Fitting 405nm Nuclear Polarization Probe Spectra

This collection of scripts is used to extract and analyze data gathered by TRINAT's nuclear acquisition to measure the nuclear polarization of trapped potassium atoms.

Files

- `_constants.py`: Stores any relevant physical constants (e.g. μ_B) in a single place.
- `_load.py`: Contains three functions: `load_data()`, `compute_observables()` and `generate_histograms()`:
  - `load_data()` extracts relevant data from the given `.root` file. It also converts units where needed (`TDC_*` $\rightarrow ns$, `TTTL_*` $\rightarrow \mu s$). The function returns a dictionary of NumPy arrays.
  - `compute_observables()` takes the dictionary generated by `load_data()` and computes relevant observables (e.g. `Y = TDC_ION_MCP_LE[0] - TDC_PHOTO_DIODE_LE[0]`, ...)
  - `generate_histograms()` plots the data extracted using `load_data()` after running `compute_observables()`. It plots the data before and after applying cuts.
- `_models.py`: Stores models used to fit the data (currently only $F = 2 \rightarrow F'$ is implemented).
- `_physics.py`: Stores functions to compute physical quantities (e.g. nuclear polarization).
- `polarization.py`: Performs the analysis using functions defined in the files above. Loads data, makes cuts and fits the model to the generated spectra. Generates a printout of the fit parameters and saves them to a `.txt` file.

How to use this script

1. Make a new folder and copy the `.py` files from one of the examples.
2. Move the relevant `.root` files into the folder (one for FLIP and one for NORM polarization).
3. Set `OUTPUT_PATH` and `path_flip` and `path_norm` to point to the correct `.root` files.
4. Set the measurement parameters `V_MIN`, `V_MAX`, `STEPS` and `LOCKPOINT`. These should be recorded for every measurement.
5. Run the script. The script must be run from the Polarization directory (running `pwd` should print `/{some path}/Polarization`). On Linux the script can be run by simply typing `python /{some path}/Polarization/{folder name}/polarization.py` (this assumes that the python installation is properly set up on the system).

    - When using conda ensure that the correct environment is used by running `conda activate {environment name}` before executing the python script.
6. Check the histograms generated in `.root` files called `histograms_*.root` in the output folder to make sure that cuts are done properly. Adjust `CUTS` if necessary and run again.
7. Once the histograms look good adjust fit parameters until fit looks good.

Prerequisites

Python Version: 3.10.5 or higher (python is excellent with respect to backwards compatibility excluding the jump from python 2 to python 3)

On Linux systems I recommend using a conda environment. The advantages are that you can avoid messing with the python version installed natively on most systems which might affect how the system works. It also makes it easy to install ROOT with its python interface (which is needed for this code to run).

I recommend miniconda (https://docs.conda.io/en/latest/miniconda.html). Prior to installing any conda environments I recommend installing mamba (https://anaconda.org/conda-forge/mamba). Mamba speeds up how fast conda can install environments (on my machine installing root without mamba took about 15 minutes, with mamba less than a minute).

ROOT

- A ROOT install that provides access to ROOT's python interface (I recommend installing it using conda)
    - I have tested this with ROOT6. ROOT5 should also work since I am using very basic functionality, but I have not tested this.

Python Modules

- Matplotlib: Required for plotting data.
- Numpy: Used for making cuts on data and computations.
- iMinuit: Used for fitting.
- uncertainties: Used to propagate statistical uncertainties through nuclear polarization calculations.
- scipy: Obtain physical constants from the NIST CODATA database. Implementation of Voigt profile.
- tabulate: For making nice tables.

These can be installed using pip:

```
pip install matplotlib numpy iminuit uncertainties scipy tabulate
```

Input and Output

1. **fit.pdf/.png**: Image of the two spectra along with fits and residuals. In the residuals plot, $\pm$ 1-σ of the residuals is indicated by a shaded bar.
2. **histograms_*.root**: Saves the data relevant to the analysis in `.root` files. Included is data before applying cuts and after.
3. **parameters.txt**: Copy of the script output. Includes fit parameters and statistics, nuclear polarization and the cuts applied to the data.

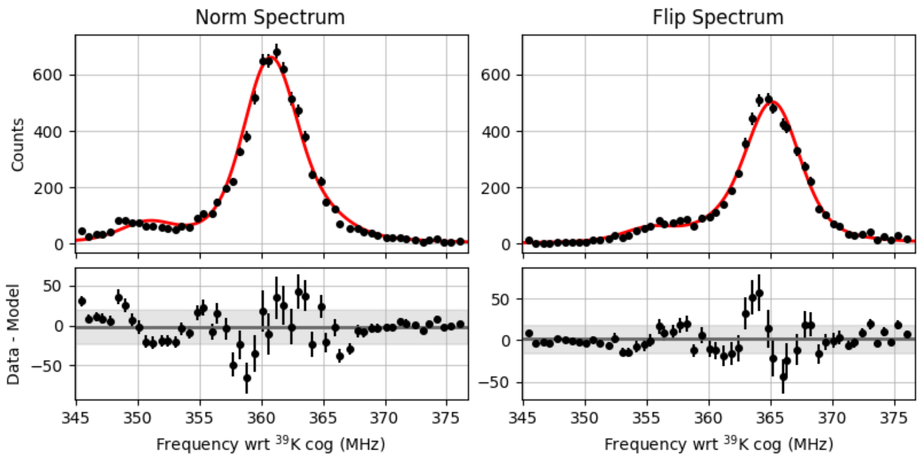

Figure 1: Example fit

Example output plot

- Top row shows data (black dots with errorbars) and fit (red line)
- Bottom row shows residuals (black dots), mean of residuals (dark grey line) and $\pm$ 1-σ region centered around the mean (grey region)

Fitting Procedure

The main problem with the polarization data is that the sublevels are identified using their frequency shifts. This is hard since the sublevel shifts depend on three parameters: x_0 which is a "frequency" shift related to the isotope shift, B the magnetic field (Zeeman effect) and the laser power (Stark shift) (in its current form the script does not account for a Stark shift and I am only mentioning it here for completeness).

Accounting for the "frequency" shift and the Zeeman shift at the same time leaves us with two unrelated parameters that both affect the frequency locations of the sublevel transitions. Essentially we have an expression with two unknowns leaving us with infinitely many solutions. One approach to circumvent this issue would be to fix one of the parameters, but the frequency shifts are usually not known to high enough precision and the magnetic field strength inside the trap is hard to measure in situ. The second approach is to fit two spectra at opposite pumping polarizations (norm and flip) simultaneously. This takes advantage of the symmetry of the Zeeman effect when probing it with π-polarized light. The magnetic field B controls the "width" of each spectrum, i.e. the splitting between the $m_F = \pm 2$ and $m_F = \pm 1$. By using spectra taken at roughly opposite polarization we essentially provide data for both $m_F = +2$ and $m_F = -2$ and hence can determine this total width more precisely.

To fit this global model the sum of two log-likelihoods is minimized. A single dataset (x_i, y_i) can be fitted to some model $f(x; \vec{\alpha})$ where $\vec{\alpha}$ is a vector of parameters by minimizing:

$$-\mathcal{L} = -\log L = -\sum_i \left[y_i \log\left(f(x_i; \vec{p}) \right) - f(x_i, \vec{p}) \right]$$

To perform a simultaneous fit we add the likelihood functions computed using the norm and flip data:

$$-\mathcal{L}^{\text{Global}} = -\mathcal{L}^{\text{Flip}} - \mathcal{L}^{\text{Norm}}$$

$$\begin{aligned} -\mathcal{L}^{\text{Global}} = &- \sum_i^{N^{\text{Flip}}} \left[y_i^{\text{Flip}} \log\left(f(x_i^{\text{Flip}}; \vec{\alpha}, \vec{b}^{\text{Flip}}) \right) - f(x_i^{\text{Flip}}; \vec{\alpha}, \vec{b}^{\text{Flip}}) \right] \\ &- \sum_i^{N^{\text{Norm}}} \left[y_i^{\text{Norm}} \log\left(f(x_i^{\text{Norm}}; \vec{\alpha}, \vec{b}^{\text{Norm}}) \right) - f(x_i^{\text{Norm}}; \vec{\alpha}, \vec{b}^{\text{Norm}}) \right] \end{aligned}$$

Here we have 3 parameter vectors $\vec{\alpha}$ and $\vec{b}$ where:

$$\vec{\alpha} = \begin{pmatrix} x_0 & h & B & s & g \end{pmatrix},$$

contains the shared parameters and

$$\vec{b} = \begin{pmatrix} a_{-2} & a_{-1} & a_0 & a_{+1} & a_{+2} \end{pmatrix},$$

contains the sublevel populations for norm polarization $\vec{b}^{\text{Norm}}$ and flip polarization $\vec{b}^{\text{Flip}}$. The model fitted to each spectrum is given by:

$$\begin{aligned} f(x; \vec{\alpha}, \vec{b}) = &\sum_{m_F=-1}^{1} s_{1,m_F} a_{m_F} V\left(f - x_0 + h - \frac{2}{3} m_F \bar{\mu}_B B, g, s \right) \\ &+ \sum_{m_F=-2}^{2} s_{2,m_F} a_{m_F} V\left(x - x_0 - \frac{1}{3} m_F \bar{\mu}_B B, g, s \right) \end{aligned}$$

Here s_{F,m_F} are the transition strengths (a weighting factor), $V(x, g, s)$ is the Voigt profile (a convolution of a Lorentzian and Gaussian distribution) where g and s are the widths of the Lorentzian and Gaussian part respectively and $\bar{\mu}_B = \mu_B / h$ where μ_B is the Bohr magneton.

Comments

χ^2

Note that in its current form the calculated χ^2 can be infinite. In this case the script will throw an error similar to:

```
/mnt/Secondary/Repositories/Polarization/Example 2/polarization.py:132: RuntimeWarning: divi
  chi2_flip[y_flip != 0] += y_flip * np.log(y_flip / sublevel_model(frequencies, *p_flip))
/mnt/Secondary/Repositories/Polarization/Example 2/polarization.py:132: RuntimeWarning: inva
  chi2_flip[y_flip != 0] += y_flip * np.log(y_flip / sublevel_model(frequencies, *p_flip))
Traceback (most recent call last):
  File "/mnt/Secondary/Repositories/Polarization/Example 2/polarization.py", line 132, in <m
    chi2_flip[y_flip != 0] += y_flip * np.log(y_flip / sublevel_model(frequencies, *p_flip))
ValueError: operands could not be broadcast together with shapes (45,) (53,) (45,)
```

In this case just comment out the section of code that computes the variables `chi2_flip`, `chi2_norm` and manually set `chi2` equal to zero.

Features to add

- Fitting $F \to 1'$. Currently only $F \to 2'$ is modelled. To implement this functionality consider the following steps:
  1. Implement a function `sublevel_model_F1` in `_models.py` (can be done similarily to how `sublevel_model_F2` is implemented).
  2. Implement a function `nuclear_polarization_41K_F1` in `_physics.py`. Use `nuclear_polarization_41K_F2` as a guide. Using `ufloat()` allows for easily propagating statistical uncertainties.
  3. In `polarization.py` adjust `global_poisson_likelihood` to call `sublevel_model_F1` instead of `sublevel_model_F2`. Adjust parameters names so that they make sense.
  4. Adjust the initial guess in the call to `Minuit()`. This also involves changing the parameter names. `Minuit` knows the parameter names of the cost function and so when setting the initial guess they need to be correct.
  5. Adjust parameter limits and fixed parameters if necessary.
  6. Adjust what parameters are assigned to `p_norm` and `p_flip`. In reality this only requires making sure that the correct parameter names are given and that they are in the same order as required by the `sublevel_model_F1` function.
  7. Call `nuclear_polarization_41K_F1` instead of `nuclear_polarization_41K_F2` and make sure that the arguents are correctly provided.
- Instead of χ^2 it might be better to just provide the value of the likelihood function at the minimum.