

mrsimulator

A cross-platform object-oriented open-source software package
for fast solid-state NMR spectral simulation and analysis

D. J. Srivastava, M. Giammar, M. Venetos, A. McCarthy and P. J. Grandinetti

Department of Chemistry
L'Ohio State University

Web: www.grandinetti.org

Twitter: @pjgrandinetti



<https://mrsimulator.readthedocs.io>



New National Gateway Ultrahigh Field NMR Center



Materials, Bio-solids, Solutions of proteins and RNA,
Metabolomics



Coming soon: 1.2 GHz

- Avance III HD Ascend 600 MHz
- Avance III HD Ultrashield 600 MHz
- Avance III HD Ascend 700 MHz
- Avance III HD 800 MHz
- Avance III HD Ascend 850 MHz
- Avance III HD Ascend 800 MHz
- Avance III HD Aeon 800 MHz Wide-bore
- Avance III HD Ascend 600 MHz Wide-bore DNP NMR

<https://gateway-nmr.osu.edu/>



Deepansh Srivastava

Hyperfine



Maxwell Venetos

UC Berkeley



Grandinetti Lab

Summer 2021 Tomato Harvest Party



Lexi McCarthy, Matthew Giammar

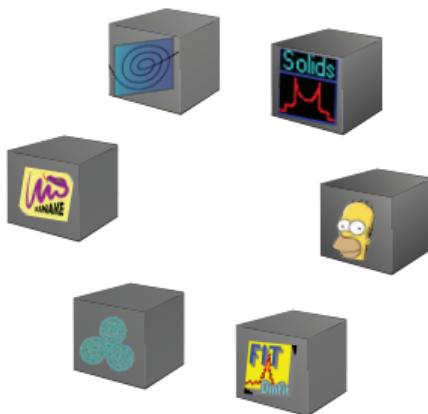
Brendan Wilson, Mark Bovee, Audrey Cowen,
Modeste Tegomoh

Email: grandinetti.1@osu.edu

<https://www.grandinetti.org/Software>

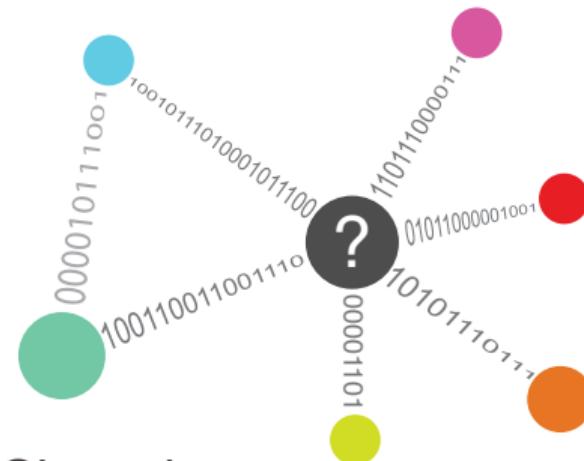
Motivation - Why another NMR simulation package?

- 1 Easy to learn and use by non-NMR specialists.
- 2 Easily connects to network of libraries and the cloud.



Monolithic

Disconnected from new data science tools

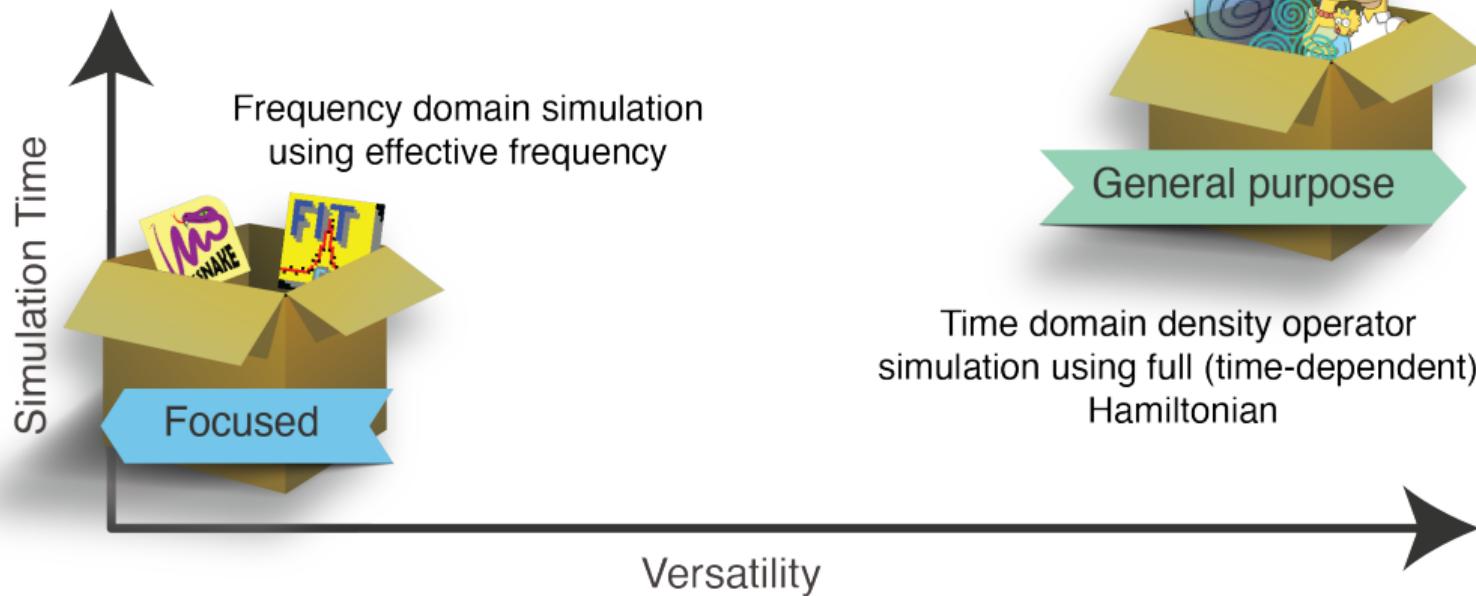


Shared

Library “plays well” with other libraries

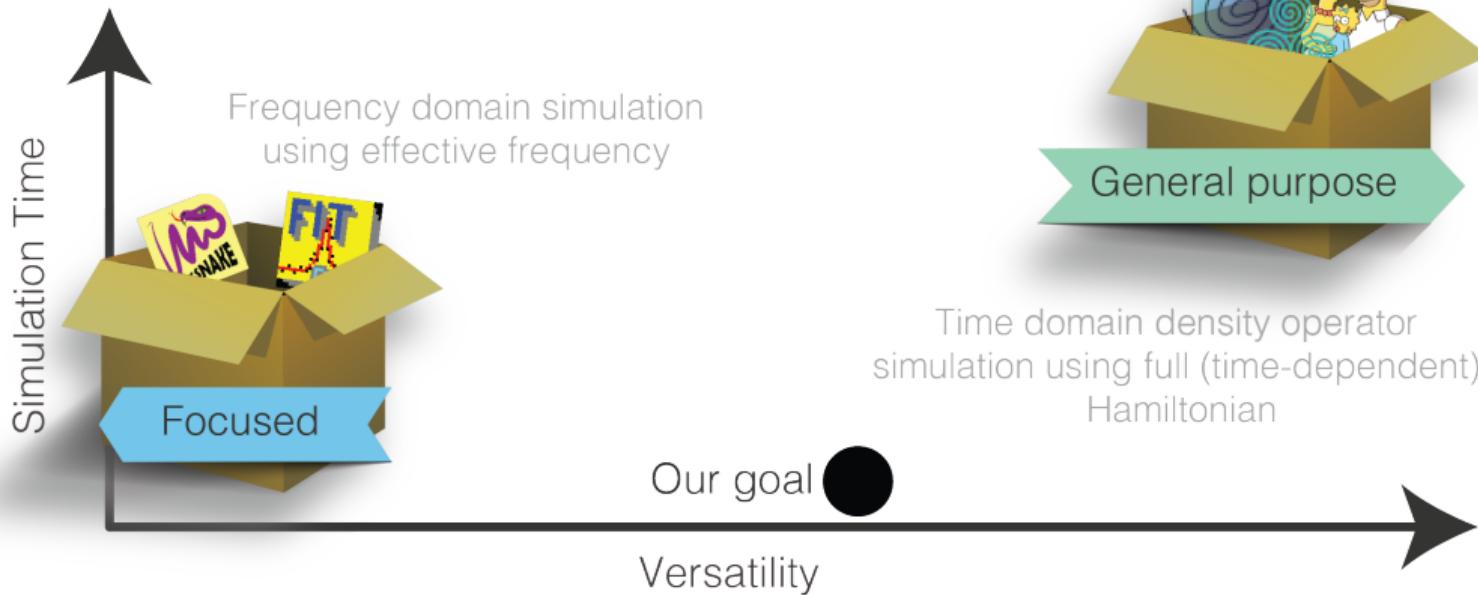
Motivation - Why another NMR simulation package?

3 Is fast and versatile



Motivation - Why another NMR simulation package?

3 Is fast and versatile



Why Python?



- Free and open-source
- Portable - can even be run in a web browser with hosted Jupyter notebook service
- Easy to learn, object-oriented and allows for fast prototype development
- Packages are easy to install
- Supported by all industry platforms
- Supported by a large global community
- Extensive libraries, such as Scikit-learn, have made Python the standard-bearer for machine learning and data science applications.
- Can be used to develop progressive web apps

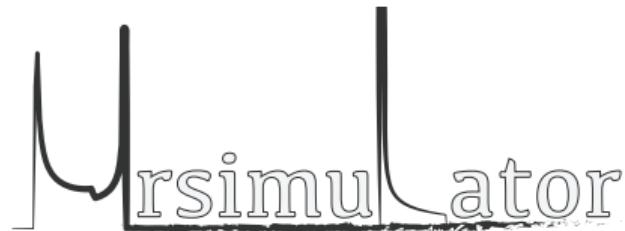
How can I get started with Python and mrsimulator?

- start with Jupyter notebooks:
 - ▶ run remotely in a web browser at Google's colab:
<https://colab.google.com>
 - ▶ run locally in a web browser. Recommend downloading Anaconda at
<https://www.anaconda.com>
- ask us to do a Zoom demo of mrsimulator at your next research group meeting.

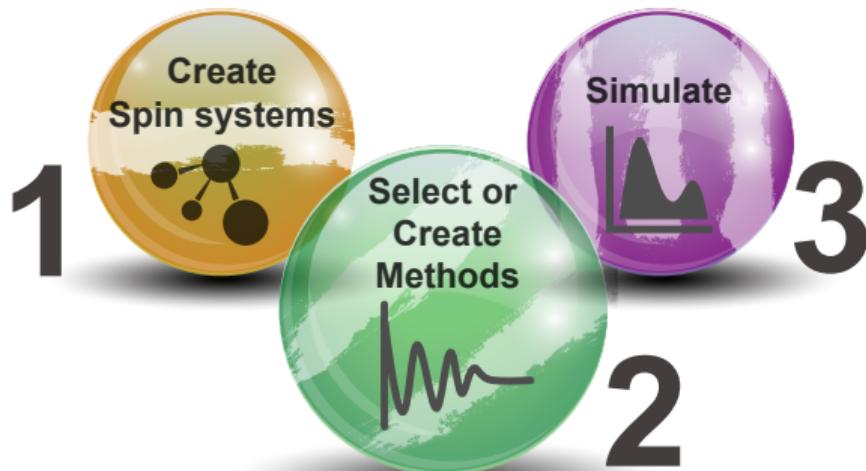
grandinetti.1@osu.edu

“pip install mrsimulator”

inside Jupyter Notebook*: “!pip install mrsimulator”



A Python toolbox for simulating
solid-state NMR spectra



*If using Google colab, first update numpy library with “!pip install -U numpy”

Example of an mrsimulator script

```
# Define Sites, Coupling, and Spin System
from mrsimulator import Site, Coupling, SpinSystem
C_site = Site(isotope = "13C", isotropic_chemical_shift = 100.0, shielding_symmetric=dict(zeta=70., eta=0.5))
H_site = Site(isotope = "1H")
coupling = Coupling(site_index = [0,1], dipolar = dict(D=-2e4))
spin_system = SpinSystem(sites = [C_site,H_site], couplings = [coupling])

# Select Method
from mrsimulator.methods import BlochDecaySpectrum
method = BlochDecaySpectrum(channels=["13C"],
                             magnetic_flux_density=9.4, # in T
                             rotor_frequency=3000,      # in Hz
                             spectral_dimensions=[{"count": 2048, "spectral_width": 8.0e4}])

# Simulate Spectrum
from mrsimulator import Simulator
# Create Simulator with lists of spin_systems and methods
sim = Simulator(spin_systems=[spin_system],methods=[method])
sim.run() # simulate.

# Do post-simulation signal processing
from mrsimulator import signal_processing as sp
processor = sp.SignalProcessor(operations=[ sp.IFFT(), sp.apodization.Exponential(FWHM="200 Hz"), sp.FFT(),])
processed_data = processor.apply_operations(data=sim.methods[0].simulation)
```

Example of an mrsimulator script

```
# Define Sites, Coupling, and Spin System
from mrsimulator import Site, Coupling, SpinSystem
C_site = Site(isotope = "13C", isotropic_chemical_shift = 100.0, shielding_symmetric=dict(zeta=70., eta=0.5))
H_site = Site(isotope = "1H")
coupling = Coupling(site_index = [0,1], dipolar = dict(D=-2e4))
spin_system = SpinSystem(sites = [C_site,H_site], couplings = [coupling])

# Select Method
from mrsimulator.methods import BlochDecaySpectrum
method = BlochDecaySpectrum(channels=["13C"],
                             magnetic_flux_density=9.4, # in T
                             rotor_frequency=3000,      # in Hz
                             spectral_dimensions=[{"count": 2048, "spectral_width": 8.0e4}])

# Simulate Spectrum
from mrsimulator import Simulator
# Create Simulator with lists of spin_systems and methods
sim = Simulator(spin_systems=[spin_system],methods=[method])
sim.run() # simulate.

# Do post-simulation signal processing
from mrsimulator import signal_processing as sp
processor = sp.SignalProcessor(operations=[ sp.IFFT(), sp.apodization.Exponential(FWHM="200 Hz"), sp.FFT(),])
processed_data = processor.apply_operations(data=sim.methods[0].simulation)
```

Example of an mrsimulator script

```
# Define Sites, Coupling, and Spin System
from mrsimulator import Site, Coupling, SpinSystem
C_site = Site(isotope = "13C", isotropic_chemical_shift = 100.0, shielding_symmetric=dict(zeta=70., eta=0.5))
H_site = Site(isotope = "1H")
coupling = Coupling(site_index = [0,1], dipolar = dict(D=-2e4))
spin_system = SpinSystem(sites = [C_site,H_site], couplings = [coupling])

# Select Method
from mrsimulator.methods import BlochDecaySpectrum
method = BlochDecaySpectrum(channels=["13C"],
                             magnetic_flux_density=9.4, # in T
                             rotor_frequency=3000,      # in Hz
                             spectral_dimensions=[{"count": 2048, "spectral_width": 8.0e4}])

# Simulate Spectrum
from mrsimulator import Simulator
# Create Simulator with lists of spin_systems and methods
sim = Simulator(spin_systems=[spin_system],methods=[method])
sim.run()    # simulate.

# Do post-simulation signal processing
from mrsimulator import signal_processing as sp
processor = sp.SignalProcessor(operations=[ sp.IFFT(), sp.apodization.Exponential(FWHM="200 Hz"), sp.FFT(),])
processed_data = processor.apply_operations(data=sim.methods[0].simulation)
```

Example of an mrsimulator script

```
# Define Sites, Coupling, and Spin System
from mrsimulator import Site, Coupling, SpinSystem
C_site = Site(isotope = "13C", isotropic_chemical_shift = 100.0, shielding_symmetric=dict(zeta=70., eta=0.5))
H_site = Site(isotope = "1H")
coupling = Coupling(site_index = [0,1], dipolar = dict(D=-2e4))
spin_system = SpinSystem(sites = [C_site,H_site], couplings = [coupling])

# Select Method
from mrsimulator.methods import BlochDecaySpectrum
method = BlochDecaySpectrum(channels=["13C"],
                             magnetic_flux_density=9.4,    # in T
                             rotor_frequency=3000,        # in Hz
                             spectral_dimensions=[{"count": 2048, "spectral_width": 8.0e4}])

# Simulate Spectrum
from mrsimulator import Simulator
# Create Simulator with lists of spin_systems and methods
sim = Simulator(spin_systems=[spin_system],methods=[method])
sim.run()    # simulate.

# Do post-simulation signal processing
from mrsimulator import signal_processing as sp
processor = sp.SignalProcessor(operations=[ sp.IFFT(), sp.apodization.Exponential(FWHM="200 Hz"), sp.FFT(),])
processed_data = processor.apply_operations(data=sim.methods[0].simulation)
```

Example of an mrsimulator script

```
# Define Sites, Coupling, and Spin System
from mrsimulator import Site, Coupling, SpinSystem
C_site = Site(isotope = "13C", isotropic_chemical_shift = 100.0, shielding_symmetric=dict(zeta=70., eta=0.5))
H_site = Site(isotope = "1H")
coupling = Coupling(site_index = [0,1], dipolar = dict(D=-2e4))
spin_system = SpinSystem(sites = [C_site,H_site], couplings = [coupling])

# Select Method
from mrsimulator.methods import BlochDecaySpectrum
method = BlochDecaySpectrum(channels=["13C"],
                             magnetic_flux_density=9.4, # in T
                             rotor_frequency=3000,      # in Hz
                             spectral_dimensions=[{"count": 2048, "spectral_width": 8.0e4}])

# Simulate Spectrum
from mrsimulator import Simulator
# Create Simulator with lists of spin_systems and methods
sim = Simulator(spin_systems=[spin_system],methods=[method])
sim.run()    # simulate.

# Do post-simulation signal processing
from mrsimulator import signal_processing as sp
processor = sp.SignalProcessor(operations=[ sp.IFFT(), sp.apodization.Exponential(FWHM="200 Hz"), sp.FFT(),])
processed_data = processor.apply_operations(data=sim.methods[0].simulation)
```

Example of an mrsimulator script

```
# Define Sites, Coupling, and Spin System
from mrsimulator import Site, Coupling, SpinSystem
C_site = Site(isotope = "13C", isotropic_chemical_shift = 100.0, shielding_symmetric=dict(zeta=70., eta=0.5))
H_site = Site(isotope = "1H")
coupling = Coupling(site_index = [0,1], dipolar = dict(D=-2e4))
spin_system = SpinSystem(sites = [C_site,H_site], couplings = [coupling])

# Select Method
from mrsimulator.methods import BlochDecaySpectrum
method = BlochDecaySpectrum(channels=["13C"],
                             magnetic_flux_density=9.4,    # in T
                             rotor_frequency=3000,        # in Hz
                             spectral_dimensions=[{"count": 2048, "spectral_width": 8.0e4}])

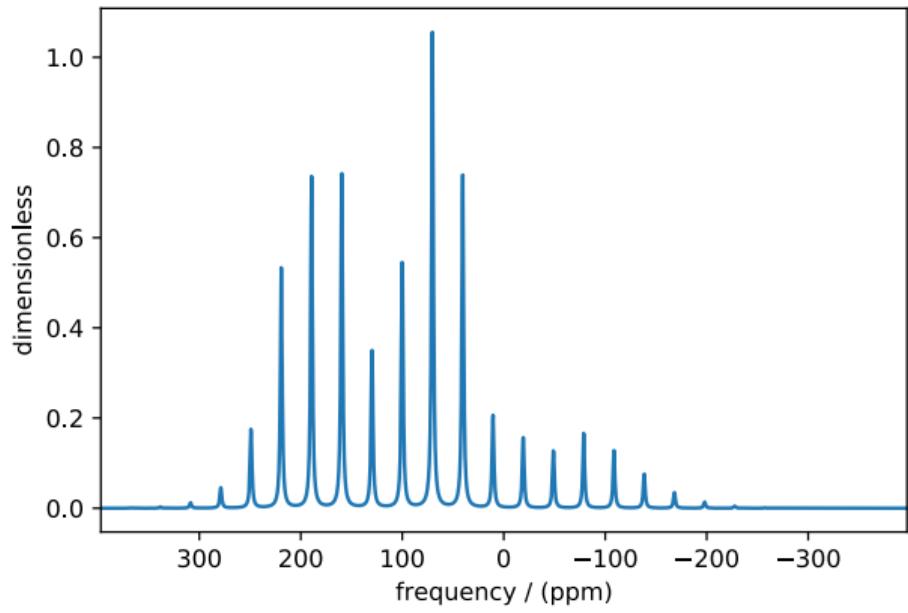
# Simulate Spectrum
from mrsimulator import Simulator
# Create Simulator with lists of spin_systems and methods
sim = Simulator(spin_systems=[spin_system],methods=[method])
sim.run()    # simulate.

# Do post-simulation signal processing
from mrsimulator import signal_processing as sp
processor = sp.SignalProcessor(operations=[ sp.IFFT(), sp.apodization.Exponential(FWHM="200 Hz"), sp.FFT(),])
processed_data = processor.apply_operations(data=sim.methods[0].simulation)
```

Matplotlib - Python library for creating static, animated, and interactive visualizations

<https://matplotlib.org>

```
import matplotlib.pyplot as plt
plt.figure()
ax = plt.subplot(projection="csdm")
ax.plot(processed_data.real)
ax.invert_xaxis()
plt.savefig("plot.pdf")
plt.show()
```



The **SpinSystem** object is collection of **Site** and **Coupling** objects

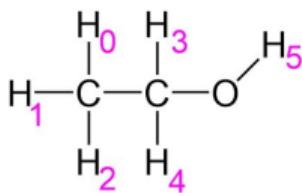
```
# methyl proton site
H_CH3 = Site(isotope='1H', isotropic_chemical_shift=1.226)
# methylene proton site
H_CH2 = Site(isotope='1H', isotropic_chemical_shift=2.61)
# hydroxyl proton site
H_OH = Site(isotope='1H', isotropic_chemical_shift=3.687)
sites = [H_CH3, H_CH3, H_CH3, H_CH2, H_CH2, H_OH] # Put sites into list

HH_coupling_1 = Coupling(site_index=[0, 3], isotropic_j=7)
HH_coupling_2 = Coupling(site_index=[0, 4], isotropic_j=7)
HH_coupling_3 = Coupling(site_index=[1, 3], isotropic_j=7)
HH_coupling_4 = Coupling(site_index=[1, 4], isotropic_j=7)
HH_coupling_5 = Coupling(site_index=[2, 3], isotropic_j=7)
HH_coupling_6 = Coupling(site_index=[2, 4], isotropic_j=7)
# Put couplings into list
couplings = [HH_coupling_1, HH_coupling_2, HH_coupling_3,
             HH_coupling_4, HH_coupling_5, HH_coupling_6,]

isotopomer1 = SpinSystem(sites=sites, couplings=couplings, abundance=97.812)
```

The **SpinSystem** object is collection of **Site** and **Coupling** objects

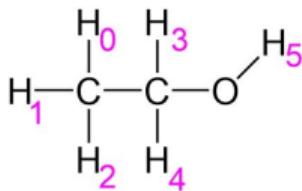
```
# methyl proton site
H_CH3 = Site(isotope='1H', isotropic_chemical_shift=1.226)
# methylene proton site
H_CH2 = Site(isotope='1H', isotropic_chemical_shift=2.61)
# hydroxyl proton site
H_OH = Site(isotope='1H', isotropic_chemical_shift=3.687)
sites = [H_CH3, H_CH3, H_CH3, H_CH2, H_CH2, H_OH] # Put sites into list
```



```
HH_coupling_1 = Coupling(site_index=[0, 3], isotropic_j=7)
HH_coupling_2 = Coupling(site_index=[0, 4], isotropic_j=7)
HH_coupling_3 = Coupling(site_index=[1, 3], isotropic_j=7)
HH_coupling_4 = Coupling(site_index=[1, 4], isotropic_j=7)
HH_coupling_5 = Coupling(site_index=[2, 3], isotropic_j=7)
HH_coupling_6 = Coupling(site_index=[2, 4], isotropic_j=7)
# Put couplings into list
couplings = [HH_coupling_1, HH_coupling_2, HH_coupling_3,
             HH_coupling_4, HH_coupling_5, HH_coupling_6,]

isotopomer1 = SpinSystem(sites=sites, couplings=couplings, abundance=97.812)
```

The **SpinSystem** object is collection of **Site** and **Coupling** objects



```
# methyl proton site
H_CH3 = Site(isotope='1H', isotropic_chemical_shift=1.226)
# methylene proton site
H_CH2 = Site(isotope='1H', isotropic_chemical_shift=2.61)
# hydroxyl proton site
H_OH = Site(isotope='1H', isotropic_chemical_shift=3.687)
sites = [H_CH3, H_CH3, H_CH3, H_CH2, H_OH] # Put sites into list

HH_coupling_1 = Coupling(site_index=[0, 3], isotropic_j=7)
HH_coupling_2 = Coupling(site_index=[0, 4], isotropic_j=7)
HH_coupling_3 = Coupling(site_index=[1, 3], isotropic_j=7)
HH_coupling_4 = Coupling(site_index=[1, 4], isotropic_j=7)
HH_coupling_5 = Coupling(site_index=[2, 3], isotropic_j=7)
HH_coupling_6 = Coupling(site_index=[2, 4], isotropic_j=7)
# Put couplings into list
couplings = [HH_coupling_1, HH_coupling_2, HH_coupling_3,
             HH_coupling_4, HH_coupling_5, HH_coupling_6,]

isotopomer1 = SpinSystem(sites=sites, couplings=couplings, abundance=97.812)
```

The **SpinSystem** object is collection of **Site** and **Coupling** objects

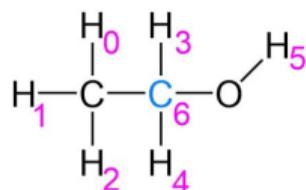
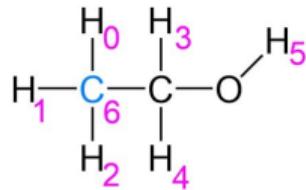
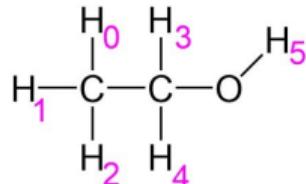
```
# methyl proton site
H_CH3 = Site(isotope='1H', isotropic_chemical_shift=1.226)
# methylene proton site
H_CH2 = Site(isotope='1H', isotropic_chemical_shift=2.61)
# hydroxyl proton site
H_OH = Site(isotope='1H', isotropic_chemical_shift=3.687)
sites = [H_CH3, H_CH3, H_CH3, H_CH2, H_CH2, H_OH] # Put sites into list

HH_coupling_1 = Coupling(site_index=[0, 3], isotropic_j=7)
HH_coupling_2 = Coupling(site_index=[0, 4], isotropic_j=7)
HH_coupling_3 = Coupling(site_index=[1, 3], isotropic_j=7)
HH_coupling_4 = Coupling(site_index=[1, 4], isotropic_j=7)
HH_coupling_5 = Coupling(site_index=[2, 3], isotropic_j=7)
HH_coupling_6 = Coupling(site_index=[2, 4], isotropic_j=7)
# Put couplings into list
couplings = [HH_coupling_1, HH_coupling_2, HH_coupling_3,
             HH_coupling_4, HH_coupling_5, HH_coupling_6,]

isotopomer1 = SpinSystem(sites=sites, couplings=couplings, abundance=97.812)
```

Simulator accepts a collection of SpinSystem objects

The ^1H spectrum of ethanol also contains resonances from other isotopomers



```
# Create first C13 isotopomer
C_CH3 = Site(isotope="13C", isotropic_chemical_shift=18)
iso2_sites = sites + C_CH3

CH3_coupling_1 = Coupling(site_index=[0, 6], isotropic_j=125)
CH3_coupling_2 = Coupling(site_index=[1, 6], isotropic_j=125)
CH3_coupling_3 = Coupling(site_index=[2, 6], isotropic_j=125)
iso2_coupleings = coupleings + [CH3_coupling_1, CH3_coupling_2, CH3_coupling_3]

isotopomer2 = SpinSystem(sites=iso2_sites, couplings=iso2_coupleings, abundance=1.088)

# Create second C13 isotopomer
C_CH2 = Site(isotope="13C", isotropic_chemical_shift=58)
iso3_sites = sites + C_CH2

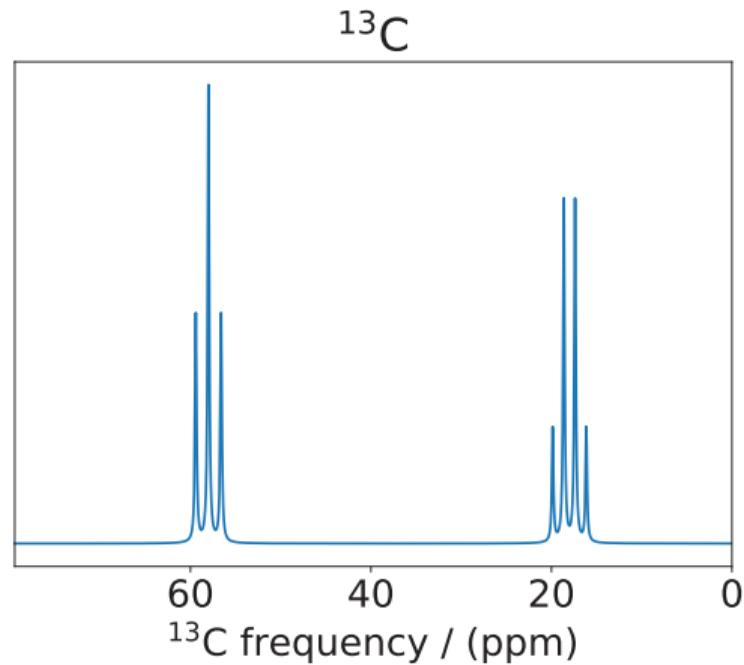
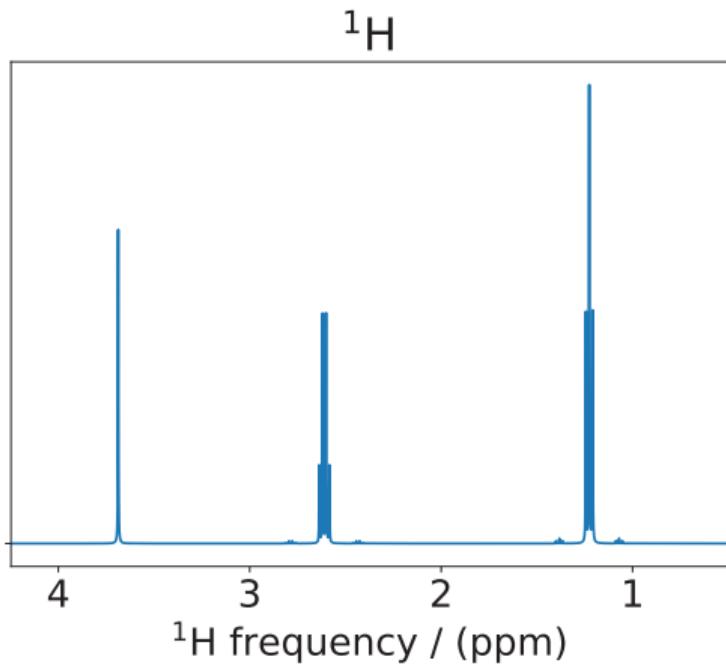
CH2_coupling_1 = Coupling(site_index=[3, 6], isotropic_j=141)
CH2_coupling_2 = Coupling(site_index=[4, 6], isotropic_j=141)
iso3_coupleings = coupleings + [CH2_coupling_1, CH2_coupling_2]

isotopomer3 = SpinSystem(sites=iso3_sites, couplings=iso3_coupleings, abundance=1.088)

sim = Simulator()
sim.spin_systems = [isotopomer1, isotopomer2, isotopomer3]
```

Simulator accepts a collection of SpinSystem objects

The ^1H spectrum of ethanol also contains resonances from other isotopomers



The **Method** object holds a collection of **SpectralDimension** objects

The **Method** object holds a collection of **SpectralDimension** objects

- Each **SpectralDimension** object describes a dimension in a multi-dimensional spectrum.

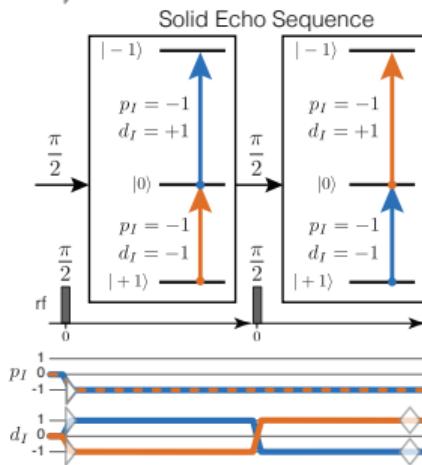
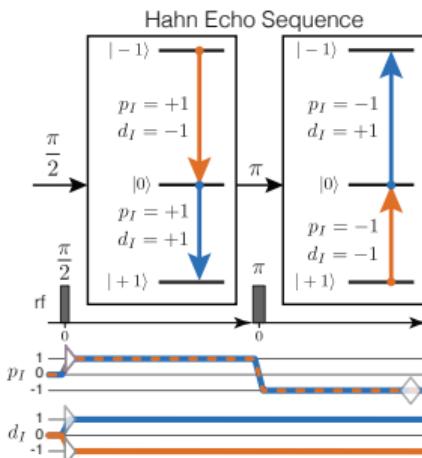
The **Method** object holds a collection of **SpectralDimension** objects

- Each **SpectralDimension** object describes a dimension in a multi-dimensional spectrum.
- Inside a **SpectralDimension** you create a sequence of events. Inside each **Event** you can
 - ▶ select among a standard list of NMR frequency contributions
 - ▶ adjust parameters, e.g., rotor speed or angle, B_0 , etc.
 - ▶ select transitions based on their p or d symmetries, i.e., $p = m_f - m_i$ and $d = m_f^2 - m_i^2$
 - ▶ modulate transition amplitude with a constant duration event, , e.g., $\tau = 1/(4J)$.
 - ▶ transfer coherence among transitions with a rotation, e.g., $\pi/2$, π , etc.
 - ▶ calculate a spectrum with amplitudes and frequencies determined by sequence of prior events.

Hahn vs Solid Echo Methods on Deuterium

```
hahn_echo = Method(  
    channels=["2H"],  
    magnetic_flux_density=9.4, # in T  
    spectral_dimensions=[  
        {"count": 512,  
         "spectral_width": 2e4, # in Hz  
         "events": [  
             SpectralEvent(fraction=0.5, transition_query=[{"ch1": {"P": [1]}]}),  
             MixingEvent(mixing_query={"ch1": {"tip_angle": np.pi, "phase": 0}}),  
             SpectralEvent(fraction=0.5, transition_query=[{"ch1": {"P": [-1]}]}])]  
    ]  
)
```

```
solid_echo = Method(  
    channels=["2H"],  
    magnetic_flux_density=9.4, # in T  
    spectral_dimensions=[  
        {"count": 512,  
         "spectral_width": 2e4, # in Hz  
         "events": [  
             SpectralEvent(fraction=0.5, transition_query=[{"ch1": {"P": [-1]}]}),  
             MixingEvent(mixing_query={"ch1": {"tip_angle": np.pi/2, "phase": 0}}),  
             SpectralEvent(fraction=0.5, transition_query=[{"ch1": {"P": [-1]}]}])]  
    ]  
)
```

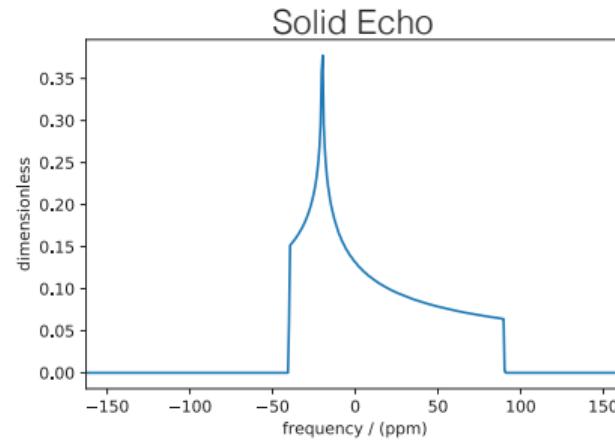
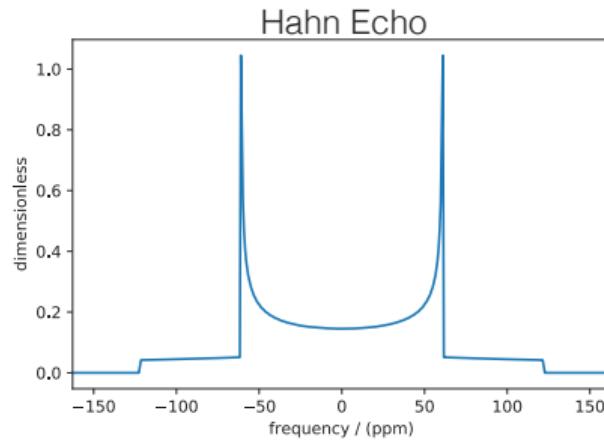


Hahn vs Solid Echo Methods on Deuterium

Simulator accepts a collection of Method objects

```
spin_system = SpinSystem(sites=[Site(isotope="2H",
                                      isotropic_chemical_shift=10, # in ppm
                                      shielding_symmetric={"zeta": -80, "eta": 0.25}, # zeta in ppm
                                      quadrupolar={"Cq": 10e3, "eta": 0.0})])

sim = Simulator(spin_systems=[spin_system], methods=[hahn_echo, solid_echo])
sim.run()
```



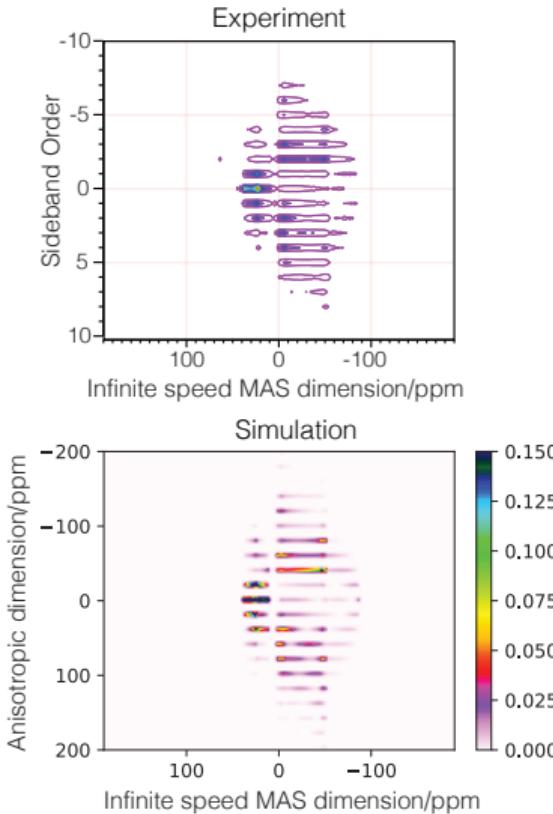
mrsimulator has pre-built specialized Methods for easier setup

- **BlochDecaySpectrum** - Bloch Decay 1D spectrum
- **BlochCTDecaySpectrum** - Bloch Decay 1D spectrum of quadrupolar central transition
- **ThreeQ_VAS** - 3-quantum 2D spectrum of half-integer quadrupoles
- **FiveQ_VAS** - 5-quantum 2D spectrum of half-integer quadrupoles
- **SevenQ_VAS** - 7-quantum 2D spectrum of half-integer quadrupoles
- **ST1_VAS** - 1st inner satellite transition 2D spectrum of half-integer quadrupoles
- **ST2_VAS** - 2nd inner satellite transition 2D spectrum of half-integer quadrupoles
- **SSB2D** - Finite to infinite MAS speed correlation 2D spectrum.
- **Method1D** - generic 1D method
- **Method2D** - generic 2D method

^{87}Rb TOP-QMATE-PIETA spectra of Rb_2SO_4

Walder et al. J. Chem. Phys. 138, 174203 (2013)

```
qmat = SSB2D(
    channels=["87Rb"],
    magnetic_flux_density=9.4,
    rotor_frequency=2604,
    spectral_dimensions=[
        {
            "count": 32 * 4,
            "spectral_width": 2604 * 32, # in Hz
            "label": "Anisotropic dimension",
        },
        {
            "count": 512,
            "spectral_width": 50000, # in Hz
            "label": "Infinite Speed MAS dimension",
        },
    ],
)
```

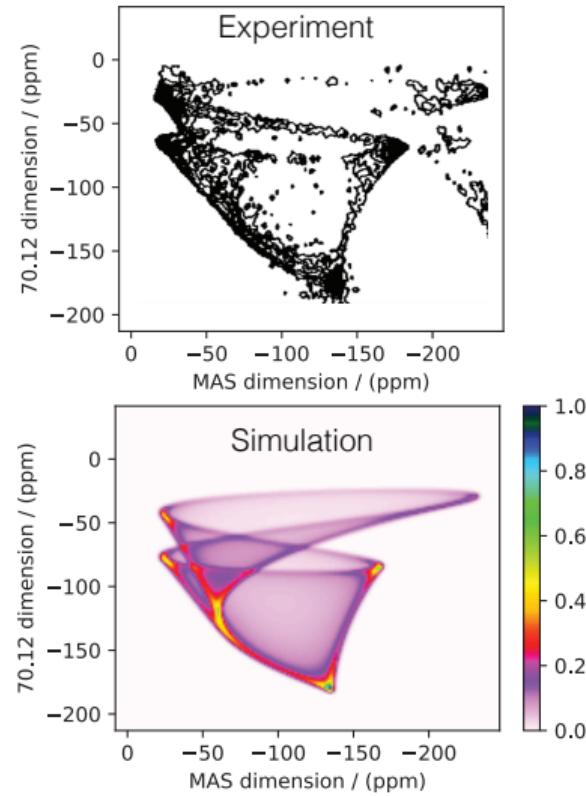


Simulation time: 30 milliseconds

^{87}Rb 2D Switched-Angle Spinning (SAS) of Rb_2CrO_4

Shore et al., J. Chem. Phys., 105, 9412 (1996)

```
sas = Method2D(channels=["87Rb"],  
    magnetic_flux_density=4.2,  
    spectral_dimensions=[  
        {"count": 256,  
         "spectral_width": 1.5e4, # in Hz  
         "reference_offset": -5e3, # in Hz  
         "label": "70.12 dimension",  
         "events": [{"  
             "rotor_angle": 70.12 * 3.14159 / 180,  
             "transition_query": [{"ch1": {"P": [-1], "D": [0]}},]  
         }],  
        },  
        {"count": 512,  
         "spectral_width": 15e3, # in Hz  
         "reference_offset": -7e3, # in Hz  
         "label": "MAS dimension",  
         "events": [{"  
             "rotor_angle": 54.74 * 3.14159 / 180,  
             "transition_query": [{"ch1": {"P": [-1], "D": [0]}},]  
         }],  
        },  
    ],  
)
```



Simulation time: 40 milliseconds

mrsimulator:docs v0.6.1

mrsimulator

A python toolbox for simulating solid-state NMR spectra

Search... Go

Getting Started
Installations
Introduction to Spin Systems
The Basics
Uncoupled Spin System: Using objects
Coupled Spin System: Using objects
Configuring Simulator object
mrsimulator I/O
Signal Processing
Signal Processing
Models
Czjzek distribution
Extended Czjzek distribution
Examples and Benchmarks

Simulation Examples

- 1D NMR simulation (small molecules/crystalline solids)
- 1D NMR simulation (macromolecules/amorphous solids)
- 2D NMR simulation (Crystalline solids)
- 2D NMR simulation (Disordered/Amorphous solids)

Fitting Examples (Least Squares)
Performance benchmark
Theory
How does **mrsimulator** work?

previous next index

Simulation Examples

In this section, we use the `mrsimulator` tools to create spin systems and simulate spectrum with practical/experimental applications. The examples illustrate

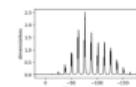
- building spin systems (uncoupled and weakly-coupled),
- building NMR methods,
- simulating spectrum, and
- processing spectrum (e.g. adding line-broadening).

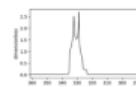
For mrsimulator applications related to least-squares fitting, see the [Fitting Examples \(Least Squares\)](#).

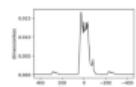
1D NMR simulation (small molecules/crystalline solids)

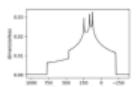
The following examples are the NMR spectrum simulation of small molecules and crystalline solids for the following methods:

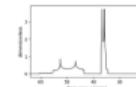
- Bloch decay method (`BlochDecaySpectrum`),
- Central transition selective Bloch decay method (`BlochDecayCTSpectrum`).
- Generic one-dimensional method (`Method1D`).

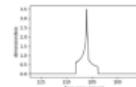

Figure 26
Wollastonite, ^{29}Si ($I=1/2$)

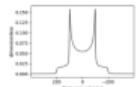

Figure 27
Potassium Sulfate, ^{35}S ($I=3/2$)

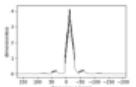

Figure 28
Coesite, ^{17}O ($I=5/2$)


Figure 29
Non-coincident Quad and CSA, ^{17}O ($I=5/2$)


Figure 30
Arbitrary spin transition (single-quantum)


Figure 31
Arbitrary spin transition (multi-quantum)


Figure 32
Coupled spin-1/2 (Static dipolar spectrum)

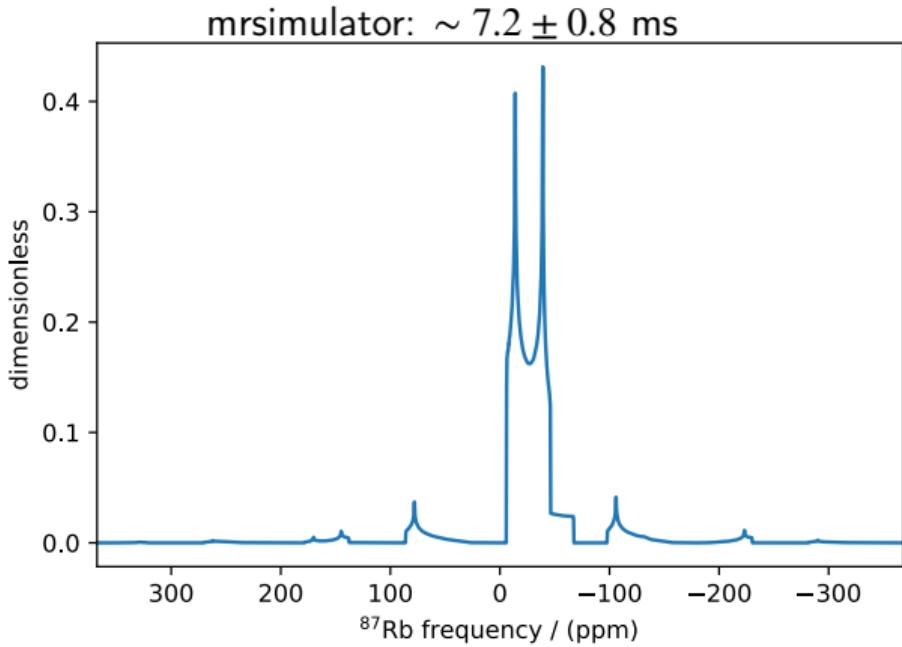

Figure 33
Coupled spins 5/2-9/2 (Quad + J-coupling) v: stable

<https://mrsimulator.readthedocs.io/en/stable/theory/models.html>

Benchmarks

How fast is mrsimulator compared to other NMR simulation packages?

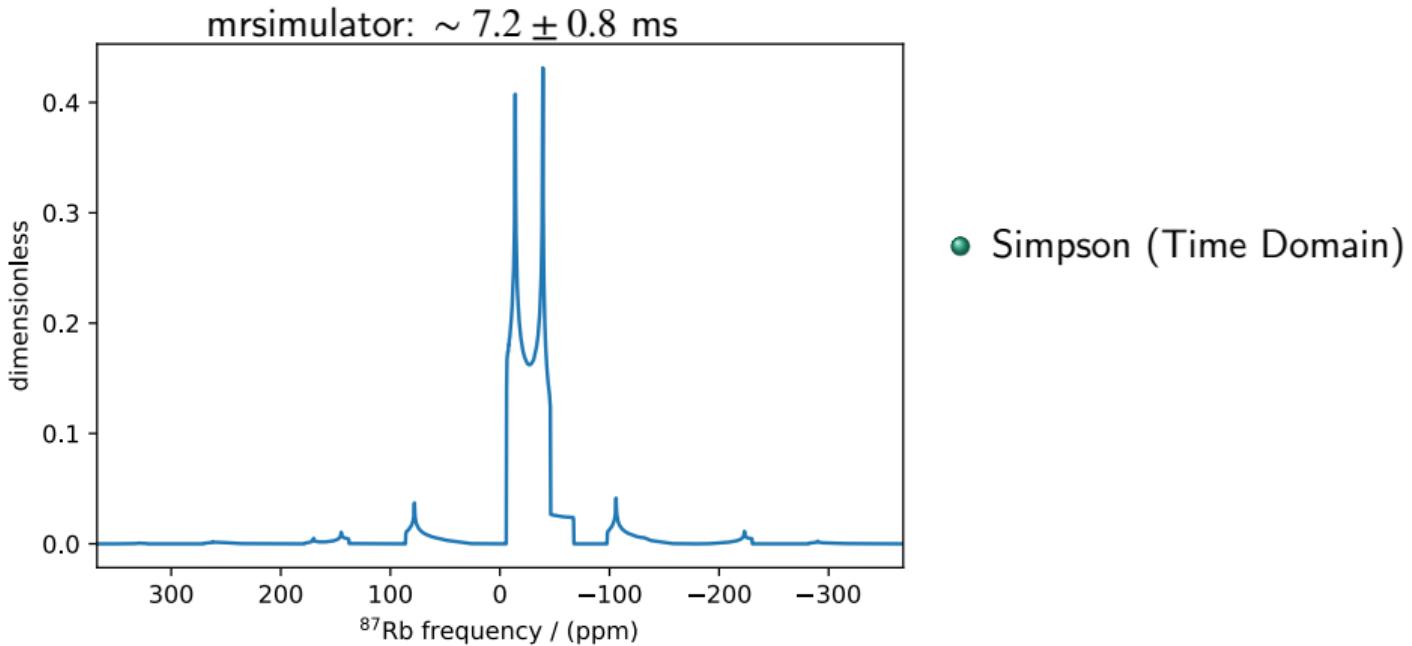
test case: central transition spectrum of half-integer quadrupole with finite spinning speed.



Benchmarks

How fast is mrsimulator compared to other NMR simulation packages?

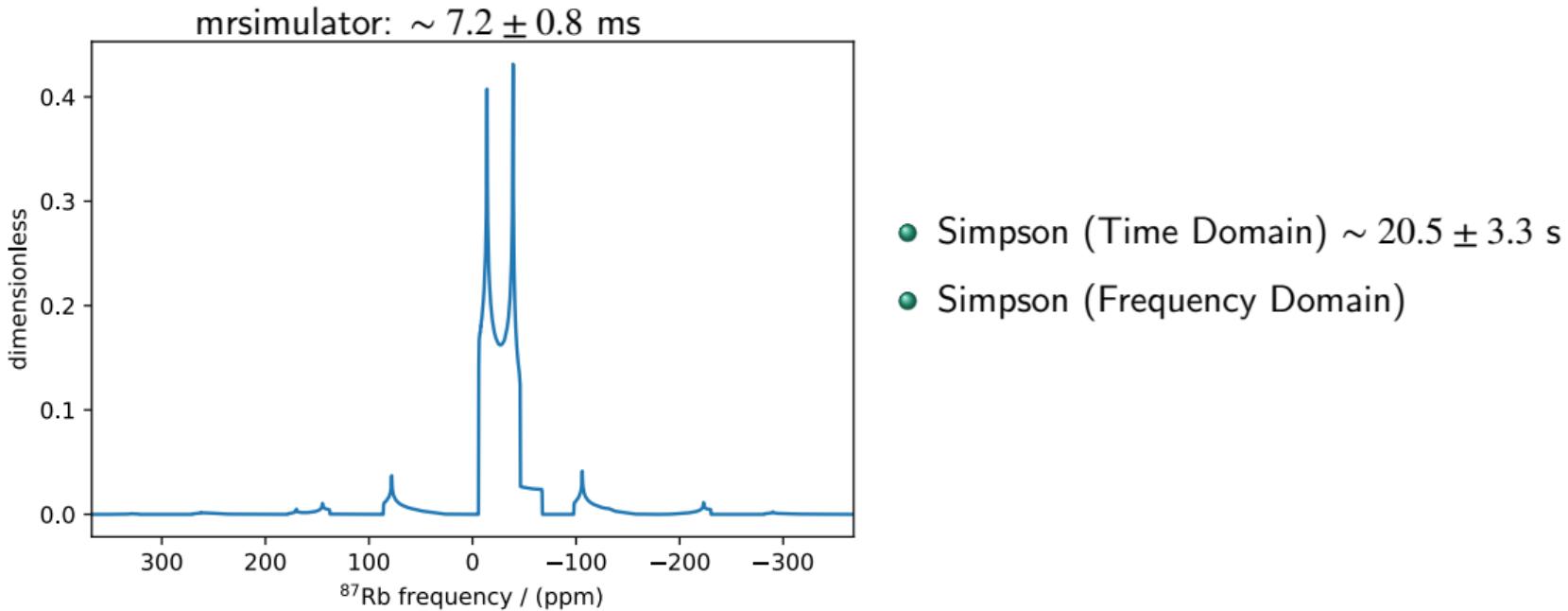
test case: central transition spectrum of half-integer quadrupole with finite spinning speed.



Benchmarks

How fast is mrsimulator compared to other NMR simulation packages?

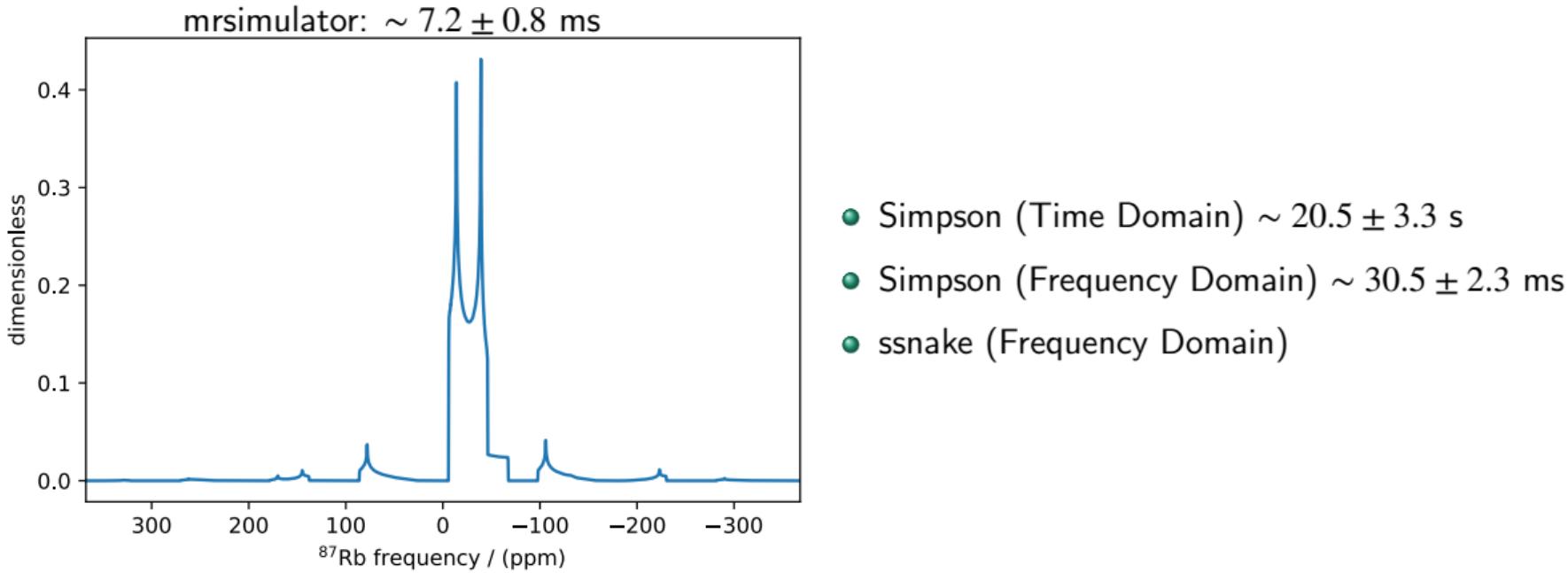
test case: central transition spectrum of half-integer quadrupole with finite spinning speed.



Benchmarks

How fast is mrsimulator compared to other NMR simulation packages?

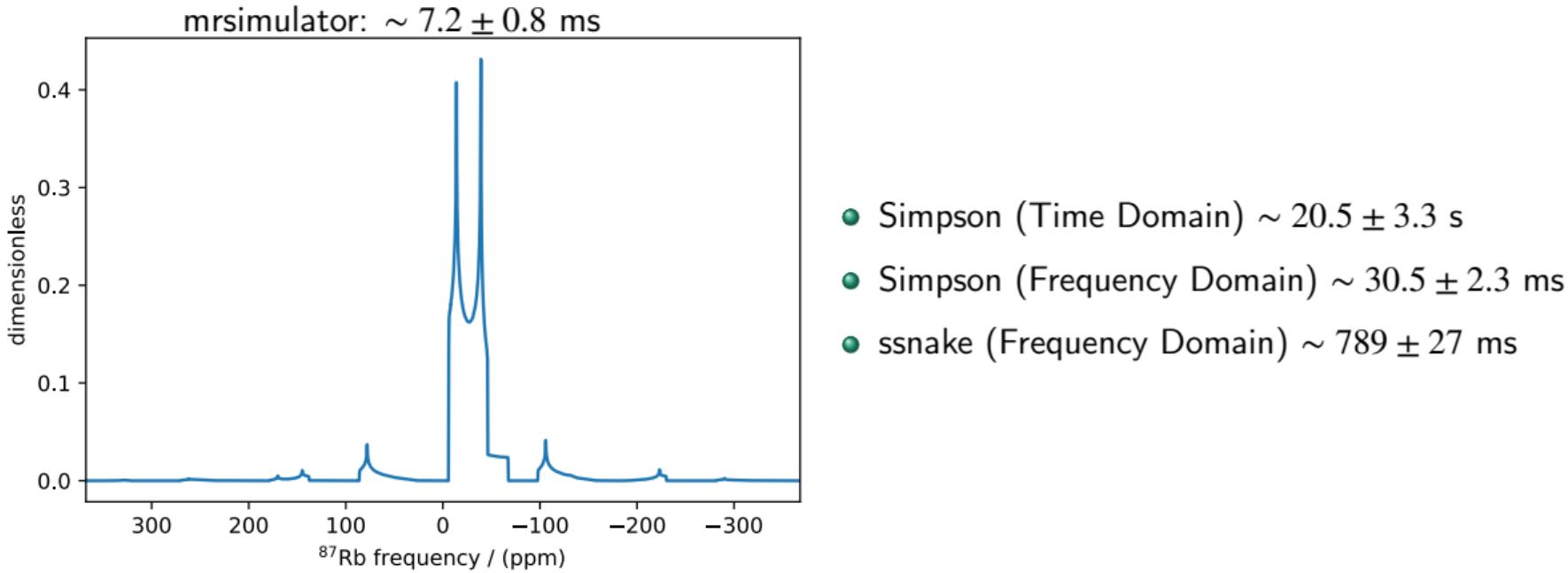
test case: central transition spectrum of half-integer quadrupole with finite spinning speed.



Benchmarks

How fast is mrsimulator compared to other NMR simulation packages?

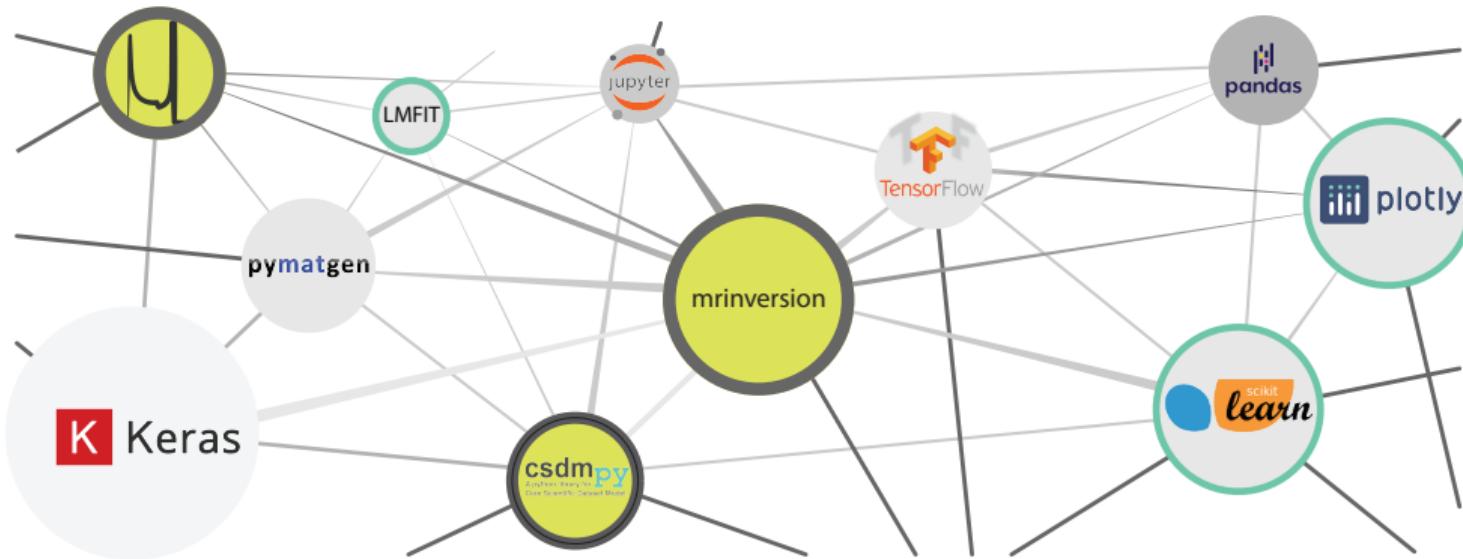
test case: central transition spectrum of half-integer quadrupole with finite spinning speed.



Building a Network of Software Libraries for Solid-State NMR



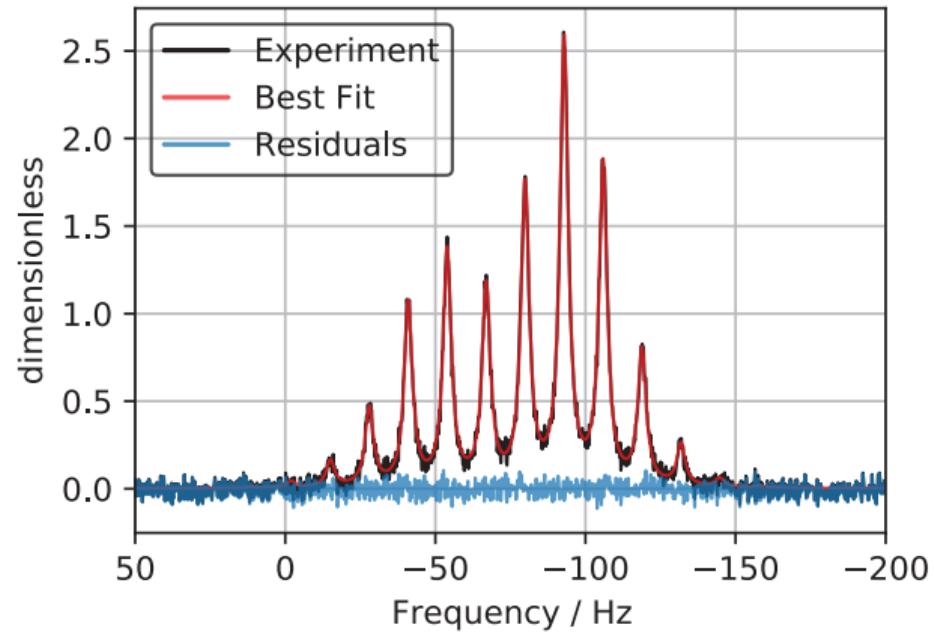
A platform for modern-day scientific tools



Wanted: Python programmers with solid-state NMR expertise

Solve least-squares problems with mrsimulator + LMFit

<https://lmfit.github.io/lmfit-py>



Three steps in least-squares calculations with mrsimulator + LMFit

First: Import the experimental spectrum.

```
import csdmpy as csm
file = "https://sandbox.zenodo.org/record/835664/files/synthetic_cuspidine_test.csdf"
experiment = csm.load(file)
sigma = 0.03383338 # standard deviation of noise from the dataset
experiment = experiment.real # For spectral fitting, use real part of the complex dataset
experiment.x[0].to("ppm", "nmr_frequency_ratio") # Convert coordinates from Hz to ppm.

from mrsimulator.utils import get_spectral_dimensions
# get the count, spectral_width, and reference_offset information from the experiment.
spectral_dims = get_spectral_dimensions(experiment)
```

Three steps in least-squares calculations with mrsimulator + LMFit

Second: Setup the Spin System, Method, and Simulation.

```
from mrsimulator.methods import BlochDecaySpectrum
from mrsimulator import Simulator, SpinSystem
from mrsimulator import signal_processing as sp

site = Site(isotope="29Si",isotropic_chemical_shift=-82.0,
            shielding_symmetric={"zeta": -63, "eta": 0.4})
spin_system = SpinSystem(sites=[site], abundance=100, name="Si Site")

MAS = BlochDecaySpectrum(channels=["29Si"],magnetic_flux_density=7.1,rotor_frequency=780,
spectral_dimensions=spectral_dims,experiment=experiment) # attach experiment to method

sim = Simulator(spin_systems=[spin_system], methods=[MAS])
sim.run()
processor = sp.SignalProcessor(operations=[sp.IFFT(),
                                            sp.apodization.Exponential(FWHM="100 Hz"),
                                            sp.FFT(),sp.Scale(factor=3)])
processed_data = processor.apply_operations(data=sim.methods[0].simulation).real
```

Three steps in least-squares calculations with mrsimulator + LMFit

Last: Create LMFit parameters from Simulator and SignalProcessor, and minimize!

```
from mrsimulator.utils import spectral_fitting as sf
params = sf.make_LMFIT_params(sim, processor)

from lmfit import Minimizer
minner = Minimizer(sf.LMFIT_min_function, params, fcn_args=(sim, processor, sigma))
result = minner.minimize()
```

Three steps in least-squares calculations with mrsimulator + LMFit

Last: Create LMFit parameters from Simulator and SignalProcessor, and minimize!

```
from mrsimulator.utils import spectral_fitting as sf
params = sf.make_LMFIT_params(sim, processor)

from lmfit import Minimizer
minner = Minimizer(sf.LMFIT_min_function, params, fcn_args=(sim, processor, sigma))
result = minner.minimize()
```

With mrsimulator+LMFit you can perform a simultaneous fit of spectra from different methods
for a single set of spin system parameters.

mrsimulator:docs v0.6.0

previous next index

Fitting Examples (Least Squares)

The `mrsimulator` library is easily integrable with other python-based libraries. In the following examples, we illustrate the use of LMFIT non-linear least-squares minimization python package to fit a simulation object to experimental data.

1D Data Fitting

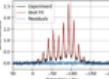


Figure 58
²⁹Si 1D MAS spinning sideband (CSA)

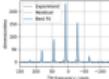


Figure 59
³¹P MAS NMR of crystalline Na₂PO₄ (CSA)

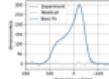


Figure 60
³¹P static NMR of crystalline Na₂PO₄ (CSA)

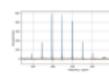


Figure 61
¹³C MAS NMR of Glycine (CSA) [1940 Hz]

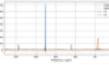


Figure 62
¹³C MAS NMR of Glycine (CSA) [5000 Hz]



Figure 63
¹³C MAS NMR of Glycine (CSA) [980 Hz]

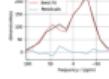


Figure 64
1D PASS/MAT sideband order cross-section

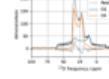


Figure 65
¹⁷O MAS NMR of crystalline Na₂SiO₃ (2nd order quad)

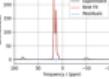


Figure 66
¹¹B MAS NMR of Lithium orthoborate crystal

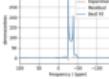


Figure 67
²³Na MAS NMR of Nasicon

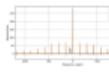


Figure 68
²⁷Al MAS NMR of YAG (1st and 2nd order Quad)

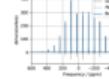


Figure 69
²H MAS NMR of Met nine

v. stable ▾

Solve linear-inversion problems with mrsimulator + mrinversion+ scikit-learn

<https://scikit-learn.org>

See recording of Deepansh Srivastava's talk
at the Global NMR Discussion Meetings
Feb 16, 2021.



Library available from pip

pip install mrinversion

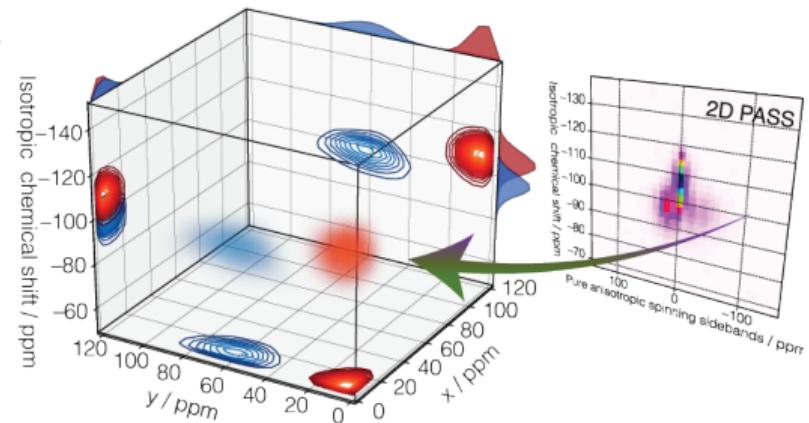
Available on Readthedocs

<https://mrinversion.readthedocs.io>



mrinversion available on Github

<https://github.com/DeepanshS/mrinversion>



D. Srivastava and P.J. Grandinetti, *J. Chem. Phys.*, **153**, 134201 (2020)

Progressive web app with mrsimulator + plotly dash

<https://plotly.com>

Available on Heroku

<https://mrsimulator.herokuapp.com>

mrsimulator
web app



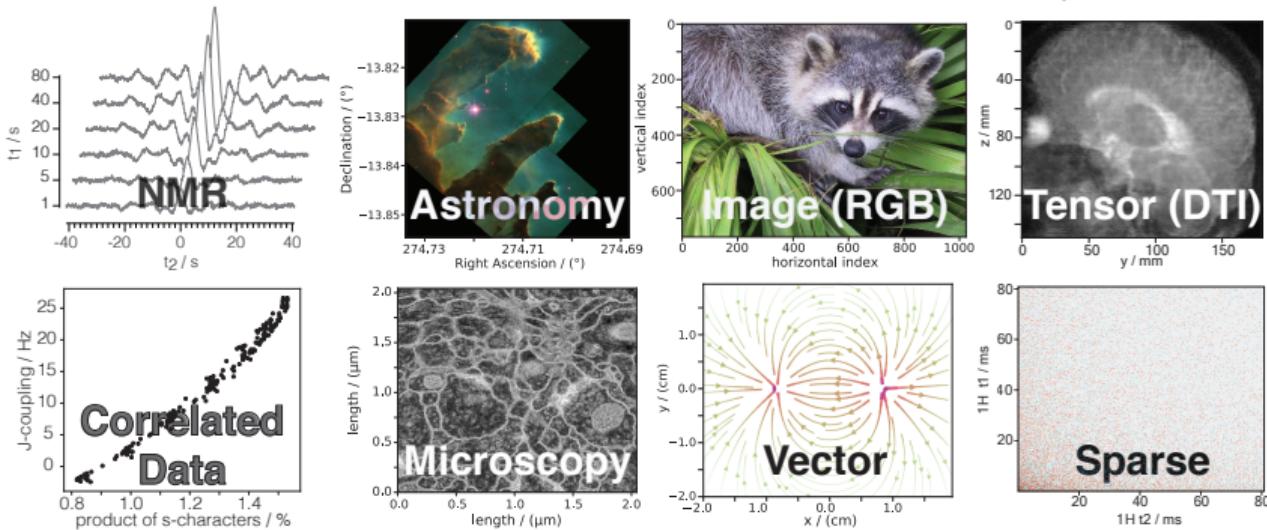
Compatible with Desktop and mobile devices

Use mrsimulator + LMFit without learning Python

mrsimulator uses the Core Scientific Dataset Model

PLOS ONE, 15(1): e0225953 (2020), Srivastava, Vosegaard, Massiot, and Grandinetti

- encodes a wide variety of multi-dimensional and correlated datasets (not just NMR).



- encapsulates data and minimum metadata needed to represent data in coordinate system.
- is human readable.
- can be a re-usable building block in a hierarchical description of more sophisticated portable scientific dataset file standards.

Need a FAIR dataset file format? Consider the Core Scientific Dataset Model
Flexible, modern, and open source file format for multi-dimensional datasets

CSDM compliant NMR software and packages

Mrsimulator • DMFit • Simpson • easyNMR
RMN • Mrinversion • NMRGlue • csdmpy

PLOS ONE, 15(1): e0225953 (2020),
Srivastava, Vosegaard, Massiot, and Grandinetti

csdmpy

A python library for
Core Scientific Dataset Model

Library available from pip
pip install csdmpy

Available on Readthedocs
csdmpy.readthedocs.io

Source code available on Github
github.com/DeepanshS/csdmpy

Summary: Fast NMR spectra simulation in Python with **mrsimulator**

Easy Install

Library available from pip

```
pip install mrsimulator
```

Fully documented with Examples

Available on Readthedocs

mrsimulator.readthedocs.io



mrsimulator

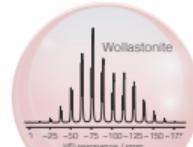
Open Source And Free

Source code available on Github

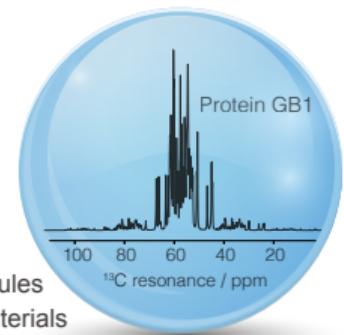
github.com/DeepanshS/mrsimulator



Small molecules
Crystalline materials



Macro molecules
Amorphous Materials



Email: grandinetti.1@osu.edu

<https://www.grandinetti.org/Software>