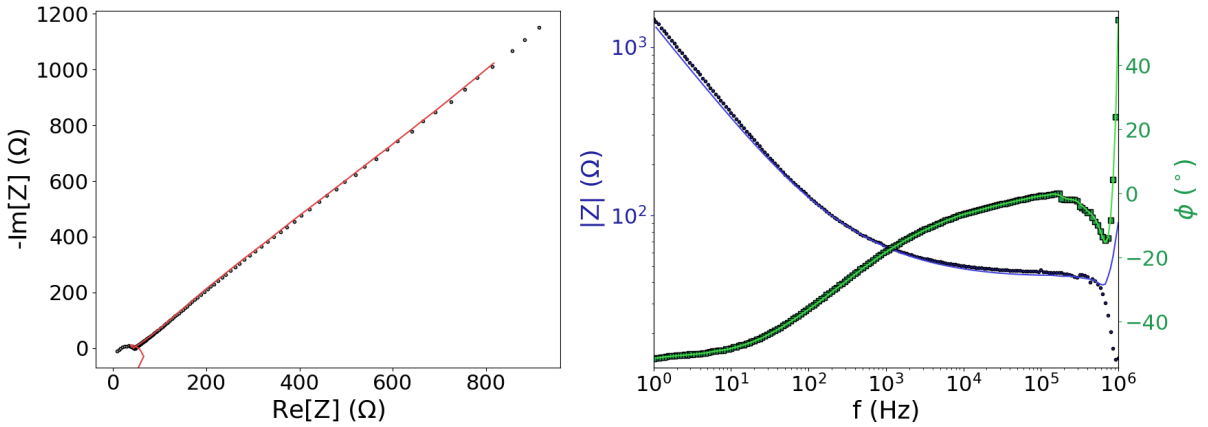

Figure 16: Z-HIT transform of the spectrum in Figure 15.

5. The phase data are well-smoothed over the entire frequency range, but the calculated amplitude increases instead of decreasing, as the measured data does, above ~300 kHz. Therefore, the data at the highest frequencies cannot be trusted. Knowing this, the Z-HIT window can be closed with `Reject and close`.

6. Now it is time to define an equivalent circuit model. Because the phase angle goes up at high frequencies, an inductor should be included. Other than that, the Randles circuit (with a CPE instead of a Warburg element for the low-frequency region) will suffice. This circuit can be drawn via `Circuit>Draw circuit`; see Figure 17. With `Use circuit and close`, the circuit is set.

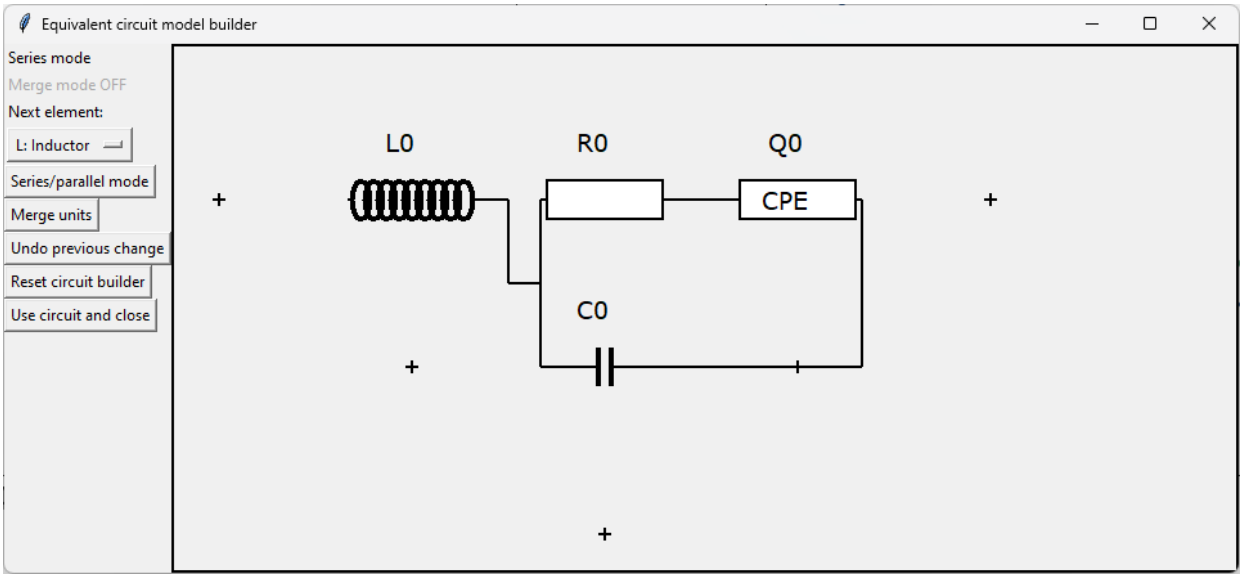

Figure 17: Chosen circuit for $RbOH{\cdot}xH_2O$.

7. Since this is quite a simple circuit, using `Calculate>Automatic initial guess`, a reason-

able initial guess can be obtained. Note that this method is subject to some randomness; the result may change (for better or worse) by pressing `Automatic initial guess` multiple times. After doing this twice, the result could look like the one in Figure 18.

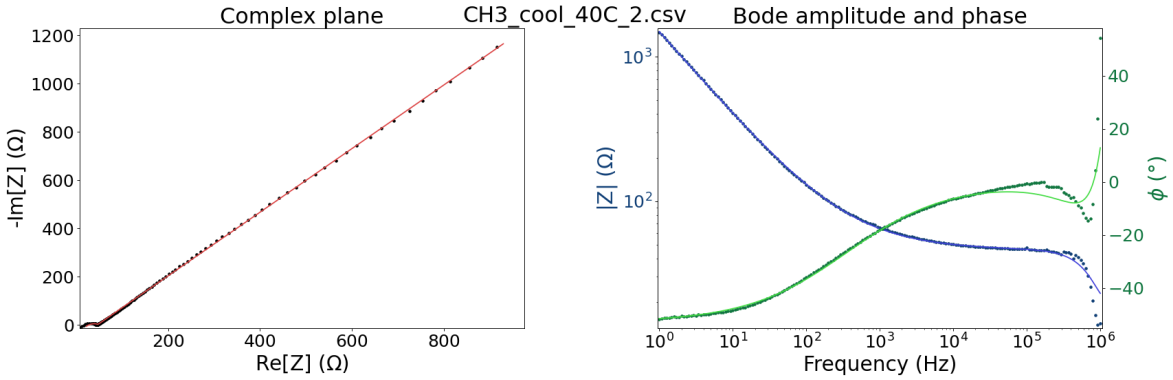

Figure 18: Initial guess for $RbOH{\cdot}xH_2O$.

8. The amplitude shoulder on the right-hand side looks as if it could be better captured by the model, although we know to doubt this feature due to the Z-HIT analysis. By selecting `C0` in the parameter list box and pressing `Adjust controls`, the slider can be moved to the right to slightly improve the fit. Figure 19 results.

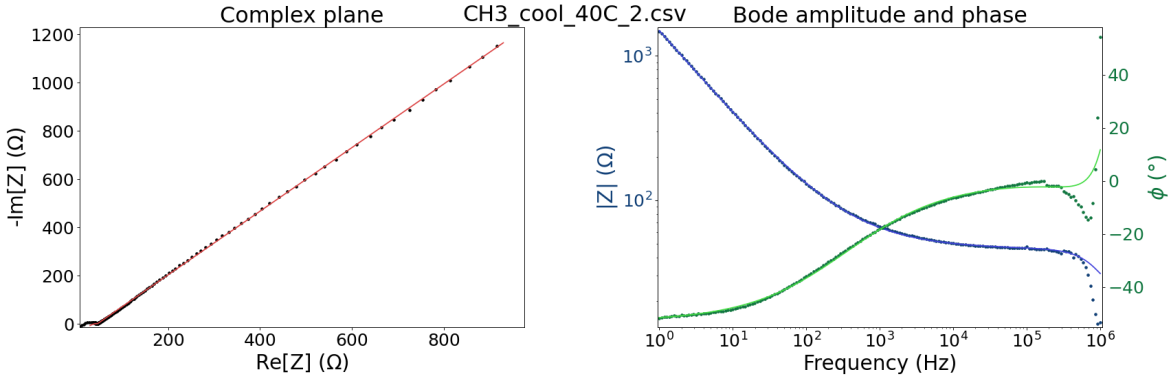

Figure 19: Manual adjustment to the $RbOH{\cdot}xH_2O$ spectrum.

9. To finalize the fit, a refinement is in order. Using `Calculate>Simple refinement`, Figure 20 is obtained. Apparently, the change in step (8) was not an improvement to the overall fit.

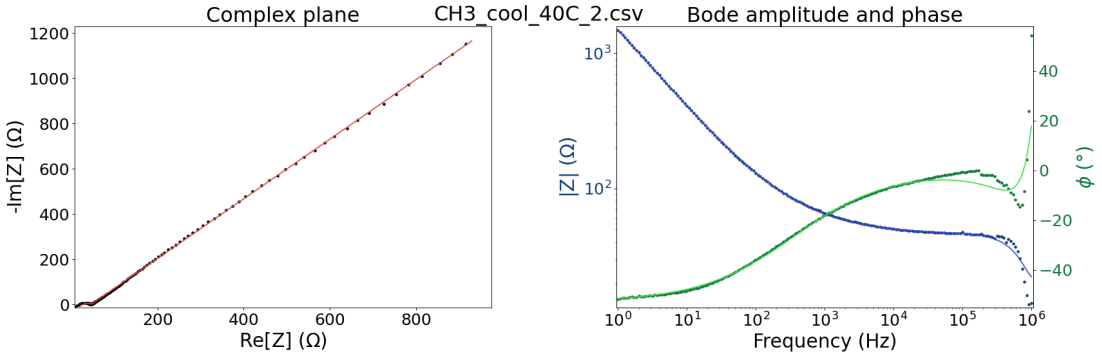

Figure 20: Refinement of the fit of the $RbOH \cdot xH_2O$ spectrum.

10. To verify that the fit is of good quality, it is worth looking at the result in a few different plot types. For example, changing the right-hand side plot to `Y' and Y'' vs. frequency` (Figure 21) reveals a few measurement artifacts around 100 kHz and the apparent breakdown of the model above 600 kHz.

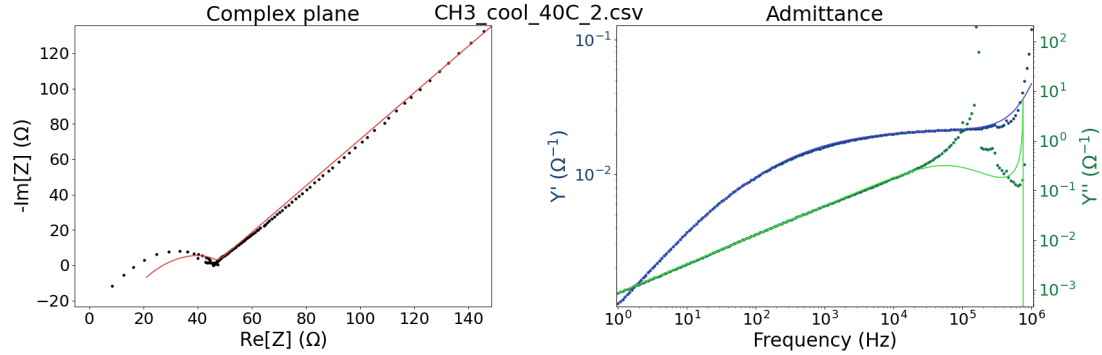

Figure 21: Alternative view of the $RbOH \cdot xH_2O$ spectrum.

11. Now that the spectrum has been validated and modeled, and we understand the limitations of the result (breakdown at high frequencies), the result can be saved. Go to `File>Save result...` and save the result.

12. DEC*i*M can now be closed; the result can be reloaded with `File>Load result...`.

10.2 Example 2: graphite||lithium half-cell

In this example, a graphite||lithium (C||Li) half-cell's impedance spectrum, recorded after a series of charge-discharge cycles, is analyzed by manually fitting and then automatically refining an equivalent circuit model.

1. Start DEC*i*M with `python DECiM.py` (or the equivalent on your system for starting a Python 3 script) in the `src` folder.

2. Go to `File>Load data...` and navigate to the `examples` folder (`path_to_decim/DECiM/examples`). Go to `Li_Graphite_Half_Cell` and open the text file (.txt, not .recm2) that's inside. The data should now appear in the main DEC*i*M window.

3. Because the spectrum is quite complex, it is useful to get an impression of the characteristic frequencies of the features. Selecting `Plot>Mark frequencies that are integer powers of 10` results in Figure 22.

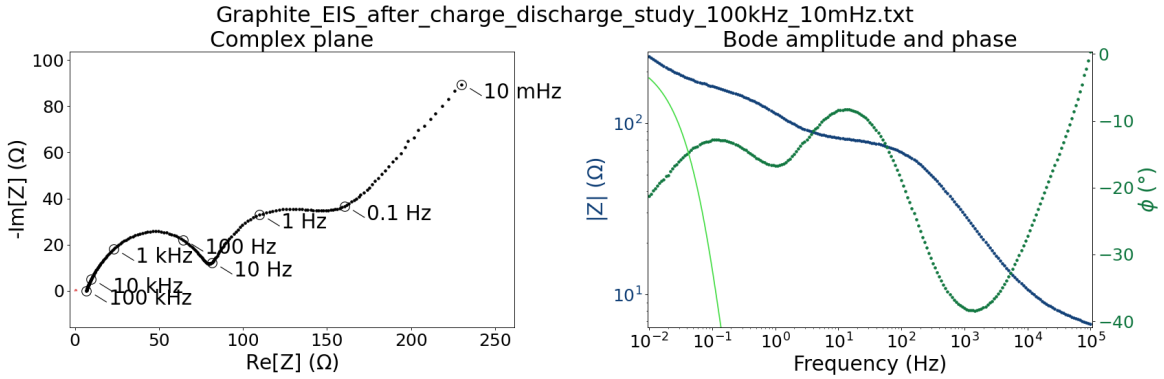

Figure 22: Impedance spectrum of a graphite||lithium half-cell.

4. The impedance spectrum has a high-frequency semicircle (100 kHz–10 Hz), a medium-frequency semicircle (10 Hz–100 mHz), and a straight line (<100 mHz). There is also a gap between the origin and the start of the high-frequency semicircle due to the external circuit. An appropriate choice of equivalent circuit for this spectrum is `{L0R0(R1Q1)({R2Q0}Q2)}`. L_0 and R_0 represent the external circuit and possibly electrolyte resistance, (R_1Q_1) and (R_2Q_2) represent two independent processes (likely charge transfers at the two electrodes, given the low characteristic frequencies), and Q_0 represents polarisation (straight line). Typing the circuit string `{L0R0(R1Q1)({R2Q0}Q2)}` into `Circuit>Type circuit` will populate the `Parameter` list box on the left, below the plots, as in Figure 23.

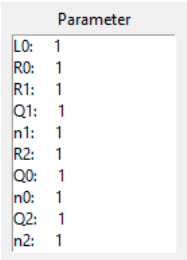

Figure 23: List of parameters for `{L0R0(R1Q1)({R2Q0}Q2)}`.

5. The `Plot>Mark frequencies that are integer powers of 10` option should now be clicked again to toggle it off. Also `Plot>Reset View` to zoom back in on the whole spectrum.

6. The parameters' values are now all 1; they should be set to reasonable starting values. For L_0, this is 10^{-12} H. For R_0, this is the distance on the real axis between the origin and the

start of the high-frequency semicircle; this is ca. 7 Ω. R_1 should be roughly the diameter of the high-frequency semircircle (ca. 70 Ω) and R_2 should be roughly the diameter of the intermediate-frequency semicircle (ca. 80 Ω). All Q and n can remain equal to 1. The desired starting values can be quickly set by first selecting the right parameter and then typing the value into the `Direct input` field and pressing the `Set` button. Alternatively, `Adjust controls` may be pressed, and the slider may then be moved.

7. Now, the constant phase elements' parameters can be adjusted. First, adjust Q_1 by selecting it, pressing `Adjust controls` and moving the slider to the right. The high-frequency semicircle should now be fitted to some extent by the red line and the first shoulder of |Z| should be touched by the model curve. The result should look like Figure 24, with $Q_1 = 1.1 \times 10^{-5}$ F.

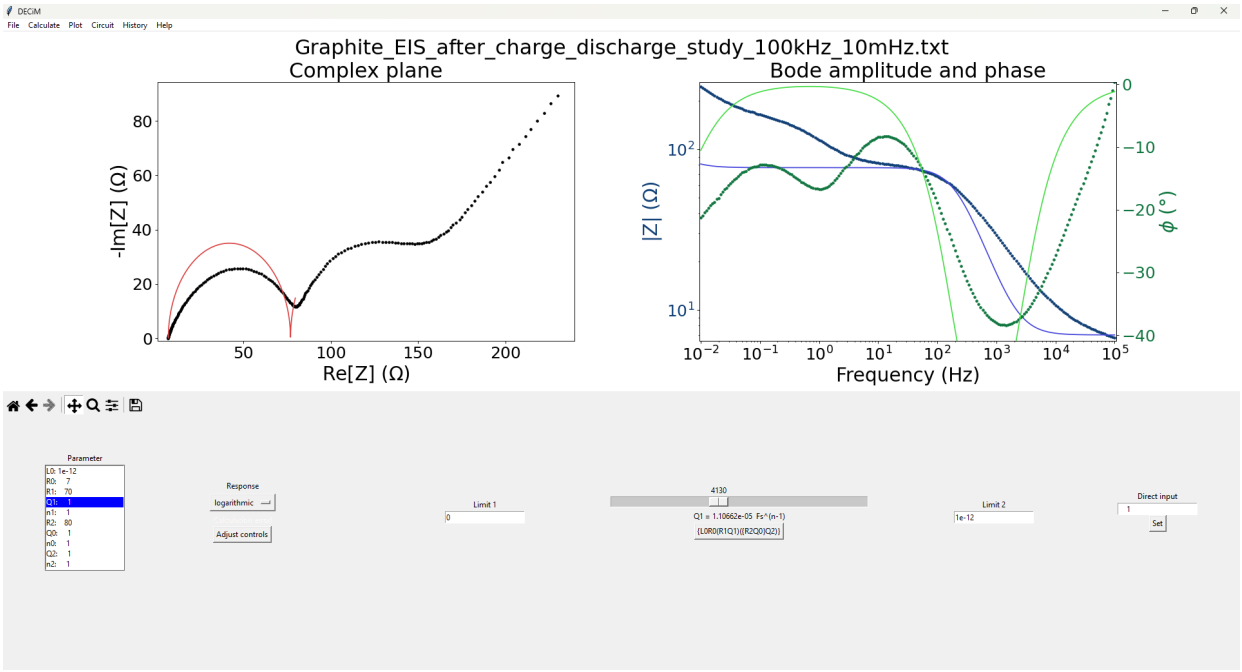

Figure 24: The result of the first few manual fitting steps for the C||Li half-cell.

8. Q_2 can now also be adjusted with the slider for a rough fit of the intermediate-frequency semicircle. The result should look like Figure 25.

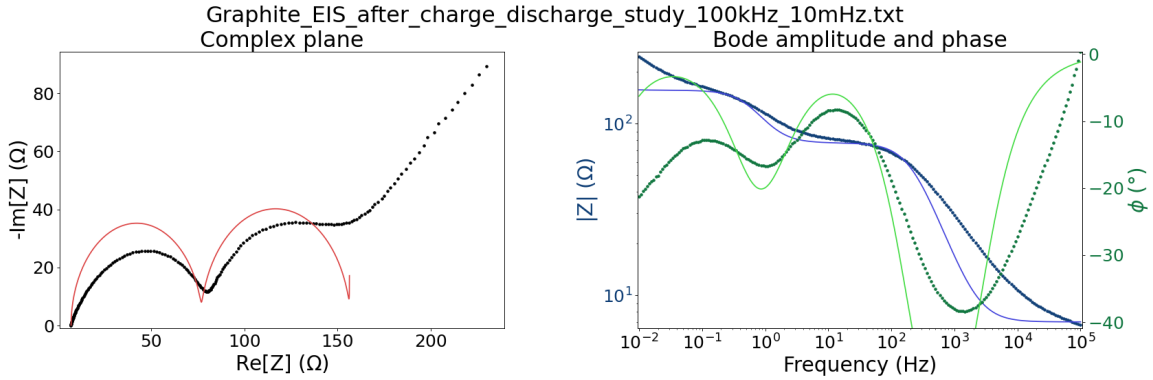

Figure 25: C||Li half-cell spectrum after tuning Q_2.

9. Now, the CPE exponent n_0 should be adjusted to match the angle of the low-frequency line and Q_0 should then be adjusted to roughly match its length, resulting in Figure 26. The parameters are those in Figure 27.

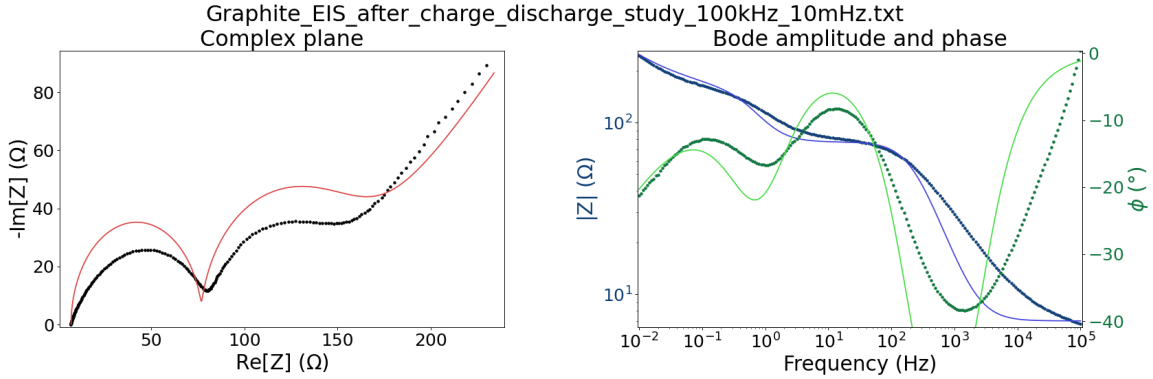

Figure 26: C||Li half-cell spectrum after tuning n_0 and Q_0.

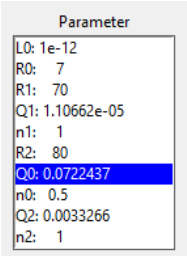

Figure 27: Parameters after tuning n_0 and Q_0.

10. To better match the initial guess to the data in the complex plane, the CPE exponents n_1 and n_2 should be decreased. However, decreasing n_1 will cause the model's |Z| to differ greatly from the measured data. This can be remedied by adjusting Q_1. A final, small adjustment of R_1 gives rise to Figure 28.

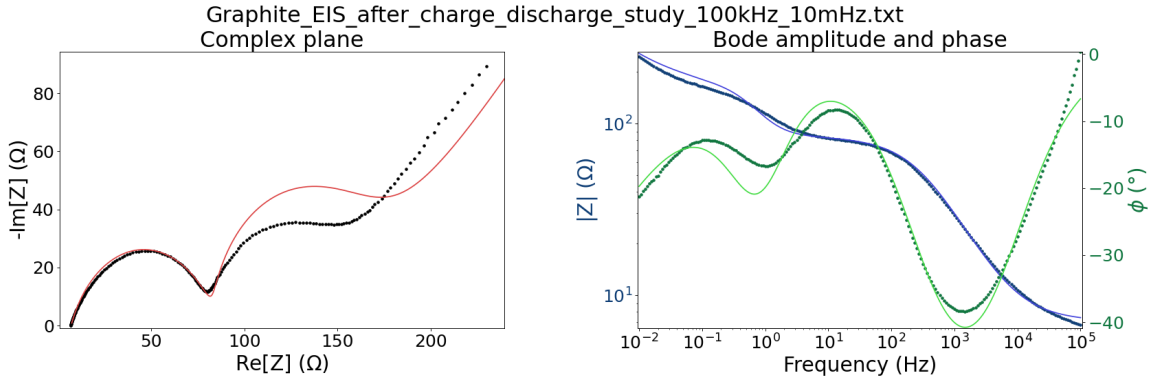

Figure 28: C||Li impedance spectrum after tuning n_1, Q_1, and R_1.

11. Adjusting n_2, Q_2 and R_2 in the same order as n_1, Q_1 and R_1 leads to Figure 29. The parameters are in Figure 30.

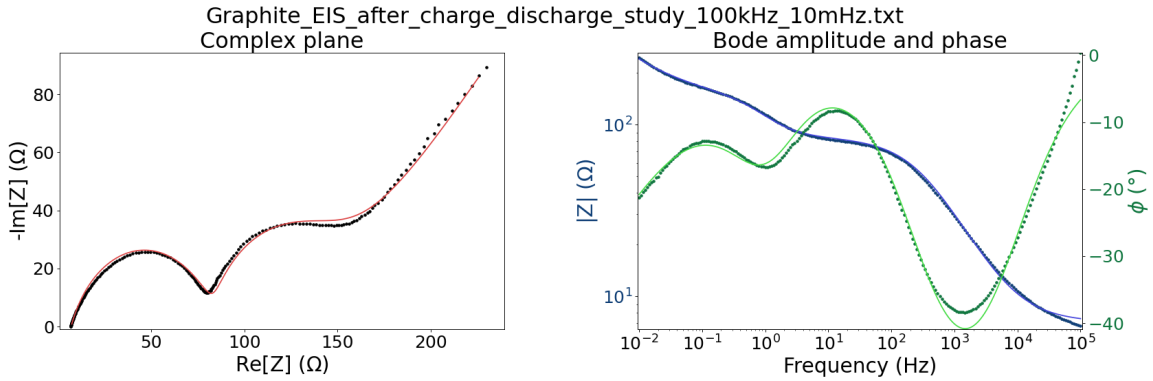

Figure 29: C||Li impedance spectrum after tuning n_2, Q_2, and R_2.

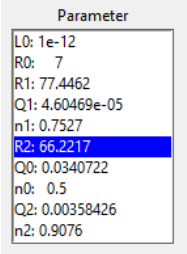

Figure 30: C||Li impedance spectrum after tuning n_2, Q_2, and R_2.

12. Now, the final corrections before automatic refinement should be made. First, L_0 should be increased to around 1.7×10^{-6} H to better describe the phase data at the highest frequencies. On the other end of the spectrum, Q_0 should be decreased to 3.2×10^{-2} Fs^{n-1}. Then, Q_2 can be decreased as well, to 2.4×10^{-3} Fs^{n-1}. The result is Figure 31.

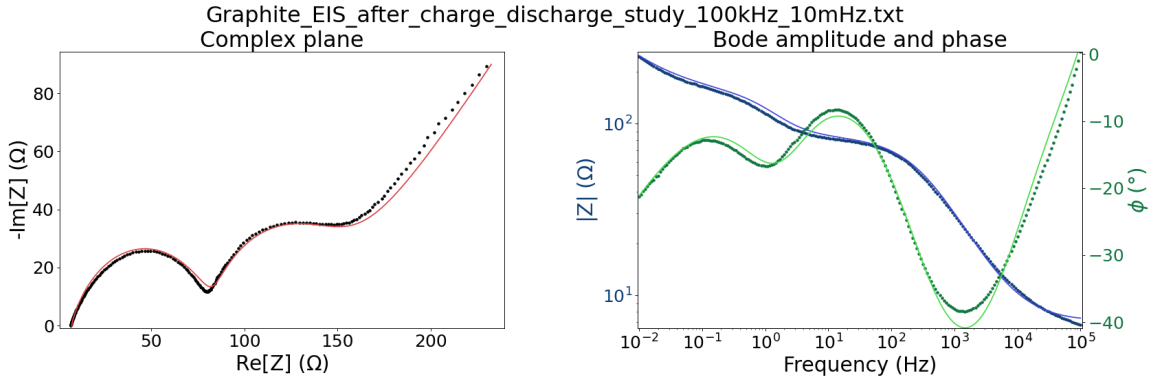

Figure 31: C||Li impedance spectrum after tuning n_2, Q_2, and R_2.

13. It now appears that R_2 is too large, so it should be decreased. Then, n_2 should be increased to raise the the semicircle without widening it. Finally, n_0 and Q_0 should be tweaked to correct the low-frequency part of the spectrum. This leads to Figure 32. The parameters are in Figure 33.

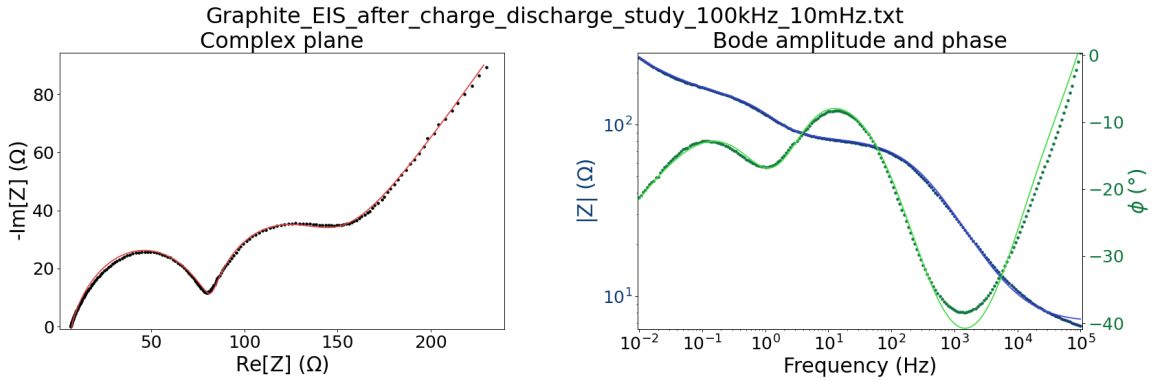

Figure 32: C||Li impedance spectrum before moving on to advanced refinement.

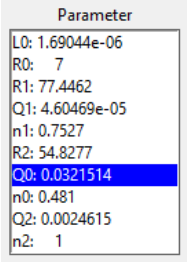

Figure 33: Parameters before moving on to advanced refinement.

14. Now, it is time to move on to automatic refinement via `Calculate>Advanced refinement...`. This will open a window with plots as in Figure 34. In the win-

dow, the `SciPy: Nelder-Mead` optimizer and `Calculated modulus` weighting scheme (modulus weighting based on the model curve) should be selected.

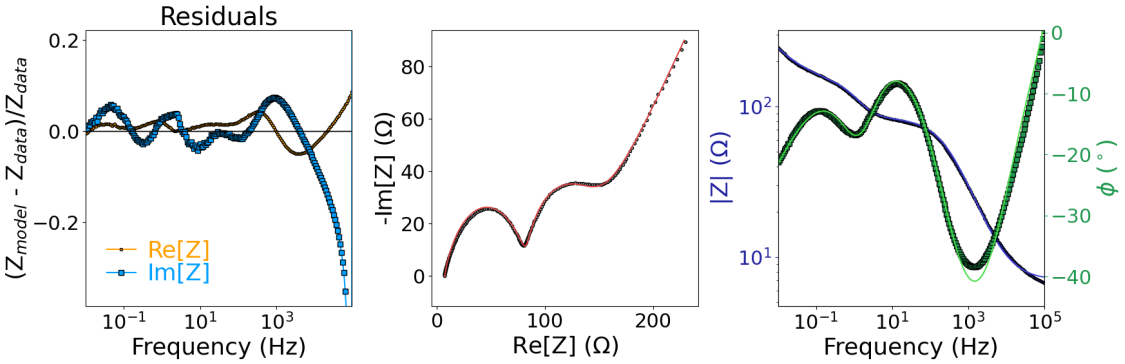

Figure 34: Plots in the refinement window before any refinement has been done.

15. The goal of the refinement is to improve the residuals. This will be done in several steps. First, the high-frequency part of the model should be optimized by selecting L_0, R_0, R_1, Q_1, and n_1 on the left and setting the lower frequency limit to 10 Hz in the centre. The settings should look as in Figure 35.

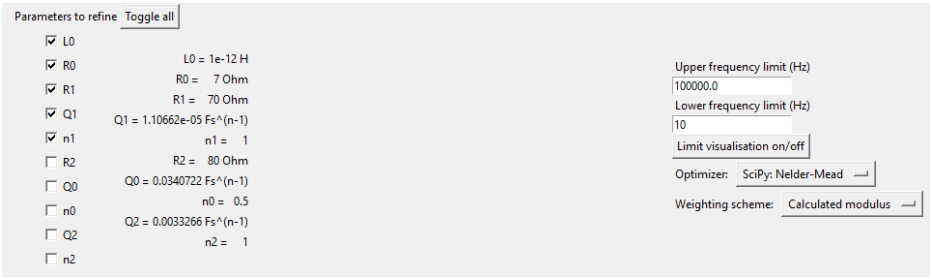

Figure 35: Refinement settings.

16. Pressing `Refine solution` on the right results in Figure 36.

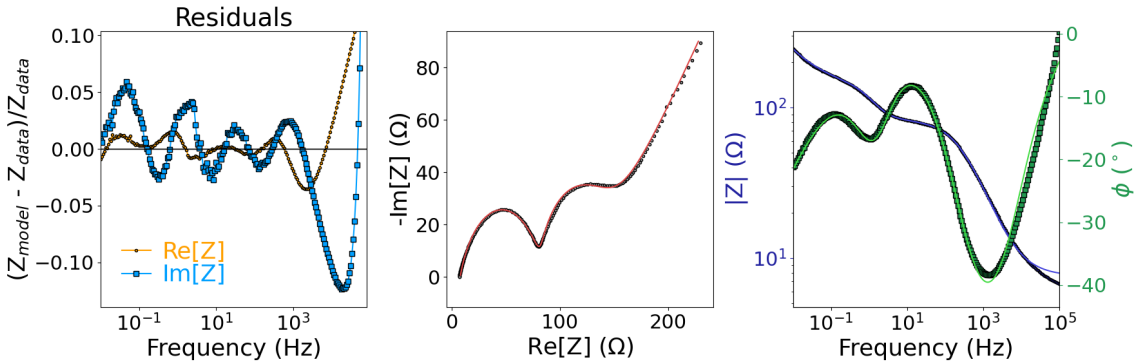

Figure 36: High-frequency refinement result.

17. Now, the lower frequencies should be treated. Because there are no points where Re[Z] = 0 or Im[Z] = 0 in this region, the weighting scheme can be set to `Observed proportional` (proportional weighting based on measured data). Refining with the upper frequency limit set to 10 Hz, the lower frequency limit set to 0.009 Hz, and R_2, Q_2, n_2, Q_0, and n_0 selected, results in Figure 37.

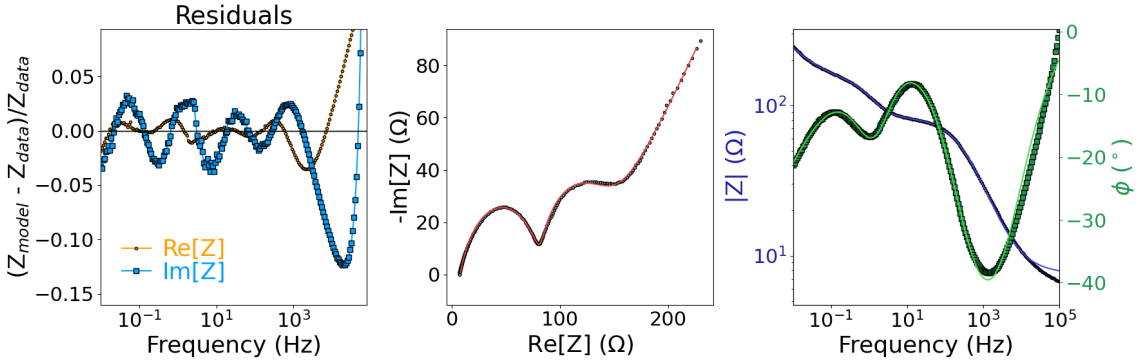

Figure 37: Low-frequency refinement result.

18. The final refinement step covers the entire frequency range and all parameters. With the `Calculated modulus` weighting scheme, the result is Figure 38.

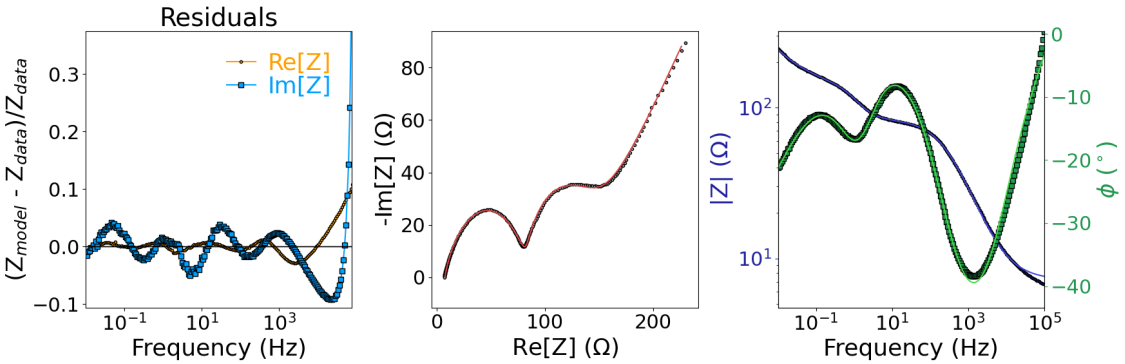

Figure 38: Final refinement result.

19. The result can be saved to a .recm2 file with `File>Save result...` This file can be loaded again via `File>Load result...`, which will load both the data and model.

10.3 Short examples

10.3.1 Changing the data file format for loading

In this example, the loading of a data file with (f, $|Z|$, ϕ) data instead of the standard (f, Re[Z], Im[Z]) is demonstrated.

1. Go to `Bode_format_data` in the `examples` folder (`path_to_decim/DECiM/examples`). Open the `CH4_H2.txt` file in a text editor to check the format. You will see that it is formatted in three columns: each row has (f, $|Z|$, ϕ) data instead of the standard (f, Re[Z], Im[Z]).
2. Start DEC*i*M with `python DECiM.py` (or the equivalent on your system for starting a Python 3 script) in the `src` folder.
3. Click `File>Specify data file layout...` and select the settings indicated in Figure 39. Then, click `Save and exit`.

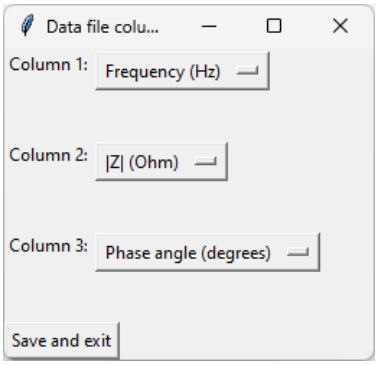

Figure 39: Modifying the data file layout specification.

4. Go to `File>Load data...` to open the file from step 1 in DEC*i*M. The data should now appear in the main DEC*i*M window (Figure 40).

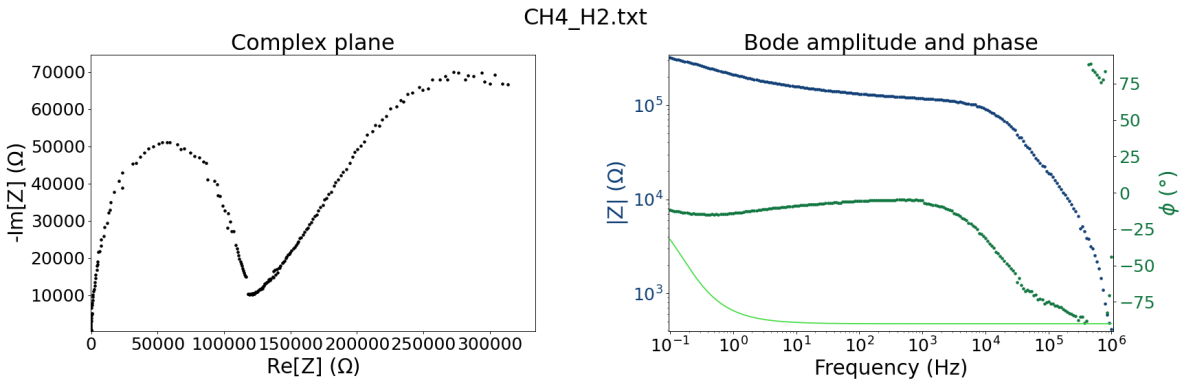

Figure 40: Spectrum loaded from (f, $|Z|$, ϕ) data.

5. To go back to the standard loading settings, go to `File>Specify data file layout...` and select the settings indicated in Figure 41. If you do not do this, DEC*i*M will continue to expect (f, $|Z|$, ϕ) data even after closing the program.

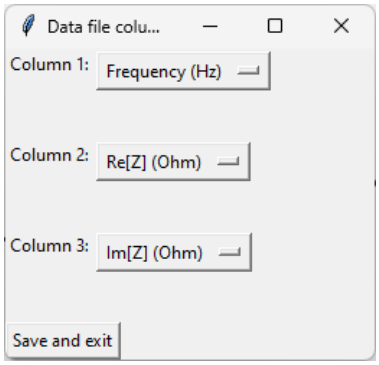

Figure 41: Restoring the default data file layout specification.

11 Copyright information

DEC*i*M is distributed under the MIT license.

11.1 DEC*i*M license

The MIT License (MIT)

Copyright 2024 Utrecht University

12 Citing

12.1 Scientific publication

Please cite the publication referenced on the DEC*i*M download page if you publish work in which you made use of DEC*i*M for data analysis.

References

1. Python Software Foundation. Python 3, 2023.
2. Harris, C. R.; Millman, K. J.; Walt, S. J. van der; Gommers, R.; Virtanen, P.; Cournapeau, D.; et al. Array programming with NumPy. *Nature* **2020**, *585*(7825), 357–362.
3. Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **2020**, *17*, 261–272.
4. Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* **2007**, *9*(3), 90–95.
5. DeepMind; Babuschkin, I.; Baumli, K.; Bell, A.; Bhupatiraju, S.; Bruce, J.; et al. The DeepMind JAX Ecosystem, 2020.
6. Rodenburg, H. P.; Ngene, P. DECiM, 2024.
7. Boukamp, B. A. *Equivalent Circuit Users Manual*; 1989.
8. Schiller, C. A.; Richter, F.; Gülzow, E.; Wagner, N. Validation and evaluation of electrochemical impedance spectra of systems with states that change with time. *Phys. Chem. Chem. Phys.* **2001**, *3*, 374–378.
9. Savitzky, Abraham.; Golay, M. J. E. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry* **1964**, *36*(8), 1627–1639.