Article as accepted for publication in *Journal of Magnetic Resonance*.
License: CC-BY-NC-ND
The final published article can be accessed via its DOI: 10.1016/j.jmr.2021.107140
Please cite this article as:

M. Schröder, T. Biskup: *J. Magn. Reson.* 335:107140, 2022

cwepr – a Python package for analysing cw-EPR data focussing on reproducibility and simple usage

Mirjam Schröder[a], Till Biskup[b,c,*]

[a] *Leibnitz-Institut für Katalyse e.V., Albert-Einstein-Straße 29a, 18059 Rostock, Germany*
[b] *Institut für Physikalische Chemie, Albert-Ludwigs-Universität Freiburg, Albertstraße 21, 79104 Freiburg, Germany*
[c] *Current Address: Physikalische Chemie, Universität des Saarlandes, Campus B2 2, 66123 Saarbrücken, Germany*

Abstract

Reproducibility is at the heart of science. Nevertheless, with the advent of computer-based data processing and analysis, most spectroscopists have a hard time fully reproducing a figure from last year's publication starting from the raw data. Unfortunately, this renders their work eventually unscientific. To change this, we need to develop analysis tools that relieve their users from having to trace each processing and analysis step. Furthermore, these tools need to be modular, extendible, and easy to use in order to get used. To this end, we present here the open-source Python package cwepr based on the ASpecD framework for reproducible analysis of spectroscopic data. This package follows best practices of both, science and software development. Key features include an automatically generated gap-less record of each individual processing and analysis step from the raw data to the final published figure. Additionally, it provides a powerful user interface requiring no programming skills of the user. Due to its code quality, modularity, and extensive documentation, it can be easily extended and is actively developed by spectroscopists working in the field. We expect this approach to have a high impact in the field and to help fighting the looming reproducibility crisis in spectroscopy.

Keywords: reproducible research, electron paramagnetic resonance spectroscopy, software, data analysis

1. Introduction

Why does reproducibility matter, and what are its prerequisites? Why is cw-EPR spectroscopy still relevant, despite pulsed EPR methods being routinely available? Why yet another software package for analysing cw-EPR data, and what makes this one different? We will briefly address these three questions before presenting the cwepr Python package for analysing cw-EPR data focussing on reproducibility and simple usage.

Reproducibility. Reproducibility of scientific results is not only fundamental to science [1, 2, 3], but as well to the fact-based further development and survival of mankind [4]. Problems with the lack of reproducibility have long been known [5, 6, 7, 8, 9, 10, 11, 12, 13, 14], not only in context of software-based data analysis [15, 16, 17, 18]. However, only recently general guidelines and rules, namely the FAIR principles [19], have been developed and are now increasingly being supported [20] and enforced [21] by funding bodies.

The prerequisites for reproducible research start with obtaining all relevant metadata during data acquisition. Next is a gap-less record of each data processing step from the raw data to the final publication. And finally, data need to be present in formats applicable to long-term storage. As an overarching principle, as many aspects as possible need to be automated. Only this ensures all relevant information to be recorded and allows us to focus on the intellectual rather than routine tasks of science [22, 23]. Fig. 1 provides an overview of the workflow of reproducible data analysis.

cw-EPR spectroscopy. With the advent of pulsed EPR methods, cw-EPR spectroscopy is sometimes

*Corresponding author
Email address: research@till-biskup.de (Till Biskup)

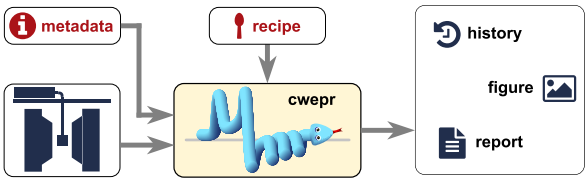

Figure 1: Data collection and analysis pipeline focussing on reproducibility. Data acquisition (left) is followed by data processing and analysis using the cwepr package (centre) and results in graphical representations, reports, and a gap-less history (right). Only metadata and recipe require user input.

considered to be of minor relevance. However, for good reasons cw-EPR spectroscopy is still the method of choice to address highly relevant questions in many areas from medicine to materials science. Probably the most important reason is the generally short relaxation times of paramagnetic species. Thus, pulsed EPR methods [24] usually require cryogenic sample temperatures, while cw-EPR spectra can often be recorded for samples kept at room temperature. This allows for *in-vivo*, *in-situ*, and high-temperature measurements relevant for biological as well as materials science applications [25]. Furthermore, there are paramagnetic species that simply escape detection by pulsed EPR methods and that can only be detected by cw-EPR spectroscopy [26]. And eventually, quantitative analyses of paramagnetic species are not regularly possible using pulsed methods, but highly relevant not only in semiconductor research [27, 28, 29]. However, cw-EPR spectroscopy can only develop its full potential if it is applied in a routine fashion [30]. This requires robust workflows covering every step from data acquisition to presenting the results of the analysis.

Analysis software. Analysing cw-EPR data is much more complex as it may appear, and far too much time is spent reinventing the wheel over and over again. Simulating EPR spectra can be considered solved with EasySpin [31] having become the *de facto* standard. However, simulating and fitting EPR spectra is only one aspect of data analysis, and it nearly always requires extensive preprocessing of the raw data. While many programs can import data from different formats, they usually lack a *unified representation* of the data and their accompanying metadata, making parameter-dependent processing unnecessarily complicated. As an example, the microwave frequency is abbreviated 'MF', 'MWFQ', and 'MwFreq' in different file formats and stored in parameters of this name using, *e.g.*, the EasySpin `eprload` function. Furthermore, the authors know of no software solution focussing on reproducibility and a gap-less (and replayable) record of each data processing step that covers the entire process from raw data to final publication.

To be successful, the software needs to be freely available, platform-independent, modular and thus easily extendible, well-documented and easy to use. Furthermore, it needs to be capable of dealing with both routine and complex analysis tasks, and providing obvious advantages over existing solutions and the usual habit of writing individual scripts for each analysed series of datasets. Additionally, it should allow users without programming skills to perform even complex tasks in a straight-forward manner, while making it easy for more experienced programmers to extend the software according to their needs.

The article is organised as follows: First, we briefly review the cw-EPR method focussing on the routine tasks necessary during parameter optimisation and data acquisition. Next, we give an overview of the cwepr Python package and the concepts underlying its design. Finally, we showcase its capabilities by presenting a series of real-life examples of analysing cw-EPR data.

2. Practical aspects of cw-EPR spectroscopy

An introduction to EPR spectroscopy is clearly beyond the scope of the present article, and the interested reader is referred to the literature for both general [32, 33, 34, 35, 36, 37] and more theoretical [38, 39, 40] aspects. Probably the best introduction into practical aspects of cw-EPR spectroscopy is the book by Eaton *et al.* on quantitative EPR spectroscopy [41]. Here, we will briefly mention those practical aspects of the method directly connected to both, reproducibility as well as data processing and analysis, that are directly relevant for using the cwepr Python package described hereafter.

While not specific to cw-EPR spectroscopy, it is important to mention that reproducible science does not start once the data have been acquired, but even before. This requires implementing a workflow that guarantees recording of all relevant metadata of an experiment as well. Depending on the hardware used, many parameters are already recorded and saved together with the actual data. However, important information not regularly entering the different vendor file formats ranges from

general information about the sample temperature (and cryostat, if any) to probehead and tube used, not to mention the purpose of the whole measurement. Writing this information by hand in a lab notebook does not allow for accessing it automatically during data analysis. Hence, a simple file format designed to be easily *written* by humans while retaining machine-readability together with the habit of collecting all these pieces of relevant information during data acquision is the first necessary step in a long journey towards complete reproducibility. An example of such a file format (termed info file) is discussed in the Supporting Information.

Aspects more specific for cw-EPR spectroscopy are spectrometer calibration and parameter optimisation. If a setup does not get changed regularly, it can usually be considered properly calibrated. However, as soon as a probehead is changed or a cryostat inserted or removed, at least the external magnetic field needs to be recalibrated or alternatively a field standard measured directly before or after measuring the actual samples. For further calibration tasks such as modulation amplitude, the reader is referred to either the hardware manufacturer's manuals or again the Eaton book [41].

The most important parameters that need to be optimised for each sample individually are magnetic field range, modulation amplitude, and microwave power. While optimising the first two is usually straight-forward using visual inspection of the recorded spectra, finding the optimal microwave power level without even slightly saturating the signal is already more involved. While some vendor-specific software comes equipped with analysis routines for this purpose, it is otherwise a first example how a robust and easy-to-use software package comes in quite handy.

Once the parameters have been optimised and the cw-EPR spectra obtained for the sample, together with the relevant metadata, data processing and analysis can begin. As a bare minimum, this normally consists of a (polynomial) baseline correction and a frequency correction to the same microwave frequency as prerequisite for meaningful comparison between different recorded spectra. A magnetic field correction is necessary in case of the spectrometer not being calibrated with respect to its external magnetic field. In its simplest form, a constant magnetic field offset is extracted from the recorded EPR signal of a standard sample with accurately known g value and narrow and isotropic line. Details of how to achieve these and other steps are provided in the examples section below.

Another frequent requirement, particularly for spectra recorded at low temperature or with weak signals, is subtracting a background from either or both of probehead and sample tube used. Therefore, recording cw-EPR data for an empty tube or the empty probehead under as identical conditions as possible with respect to the recordings of the actual sample should be a routine task, too. Needless to mention that these data should be recorded for a slightly broader external magnetic field range as the actual data, as due to the different microwave frequency data will be shifted after frequency correction. Due to the unavoidably different microwave frequency of signal and background spectrum, a common axis range needs to be extracted for both datasets and the data interpolated to a common grid. No rocket science at all, but considerably more involved than a simple subtraction of two vectors, and a recurring task that should be solved once and forever in code in a package for data analysis and nothing to think about.

Other straight-forward tasks in the analysis of cw-EPR data are normalisation (to maximum, minimum, amplitude, or area), peak finding, extracting peak-to-peak distances, integrating, averaging, and filtering. Converting the magnetic field axis to a g axis or adding a g axis as a second axis, though less frequent, is similarly simple. Regarding filtering, *i.e.* smoothing, a word of caution shall be added: While data shall never be smoothed to look more pleasing to the eye, filtering can be an important prerequisite for both, extracting basic characteristics such as peak positions, as well as dramatically enhancing the quality of fitting spectral simulations to data. Therefore, it is imperative to provide a full record of each individual data processing and analysis step including the full set of explicit and implicit parameters used, to allow others to independently reproduce, understand, and judge both, the analysis as well as the quality and reliability of the arguments based on its results.

Power saturation analysis, *i.e.* the procedure usually employed to optimise the microwave power of an experiment, as well as quantiative EPR spectroscopy (here understood as extracting relative or absolute concentrations of paramagnetic centres in a sample) can be regarded as more complex tasks involving a series of different processing and analysis steps. Here, easily obtaining graphical representations of the data after each individual step is even more important to be able to judge the quality and

reliability of the analysis.

While usually, full analysis of cw-EPR results requires to fit spectral simulations to the experimental data to extract the spin Hamiltonian parameters and getting access to the underlying physics, fitting is an entirely different topic on its own. The cwepr Python package described here does not and will not have simulation and fitting capabilities itself, but it provides general interfaces for both, as fitting spectral simulations can be regarded as one special form of an analysis task. Python packages for both, spectral simulations [42] and fitting [43], are currently being developed, again with a focus on modular, open-source software designed for a maximum of reproducibility.

After this brief survey of processing and analysis tasks regularly encountered when dealing with cw-EPR spectra, the stage is prepared to introduce the cwepr Python package for analysing cw-EPR data focussing on reproducibility and simple usage.

3. The cwepr Python package

The bar has been set pretty high for the cwepr Python package, as it should be open-source, modular, easy to use, and guarantee full reproducibility. At the same time, it should allow to perform all the routine processing and analysis tasks regularly encountered for cw-EPR spectroscopy and mentioned in the previous section. We are convinced that the cwepr package can fulfil all these promises. Nevertheless, rather than persuading people to use a particular piece of software, it is much more important to highlight the significance of truly reproducible research and the need to develop and apply strategies that turn it into a reality. Therefore, not only the features, but the underlying ideas will be briefly described below, as they are much more generally applicable.

One of the particular strengths of the cwepr Python package is its simple user interface. As the package is based on the ASpecD framework [44], it supports 'recipe-driven data analysis': The user creates a simple, structured text file containing a list of datasets to load and a list of tasks to perform on these datasets. A first example of such a recipe is provided in Listing 1. Getting served the results of 'cooking' this recipe is as simple as issuing a single command in the terminal.

The idea behind recipe-driven data analysis is to reduce complexity and to allow the user to focus on the actual science, namely data processing and analysis. Usually, we have an idea which tasks we want to perform on a dataset, and we will even have ideas which parameters we would need for the individual tasks. All this enters the recipe in a highly structured and obvious way. While the details of the individual tasks will be discussed below, the recipe presented in Listing 1 should be pretty self-explanatory (not only) for a spectroscopist used to dealing with cw-EPR data.

Listing 1: Example of a recipe used for recipe-driven data analysis within the cwepr Python package. Here, a list of datasets is followed by a list of tasks. The user needs no programming skills, but can fully focus on the tasks to be performed. 'Cooking' this recipe is a matter of issuing a single command on the terminal.

```
format:
  type: ASpecD recipe
  version: '0.2'

settings:
  default_package: cwepr

datasets:
  - /path/to/first/dataset
  - /path/to/second/dataset

tasks:
  - kind: processing
    type: FrequencyCorrection
    properties:
      parameters:
        frequency: 9.5
  - kind: processing
    type: BaselineCorrection
  - kind: singleplot
    type: SinglePlotter1D
    properties:
      filename:
        - first-dataset.pdf
        - second-dataset.pdf
```

But what about reproducibility? Upon 'cooking' the recipe presented in Listing 1 and serving its results, a history will be written detailing each individual step. For a first impression, cf. Listing 2. As this is a valid recipe in itself, it serves a dual purpose: (i) it contains all information necessary to fully reproduce the analysis, including the list and version of all relevant Python packages and all explicit and implicit parameters, and (ii) it can be used to automatically rerun the analysis.

Providing an extensive user manual is beyond the scope of a journal article, and the interested reader is referred to the extensive user and developer documentation available online for both, the

Listing 2: Excerpt of the history automatically written by serving the example recipe displayed in Listing 1. Notable are the automatically added blocks at the top containing information on the time of execution as well as the system used, including version numbers of all relevant Python packages. Furthermore, as the baseline correction results in different coefficients for each of the two datasets, those are separately presented for each individual dataset.

```
info:
  start: '2021-11-26T09:03:52'
  end: '2021-11-26T09:03:57'
system_info:
  python:
    version: "3.7.3 ..."
  packages:
    aspecd: 0.6.4
    jinja2: 3.0.2
    matplotlib: 3.4.3
    numpy: 1.21.3
    scipy: 1.7.1
    oyaml: '1.0'
    asdf: 2.8.1
    bibrecord: 0.1.0
    cwepr: 0.2.0
  platform: Linux-4.19.0-18-...
  user:
    login: johndoe
format:
  type: ASpecD recipe
  version: '0.2'
settings:
  default_package: cwepr
  # ...
datasets:
- /path/to/first/dataset
- /path/to/second/dataset
tasks:
- kind: processing
  type: BaselineCorrection
  properties:
    parameters:
      kind: polynomial
      order: 0
      coefficients:
      - -0.06901404916763308
      fit_area:
      - 10
      - 10
      axis: 0
  apply_to:
  - /path/to/first/dataset
- kind: processing
  type: BaselineCorrection
  properties:
    parameters:
      kind: polynomial
      order: 0
      coefficients:
      - -0.07042420227050784
      fit_area:
      # ... remainder same as above
  apply_to:
  - /path/to/second/dataset
# ...
```

cwepr package [45] and the ASpecD framework [44] it is based upon. An overview can be found in the Supporting Information. For details on how ASpecD and derived packages are implemented, see Ref. [46]. Here, we briefly describe the underlying concepts, and in the next section, we showcase the capabilites of the package presenting a few real-life examples of analysing cw-EPR data.

3.1. Data import and supported formats

Data are represented within the cwepr package as 'datasets', *i.e.* the unit of (numerical) data and accompanying metadata. As mentioned above, a lot of crucial parameters are usually recorded by the vendor-specific software and stored in the respective data formats. However, some essential information regularly remains unaccounted for, such as details regarding the sample, the purpose of the measurement, and probehead and cooling system used. In any case, it is the responsibility of the scientist performing the measurements to record the missing information *during data acquisition*, at best in an electronic format that can be read directly by the analysis software.

In terms of vendor file formats, the cwepr Python package currently supports the different Bruker file formats for the old ESP and EMX spectrometer series as well as the newer BES3T format. Additionally, Magnettech XML files can be read, and as a last resort, bare text files (CSV and alike) can be imported. The latter, however, usually lack any metadata. Thanks to the highly modular architecture of the cwepr package, adding importers for additional file formats is simple and straight-forward. Details can be found in the package documentation available online [45]. What is much more relevant for the user of the package: File formats will be auto-detected and the respective importer chosen.[1]

3.2. Data processing

Data processing is a necessary prerequisite for data analysis, and therefore separate from the latter. The difference between processing and analysis in context of the cwepr Python package: Processing steps operate on datasets and always return (modified) datasets, while analysis steps operate on datasets and extract information, but may return

[1] In software engineering terms, a factory pattern [47] is used here.

everything from a scalar value to a full (calculated) dataset, depending on the type of analysis step.

The processing currently available within the cwepr Python package can be categorised further: corrections, simple algebra, normalisation, handling two-dimensional datasets, and working with multiple datasets (dataset algebra).

3.3. Data analysis

In nearly all cases, data analysis needs to be preceded with data processing. While processing steps can often be automated to a large extend and are rather generally applicable, data analysis is usually much more focussed on individual types of measurements and the actual questions at hand.

Most important, the analysis steps provided by the cwepr package are generally meant as basic building blocks to be used in arbitrarily complex overall analyses, consisting of large lists of processing and analysis steps, usually interspersed with plotting steps for graphical feedback. Eventually, the possibilities are only limited by the user's creativity and imagination. This is the focus and power of the cwepr package: freeing the users from dealing with the implementation details of each individual processing and analysis step and allowing them to creatively combine the different tasks in a fully transparent manner. For more complicated and time-consuming tasks, the analysis can even be run fully unattended.

3.4. Data representation: plotting

Graphically representing the results of processing (and analysis) steps is of paramount importance in data analysis, not only as means of finally presenting the results, but for ensuring that the individual tasks performed on the data give sensible results. Furthermore, given that datasets contain both, (numerical) data as well as accompanying metadata, correct axes labels can (and will) be created fully automatically. By using the Matplotlib library [48] of the scientific Python stack, publication-quality figures are readily available.

While the plotters provide sensible defaults, the appearance of figures can be controlled in quite some detail. Furthermore, predefined styles can be applied. For an example of the capabilities of styling, though admittedly with limited use in serious scientific publications, cf. Fig. 2. More details including how to define own styles can be found in the Matplotlib documentation available online [49].

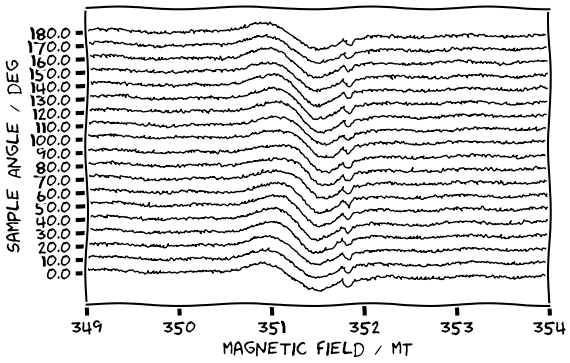

Figure 2: Applying predefined styles to a graphical representation. A (not so serious) option provided by the Matplotlib library is the 'xkcd' style named after the famous web comic created by Randall Munroe (`https://xkcd.org/`).

3.5. Report generation: accessing information

While plotters are an excellent way to obtain publication-quality figures without hassle, and the recipe history automatically created contains all information necessary to fully reproduce and replay the tasks, there is a lot more of information contained in the datasets and potentially the recipes as well. The latter is even more true in light of recipes supporting adding comments to each individual task, as well as figure captions to plotters. Hence, being able to automatically create well-formatted reports using pre-defined templates opens an entirely new dimension in terms of comparing different datasets and workflows, besides presenting the results of the research. An example of a report presenting all information contained in a dataset is provided in the Supporting Information.

Automatically generating not only the figures for a manuscript or thesis, but as well the captions, and having them included in the main text, is only one possible application. Besides that, generating reports of (complex) routine processing and analysis workflows for individual datasets provides means to easily compare the results. Never underestimate the power of well-formatted and uniform reports allowing to focus on the differences rather than having to find the parameters to compare in different places.

3.6. Data export and supported formats

As most data processing and analysis tasks are not too time-consuming and can always be repeated starting from the raw data, saving the re-

sulting processed datasets may not be an immediate need. Nevertheless, for more complex tasks this may change. Therefore, two particular formats for datasets are supported by the cwepr package by means of the underlying ASpecD framework: the 'Advanced Scientific Data Format' (ASDF) [50] and a format particularly developed for the ASpecD framework and termed 'ASpecD Dataset Format' (ADF). Both are fully self-contained, *i.e.*, come with their own specification, and are thus platform-independent, relying on well-developed standards. Furthermore, for a maximum of interoperability, data can be exported to plain text. Note, however, that in this case usually all metadata accompanying the data will be lost, rendering this a choice of last resort. As with data import, writing own exporters is both, straight-forward and simple. Details can be found in the documentation available online [45].

After this general overview of the (still growing) functionality of the cwepr Python package for analysing cw-EPR data, the next section provides a series of real-world examples.

4. Examples

Each of the following examples focusses on a particular task of processing and analysing cw-EPR spectra and highlights specific aspects of the cwepr Python package. All examples operate on real data, although the origin and exact context of these data do not matter, hence no details in this regard will be given. While the recipes presented in this section are shortened to highlight only the crucial aspects for clarity, the full working recipes are provided in the Supporting Information. Further details may be found in the growing list of examples in the documentation of the cwepr Python package available online [45].

4.1. Compare a series of recorded spectra

A standard situation in cw-EPR spectroscopy is to record spectra for a series of samples, usually with at least one parameter of the sample varied, but with comparable experimental conditions. Therefore, a quick overview of the data recorded is the first step towards data analysis.

To have a meaningful comparison of the data obtained for the different samples, a series of standard processing steps are necessary: baseline correction, frequency correction, and probably normalisation. Afterwards, all spectra should be plotted on top of each other into one common axis. The crucial steps are summarised in Listing 3. Normalisation to amplitude is used here for easier comparison of differences in the line shape. The result of these steps is displayed in the top panel of Fig. 3.

Listing 3: Key steps of data processing for comparing cw-EPR spectra recorded for a series of samples under mostly identical experimental conditions.

```
- kind: processing
  type: BaselineCorrection
- kind: processing
  type: FrequencyCorrection
  properties:
    parameters:
      frequency: 9.48
- kind: processing
  type: Normalisation
  properties:
    parameters:
      kind: amplitude
- kind: multiplot
  type: MultiPlotter1D
```

Listing 4: Applying a filter to a series of data for smoothing. The Savitzky-Golay filter is particularly useful in this case, as it does not distort the spectral shape.

```
- kind: processing
  type: Filtering
  properties:
    parameters:
      type: savgol
      window_length: 501
      order: 5
```

Although the data shown here have already a fairly good signal-to-noise ratio, filtering (*i.e.* smoothing) the data can help with investigating the rather subtle differences in the line shape. Therefore, in this case, a Savitzky-Golay filter [51] has been applied, cf. Listing 4. Note that the rather large window applied here (501 points) is due to the data having been recorded with a spectrometer using a high sampling rate of the magnetic field axis. While filtering clearly has its merits, one should never present only the filtered data, but always the unfiltered data as a comparison, as in Fig. 3.

This is a good example for the need of creating a figure that consists of several panels. Even this can be elegantly done within a recipe by defining individual plotters for each of the panels and using a special plotter (a `CompositePlotter`) to create the final figure, calling out to the individual plotters

Listing 5: Creating a figure consisting of several panels using individual plotters and a `CompositePlotter`. Key is to define labels for the individual plotters in the `result` field and use these labels to refer to the plotters in the `CompositePlotter`. Grid dimensions and subplot locations are explicitly given in the latter and are thus entirely flexible.

```
- kind: multiplot
  type: MultiPlotter1D
  apply_to:
    # List of unfiltered datasets
  result: unfiltered
- kind: multiplot
  type: MultiPlotter1D
  apply_to:
    # List of filtered datasets
  result: filtered
- kind: compositeplot
  type: CompositePlotter
  properties:
    plotter:
    - unfiltered
    - filtered
    filename: original-vs-filtered.pdf
    grid_dimensions: [2,1]
    subplot_locations:
    - [0, 0, 1, 1]
    - [1, 0, 1, 1]
```

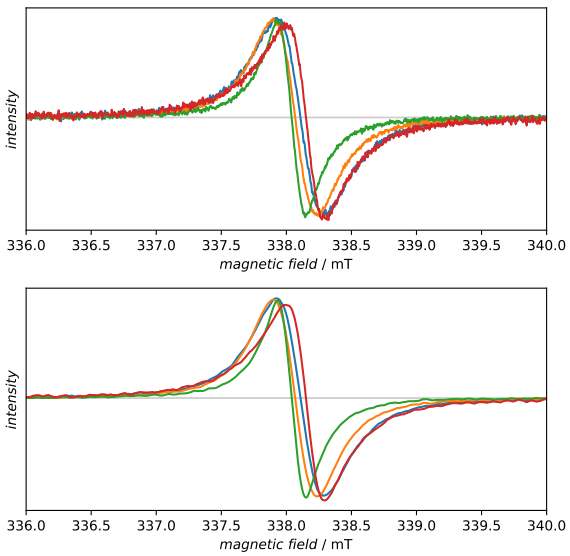

Figure 3: Comparing cw-EPR spectra of a series of samples under comparable experimental conditions. The top panel shows the baseline and frequency-corrected and amplitude-normalised data. In the bottom panel, the result of filtering is shown. This filtering allows for a more detailed investigation of the subtle differences. Nevertheless, never only filtered data should be presented.

for the individual panels. A stripped-down example is given in Listing 5, for the full recipe see the Supporting Information.

4.2. Power saturation analysis

As mentioned earlier, one important parameter to optimise in cw-EPR spectroscopy is the microwave power, as saturation of the EPR signal will result in line broadening. To find the optimal microwave power level, one usually performs a series of measurements with systematically varied power and afterwards plots the cw-EPR signal amplitude as a function of the square root of the microwave power, resulting in a power saturation curve. A linear dependence between these two quantities is characteristic for non-saturating conditions, whereas deviation from this linearity reveals the onset of saturation. A recipe containing all steps necessary for a full power saturation analysis, including graphical representation of the results, is shown in Listing 6.

Here, not only the power saturation curve is plotted, but a linear regression covering the first n points (here $n = 5$) as well. The different reading points are represented by asterisks for clarity. While all this can be readily done using a normal multiplotter as in the previous example, having a second axis with the actual microwave power rather than its square root comes in quite handy when determining the optimal value for further measurements. This can be done using the special `PowerSweepAnalysisPlotter`. The result of a slightly more complex plot is shown in Fig. 4.

4.3. Subtracting a recorded background signal

Background signals originating from tube and/or probehead are a typical issue of cw-EPR spectra recorded at low temperature, with overall weak signals, or a broad magnetic field range. Usually, the background signal gets recorded separately and needs to be subtracted from the spectra of the actual samples. Two aspects make this seemingly simple operation rather complex. Upon correcting the spectra for the same microwave frequency, data need to be interpolated to a common field axis range and grid. Furthermore, subtracting the background requires adjusting the intensity of spectra and background beforehand. The latter can often be done by normalising over a restricted range of the field

Listing 6: Complete steps of a power saturation analysis. In a first step, the cw-EPR signal amplitude and square root of the microwave power are returned as calculated dataset, afterwards a linear regression performed over the first few points. The results of both are graphically represented together, using a special plotter adding a second axis with the actual microwave power values on top.

```
datasets:
- PowerSweep
tasks:
- kind: processing
  type: BaselineCorrection
- kind: singleanalysis
  type: AmplitudeVsPower
  apply_to:
  - PowerSweep
  result: power_sweep_analysis
- kind: singleanalysis
  type: PolynomialFitOnData
  properties:
    parameters:
      order: 1
      points: 5
      return_type: dataset
  apply_to:
  - power_sweep_analysis
  result: fit
- kind: multiplot
  type: PowerSweepAnalysisPlotter
  properties:
    properties:
      drawings:
        - marker: '*'
        - color: red
      grid:
        show: true
        axis: both
      axes:
        title: Overview
        ylabel: '$EPR\ amplitude$'
        yticklabels: []
  apply_to:
  - power_sweep_analysis
  - fit
```

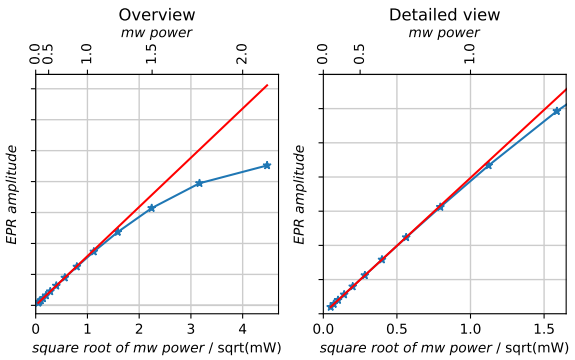

Figure 4: Power saturation curve together with a linear regression covering the first five data points. The left panel provides an overview of the entire power saturation measurement, clearly showing the onset of saturation with higher microwave power. The right panel is a detailed view, allowing to choose an optimal microwave power level.

Listing 7: Central steps to subtract the background spectrum: normalisation to the characteristic background signal at around 365 mT, interpolating to a common axis, and actual subtraction.

```
- kind: processing
  type: Normalisation
  properties:
    parameters:
      kind: amplitude
      range: [357, 375]  # in mT
      range_unit: axis
- kind: multiprocessing
  type: CommonRangeExtraction
- kind: processing
  type: DatasetAlgebra
  properties:
    parameters:
      kind: minus
      dataset: background
  apply_to:
    - compound1
    - compound2
    - compound3
```

axis. Listing 7 shows an excerpt of the recipe featuring the central steps of the baseline subtraction. The entire process is demonstrated in Fig. 5.

After baseline subtraction and frequency correction, spectra are normalised to the same amplitude in the range containing the main signal of the background. Providing the range in axes units (rather than vector indices) is both, convenient for the user and simple to retrace. The upper right panel of Fig. 5 provides a closer look at the results of normalising. Interpolating all datasets to a common x axis range and grid requires only two lines in the recipe. The actual implementation of this step is necessarily much more verbose. Subtracting the scaled background is again straight-forward. The result is shown in the lower right panel of Fig. 5. A subsequent normalisation to the amplitude without defining a specific range normalises all spectra to their largest or overall signal (Fig. 5, bottom left).

To keep the overview of all steps performed on a single dataset, a report can be automatically generated, cf. Listing 8. Here, a LaTeX template is used that comes bundled with the cwepr package. The

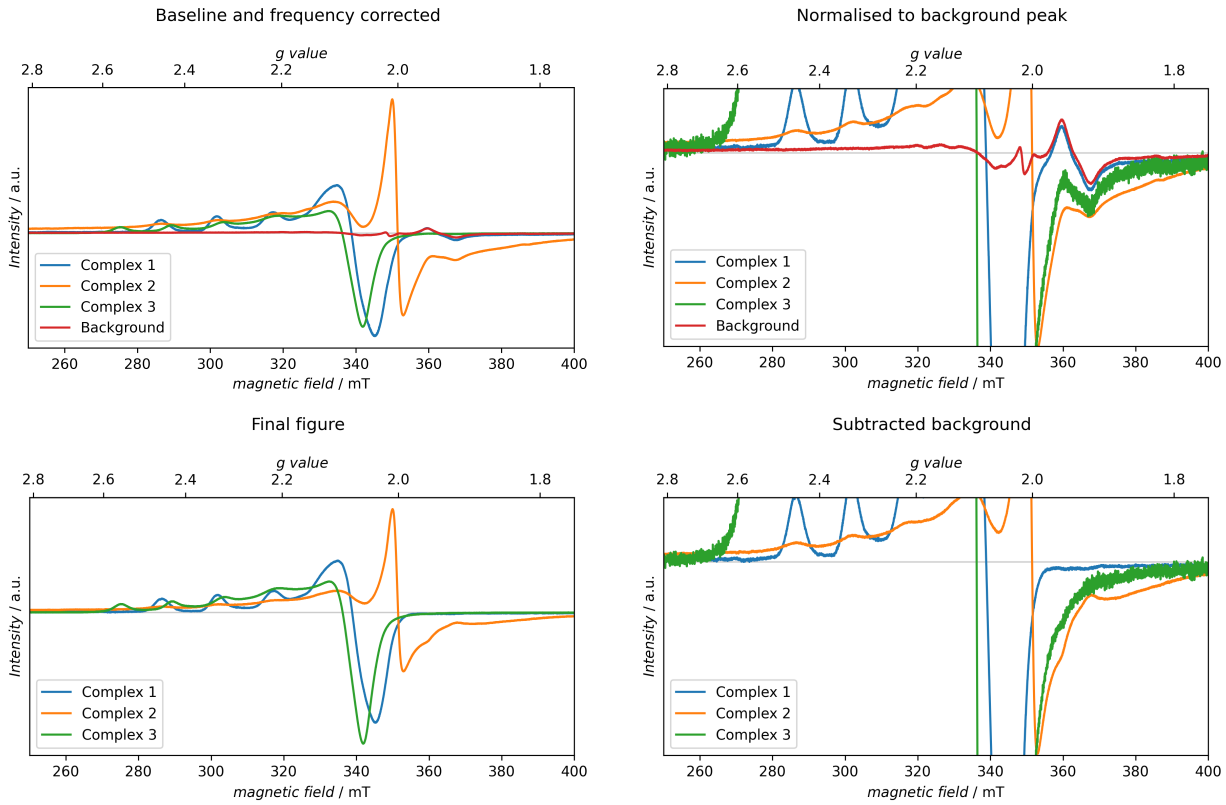

Figure 5: Steps to subtract a recorded background spectrum (red) from data. After a baseline and frequency correction (top left), all spectra are normalised to the same significant peak of the background (top right). Then, the background can be subtracted, resulting in the spectra shown in the bottom right panel. The final data are presented in the lower left panel.

report contains details of all tasks performed on the dataset, including comments and figure captions as well as all metadata contained in the dataset. Furthermore, a list of all packages and their versions for full reproducibility is included. The PDF version of the generated report as well as the full recipe are provided in the Supporting Information.

4.4. Represent angular-dependent measurements

The representation of an angular-dependent measurement (goniometer sweep) in the form shown in Fig 6 is extremely easy. Therefore, the whole recipe is presented in Listing 9. This representation may serve for both, a quick overview over the data after measuring and as a representation in a publication so that others can quickly verify the integrity of the data as well.

The two panels top left and right represent the complete 2D pattern, though in different ways. Both types of plots emphasize distinct aspects of the spectra: The form of each single slice is much more evident in the right panel whereas the change in the amplitude and the symmetry of the pattern come out better in the top left panel.

To quickly check the quality of the measurement the bottom left panel shows a direct comparison of the two slices extracted for angles of 0° and 180°,

Listing 8: Creating a well-formatted report (as a PDF document) containing all steps that have been performed on a dataset, as well as its metadata. For the actual report, see the SI.

```
- kind: report
  type: LaTeXReporter
  properties:
    template: dataset.tex
    filename:
      - report_compound1.tex
  compile: true
  apply_to:
    - compound1
```

Listing 9: Whole recipe to plot the overview representation for an angular-dependent measurement.

```
format:
  type: ASpecD recipe
  version: '0.2'
settings:
  default_package: cwepr
datasets:
  - RotationPattern-01
tasks:
- kind: processing
  type: BaselineCorrection
- kind: singleplot
  type: GoniometerSweepPlotter
  properties:
    properties:
      figure:
        dpi: 300
      axes:
        xlim: [349, 353]
```

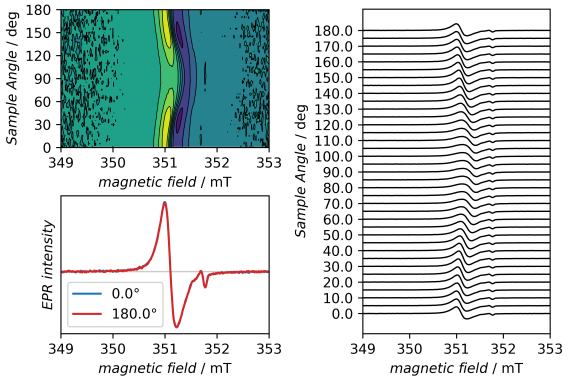

Figure 6: Different representations of a goniometer sweep: The panels top left and right represent the complete 2D surface each offering different insights into the spectra. The panel bottom left overlays the 0° and 180° spectra as a sanity check for angular mismatch, as those spectra should be the same.

respectively. These spectra should look the same due to the same relative orientation to the external magnetic field, as the interactions do not depend on its polarity.

4.5. Comparison of data recorded at X and Q band

EPR data usually consist of more or less well-resolved lines originating in part from both, g anisotropy and hyperfine interactions. Recording spectra for the same sample at different microwave frequency bands is probably the most straightforward way to discrimiate between hyperfine interaction and g anisotropy. However, meaningfully

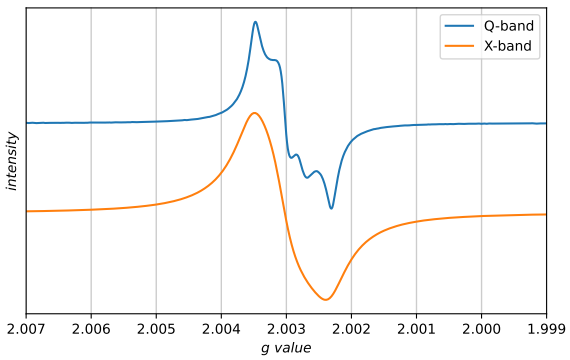

Figure 7: Comparing cw-EPR spectra of the same sample recorded at X and Q band. The splitting seen at Q band most probably originates from better resolved g anisotropy. In those cases, only conversion of the magnetic field axis to a g axis allows for directly comparing the spectra obtained at different fields and frequencies.

comparing the spectra recorded at these different magnetic fields and microwave frequencies usually requires to convert the magnetic field axis to a g axis.

Furthermore, let us assume that both spectrometers used, here operating at X and Q-band frequencies and fields, are not calibrated with respect to their external magnetic field, hence in both cases, a field standard with known g value has been recorded in addition to the sample of interest.

For both pairs of spectra, sample of interest and field standard, one needs to correct for a common frequency and apply a field calibration. Afterwards, the magnetic field axis can be converted to a g axis and the results plotted together in one axis. The crucial steps of the recipe are presented in Listing 10. For a complete recipe, see the Supporting Information. The results of the plot are shown in Fig. 7. To highlight a few aspects: Field correction is a two-step process consisting of an analysis step (`FieldCalibration`) and a processing step (`FieldCorrection`). The analysis step obtains a field offset value by comparing the measured magnetic field position of the spectral line with the value theoretically expected from its known g value.

4.6. Outlook: frontend for recipes

Although the YAML file format is particularly simple to write by hand, the mere number of options that can be set for certain tasks (in particular plotting tasks) can be daunting. Furthermore, a tool helping with automatically creating a recipe

Listing 10: Key steps of data processing for comparing cw-EPR spectra of the same sample recorded at X and Q band. Note that field correction is a two-step process and that the result of the FieldCalibration step (`field-offset-X`) is fed into the FieldCorrection step.

```
- kind: processing
  type: FrequencyCorrection
  properties:
    parameters:
      frequency: 9.5
  apply_to:
  - Sample-X
  - LiLiF-X
- kind: singleanalysis
  type: FieldCalibration
  properties:
    parameters:
      standard: LiLiF
  apply_to:
  - LiLiF-X
  result: field-offset-X
- kind: processing
  type: FieldCorrection
  properties:
    parameters:
      offset: field-offset-X
  apply_to:
  - Sample-X
### Repeat for sample recorded at Q band
- kind: processing
  type: GAxisCreation
  apply_to:
  - Sample-X
  - Sample-Q
- kind: multiplot
  type: MultiPlotter1DStacked
  properties:
    filename: x-q-comparison.pdf
  apply_to:
  - Sample-Q
  - Sample-X
```

from its building blocks dramatically reduces the chance of introducing errors. To this end, we are currently developing a web frontend running locally and built using the Python Flask web framework. A few details and a preview of a working prototype are given in the Supporting Information.

5. Conclusions

In summary, we have presented an open-source Python software package for the fully reproducible processing and analysis of cw-EPR data designed with a focus on usability. By using the cwepr package, scientists can focus on the analysis itself, liberated from caring both, about the intricate details of the implementation of each individual processing and analysis step as well as reproducibility of the results. Furthermore, no programming skills are required for using the package, while thanks to the modular nature and extensive documentation those with programming skills will find it easy to further extend the package. Due to its open-source nature, the cwepr Python package welcomes contributions from the community. Further details can be found in the documentation available online. Taken together, we envision the use of tools such as the cwepr Python package to change both, the way data will be analysed in spectroscopy and the approach taken towards fully reproducible research.

Supporting Information

Comments on reproducibility of scientific results; details of the info file format for storing metadata; overview of the cwepr package; details of how to extend the cwepr package; full recipes from the example section; PDF output of a report generated on a dataset; details of the graphical frontend for recipes currently in development.

Author information

The authors declare no competing financial interest.

Acknowledgements

TB thanks T. Berthold for shaping his ideas on how and why to record metadata during data acquisition, and B. Paulus for help with both crafting the info file format and implementing its actual use in the lab. J. Popp had crucial impact on the development of the ASpecD framework underlying the cwepr package. P. Kirchner contributed to an initial implementation of the cwepr Python package.

Software availability

The cwepr Python package is provided open-source and free of charge under a BSD license and can be referenced using the following DOI: 10.5281/zenodo.4896687. Further details can be found on its website (https://docs.cwepr.de/) together with a detailed documentation for both, users and developers. The cwepr Python package is available via the Python Package Index

(PyPI) (https://pypi.org/project/cwepr/), facilitating installation, and the source code is provided on GitHub (https://github.com/tillbiskup/cwepr). The package welcomes contributions from the community. Further details can be found in the documentation available online.

References

[1] I. Newton, "If I've seen further, it's by standing on the shoulders of giants", 1676. Letter to Robert Hooke, 5th February 1676.

[2] R. K. Merton, On the Shoulders of Giants. A Shandean Postscript. The Post-Italianate Edition, The University of Chicago Press, Chicago, 1993.

[3] K. Popper, Logik der Forschung, 11th ed., Mohr Siebeck, Tübingen, 2005.

[4] V. Masson-Delmotte, P. Zhai, A. Pirani, S. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M. Gomis, M. Huang, K. Leitzell, E. Lonnoy, J. Matthews, T. Maycock, T. Waterfield, O. Yelekçi, R. Yu, B. Zhou (Eds.), IPCC, 2021: Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change, Cambridge University Press, 2021. In press.

[5] G. Wilson, What should computer scientists teach to physical scientists and engineers?, IEEE Comput. Sci. Eng. 3 (1996) 46–55.

[6] G. Wilson, Software carpentry. getting scientists to write better code by making them more productive, Comput. Sci. Eng. 8 (2006) 66–69.

[7] C. Goble, Better software, better research, IEEE Internet Comput. 18 (2014) 4–8.

[8] Z. Merali, ...why scientific programming does not compute, Nature 467 (2010) 775–777.

[9] S. M. Baxter, S. W. Day, J. S. Fetrow, S. J. Reisinger, Scientific software development is not an oxymoron, PLoS Comput. Biol. 2 (2006) e87.

[10] R. D. Peng, Reproducible research in computational science, Science 334 (2011) 1226–1227.

[11] G. K. Sandve, A. Nekrutenko, J. Taylor, E. Hovig, Ten simple rules for reproducible computational research, PLoS Comput. Biol. 9 (2013) e1003285.

[12] R. J. LeVeque, I. M. Mitchell, V. Stodden, Reproducible research for scientific computing: Tools and strategies for changing the culture, Comput. Sci. Eng. 11 (2012) 13–17.

[13] J. M. Perkel, A toolkit for data transparency, Nature 560 (2018) 513–515. doi:10.1038/d41586-018-05990-5.

[14] Yale Law School Roundtable on Data and Code Sharing, Reproducible research, Comput. Sci. Eng. 12 (2010) 8–13.

[15] M. Baker, Is there a reproducibility crisis?, Nature 533 (2016) 452–454. doi:10.1038/533452a.

[16] D. L. Donoho, A. Maleki, M. Shahram, I. U. Rahman, V. Stodden, Reproducible research in computational harmonic analysis, Comput. Sci. Eng. 11 (2009) 8–18.

[17] B. Lawlor, P. Walsh, Engineering bioinformatics. building reliability, performance and productivity into bioinformatics software, Bioengineered 6 (2015) 193–203.

[18] R. D. Peng, F. Dominici, S. L. Zeger, Reproducible epidemiologic research, Am. J. Epidemiol. 163 (2006) 783–789.

[19] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, B. Mons, The FAIR Guiding Principles for scientific data management and stewardship, Sci. Data 3 (2016) 160018. doi:10.1038/sdata.2016.18.

[20] C. Steinbeck, O. Koepler, F. Bach, S. Herres-Pawlis, N. Jung, J. C. Liermann, S. Neumann, M. Razum, C. Baldauf, F. Biedermann, T. W. Bocklitz, F. Boehm, F. Broda, P. Czodrowski, T. Engel, M. G. Hicks, S. M. Kast, C. Kettner, W. Koch, G. Lanza, A. Link, R. A. Mata, W. E. Nagel, A. Porzel, N. Schlörer, T. Schulze, H.-G. Weinig, W. Wenzel, L. A. Wessjohann, S. Wulle, NFDI4Chem - towards a national research data infrastructure for chemistry in Germany, Research Ideas and Outcomes 6 (2020) e55852. doi:10.3897/rio.6.e55852.

[21] Deutsche Forschungsgemeinschaft, Guidelines for Safeguarding Good Research Practice. Code of Conduct, 2019. doi:10.5281/zenodo.3923602.

[22] A. N. Whitehead, An Introduction to Mathematics, Dover Publications, Mineola, 2017 (1911). 'Civilization advances by extending the number of important operations which we can perform without thinking about them.' (p. 34).

[23] E. W. Dijkstra, EWD447: On the role of scientific thought, Springer-Verlag, New York, 1982, pp. 60–66.

[24] A. Schweiger, G. Jeschke, Principles of pulse electron paramagnetic resonance, Oxford University Press, Oxford, 1991.

[25] S. A. Bonke, T. Risse, A. Schnegg, A. Brückner, In situ electron paramagnetic resonance spectroscopy for catalysis, Nat. Rev. Methods Primers 1 (2021) 33. doi:10.1038/s43586-021-00031-4.

[26] M. Arvind, C. E. Tait, M. Guerrini, J. Krumland, A. M. Valencia, C. Cocchi, A. E. Mansour, N. Koch, S. Barlow, S. R. Marder, J. Behrends, D. Neher, Quantitative analysis of doping-induced polarons and charge-transfer complexes of poly(3-hexylthiophene) in solution, J. Phys. Chem. B 124 (2020) 7694–7708. doi:10.1021/acs.jpcb.0c03517.

[27] Y. Shin, M. Massetti, H. Komber, T. Biskup, D. Nava, G. Lanzani, M. Caironi, M. Sommer, Improving miscibility of a naphthalene diimide-bithiophene copolymer with n-type dopants through the incorporation of "kinked" monomers, Adv. Electron. Mater. 4 (2018) 1700581. doi:10.1002/aelm.201700581.

[28] D. Kiefer, A. Giovannitti, H. Sun, T. Biskup, A. Hofmann, M. Koopmans, C. Cendra, S. Weber, J. A. Koster, E. Olsson, J. Rivnay, S. Fabiano, I. McCulloch, C. Müller, Enhanced n-doping efficiency of a naphthalenediimide-based copolymer through polar side chains for organic thermoelectrics, ACS Energy

Lett. 3 (2018) 278–285. doi:10.1021/acsenergylett.7b01146.

[29] S. B. Schmidt, T. Biskup, X. Jiao, C. R. McNeill, M. Sommer, Controlling intermolecular redox-doping of naphthalene diimides, J. Mater. Chem. C 7 (2019) 4466–4474. doi:10.1039/c9tc00721k.

[30] T. Biskup, Doping of organic semiconductors: Insights from EPR spectroscopy, Appl. Phys. Lett. 119 (2021) 010503. doi:10.1063/5.0054685.

[31] S. Stoll, A. Schweiger, EasySpin, a comprehensive software package for spectral simulation and analysis in EPR, J. Magn. Reson. 178 (2006) 42–55.

[32] A. Carrington, A. D. McLachlan, Introduction to Magnetic Resonance. With Applications To Chemistry and Chemical Physics, Harper & Row, New York, 1967.

[33] N. M. Atherton, Principles of Electron Spin Resonance, Ellis Horwood Ltd., Chichester, 1993.

[34] J. A. Weil, J. R. Bolton, Electron Paramagnetic Resonance: Elementary Theory and Practical Applications, second edition ed., John Wiley & Sons, Inc., Hoboken, 2007.

[35] M. Brustolon, E. Giamello, Electron Paramagnetic Resonance: A Practitioner's Toolkit, Wiley, Hoboken, 2009.

[36] V. Chechik, E. Carter, D. Murphy, Electron Paramagnetic Resonance, Oxford University Press, Oxford, UK, 2016.

[37] D. Goldfarb, S. Stoll (Eds.), EPR Spectroscopy: Fundamentals and Methods, John Wiley & Sons, Chichester, UK, 2018.

[38] A. Abragam, Principles of Nuclear Magnetism, Oxford University Press, Oxford, UK, 1961.

[39] C. P. Poole, H. A. Farach, Theory of Magnetic Resonance, John Wiley & Sons, New York, 1987.

[40] C. P. Slichter, Principles of Magnetic Resonance, Harper & Row, New York, 1963.

[41] G. E. Eaton, S. S. Eaton, D. P. Barr, R. T. Weber, Quantitative EPR, Springer, Wien, 2010.

[42] T. Biskup, SpinPy Python package, 2021. URL: https://docs.spinpy.de/.

[43] T. Biskup, FitPy Python package, 2021. URL: https://docs.fitpy.de/.

[44] T. Biskup, ASpecD framework, 2021. URL: https://docs.aspecd.de/. doi:10.5281/zenodo.4717937.

[45] M. Schröder, T. Biskup, cwepr Python package, 2021. URL: https://docs.cwepr.de/. doi:10.5281/zenodo.4896687.

[46] J. Popp, T. Biskup, ASpecD: A modular framework for the analysis of spectroscopic data focussing on reproducibility and good scientific practice, ChemRxiv (2021). doi:10.26434/chemrxiv-2021-6jt1l.

[47] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns. Elements of Reusable Object-Oriented Software, Addison-Wesley, Boston, 1995.

[48] J. D. Hunter, Matplotlib: a 2D graphics environment, Comput. Sci. Eng. 9 (2007) 90–95. doi:10.1109/MCSE.2007.55.

[49] J. Hunter, D. Dale, E. Firing, M. Droettboom, Matplotlib development team, Matplotlib documentation, 2021. URL: https://matplotlib.org/.

[50] P. Greenfield, M. Droettboom, E. Bray, ASDF: a new data format for astronomy, Astron. Comput. 12 (2015) 240–251. doi:10.1016/j.ascom.2015.06.004.

[51] A. Savitzky, M. J. E. Golay, Smoothing and differentiation of data by simplified least squares procedures, Anal. Chem. 36 (1964) 1627–1639.