Zfit Impedance Equivalent Circuit Fitter

v2.0 – 2 Sep 2019

1 Introduction and Installation

Zfit is a Python program with a PyQt graphical user interface. It is used to find component values for the best fit of an equivalent circuit to impedance data over a range of frequencies. You can find equivalent circuit models for transducers, electronic component networks, cables, transformers, etc. very accurately this way. The resulting equivalent circuit may be used in a circuit simulator or for further analysis.

The impedance data may come from an impedance analyzer or may be drawn as curves in Zfit. Measured impedance input data is in the form of a `.csv` text file with added header and data segmentation lines.

The equivalent circuit is modeled using equations in Python with convenient reference designators (like Z = Rs + j*w*Ls) and Zfit reports the best fit using your designators and engineering units rounded to N significant digits (like Rs = 104m, Lp = 13.2u).

There are many fitting options to help you model your data. Zfit uses the Python `lmfit` package, which provides nine different fitting methods (algorithms). You can choose to automatically normalize/denormalize your problem so the numerical range is reduced for easier fitting. Zfit will also fit to the log of your impedance magnitude data, which can be helpful if it ranges over several orders of magnitude.

Zfit is a Windows application, developed under Windows 10. It uses PyQt for its GUI, so it may be portable to other environments also. I have tried unsuccessfully to make a complete portable operating environment for it, so you need to install Python plus a few packages. The steps below describe the development environment I used, but Zfit should work with other versions too.

Install 32-bit Python version 3.7.3 from https://www.python.org/downloads/release/python-373/.

Add Python to your PATH if you are prompted, and disable the PATH length limit.

Packages may be installed using `pip install -r requirements.txt` with the `requirements.txt` file included in the Zfit distribution.

Open `zfit_constants.py` with a text editor and change the `EDITOR` and `EDIT_PATH` strings to point to your favorite text editor. Note the double backslashes in the

EDIT_PATH string. You can also customize other things like color and parameter numerical precision in this file.

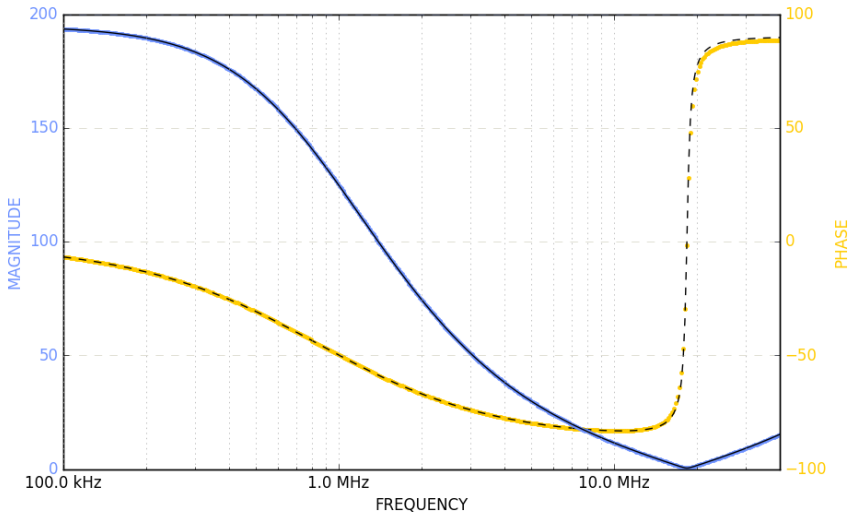
To start Zfit, double-click Zfit.pyw. If you don't have some of the required Python packages installed it will let you know, and you can install them using pip install xxx.

Zfit keyboard shortcuts are shown underlined in the app. The shortcut keys are simply the letters (without Alt, Ctrl, or Shift).

Please note that the model file structure has changed for Zfit 2.0. If you are an earlier Zfit user and have created some custom models you will need to convert them for use with 2.0. I have included some scripts to help with this, named #ModelConvert*.py in the Models folder. These scripts are to be run from Python, not from within Zfit. Modify the scripts to your needs, starting with #ModelConvert.py. This script should take care of most of your models, and you can use variations of the other scripts as required. You can simply examine 1.1 and 2.0 models and make changes by hand, too.

2 Quick Demos

For a quick demonstration, start Zfit, click Data File, and select Sample ls(cpr).csv (in the Test Data Files folder) as your data file. You'll see an impedance scan from an HP 4194 impedance analyzer of a resistor in parallel with a capacitor on the end of a few inches of wire. You can click Log on the frequency axis to match the analyzer's log sweep. Click Model Script and select ls(cpr) (in the Models folder) for your model script (L in series with (C in parallel with R)). Turn Norm/Denorm ON, and click Model (or press M). After a moment you'll see black lines overlaid on the data and Ls, Cp, and Rp component values in the Parameters sidebar, as shown at right. The black line plots represent the impedance of the Ls + (Cp || Rp) network using the parameter values shown in the Parameters box, and it lines up almost exactly with the measured data.



Experiment with different fitting methods, Norm/Denorm on/off, and log or linear magnitude. Fitting results will vary; sometimes you won't get a good fit and the resulting black lines will not match the data. The selected model script using the parameters in the right-hand panel is always what creates the black modeled lines.

2.1 Editing Model Scripts

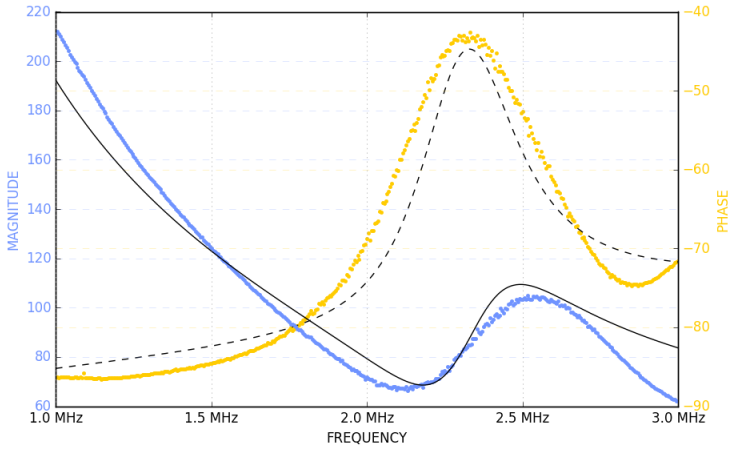
Click the Edit button next to the name of the model script and the script itself will open in the text editor you specified when you installed Zfit. You can make changes to the model script, save, and re-run Zfit Model to see the effect, leaving the editor open. Iterating like this makes the modeling process quick and efficient. In section 3 below we'll cover this in more detail.

2.2 Locking Parameters

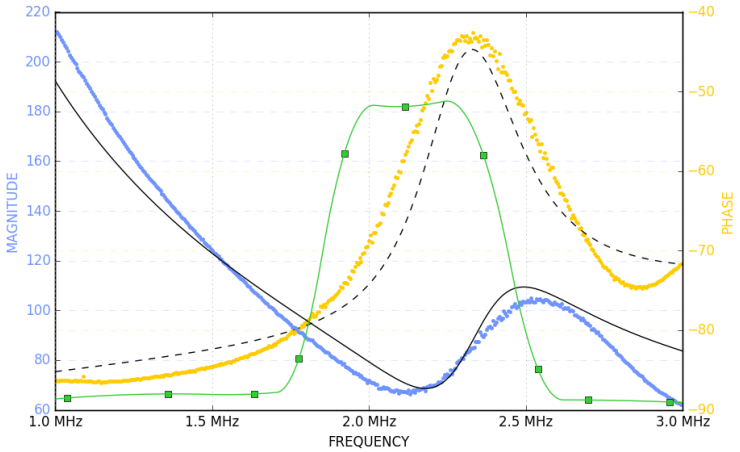
The parameters which result from a good fit to the example above will be something like Ls = 76.77n, Cp = 974.5p, and Rp = 194.8. Check the Locked box, and the parameters turn red. Now open the file `params.csv` in the Zfit folder with a text editor and change the values to Ls = 4e-08, Cp = 9e-10, and Rp = 200. Save, return to Zfit, and click Model again. Zfit now reads the parameters you've specified, displays them in the Parameters box, and uses them in your model script. You can do this to see how your model behaves with different component values, or to compare the model to a different impedance data file.

2.3 Weighting

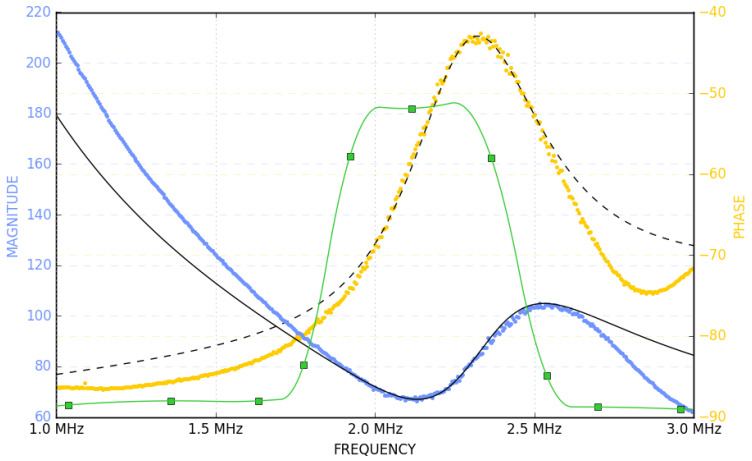
Select data file `Transducer.csv`. This is an impedance scan of an ultrasound transducer. Select `xdcr1` as your model script and click Model. You will see a fair fit to the data, as close as Zfit can get with the `xdcr1` circuit model.



Ensure that “Weight” is selected in the Drawing box at the top. Click a few times on the plot as shown and then Draw Weight to get a plot similar to the green line here. You can right-click to delete points, click to add others, and Draw Weight again.



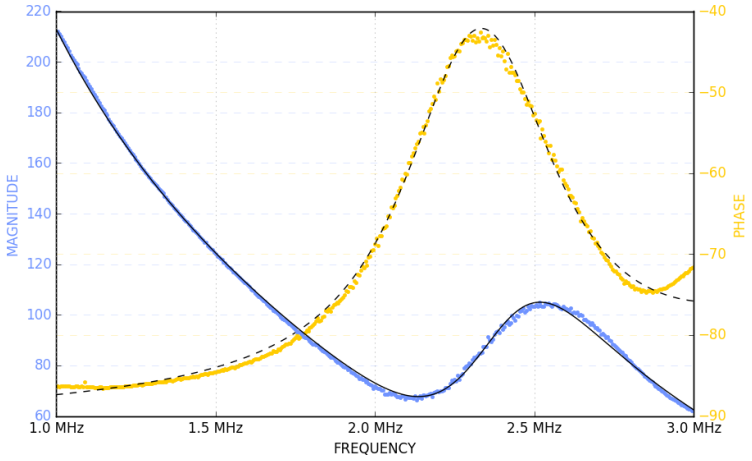
Click Model again and observe how the fit is better now in the region where the weight is higher.



The Weight Y axis is not shown, but it runs from 1 at the bottom to 100 at the top. It allows you to emphasize certain regions as more important for Zfit to match.

Experiment with right-clicking to delete points, clicking to add new ones, and repeating Draw Weight to create a new weighting function. You can Clear Weight before saving plots so the weighting function doesn’t appear in your plot, or you can leave it in the plot.

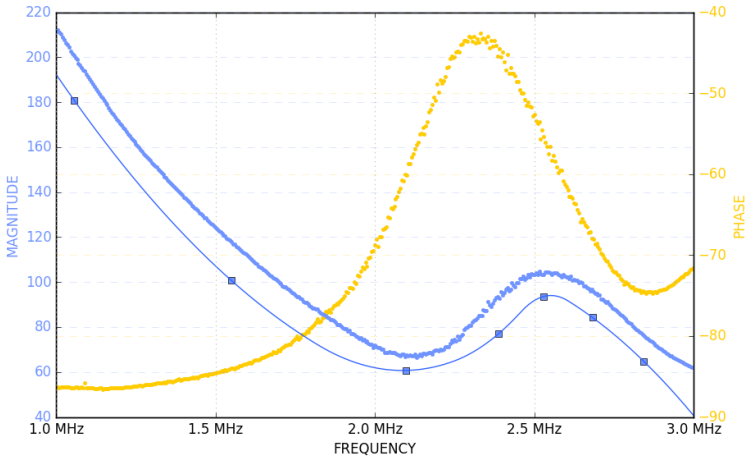
Incidentally, if you now choose model script `xdcr2` with this same data file and click Model with no weighting you will see an example of the benefit of fitting to a better model.



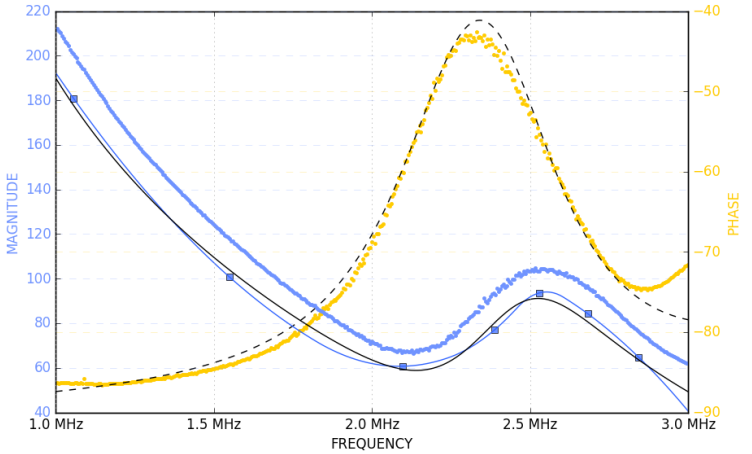
2.4 Model Your Own Impedance Curves

Continuing from the previous section, in the Drawing box select Magnitude, Clear, Phase, Clear. This clears any modeling already performed, or drawn curves.

Now select Magnitude, click a few points which are lower than the data, and click Draw Magnitude. Your display should look something like this.



Now click Model and Zfit will fit to the new magnitude you've drawn. See how it now follows your drawn line rather than the magnitude data.



Phase curves may also be drawn in a similar fashion. Whenever a drawn curve exists Zfit uses it, otherwise it uses the data.

The curve drawing process can be made efficient by using click and right-click to draw and erase points, D to draw, C to clear, and W, G, and P to select Weight, Magnitude, and Phase drawing plots.

To plot `.svg` and `.png` versions of the plot, including modeled and drawn curves, click Plots / Save to select the folder in which to save to the fixed file names `Zfit.png` and `Zfit.svg`. These files are overwritten each time you save, so be sure to copy them first if you want to keep them.

2.5 Multiple Segments in a Data File

Zfit allows you to concatenate several measurements of the same network under different conditions into the same data file, and it will fit to all measurements at once.

For example, you can make transformer primary impedance measurements with the secondary opened and shorted and fit the two data sets to a single transformer model.

For an example, load the model script `TestTline1, four loads.csv`. Click Next to see the four measured data segments for a 3.251 m long twisted pair transmission line with four different loads at the end. For a model script select `trans_line1`. Check Log magnitude, Norm/Denorm On, and fitting method Levenberg-Marquart. Click Model, and Next four times after modeling is finished to see the results.

The four segments of this data file correspond to impedance measurements with four different loads at the end of the line: a short (5 mΩ), a 49.9 Ω resistor, a complex load consisting of 69.16 nH in series with a parallel combination of 92.2 ohms and 820.9 pF, and an open circuit (0.5 pF). The impedance at the measurement end of the line varies widely with the different loads, but Zfit finds a good fit to all the data by fitting to all segments simultaneously.

Note that you can't draw weights or your own magnitude and phase curves with a segmented data file.

2.6 Wide Data Range Modeling

With the same data file, model script, and settings as in the previous section, try unchecking Log magnitude and Model again. The parameter results are different and the fit is not as good. Zfit will fit to the log of your magnitude data if Log magnitude is checked.

With data whose impedance magnitude ranges over more than a couple orders of magnitude, a better fit is often obtained with Log magnitude checked.

2.7 Modeling Using Loop or Node Matrices

Now select Data File `MatrixSolnExptSwept` and Model Script `MatrixSolnExptModel`. Select Log magnitude and frequency, Norm/Denorm on, and Levenberg-Marquardt fitting method. Click Model to fit to the data.

Click Edit and examine the model script. The model is not a set of impedance equations like you would normally write for a network, but rather a matrix solution for the current entering the network which is converted to impedance by dividing into the driving voltage (1.0). A matrix model like this can be derived directly from loop current or node voltage analysis, and is particularly useful for coupled windings with a coupling factor less than one.

See MatrixSolnExpt.html in folder Models\Data for a schematic and the node analysis of this network. (The original Jupyter notebook is also located there as a .ipynb file.) Using Kirchoff's Voltage Law at each node plus the terminal voltage equations for the coupled inductors, the coefficient matrix A and the output matrix B can be written directly from the schematic as shown in the sketch in Out [2]. Arbitrary test values for components and frequency are defined in [3], and matrices A and B are copied from the sketch in [4] and [5]. The Python linear algebra solver is used in [7] to find the magnitude and phase of the voltage and current variables in the network at the given operating frequency, for checking against a SPICE simulator.

LTSpice circuit MatrixSolnExpt.asc is also found in folder Models\Data, and is used to verify the matrix solution at the test frequency. Running the simulation and measuring the node voltages and loop currents, you will see they match the calculations from the Jupyter notebook. MatrixSolnExptSwept.asc was then used to create swept impedance data in LTSpice as follows: run the simulation, select the output plot window V(n001)/-i(V1), do File / Export and select Cartesian format. The resulting text file was converted from real/imaginary to magnitude/phase format and became the MatrixSolnExptSwept data file. This data file thus contains the impedance values you would get from measuring the test circuit with an impedance analyzer in place of the 1.0 V voltage source. When Zfit fits the matrix-derived model to the data in this file, it finds the same parameter values for L1, R1, L2, C1, and M (or k) as were in the original LTSpice simulation.

This shows how you can fit to a model which is best expressed by a matrix rather than by complex equations. You can use this example as a way to get started.

3 Modeling Details

In order to successfully fit a model of an equivalent circuit to your data, you need:

- A suitable equivalent circuit (model) for the data
- Estimates for the initial circuit component values to "seed" the fitter
- Allowable ranges for the components

All these are contained in a Zfit model script. Some typical scripts are found in the Models folder. Generally you will click Edit to open the model script for your circuit in a text editor alongside Zfit so you can iteratively adjust parameters, save, and click Model again in Zfit. Zfit reloads your model script each time you click Model.

Let's take a closer look at the process. Select `Sample ls(cpr)` again and Model Script `ls(cpr)`, and click Edit to look at the model script.

There are two main parts to a model script: the `PARAMS` list and the model. The components in your model are defined and delimited in `PARAMS`. Each entry contains:

- the string name of the component as you wish to call it ("`name`")
- the initial estimate of the component value ("`init`")
- whether to allow the component to vary during fitting or to remain fixed ("`vary`")
- the minimum and maximum value range for the component ("`min`" and "`max`")

You can name the components whatever you like, but there is a restriction if you want to use Norm/Denorm: capacitors, inductors, resistors, conductances, and mutual inductances must start with `C`, `L`, `R`, `G`, and `M` respectively. This tells Zfit how to apply frequency and impedance normalization to each component.

The initial value should generally be within an order of magnitude or so of what you expect, although this varies with model and data. The min and max values may be `None` to eliminate any restriction, but it is usually helpful to at least make min equal a small positive number. With difficult fits you may need to reduce the min/max range.

The `model()` function is where you write the equations for your circuit impedance. This involves assigning individual variable names to your `PARAMS` elements (like `Ls = params['Ls']`), and then using the variable names in the equations of your model. (In Python you can use single or double quotes around strings, so `'Ls'` is the same as "`Ls`".)

The equations in `model()` can be written however you like. In this example the admittance `Ycp` of the parallel combination of `Cp` and `Rp` is calculated first, and then the impedance of that network is put in series with the impedance of `Ls`. All you need to do is return the final `Z` value from the `model` function. Use `j` for the imaginary operator and `w` for omega (radian frequency). Impedance of a capacitor, for example, is `1/(j*w*C)` and an inductor is `j*w*L`, and admittances are the reciprocals. The `numpy` module of Python will apply your equations for all values of frequency in your input array, much like Matlab does, and return an array for `Z`.

All your `model()` function needs to do is return an array `Z` which corresponds to the input variables `w` (radian frequency) and the parameters defined in `PARAMS`. (You can also pass arbitrary values or loads to the model, see below.) The fitting algorithm will

call your model repeatedly while varying PARAMS according to the boundaries you've set until the error between your model Z and the data file Z is a minimum.

Try modifying the initial and min/max values in PARAMS, saving ls(cpr).py, and clicking Model again in Zfit, just to see the effect with this example. Try setting "vary" = False for Ls, or artificial "min" and "max" values.

Model rs(c+load) is a simple example of passing a load to a model. The load is placed in parallel with C and the impedance of that network is placed in series with R. This model can be used with segmented impedance data, obtained with different loads (see section 3.3).

You may also end up writing complex models which require a more complicated model() function. Model script trans_line1.py is an example of this, where a load value at the end of a transmission line is defined by model argument load. Function segment_z() is called repeatedly in the model to calculate the total impedance of the segments comprising the cable. Constants LENGTH, SEGMENTS, and BRANCHES are also defined as part of the model script. Take a look at LTSpice schematic LineSegment.asc in Models\Data to see the schematic for one segment of the line this model represents. As long as the model() function returns the network impedance array Z Zfit will make use of it properly.

3.1 Locking or Setting Parameters

Sometimes you want to fit component values (parameters) from one set of impedance data and then see how closely those parameters match another set of data (a different frequency range for example). You'd like to lock the parameters which result from one modeling session to use with another. Or, perhaps you'd like to manually convert the exact parameters determined by Zfit to the nearest 5% component values and see how well the model still fits, or experiment with varying parameter values to see the effect.

For this you can use the "Locked" check box under the Parameters display box. Any time you click Model with Locked unchecked, the resulting parameters are written to a file named params.csv in the local Zfit folder. When you check Locked, the parameters are now *read* from params.csv instead. You can select a new data file and/or model script and Zfit will use the values in params.csv instead of fitting new ones. Of course, the component names must be the same if you switch model scripts.

One example of the utility of this is fitting to a broad frequency range and then seeing how good the fit is in a subset of that range. You can manually extract a subset of your

original .csv data file for that. Run Model on the full range, check Locked, switch to your subset data file, and press Model again.

The Locked checkbox also allows you to modify params.csv with a text editor while parameters are locked, so you can explore the effects of varying parameters manually. Check Locked, make your modifications to params.csv, save, and click Model again in Zfit to see the effect. Your manually-entered parameter values are also displayed in the Parameters box. You can do this to explore the effects of tolerance or using standard component values.

3.2 Data File Format

Zfit data files are .csv text files with a header first line and subsequent lines in the form freq, mag, phase floating point numbers. See any of the data files in the Test Data Files folder for examples. You can get a free utility for creating Zfit-compatible data files directly from an HP 4194 impedance analyzer from http://exality.com/beautiful-hp-4194a-plots

3.3 Segmented Data Files

A segmented data file is of the form:

[header line]
 [condition 1] [# comment]
data block
 [condition 2] [# comment]
data block
…

It is the same as a standard Zfit data file except for the lines. The keyword (with angle brackets) separates data segments in the file and describes the condition which applies to the following data block. Data blocks follow the standard Zfit format of multiple lines of frequency, Z magnitude, and Z phase separated by commas. You can have as many segments as you want. See Test Data Files\Samplers(c+load).csv for an example with three segments.

The condition described on each line may be a varying load, etc., whatever you varied to get the different data segments. The condition is passed to the model() function as the load argument, but it doesn't need to actually be a load. The condition is in the form of a Python expression, which may include j for the complex operator and w for omega, radian frequency. A # comment is optional, to describe the segment.

When you load a segmented data file into Zfit for modeling, the Segment and Next label and button are activated. By clicking Next you cycle through each segment of the data file.

3.4 Fitting Tips

- The most important thing in modeling is that your equivalent circuit be suitable for the impedance data you're fitting. If you just can't get a good fit, it is likely that there is a mistake in your `model()` function or that it doesn't represent the data well.
- Double check your model initial values and min/max. Try adjusting initial values by factors of ten if you don't know roughly what the final values should be.
- Try loosening the min and max values. The fitting algorithm needs to have room to maneuver while solving.
- Start off with min and max values of `1e-12` and `None`. If your model is good and your initial values are reasonable, this generally converges to a good fit.
- If two components have a similar effect Zfit can't determine the proportioning between them. For example, two series resistors may fit to 500 and 500 ohms, or 100 and 900, etc., with the same result. A resistor in series with an inductor with skin effect is another example where results are similar for different component values.
- If a parameter is stuck on an initial value or near one of the min/max limits, it is a good indication that the best fit is not being found. Try a different fitting method, different initial values, or looser bounds limits.
- Different `init` values can provide different good solutions, some slightly better.
- When a decent fit is found, try different `init` values above and below the previous fit, to approach the fit from the "other side".
- For complicated models, different methods or initial guesses can provide apparently identical fits with different resulting component values.
- Some fitting methods seem to hang sometimes at the end of fitting, but eventually finish.
- Try fitting with log magnitude checked if there is over about a factor of 20 in your range of impedance magnitudes.
- Norm/Denorm mode usually provides the best fitting, but be careful how you use it. Zfit scales frequency and component values in this mode, and the only way it knows how to scale the component values is if you name them starting with R, C, L, G, or M. Using fixed scale factors or other component types in your model, building fixed

R, L, C, G, or M values into the model, etc., means you will need to fit with Norm/Denorm off.

- If you suspect your model isn't correct, try loading values into `params.csv` for which you know what the model response should be, select Locked in Zfit, and click Model. The "modeled" lines will then show what your model does with the parameters fixed in `params.csv`. If you enter your model in a SPICE simulator with the same parameters, it should show the same magnitude and phase traces as you see in Zfit if your Zfit model is correct. To plot complex impedance in SPICE, drive the network with an AC voltage source of 1.0 and plot (the voltage source voltage node) / (the current *into* your network).
- Some impedance data is inherently difficult to fit. `Sample rs(c+load)` in the `Test Data Files` folder is an example. Try fitting this data with the `rs(c + load)` model. Most fitting methods don't work well, and varying the initial component values much more than about a factor of two won't fit at all. Fitting algorithms operate as minimizers between your data and your model, using repeated estimates of model parameters. It's like putting a marble in a concave bowl shape and letting gravity find the lowest point of the bowl. If the bowl is mostly flat with just a sharp dip, there are many places you can put the marble where it won't find the dip. Some bowls have multiple dips, too, so the minimum you find depends on where you place the marble (your component initial conditions).
- Zfit can do the best job when each component in your model affects the impedance strongly. When a component has little effect on the impedance data, Zfit can't determine an accurate value for the component.

4 Mag/Phase vs Real/Imag

Modeling in Zfit involves finding a least-squared-error fit between the real and imaginary parts of the model equations and the data (or drawn curve), not the magnitude and phase. Magnitude and phase are generally more natural and useful for engineers, but fitting data to real and imaginary components of impedance results in a more robust fitter, and a more uniform one in a sense.

Magnitude and phase are not "symmetrical" or equivalent. Small changes in real and imaginary parts can result in small magnitude changes but large phase changes. There is also the pesky discontinuity in phase at 2π radians.

Because of this it may appear that Zfit fits more tightly to magnitude than phase. It's actually more accurate to say that the phase is a more sensitive measure of the real and imaginary components of impedance, which is what Zfit uses.

5 ENDNOTES

Zfit is released to the public domain and may be used freely for whatever purpose you like. There is no support and no express or implied warranty, but suggestions and bug reports are welcome. I'm not a sophisticated Python programmer, so I'm sure there are plenty of gaffes and blunders which the Python expert will find.

Gerrit Barrere
exality.com
gerrit@exality.com