

PyMossFit: A Google Colab option for Mössbauer Spectra Fitting

Fabio D. Saccone^{1,2,3}

¹Departamento de Física, Facultad de Ingeniería, Universidad de Buenos Aires, Av. Paseo
Colón 850, Buenos Aires. ARGENTINA

²YPF Tecnología S. A., Av. del Petróleo S/N, Berisso, ARGENTINA

³CONICET, Godoy Cruz 2290, Buenos Aires. ARGENTINA

June 20, 2025

Abstract

This article introduces the main characteristics of PyMossFit, a software for Mössbauer spectra fit. It is explained how each utility of their code sections works.

Based on the `Lmfit` python package, it is a robust data fitting tool. Designed to run as a `Jupyter` notebook at the `Google Colab` cloud, it also allows us to work from multiple devices and operating systems. Additionally, it facilitates that fitting procedure can be performed in a collaborative way by researchers.

The software performs the folding of raw data with a discrete Fourier transform. Data smoothing is available with the use of a Savitzky-Golay algorithm. Likewise, a K-nearest neighbor algorithm helps us to determine the present phases by matching the correlations of hyperfine parameters from a local database.

1 Introduction

Mössbauer Spectroscopy [Wikipedia contributors, 2025] is a highly specialized technique that investigates the resonant absorption of γ rays by atomic nuclei. The Mössbauer effect, observed primarily in isotopes such as iron-57 (^{57}Fe), provides insight into hyperfine interactions, including isomer shifts, quadrupole splittings, and magnetic hyperfine fields. These parameters offer detailed information about the electronic, magnetic, and structural environment of the sample, making the technique invaluable in materials science, chemistry, and condensed matter physics. The gamma ray of resonant absorption in ^{57}Fe nuclei is **14.4 keV**.

In a typical Mössbauer experiment, a radioactive source emits gamma rays, which are absorbed by the nuclei in the sample (see Figure1). Resonant absorption occurs, in a small fraction, when γ rays hit the probe with recoilless.

The detector measures the intensity of the transmitted radiation as a function of the velocity of the gamma-ray source. This results in a Mössbauer spectrum, typically characterized by sharp peaks or dips at resonance frequencies corresponding to different hyperfine interactions within the sample. The most common experimental setup corresponds to **Transmission Geometry** (in this case, the observed lines come from a reduction of 14.4 keV gamma counting in detector, as compared to the background signal). The other option is the **Conversion Electron Mössbauer Spectroscopy (CEMS)** that corresponds to detection of back scattered electrons after gamma absorption.

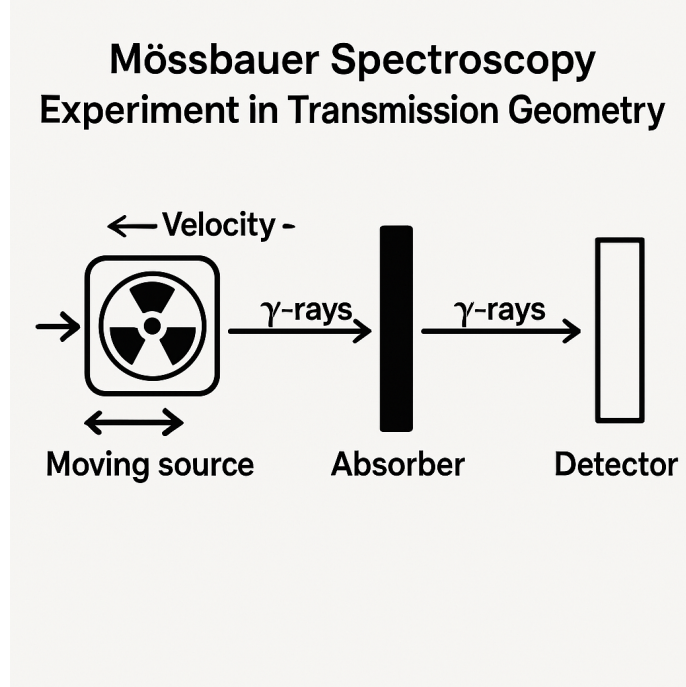


Figure 1: Scheme of a typical Mossbauer experiment in transmission geometry

1.1 Statement of need

Recently, in a review by Grandjean et al. [Grandjean and Long, 2021], the authors made a series of suggestions about how good measurements should be taken, which could be a good practice for Mössbauer data treatment and its corresponding fitting presentation. Usually, the scientists working with Mössbauer Spectroscopy manage their data and fit them with not open source softwares that run on Windows OS with poor upgrade services. The needs of an open source option with less OS or package dependency has motivated this work.

In this sense, Google Colab is a useful tool for a collaborative team job in data analysis with the advantage of no additional packages locally installed,

also compatible with the fact that the user can run their codes from multiple devices.

2 Mössbauer Spectra and Curve Shapes

The shape of a Mössbauer spectrum varies depending on the nature of the hyperfine interactions in the sample. For example, a simple paramagnetic material might produce a single absorption peak (Lorentzian or Gaussian) due to the isomer shift. Materials with quadrupole splitting generate a doublet (two peaks), while materials experiencing magnetic hyperfine interactions often exhibit more complex sextet patterns. These spectral features often overlap, making it difficult to isolate and quantify each individual component.

2.1 Curve Fitting and Least Squares Method

Curve fitting plays a crucial role in Mössbauer spectral analysis, as it allows for the decomposition of these complex spectra into individual contributions, each associated with specific hyperfine parameters. The challenge lies in accurately reproducing the spectral shapes using mathematical models and adjusting the parameters until a satisfactory fit is achieved.

To accurately extract physical parameters from Mössbauer spectra, fitting procedures are applied to the experimental data. One of the most common approaches is the **least-squares method**, which minimizes the difference between the experimental data points and the theoretical model curve. The objective is to adjust the parameters of the theoretical model (such as isomer shift, quadrupole splitting, and line broadening) so that the calculated spectrum fits the experimental data as closely as possible. Then, it is required a minimization of the χ^2 , defined as:

$$\chi^2 = \sum_{i=1}^N \frac{(y_i^{exp} - y_i^{model}(\vec{p}))^2}{\epsilon_i^2} \quad (1)$$

being \vec{p} a set of selected parameters for fitting, which minimizes the distance between the experimental data, y_i^{exp} and the modeled data points y_i^{model} .

In **Python**, this process can be implemented using libraries like **Lmfit**, **SciPy**, or **NumPy**, which provide robust tools for least-squares curve fit. The general approach involves defining a model function that represents the expected shape of the Mössbauer spectrum, which could be a sum of multiple Lorentzian or Gaussian functions, depending on the number and type of spectral components. Lorentzian functions are preferred with crystalline samples, while PseudoVoigts (sum of Lorentzian and Gaussian functions) are appropriated for Fe sites in disordered materials.

The least-squares fitting algorithm iteratively adjusts the model parameters until the sum of squared residuals between the experimental and calculated spectra is minimized. An example of Lorentzian function definition for **Python**, can be seen next,

```

from scipy.optimize import curve_fit
import numpy as np

# Define the model function (e.g., a sum of Lorentzian components)
def lorentzian(x, amplitude, center, width):
    return amplitude * (width**2 / ((x - center)**2 + width**2))

# Generate synthetic data for fitting
x_data = np.linspace(-10, 10, 500) # Example velocity range
y_data = lorentzian(x_data, 1, 0, 1) + np.random.normal(0, 0.02,
    size=len(x_data))

# Perform least-squares curve fitting
initial_guess = [1, 0, 1] # Initial guess for parameters
params, covariance = curve_fit(lorentzian, x_data, y_data,
    p0=initial_guess)

# params contains the optimized parameters: amplitude, center,
    and width

```

This example demonstrates fitting a single Lorentzian function to synthetic data, though more complex models involving sums of Lorentzians or other spectral shapes can be implemented to represent real Mössbauer spectra.

3 Description of the Python code for Google Colab (PyMossFit)

The Python code is available in several Jupyter notebooks as typical examples found in practice [Saccone, 2025]. In their names, some text is added that indicates the number and type of interactions (i. e. 1Q means one quadrupolar interaction, 2X for two magnetic sites and so on). Some selected parts of it are described throughout this page.

The code is structured in three cells. The first includes the installation in the Google Colab environment of `Lmfit`, the core of data fitting. Likewise, it imports some packages of `Scipy`, `Pandas`, `Matplotlib` and `Numpy`, among others. The next step includes a Drive connection that asks the user for permission.

The second cell reads the datafile (format should be inspected previously to define "delimiter", "columns" and "skiprows" parameters). The required inputs are date (in a YYYYMMDD format) and maximum velocity associated to the extreme channels, in order to define the velocity scale.

When raw data are plotted, they look as shown in the following because of the collection of absorption resonances converted to signals in a single-channel analyzer output, while the source moves back and forth.

In this cell, spectrum folding is performed with a Discrete Fourier Transforming routine, the `Numpy fft`. The theory of this procedure corresponds to the

Nyquist-Shannon Sampling Theorem that helps to determine a folding channel from the symmetry of discrete Fourier spectra [Kong et al., 2020].

Data can also be smoothed using a Savitsky-Golay package (I use `savgol` from `Scipy`). After folding, a new datafile is saved for fit with the use of the next cell.

The next figures illustrate the partial plots of data management with this procedure.

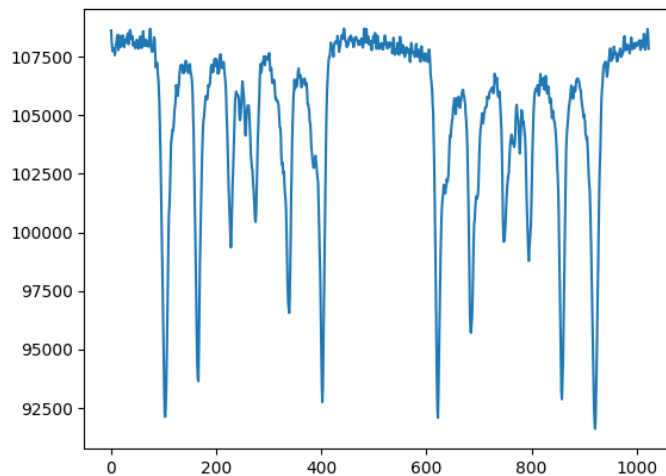


Figure 2: Unfolded plot of a raw Mossbauer experiment data

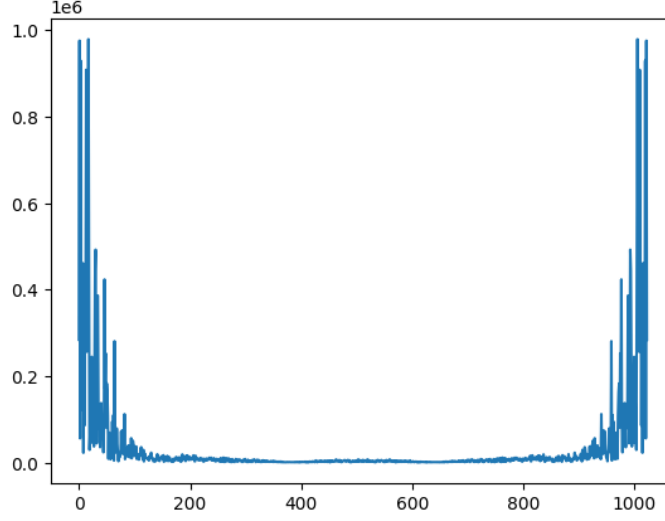


Figure 3: Discrete Fourier Transform of the raw Mossbauer experiment data plotted in the previous figure

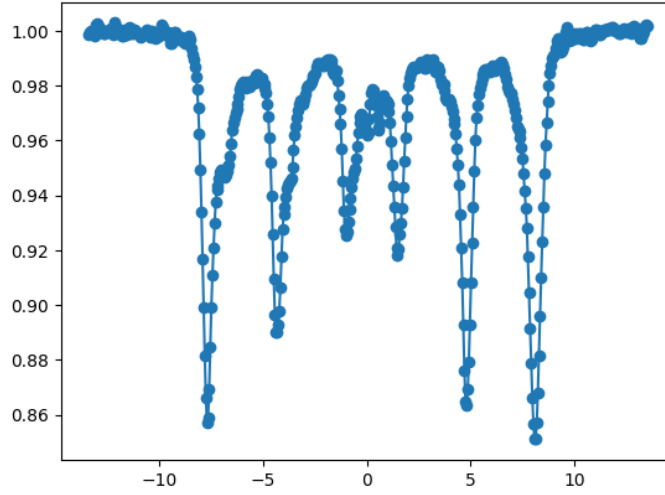
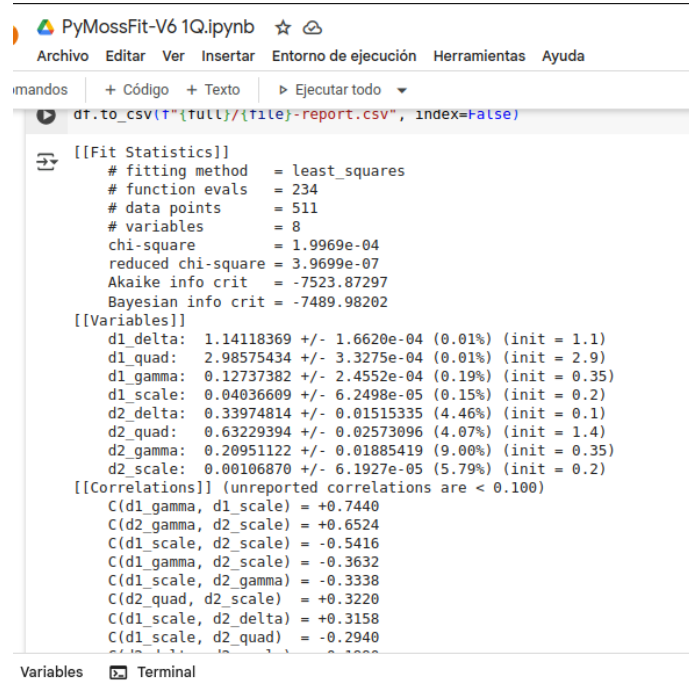


Figure 4: Final folded spectrum [Ferrari et al., 2015]

The set of physical parameters such as linewidth (Γ), isomer shifts (**IS**), quadrupole splitting (**QS**) and magnetic hyperfine field (B_{HF}) are calculated after adjusting the amplitud (**a**), full width at half maximum (**b**), centroid (**m**), line shift (**d**) and line separation (**q**). Initial set of these parameters can be fitted or fixed by selecting the "**True**" or "**False**" options, respectively.

A typical fit report of the output of this cell looks as following:



```
dt.to_csv("{}{}-report.csv".format(full_path, title), index=False)
```

```
[[Fit Statistics]]
# fitting method      = least_squares
# function evals      = 234
# data points         = 511
# variables           = 8
chi-square            = 1.9969e-04
reduced chi-square    = 3.9699e-07
Akaike info crit     = -7523.87297
Bayesian info crit   = -7489.98202

[[Variables]]
d1_delta: 1.14118369 +/- 1.6620e-04 (0.01%) (init = 1.1)
d1_quad:  2.98575434 +/- 3.3275e-04 (0.01%) (init = 2.9)
d1_gamma: 0.12737382 +/- 2.4552e-04 (0.19%) (init = 0.35)
d1_scale: 0.04036609 +/- 6.2498e-05 (0.15%) (init = 0.2)
d2_delta: 0.33974814 +/- 0.01515335 (4.46%) (init = 0.1)
d2_quad:  0.63229394 +/- 0.02573096 (4.07%) (init = 1.4)
d2_gamma: 0.20951122 +/- 0.01885419 (9.00%) (init = 0.35)
d2_scale: 0.00106870 +/- 6.1927e-05 (5.79%) (init = 0.2)

[[Correlations]] (unreported correlations are < 0.100)
C(d1_gamma, d1_scale) = +0.7440
C(d2_gamma, d2_scale) = +0.6524
C(d1_scale, d2_scale) = -0.5416
C(d1_gamma, d2_scale) = -0.3632
C(d1_scale, d2_gamma) = -0.3338
C(d2_quad, d2_scale) = +0.3220
C(d1_scale, d2_delta) = +0.3158
C(d1_scale, d2_quad) = -0.2940
```

Figure 5: An example of a fit report output of a Jupyter notebook in Google Colab

Finally, the code saves fitted parameters, experimental and modeled sub-spectra in CSV formatted output files.

3.1 Some examples

The following figures illustrate the fit results obtained from Mössbauer spectra of different samples. Each one exhibits several characteristics corresponding to the short range order neighborhood of ^{57}Fe probes.

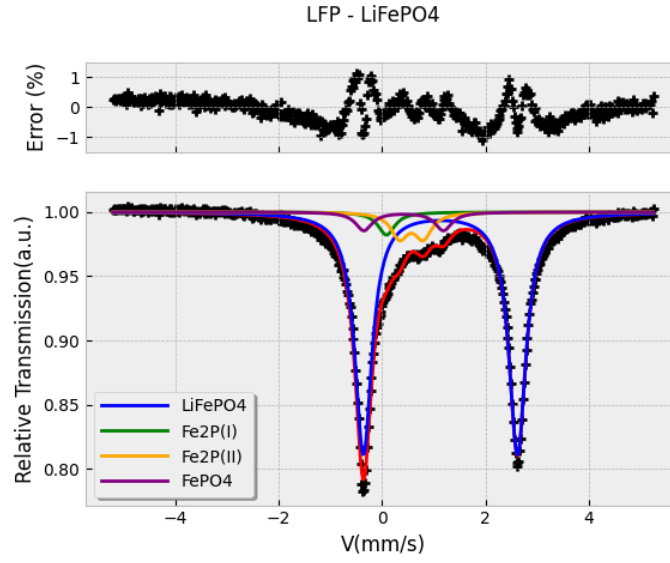


Figure 6: Mössbauer spectrum of an aged commercial LiFePO_4 active material for batteries. Residual phases were detected, such as FePO_4 and Fe_2P . This last phase, shows the presence of ^{57}Fe in two different oxidation states, Fe(I) and Fe(II) , respectively

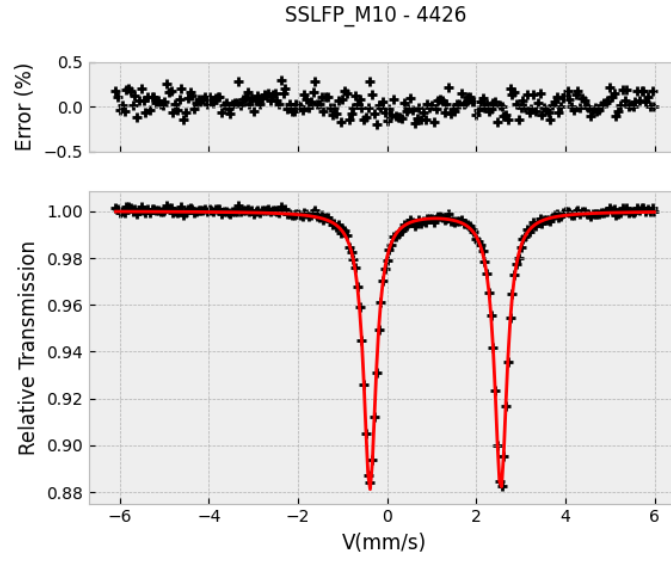


Figure 7: Mössbauer spectrum of the $LiFePO_4$ active material for batteries, after synthesis at lab scale. In this case, just $LiFePO_4$ was detected, besides the extraordinary sensitivity of the characterization technique

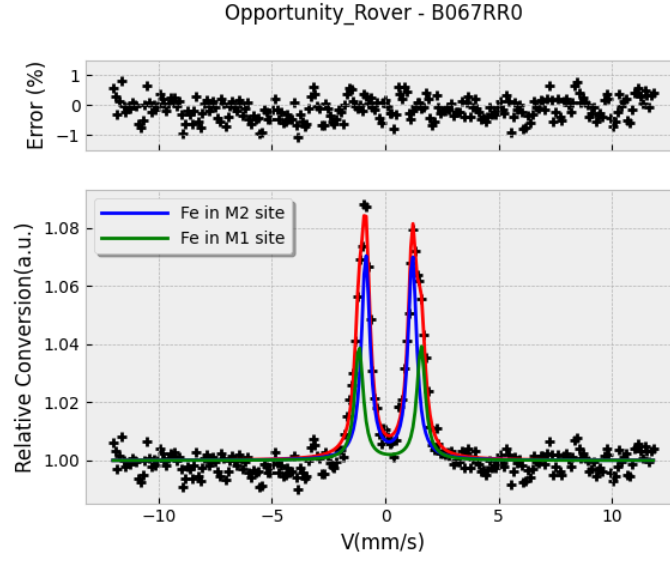


Figure 8: CEMS spectrum of pyroxene in Mars soil (Opportunity Mission, Sept. 2004, $T = 240\text{--}260\text{K}$) [Corke and Haviland, 2006]. The subspectra correspond to two different ^{57}Fe sites, M1 and M2, typically found in pyroxene [Oshtrakh et al., 2007].

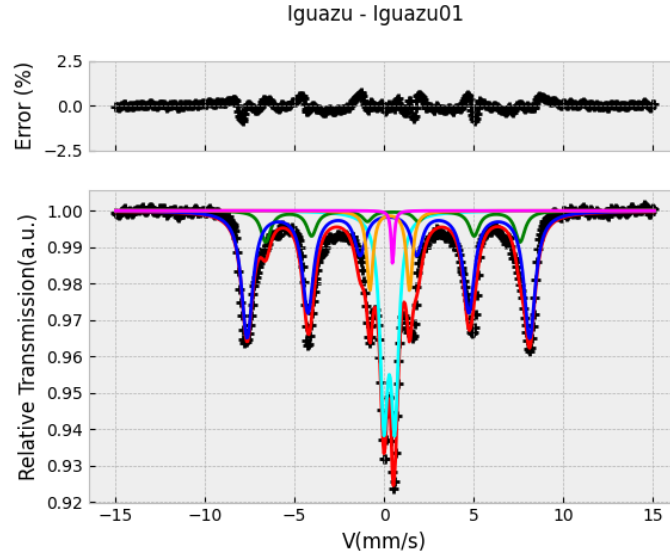
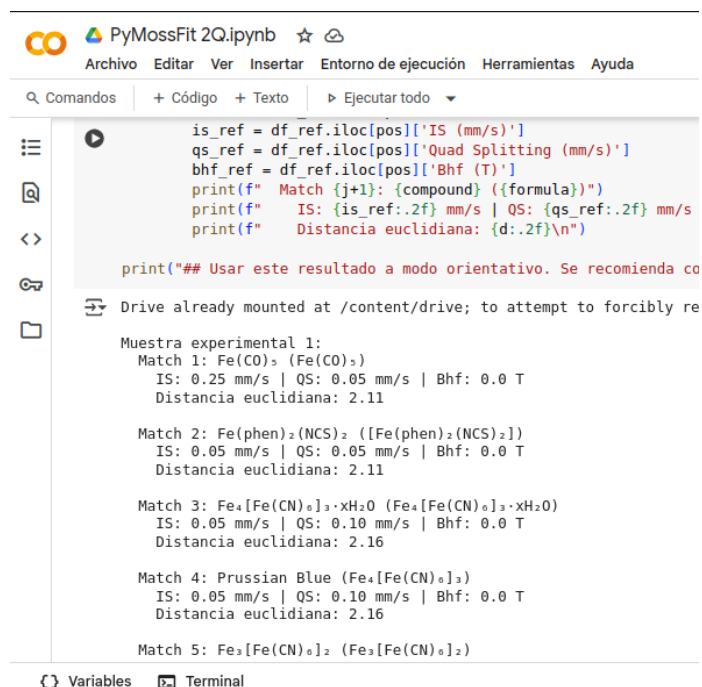


Figure 9: ^{57}Fe Mössbauer spectra of an Iguazú falls soil sample (Argentina-Brazil border)

3.2 Match of fitted parameters with a local database

A final cell allows us to identify the phases from an own database. This procedure is based on minimization of Eulerian distances of the set of parameters with the bibliographic collected ones, which can be updated or extended by the user. The Machine Learning algorithm employed to guess the present phases is based on a K-Nearest Neighbors (`sklearn.neighbors`). The output is a recommendation with a ranking of best matching suggested phases.



```
PyMossFit 2Q.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
Comandos + Código + Texto Ejecutar todo
is_ref = df_ref.iloc[pos]['IS (mm/s)']
qs_ref = df_ref.iloc[pos]['Quad Splitting (mm/s)']
bhf_ref = df_ref.iloc[pos]['Bhf (T)']
print(f" Match {j+1}: {compound} ({formula})")
print(f"      IS: {is_ref:.2f} mm/s | QS: {qs_ref:.2f} mm/s")
print(f"      Distancia euclidiana: {d:.2f}\n")

print("## Usar este resultado a modo orientativo. Se recomienda co

Drive already mounted at /content/drive; to attempt to forcibly re

Muestra experimental 1:
Match 1: Fe(CO)5 (Fe(CO)5)
IS: 0.25 mm/s | QS: 0.05 mm/s | Bhf: 0.0 T
Distancia euclidiana: 2.11

Match 2: Fe(phen)3(NCS)2 ([Fe(phen)3(NCS)2])
IS: 0.05 mm/s | QS: 0.05 mm/s | Bhf: 0.0 T
Distancia euclidiana: 2.11

Match 3: Fe4[Fe(CN)6]3·xH2O (Fe4[Fe(CN)6]3·xH2O)
IS: 0.05 mm/s | QS: 0.10 mm/s | Bhf: 0.0 T
Distancia euclidiana: 2.16

Match 4: Prussian Blue (Fe4[Fe(CN)6]3)
IS: 0.05 mm/s | QS: 0.10 mm/s | Bhf: 0.0 T
Distancia euclidiana: 2.16

Match 5: Fe3[Fe(CN)6]2 (Fe3[Fe(CN)6]2)
```

Figure 10: A match report obtained with a KNN algorithm for phase identification

References

- P. Corke and J. Haviland. Mössbauer mineralogy of rock, soil, and dust at meridiani planum, mars: Opportunity's journey across sulfate-rich outcrop, basaltic sand and dust, and hematite lag deposits. *Journal of Geophysical Research: Planets*, 111, 2006. URL <https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2006JE002791>.
- S. Ferrari, J. C. Apehsteguy, and F. D. Saccone. Structural and magnetic properties of zn doped magnetite nanoparticles obtained by wet chemical method. *IEEE Transactions on Magnetics*, 51, 2015. URL <https://ieeexplore.ieee.org/document/6975229/>.

- F. Grandjean and G. J. Long. Best practices and protocols in mössbauer spectroscopy. *Chemistry of Materials*, 33, 2021. URL <https://pubs.acs.org/doi/10.1021/acs.chemmater.1c00326>.
- Q.Q. Kong, T. Siau, and A. Bayen. Discrete fourier transform (dft), 2020. URL <https://pythonnumericalmethods.studentorg.berkeley.edu/notebooks/chapter24.02-Discrete-Fourier-Transform.html>.
- M. I. Oshtrakh, E.V. Petrova, and V.I. Grokhovsky. Determination of quadrupole splitting for ^{57}Fe in m1 and m2 sites of both olivine and pyroxene in ordinary chondrites using mössbauer spectroscopy with high velocity resolution. *Hyperfine Interactions*, 177, 2007. URL <https://link.springer.com/article/10.1007/s10751-008-9646-4>.
- F. D. Saccone. Fitting code for mössbauer spectroscopy running in jupyter colab, 2025. URL <https://doi.org/10.5281/zenodo.15428763>.
- Wikipedia contributors. Mössbauer spectroscopy — Wikipedia, the free encyclopedia, 2025. URL https://en.wikipedia.org/w/index.php?title=M%C3%B6ssbauer_spectroscopy&oldid=1279245955. [Online; accessed 19-June-2025].