scraps: an open-source Python-based analysis package for analyzing and plotting superconducting resonator data

Faustin W. Carter, Trupti Khaire, Valentyn Novosad, and Clarence Chang

***Abstract*—We present "scraps" (SuperConducting Analysis and Plotting Software), a Python package designed to aid in the analysis and visualization of large amounts of superconducting resonator data, specifically complex transmission as a function of frequency, acquired at many different temperatures and driving powers. The package includes a least-squares fitting engine as well as a Monte-Carlo Markov Chain sampler for sampling the posterior distribution given priors, marginalizing over nuisance parameters, and estimating covariances. A set of plotting tools for generating publication-quality figures is also provided in the package. We discuss the functionality of the software and provide some examples of its utility on data collected from a niobium-nitride coplanar waveguide resonator fabricated at Argonne National Laboratory.**

***Index Terms*—Superconducting resonators, Microwave measurements, Superconducting microwave devices**

I. Introduction

WE BEGIN by assuming that the reader is sufficiently interested in superconducting micro-resonators that the introduction of a new tool to help manage, analyze, and visualize large quantities of experimental data will be viewed with enthusiasm, and we proceed accordingly by dispensing with an overarching introduction to the field. Here we present our open-source Python packaged called `scraps`, which stands for "SuperConducting Resonator Analaysis and Plotting Software". `scraps` is specifically designed to automate the loading, fitting, and plotting of large amounts of transmission data with two independent variables (temperature and input power), while leaving the details of the models used in the fitting under the control of the user. It contains both a least-squares fitting engine as well as a Monte-Carlo Markov Chain (MCMC) sampler for generating posterior distributions for the parameters through Bayesian inference. We present an overview of the package features in this paper, using as an example, data taken from a niobium-nitride (NbN) coplanar-waveguide (CPW) micro-resonator chip fabricated at Argonne National Laboratory (ANL). We also briefly discuss the experimental setup we use and provide permanent links to the raw data, the Python scripts (and an archive of the current version of `scraps`) used to create each of the figures in this paper [1], [2].

Work at ANL is supported by UChicago Argonne, LLC, Operator of ANL. ANL, a U.S. DoE Office of Science Laboratory, is operated under Contract No. DE-AC02-06CH11357. We also acknowledge support from the Argonne Center for Nanoscale Materials. Partial support is provided by the the Kavli Foundation and the NSF Physics Frontier Center grant PHY-1125897 to the KICP at U. Chicago.

F.W. Carter (email: faustin.carter@gmail.com), C. Chang, T. Khaire, and V. Novosad are with Argonne National Laboratory, Argonne, IL 60439.

Manuscript submitted 9/6/2016

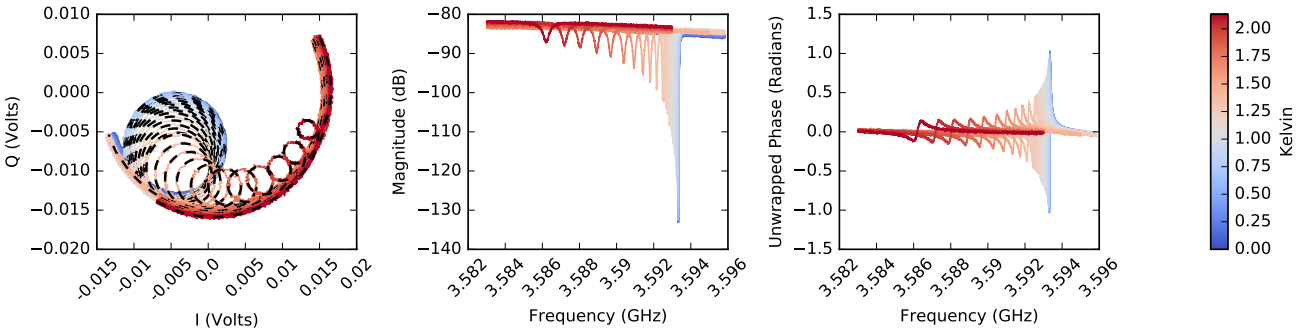
II. The target experiment

In the target experiment that `scraps` was developed to assist with, one (or more) superconducting resonator(s) are capacitively coupled to a transmission line and mounted on the cold-plate of a cryostat. The complex transmission $S_{21} = I + iQ$ (where I and Q are the in-phase and quadrature components, respectively) is measured at several different frequencies (2000 in our case), either with a pair of mixers or a vector network analyzer. The resulting data product is a set of three numbers (f, I, Q) at each of the frequency points and will be referred to as a "trace" hereafter. We use a LabVEIW™ program to sweep the temperature and power of our High Precision Devices Olympus model adiabatic demagnetization cryostat, generating transmission measurements at several hundred temperature/power combinations. (The LabVIEW™ control software for this cryostat was also developed from scratch at Argonne by one of the authors and is available in its entirety at [3], [4].)

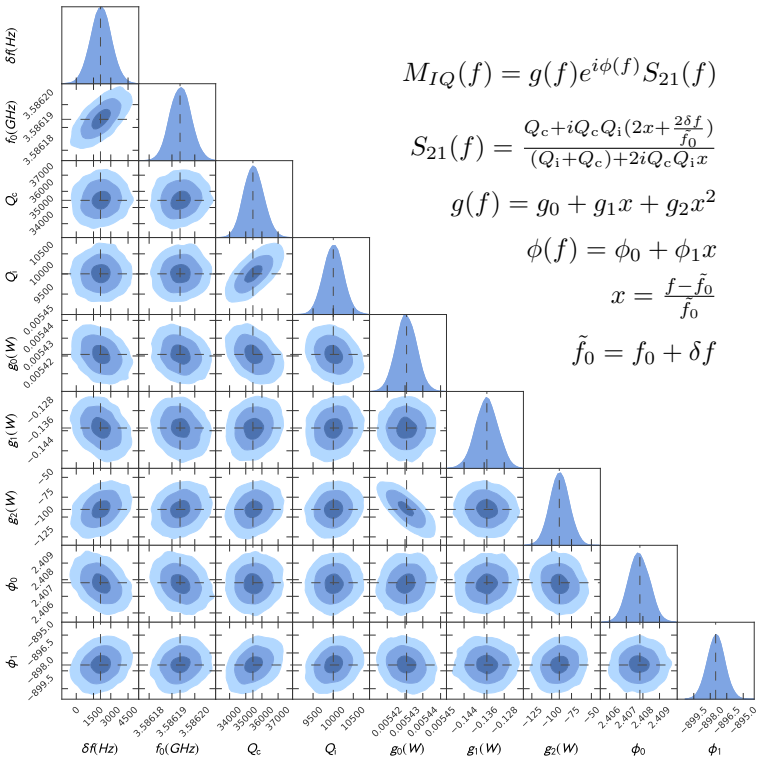
Once the experimental data has been loaded into `scraps`, high-quality figures can be generated easily. Figure 1a shows I vs. Q, magnitude, and phase of one of the resonators from our NbN chip at several different temperatures (at fixed input power). The black dashed lines superimposed on the I vs. Q curves are least-squares fits to the included resonator model (see section III-B). Figure 1c shows the resonant frequencies (f_0) and internal quality factors (Q_i) of the resonator (extracted from the fits shown in Fig. 1a) as functions of temperature at several different input powers. Finally, Fig. 1b shows the parameter covariances (via MCMC sampling with flat priors) of one of the fits shown in Fig. 1a. Figures 1a, 1c, 2a, and 2b were generated directly within `scraps` and have not been subsequently edited in any way. Figures 1b, 2c, and 2d used a package called `pygtc`, which was also developed at ANL [5].

III. Overview of scraps

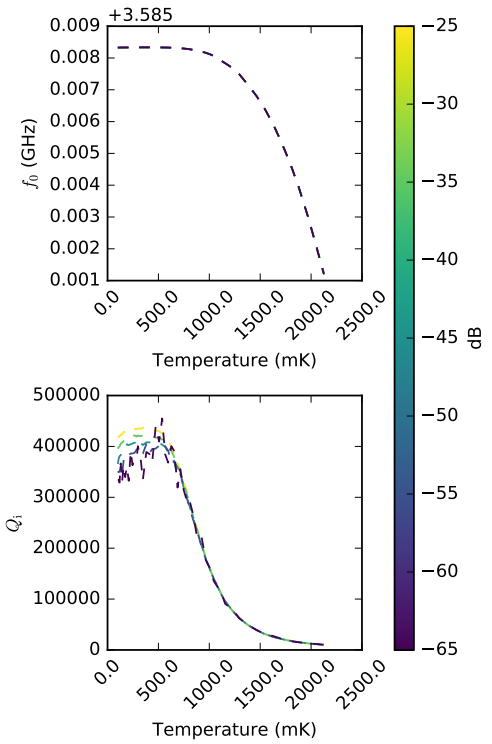
Development of `scraps` happens at GitHub in a public repository [6]. Documentation is hosted at ReadTheDocs [7], and several example tutorials have been provided to help a new user get up to speed. The latest release is hosted on PyPi, the Python Package Index [8], and installable with `pip`, the standard package manager for Python packages [9]. Anyone is welcome to download, install, reuse, and/or modify the source



(a)



(b)



(c)

Fig. 1. a) I vs Q, magnitude (uncalibrated power axis), and phase of a NbN CPW resonator at several different temperatures and fixed input power. The dashed black lines in the first panel are best fits. b) Parameter covariances generated via MCMC sampling of the posterior distribution; contours are at 1, 2, and 3 sigma. The dashed black lines indicate the best-fit values generated by the least-squares routine. Inset shows the fit model $M_{IQ}(f)$. c) f_0 and Q_i as functions of temperature at several different input powers. Power indicated is the power output from the VNA; power at the resonator was ∼60 dB lower.

code. Anyone that wishes to participate in development is encouraged to and should contact the authors either through GitHub or using the contact info on the title page of this paper.

The software handles three primary functions: 1. Loading, managing, organizing, and accessing data; 2. Fitting data to models and generating posterior distributions for the parameters; 3. Plotting and exporting the results of 1 and 2. Our design philosophy has been to provide a framework to handle the heavy lifting, but to otherwise stay out of the way of the user. In practice this means that `scraps` is *not* a standalone application for a lay-user. Rather, it is a set of tools that extends the flexibility and functionality of the Python language and is designed to be integrated into the analysis development at the code level. We have found that using `scraps` in conjunction with a JuPyter notebook running IPython [10] is a particularly efficient way to quickly create an analysis pipeline that is easy to document, share, and reuse. To underscore this point, the public GitHub repository [6] contains the JuPyter notebook used to make every figure in this paper and the documentation [7] features that notebook as "Example 3". A permanent archive of the same, including the raw data, is hosted at Zenodo [1].

A. Data organization and management

The `scraps` package is built on the `Resonator` object. There is one `Resonator` object per data trace in an experiment

(several hundred in the above example). This object has attributes describing the experimental data, as well as any state variables like temperature or power, and contains the necessary infrastructure to run least-squares fitting and MCMC sampling on the constituent data. Results of fits are stored in the `Resonator` object. The most convenient way to work with a collection of `Resonator` objects is to make a Python list object out of them. `scraps` provides a tool that will create a list of `Resonator` objects from raw data, and another that will index such a list by temperature and power, making it easy to iterate through a list and perform operations on selected objects. The data-loading tool takes, as an argument, a custom function that maps the user's data file format to the `scraps` format, and is therefore completely data-format agnostic (at the expense of some extra work by the user). A default mapping function for output generated by most Agilent/Keysight VNAs is provided with the distribution and serves as a template.

The next level of hierarchy is the `ResonatorSweep` object. A `ResonatorSweep` object subclasses a Python `dict` object, adding a custom initialization method. It takes a list of `Resonator` objects and extracts all of the fit information into two-dimensional data structures. The core data-structure functionality is provided by the `pandas` package [11]. One `ResonatorSweep` object will contain such a structure for every parameter in the model used to fit the original trace data. The `ResonatorSweep` object is indexable by temperature and power, can do interpolation to fill in missing points, may be sliced to return any subset of data desired, and gracefully handles NaN (not a number) or `None` entries. Finally, the object is able to apply external fit models to the data and stores the results of those fits locally within the object.

Loading data in from hundreds of raw text files is a computationally expensive activity that shouldn't need to be done more than once. `Resonator` and `ResonatorSweep` objects can be rapidly cached to, and loaded from, disk using Python's `cPickle` package, which stores Python objects to a binary file. A corollary of this is that it is easy to enable multithreading with `scraps`. The only hardware requirement for using `scraps` is sufficient RAM to store an entire experiment in memory. Our NbN example chip has three physical resonators on it and a 2000 point trace was measured at 54 temperatures and 6 input powers for each resonator for a total of just under 1000 total traces. `scraps` requires about 350 MB of disk space to cache the entire set of data, including fits.

B. Fitting models to data

The software is designed for flexibility in model choice and application. A `scraps` model consists of two user-defined functions. One function specifies parameter names and calculates initial guesses. The other takes a set of parameters and the data and calculates a residual (weighted or otherwise). `scraps` then handles organizing and plotting the results, primarily through the creation of a `ResonatorSweep` object as described above. We include a model for fitting complex transmission data to an asymmetric Lorentzian (see inset of Fig. 1b) as well as a very basic model for fitting f_0 and Q_i shifts with power and temperature.

1) Fitting engines: There are two fitting engines built into `scraps`. The first is a Levenberg-Marquardt least-squares fitting routine that wraps the `lmfit` package [12]. The second is a Monte-Carlo Markov Chain (MCMC) sampler that assumes the residual calculated by the fit function is Gaussian, accepts an optional prior distribution for each parameter, and generates a posterior distribution for each parameter through Bayesian inference. `scraps` implements MCMC sampling through the `emcee` package [13]. The least-squares fitter is very fast, but can become stuck in local minima within the parameter space. The MCMC sampler takes more time, but does a better job exploring the parameter space and also generates the covariances of all the fit parameters, allowing one to extract a maximum-likelihood estimate for each parameter. It is also possible to specify prior distributions for any parameter using the MCMC sampler (the default is a flat prior), a feature that is not possible in a least-squares minimization.

The fitting engines are accessed through either the `Resonator` object (for fitting transmission data as a function of frequency and generating primary fit parameters) or the `ResonatorSweep` object (for fitting primary fit parameters as functions of temperature and/or power). In either implementation one may choose to use the least-squares routine, the MCMC routine, or both. The `ResonatorSweep` object can simultaneously fit any number of primary parameters (f_0, Q_i, etc.) vs. temperature and power, each to a different model, while constraining shared parameters to have the same value between models.

2) Included models: For fitting transmission data, `scraps` includes a model called `cmplxIQ` that fits an asymmetric Lorentzian model to the I and Q data vs frequency (see inset of Fig. 1b for equation). For a derivation of the asymmetric Lorentzian equation, see [14]. This model also has an option to compensate for electrical delay, mixer offset, and slowly-varying transmission-line resonances. Figure 1c shows an example of this model applied to a list of `Resonator` objects using the least-squares engine. Figure 1b shows the posteriors for the highest-temperature fit calculated with the MCMC engine using a flat prior.

For most superconducting materials, the primary causes of f_0 and Q_i shift in a given resonator are two-level system behavior (TLS) at low temperatures and low powers [15], and changing surface impedance as the temperature nears the critical temperature [16]–[22]. The latter is typically known as the Mattis-Bardeen effect (MBD). Although analytic expressions exist for limiting cases of each effect, generally a full numerical calculation must be done to describe a real material. We provide a simple fitting model for these two effects as an example for this paper, but we caution users not to apply it blindly. For a description of the model, see [23].

As a demonstration, we fit f_0 and Q_i (independently from each other) of our NbN resonator to the above model (that combines TLS and MBD behavior). It is likely that the approximations made in this model do not apply to NbN (no dependence on mean-free-path or coherence length at a minimum), and so the numerical values of the fit parameters should be viewed with skepticism. However, the model qualitatively captures the behavior of the data and so is useful as an

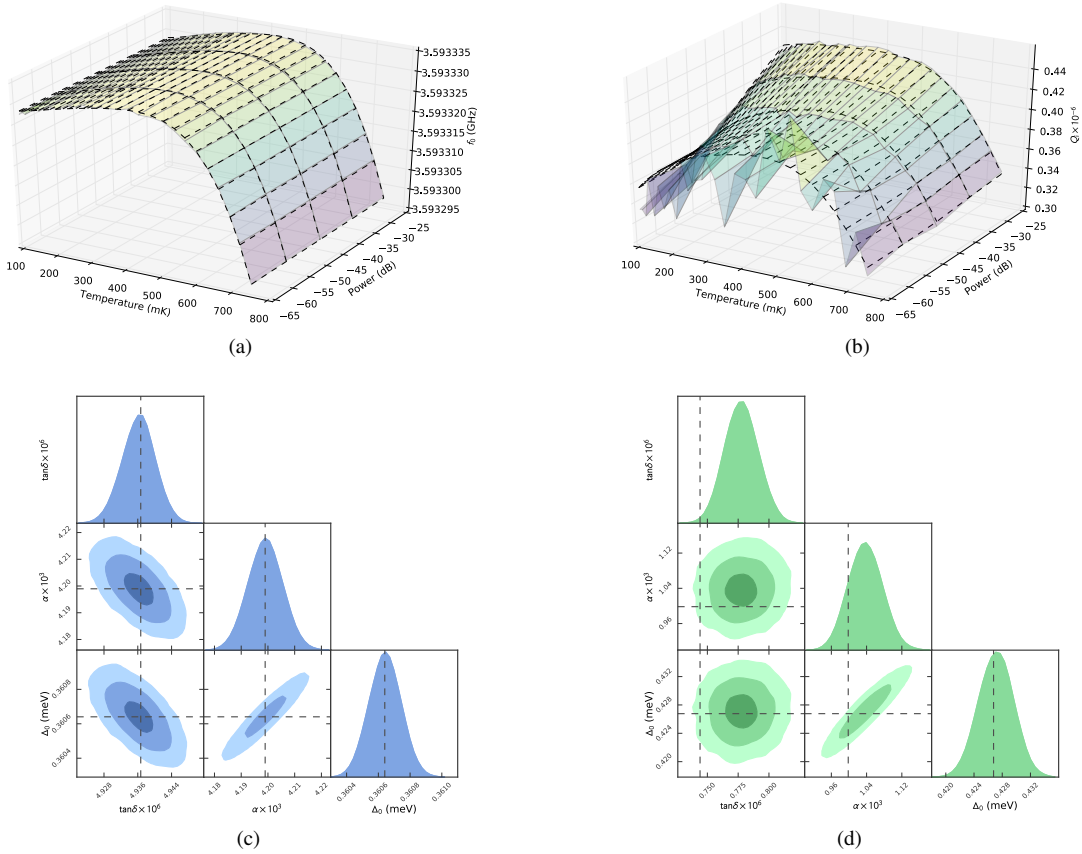


Fig. 2. The top two panels show the f_0 and Q_i surfaces as functions of temperature and power. The semi-transparent colored surfaces are produced by fitting the S_{21} traces at several different temperatures and powers. The black dashed wire-frame surfaces are the best fits (independant, not joint) to the 2D surfaces from the least-squares fitting engine. The bottom two panels show the parameter covariances, sampled with the MCMC engine, for f_0 (blue) and Q_i (green) for parameters that are shared by the two fit models. The black dashed lines correspond to the best-fit values from the least-squares algorithm.

example. The results of these fits are displayed in Fig. 2. The top panels show the f_0 and Q_i surfaces (as functions of temperature and power). The dashed black meshes superimposed on the surfaces correspond to the results of the fits. The bottom panels show the posteriors (given a flat prior), calculated by the MCMC sampler, for those parameters shared between the two models. Although the parameter values returned by the two models agree within an order of magnitude, it is clear that a joint fit would do little to resolve the tension, and so we do not demonstrate that here.

C. Plotting data

One of the core tenets of `scraps` is that it should be easy to make a beautiful plot for a paper, poster, or group meeting. To this end, we have included several plotting routines to generate a number of useful plots. All the plotting routines are built on the `matplotlib` package [24]. Usage is as simple as calling a plot function, passing it a list of `Resonator` objects or a `ResonatorSweep` object and specifying a few options. Using the functions included in the plotting tools it is possible to view almost any combination of data along any axis. `scraps` was used to generate Figs. 1a, 1c, 2a, and 2b without external editing.

IV. Conclusion and Future of Development

The `scraps` package is undergoing continual development at Argonne as we work to extend it to satisfy our analysis and plotting needs. The next feature we are working on is adding the ability to handle noise data. We are also considering extending it to handle magnetic field as an independent variable, in addition to temperature and input power, given that recent conversations with colleagues, and our own experimental data (unpublished), suggest that the resonator quality factor is a strong function of magnetic field under certain circumstances. Finally, we are working towards adding "good" models that are geared towards specific geometries and materials; this process in particular would benefit from collaboration!

We conclude by inviting the superconducting resonator community to install, test, modify, contribute to, and critique our new package. We also remind the reader that all of the code used to analyze the data and make plots for this paper, along with the raw data, is available at [1] and [2]. The latest version of the code is at [6].

Acknowledgment

We thank Drs. S. Padin, G. Wang, E. Shirokoff, R.B. Thakur, C. Posada, J. Ding, V. Yefremenko, A. Bender, and L. Bleem for helpful conversations and/or measurement advice.

REFERENCES

[1] F. W. Carter, T. S. Khaire, V. Novosad, and C. Chang, "Raw data and analysis pipeline for producing figures in F.W. Carter, et. al., 2016," Sep. 2016. [Online]. Available: http://dx.doi.org/10.5281/zenodo.61575

[2] F. W. Carter, "scraps: Scraps v0.2," Sep. 2016. [Online]. Available: http://dx.doi.org/10.5281/zenodo.61512

[3] ——, "HPD Olympus ADR cryostat control," Sep. 2016. [Online]. Available: http://dx.doi.org/10.5281/zenodo.61406

[4] ——, "Control software for sweeping ADR temperature and collecting PNA data," Sep. 2016. [Online]. Available: http://dx.doi.org/10.5281/zenodo.61407

[5] S. Bocquet and F. W. Carter, "pygtc: beautiful parameter covariance plots (aka. giant triangle confusograms)," *The Journal of Open Source Software*, vol. 1, no. 6, oct 2016. [Online]. Available: http://dx.doi.org/10.21105/joss.00046

[6] F. W. Carter, "scraps: SuperConducting Resonator Analysis and Plotting Software," 2016. [Online]. Available: http://github.com/FaustinCarter/scraps

[7] ——, "Scraps program documentation," http://scraps.readthedocs.io, 2016.

[8] "The Python Packaging Index," http://pypi.python.org/pypi.

[9] "pip: The Python Packaging Association's recommended tool for installing Python packages," http://github.com/pypa/pip.

[10] F. Pérez and B. E. Granger, "IPython: a system for interactive scientific computing," *Computing in Science and Engineering*, vol. 9, no. 3, pp. 21–29, May 2007. [Online]. Available: http://ipython.org

[11] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51 – 56.

[12] M. Newville, A. Nelson, A. Ingargiola, T. Stensitzki, D. Allan, Michał, Glenn, Y. Ram, MerlinSmiles, L. Li, G. Pasquevich, C. Deil, D. M. Fobes, Stuermer, T. Spillane, ampolloreno, stonebig, P. A. Brodtkorb, R. Clarken, K. Anagnostopoulos, A. Almarza, and B. Gamari, "lmfit-py 0.9.5," Jul. 2016. [Online]. Available: http://dx.doi.org/10.5281/zenodo.58759

[13] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, "emcee: The mcmc hammer," *Publications of the Astronomical Society of the Pacific*, vol. 125, no. 925, p. 306, 2013. [Online]. Available: http://stacks.iop.org/1538-3873/125/i=925/a=306

[14] K. Geerlings, "Improving Coherence of Superconducting Qubits and Resonators Improving Coherence of Superconducting Qubits and Resonators," Ph.D. dissertation, Yale University, 2013, see Eq. A.18.

[15] W. A. Phillips, "Two-level states in glasses," *Reports on Progress in Physics*, vol. 50, no. 12, p. 1657, 1987. [Online]. Available: http://stacks.iop.org/0034-4885/50/i=12/a=003

[16] D. C. Mattis and J. Bardeen, "Theory of the Anomalous Skin Effect in Normal and Superconducting Metals," *Physical Review*, vol. 111, no. 2, pp. 412–417, jul 1958. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRev.111.412

[17] S. B. Nam, "Theory of electromagnetic properties of superconducting and normal systems. I," *Physical Review*, vol. 156, no. 2, pp. 470–486, 1967.

[18] ——, "Theory of electromagnetic properties of strong-coupling and impure superconductors. II," *Physical Review*, vol. 156, no. 2, pp. 487–493, 1967.

[19] T. Noguchi, M. Naruse, and Y. Sekimoto, "RF conductivity and surface impedance of a superconductor taking into account the complex superconducting gap energy," *Physics Procedia*, vol. 36, pp. 318–323, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.phpro.2012.06.166

[20] D. Rainer and J. Sauls, "Strong-Coupling Theory of Superconductivity," *Superconductivity: From Basic Physics to the Latest Developments*, pp. 45–78, 1995. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/9789814503891{_}0002

[21] J. P. Turneaure, J. Halbritter, and H. A. Schwettman, "The surface impedance of superconductors and normal conductors: The Mattis-Bardeen theory," *Journal of Superconductivity*, vol. 4, no. 5, pp. 341–355, 1991.

[22] W. Zimmermann, E. H. Brandt, M. Bauer, E. Seider, and L. Genzel, "Optical conductivity of BCS superconductors with arbitrary purity," *Physica C: Superconductivity and its applications*, vol. 183, no. 1-3, pp. 99–104, 1991.

[23] J. Gao, "The Physics of Superconducting Microwave Resonators," Ph.D. dissertation, California Institute of Technology, 2008, see Eqs. 2.80, 2.88, 2.89, 5.65, 5.74, and 5.75.

[24] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.