NMR Maser Simulations with **PyOR** (A versatile NMR Simulator)

Vineeth Francis Thalakottoor Jose Chacko

Winter Chemistry Day, ENS Paris

16th December 2024

PyOR (Python On Resonance)

*Motto: **"Everybody can simulate NMR"***

PyOR is made for simulating NMR Maser and learning NMR easy for beginners. (M/Raser, Liquid State NMR, Solid State NMR (future), EPR (future))

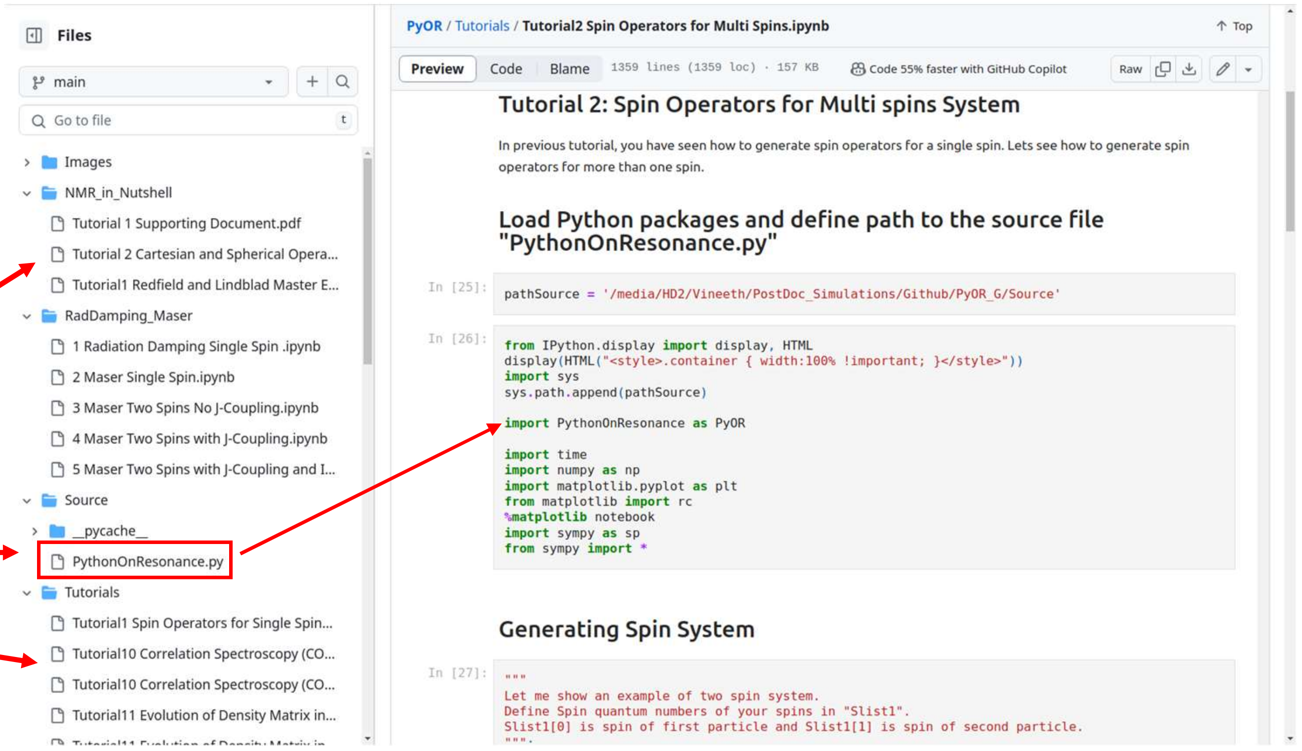
- **Python based NMR Simulator**
- **Why Python**? It's free, easy to learn, a lot packages, great online support.
- **What makes PyOR unique?** Great readability of the source code unlike other popular simulators available.
- **Versatile:** From **basic to specialized** NMR experiments
- https://github.com/VThalakottoor/PyOR-Jeener-Beta

Main Features

- Generate **Spin Operators**, (S_x, S_y S_z, S_+ and S_-) for a system with any number of particles with any spin quantum number
- Generate **Product Operator Basis (Hilbert or Liouville Vector Space)** for arbitrary system (support Zeeman, +-z basis, cartesian for spin half)
- **Hamiltonians**: Zeeman (Lab and Rotating Frame), B1, J coupling and Dipolar Coupling
- Solve **Liouville-von Neumann Equation** in ***Hilbert Space or Liouville Space***
  - **Unitary Propagation (Dense or Sparse Matrix)**
  - **Solve ODEs**
- **Relaxation**
  - ***Redfield*** Master Equation (Phenomenological, Dipolar relaxation, Random Field Fluctuation)
  - ***Lindblad*** Master Equation (Dipolar relaxation)
- **Radiation Damping and NMR Masers** (Multi-mode and J Coupling)
  - **Removed in beta version, will present in PyOR vJeener**
- *And many other functions to learn NMR Spin Physics*

How to use PyOR?

- Install Python in your computer (I prefer **Anaconda Python Distribution**)
- Download PyOR **Source code**: PythonOnResonance.py
- **Jupyter Notebook**
- PyOR and tutorials are in the **Github**
  - version: **Jeener-Beta**
  - https://github.com/VThalakottoor/PyOR-Jeener-Beta
  - **Descriptive Tutorials**
  - **Source Code**
  - **Simulation Tutorials**
- ***Modify the source code according to your need***


Files
main
Go to file
Images
NMR_in_Nutshell
Tutorial 1 Supporting Document.pdf
Tutorial 2 Cartesian and Spherical Opera...
Tutorial1 Redfield and Lindblad Master E...
RadDamping_Maser
1 Radiation Damping Single Spin .ipynb
2 Maser Single Spin.ipynb
3 Maser Two Spins No J-Coupling.ipynb
4 Maser Two Spins with J-Coupling.ipynb
5 Maser Two Spins with J-Coupling and I...
Source
__pycache__
PythonOnResonance.py
Tutorials
Tutorial1 Spin Operators for Single Spin...
Tutorial10 Correlation Spectroscopy (CO...
Tutorial10 Correlation Spectroscopy (CO...
Tutorial11 Evolution of Density Matrix in...
PyOR / Tutorials / Tutorial2 Spin Operators for Multi Spins.ipynb
Top
Preview
Code
Blame
1359 lines (1359 loc) · 157 KB
Code 55% faster with GitHub Copilot
Raw

Tutorial 2: Spin Operators for Multi spins System

In previous tutorial, you have seen how to generate spin operators for a single spin. Lets see how to generate spin operators for more than one spin.

Load Python packages and define path to the source file "PythonOnResonance.py"

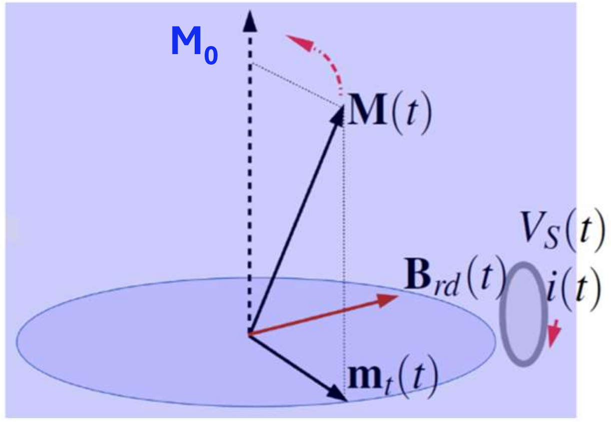
```
In [25]: pathSource = '/media/HD2/Vineeth/PostDoc_Simulations/Github/PyOR_G/Source'
```

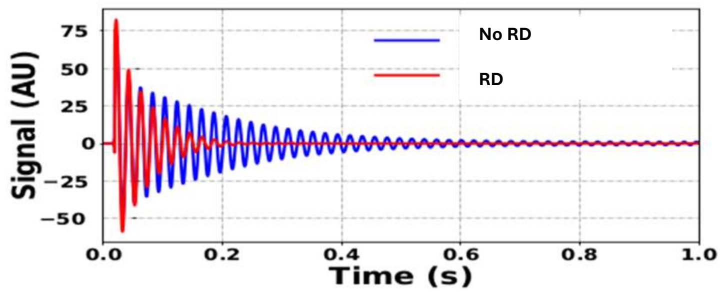
```
In [26]: from IPython.display import display, HTML
         display(HTML("<style>.container { width:100% !important; }</style>"))
         import sys
         sys.path.append(pathSource)

         import PythonOnResonance as PyOR

         import time
         import numpy as np
         import matplotlib.pyplot as plt
         from matplotlib import rc
         %matplotlib notebook
         import sympy as sp
         from sympy import *
```

Generating Spin System

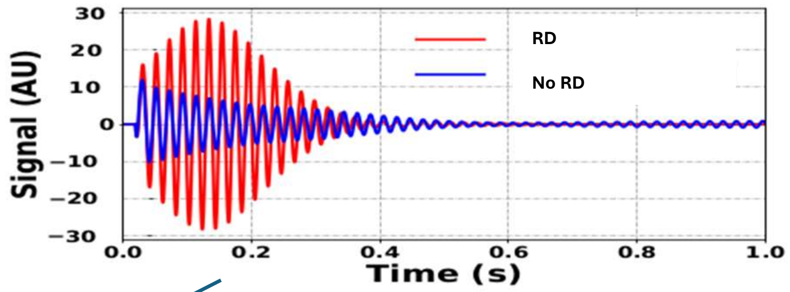
```
In [27]: """
         Let me show an example of two spin system.
         Define Spin quantum numbers of your spins in "Slist1".
         Slist1[0] is spin of first particle and Slist1[1] is spin of second particle.
```

Radiation Damping (RD): Coupling between large magnetization and resonant (detection) circuit





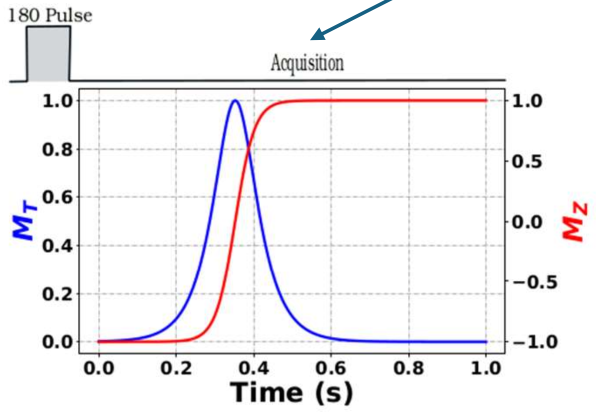
90^0 Pulse

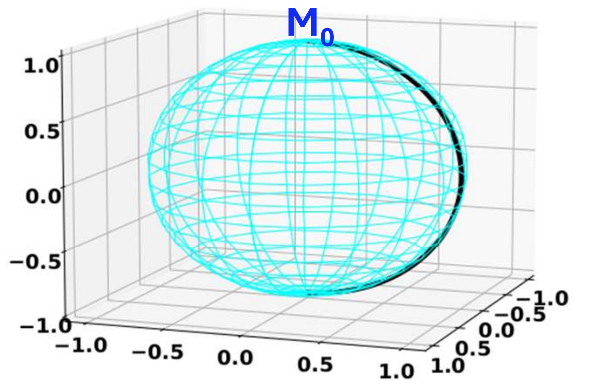


~180^0 Pulse

$$\vec{B}^{rot}_{+rd}(t) \approx j\left(\frac{\omega_{RF}\mu_o\eta L e^{-\Psi(\omega_o)}}{|Z(\omega_o)|V}\right)\int_V M^{rot}_{+}(\vec{r},t)dv' \boxed{\propto M_T}$$

RD field in rotating frame



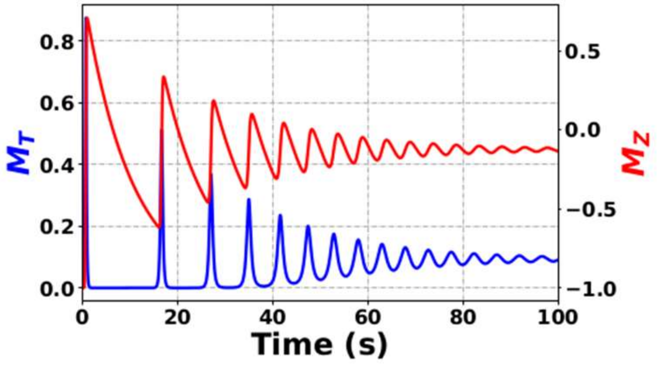


$$\frac{d}{dt}\vec{M}(\vec{r}_i) = \gamma\vec{M}(\vec{r}_i)\times(\Omega/\gamma) - \gamma_2(M_x(\vec{r}_i)\hat{x} + M_y(\vec{r}_i)\hat{y}) - \gamma_1(M_z(\vec{r}_i) - M_o)\hat{z}$$
$$+ \gamma\vec{M}(\vec{r}_i)\times\vec{B}_{rd} + \gamma\vec{M}(\vec{r}_i)\times\vec{B}_{dip}(\vec{r}_i)$$

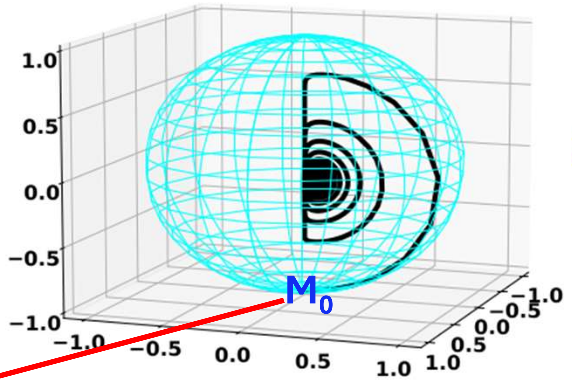
Bloch-Maxwell Equation

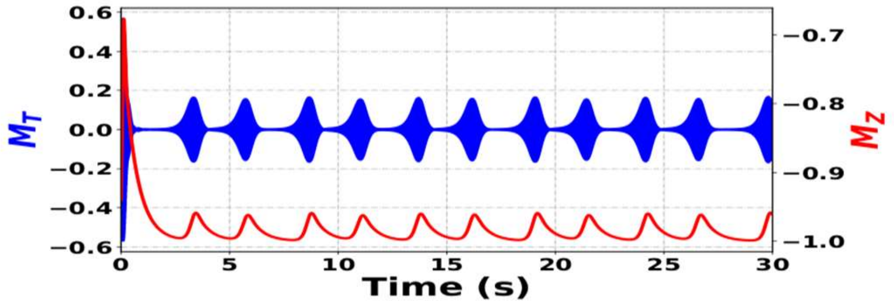
Maser: Long coherent RF signal

What if RD rotates magnetization away from equilibrium magnetization?



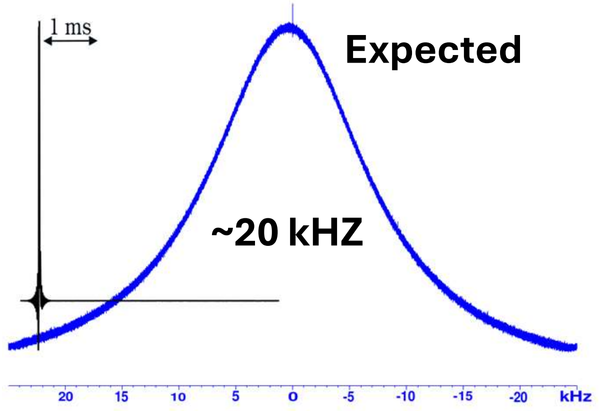
Single Spin

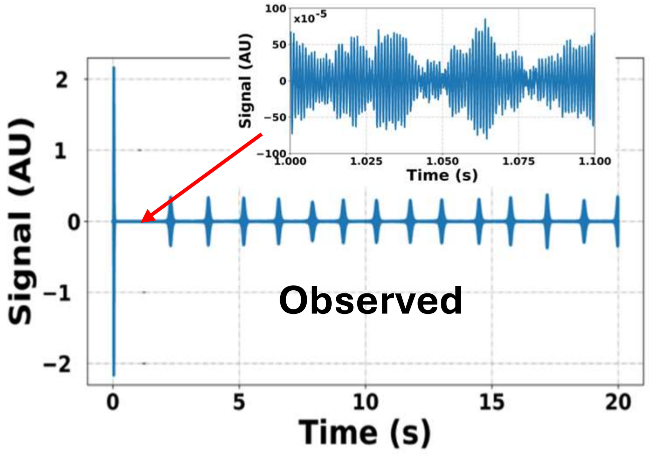


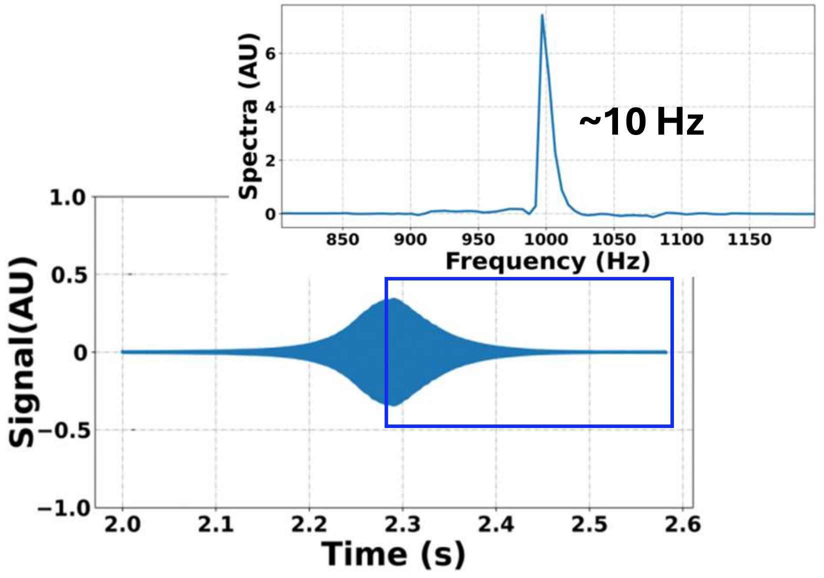


Spins with B_0 inhomogeneity

Maser from Negatively Hyperpolarized (DNP) 1H (Static Solid) @ 1.2 K



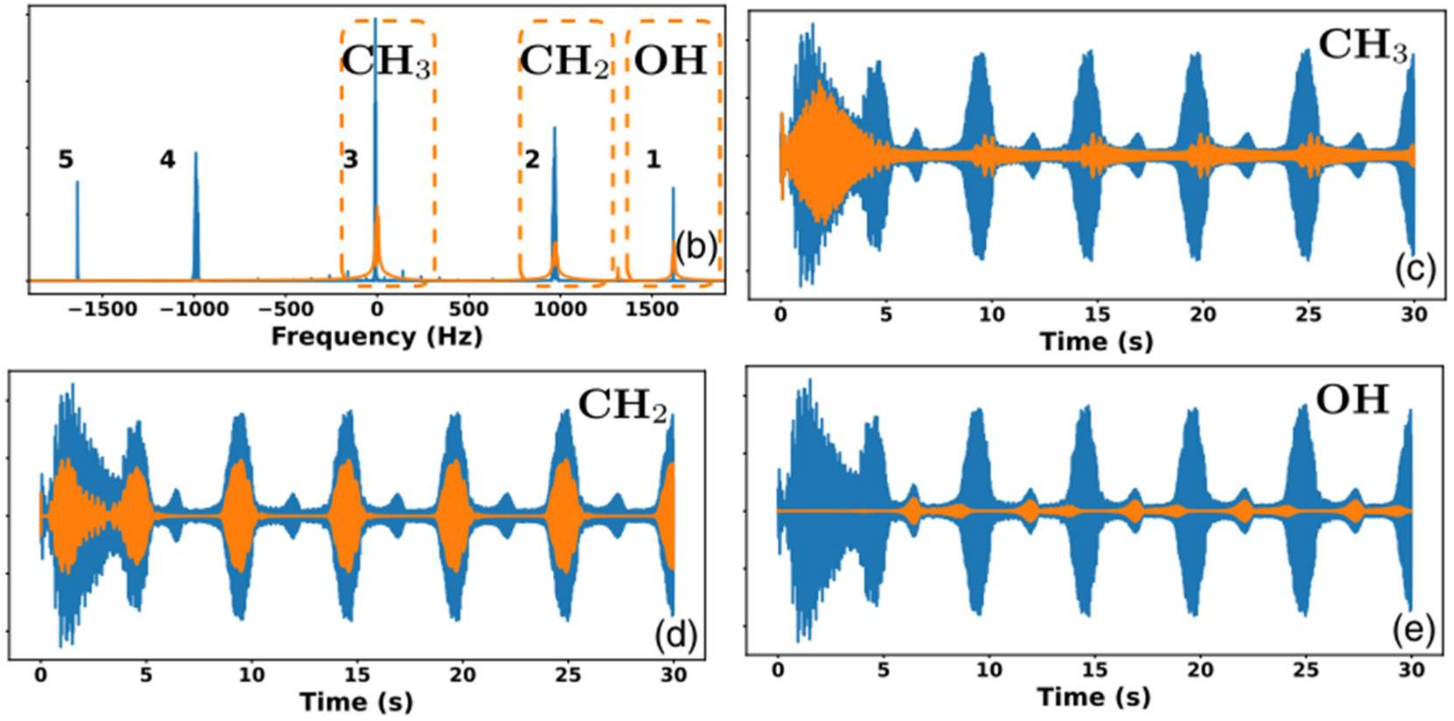


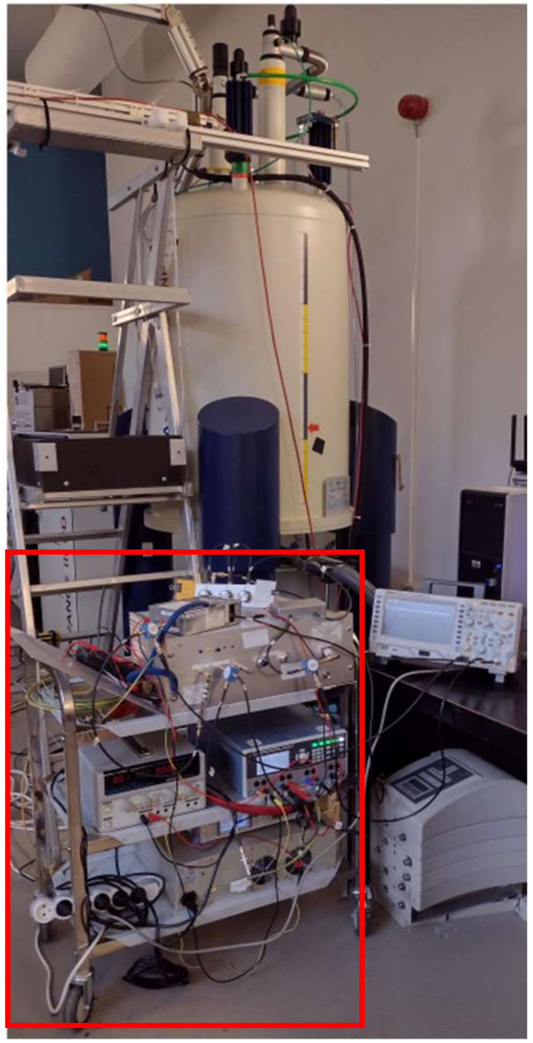


Vineeth Thalakottoor and Daniel Abergel, PCCP 2023, 25, 10392 (Hot Article 2023)

Sustained Maser with thermally polarized nuclear spins

Using an electronic feedback control unit for RD control



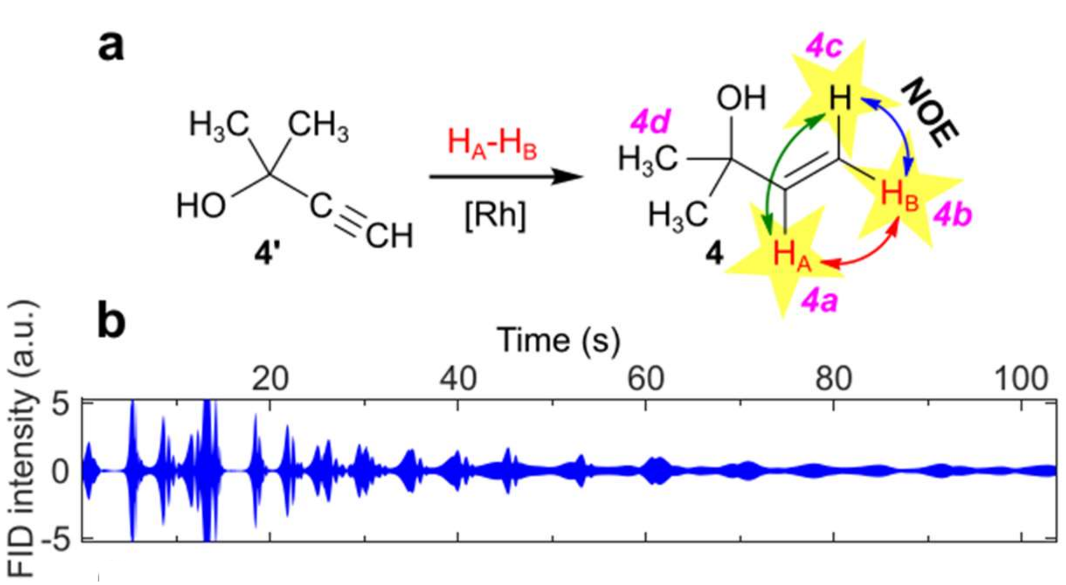


Multi-mode maser from resolved NMR lines of ethanol

Vineeth Thalakottoor, Alian Louis-Joseph and Daniel Abergel, PRL, 2024, 133, 158001

Maser Simulation using Liouville-von Neumann Equation

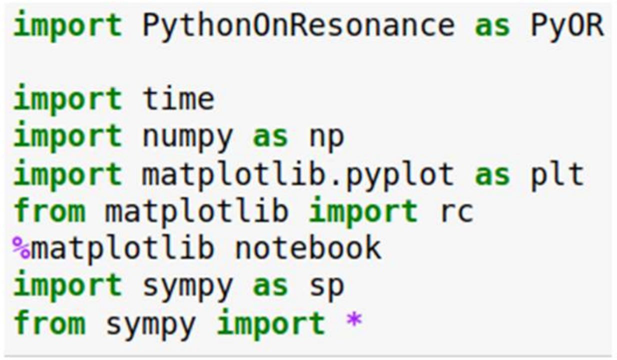
Bloch-Maxwell equation cannot address J coupling and DD relaxation



PHIP M(R)aser from thermally polarized 1H induced by cross-relaxation and J-Coupling

Ivan A. Trofimov, et.al., Communication Chemistry, 2024, 7, 235

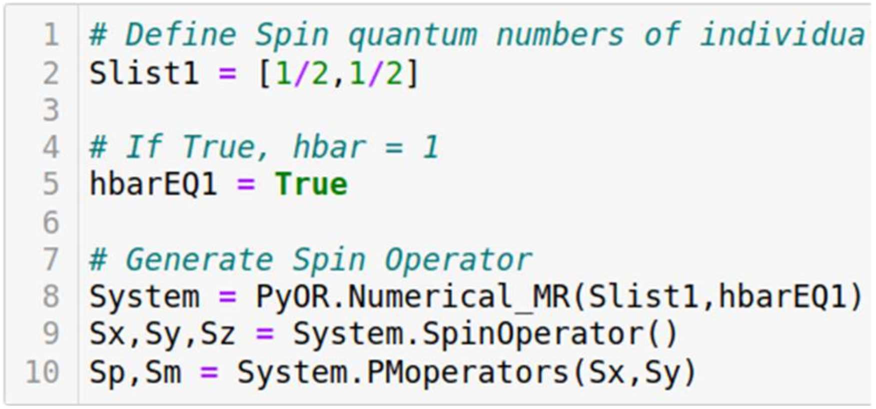
$$\frac{d\sigma(t)}{dt} = -i[\mathscr{H}_0, \sigma(t)] - \hat{\Gamma}(\sigma(t) - \sigma_0).$$



```
import PythonOnResonance as PyOR

import time
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
%matplotlib notebook
import sympy as sp
from sympy import *
```

$\boxed{1}$

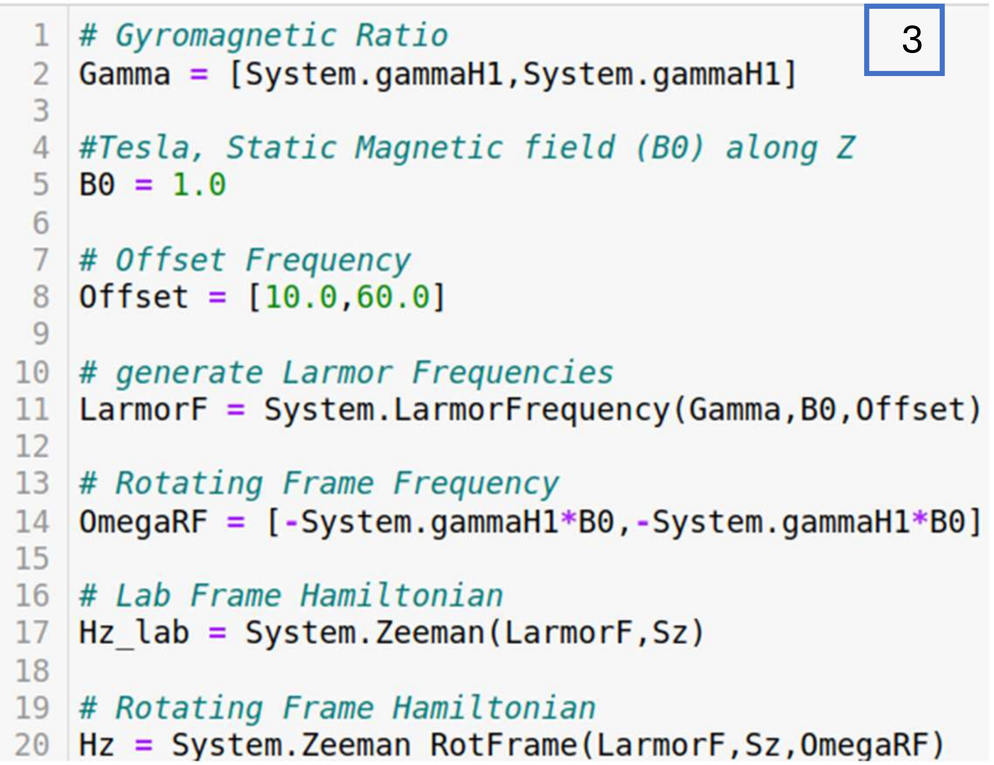
Generating Spin System



```
# Define Spin quantum numbers of individua
Slist1 = [1/2,1/2]

# If True, hbar = 1
hbarEQ1 = True

# Generate Spin Operator
System = PyOR.Numerical_MR(Slist1,hbarEQ1)
Sx,Sy,Sz = System.SpinOperator()
Sp,Sm = System.PMoperators(Sx,Sy)
```

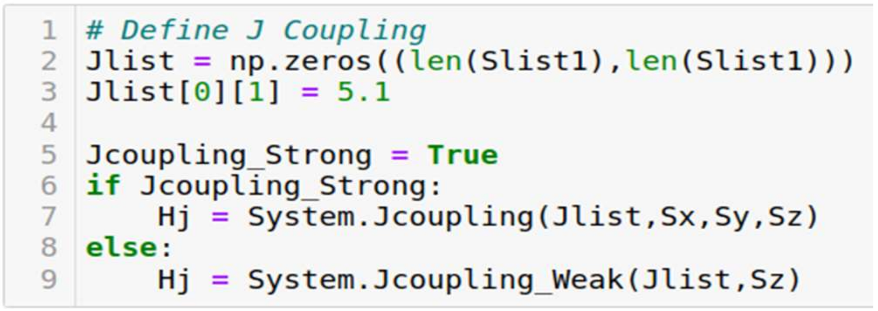
$\boxed{2}$

Zeeman Halitonian in Rotating Frame



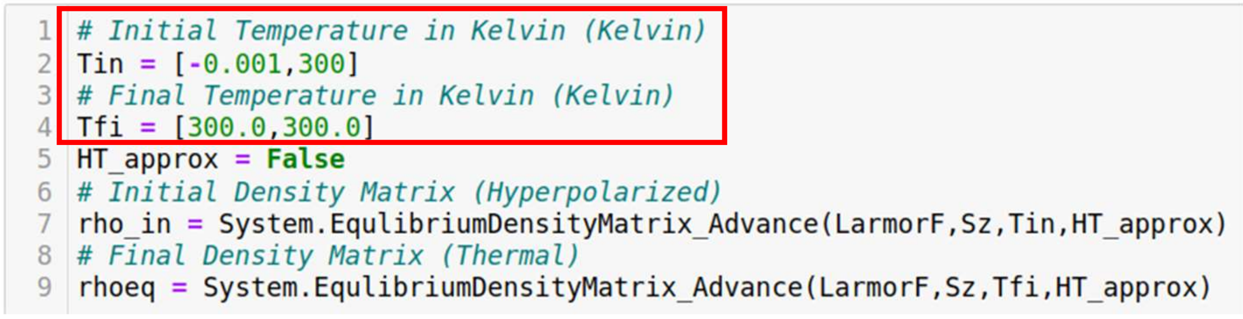
```
# Gyromagnetic Ratio
Gamma = [System.gammaH1,System.gammaH1]

#Tesla, Static Magnetic field (B0) along Z
B0 = 1.0

# Offset Frequency
Offset = [10.0,60.0]

# generate Larmor Frequencies
LarmorF = System.LarmorFrequency(Gamma,B0,Offset)

# Rotating Frame Frequency
OmegaRF = [-System.gammaH1*B0,-System.gammaH1*B0]

# Lab Frame Hamiltonian
Hz_lab = System.Zeeman(LarmorF,Sz)

# Rotating Frame Hamiltonian
Hz = System.Zeeman_RotFrame(LarmorF,Sz,OmegaRF)
```

J Coupling Hamiltonian



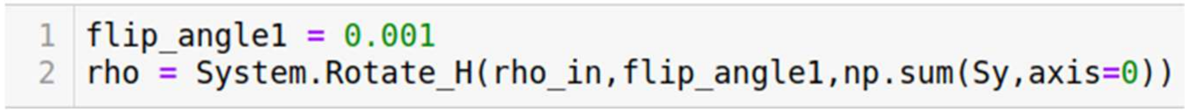
```
# Define J Coupling
Jlist = np.zeros((len(Slist1),len(Slist1)))
Jlist[0][1] = 5.1

Jcoupling_Strong = True
if Jcoupling_Strong:
    Hj = System.Jcoupling(Jlist,Sx,Sy,Sz)
else:
    Hj = System.Jcoupling_Weak(Jlist,Sz)
```

Initialize Density Matrix



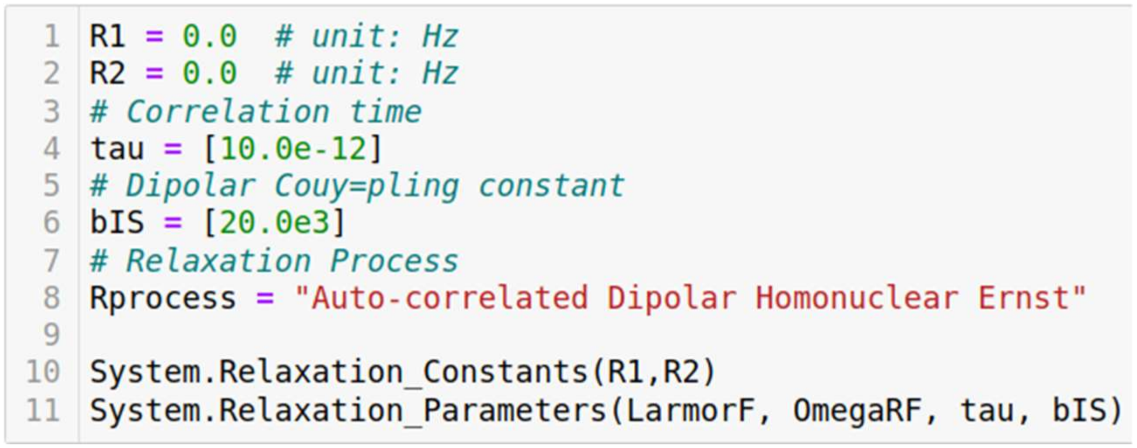
```
# Initial Temperature in Kelvin (Kelvin)
Tin = [-0.001,300]
# Final Temperature in Kelvin (Kelvin)
Tfi = [300.0,300.0]
HT_approx = False
# Initial Density Matrix (Hyperpolarized)
rho_in = System.EqulibriumDensityMatrix_Advance(LarmorF,Sz,Tin,HT_approx)
# Final Density Matrix (Thermal)
rhoeq = System.EqulibriumDensityMatrix_Advance(LarmorF,Sz,Tfi,HT_approx)
```

Initial Pulse



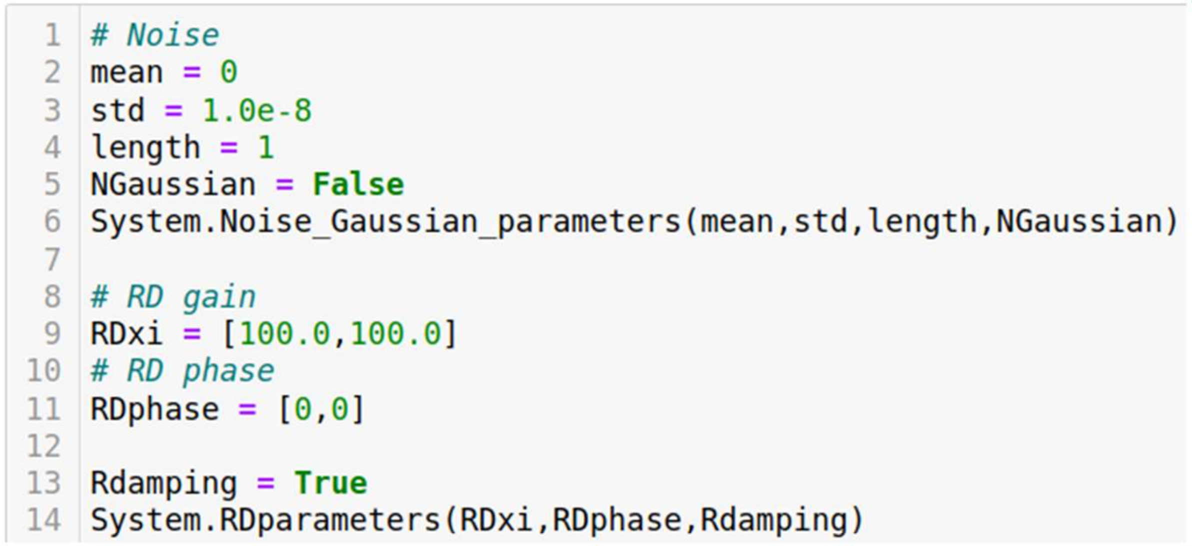
```
flip_angle1 = 0.001
rho = System.Rotate_H(rho_in,flip_angle1,np.sum(Sy,axis=0))
```

Relaxation



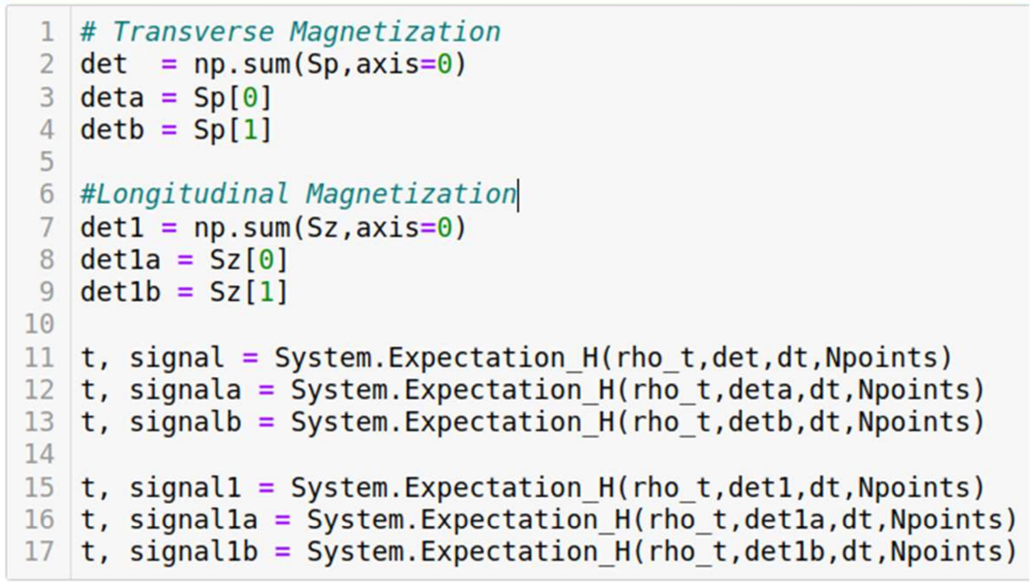
```
R1 = 0.0  # unit: Hz
R2 = 0.0  # unit: Hz
# Correlation time
tau = [10.0e-12]
# Dipolar Couy=pling constant
bIS = [20.0e3]
# Relaxation Process
Rprocess = "Auto-correlated Dipolar Homonuclear Ernst"

System.Relaxation_Constants(R1,R2)
System.Relaxation_Parameters(LarmorF, OmegaRF, tau, bIS)
```

Radiation Damping Constant 8



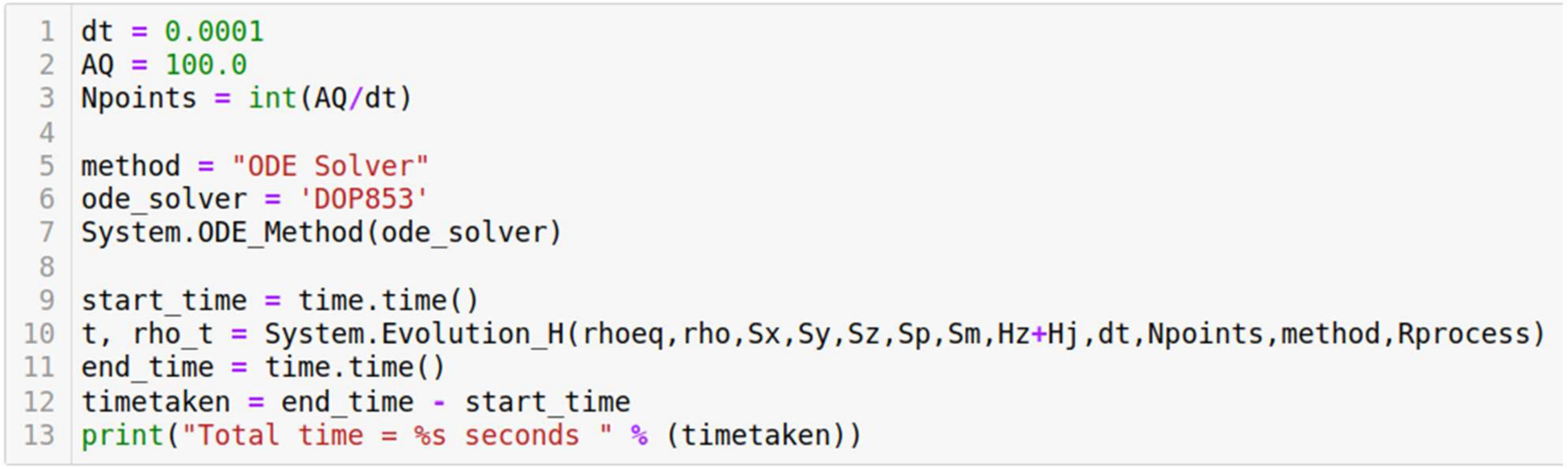
```python
# Noise
mean = 0
std = 1.0e-8
length = 1
NGaussian = False
System.Noise_Gaussian_parameters(mean,std,length,NGaussian)

# RD gain
RDxi = [100.0,100.0]
# RD phase
RDphase = [0,0]

Rdamping = True
System.RDparameters(RDxi,RDphase,Rdamping)
```

Expectation Value 10



```python
# Transverse Magnetization
det  = np.sum(Sp,axis=0)
deta = Sp[0]
detb = Sp[1]

#Longitudinal Magnetization
det1 = np.sum(Sz,axis=0)
det1a = Sz[0]
det1b = Sz[1]

t, signal = System.Expectation_H(rho_t,det,dt,Npoints)
t, signala = System.Expectation_H(rho_t,deta,dt,Npoints)
t, signalb = System.Expectation_H(rho_t,detb,dt,Npoints)

t, signal1 = System.Expectation_H(rho_t,det1,dt,Npoints)
t, signal1a = System.Expectation_H(rho_t,det1a,dt,Npoints)
t, signal1b = System.Expectation_H(rho_t,det1b,dt,Npoints)
```

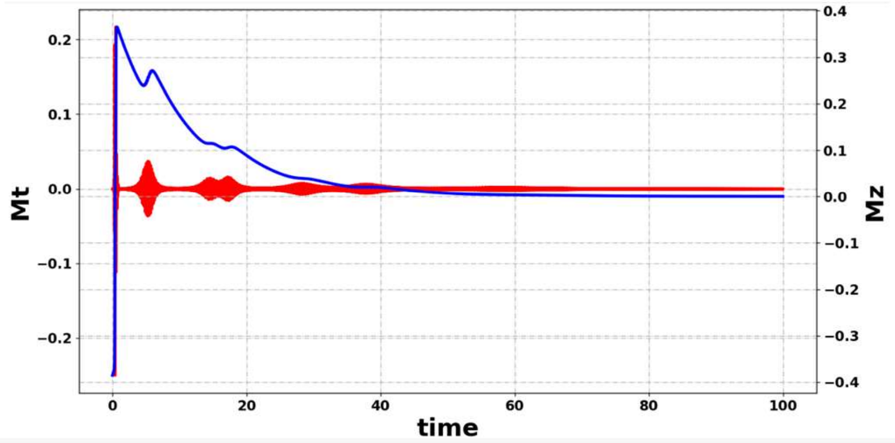
Evolution (Need ODE Solver) 9



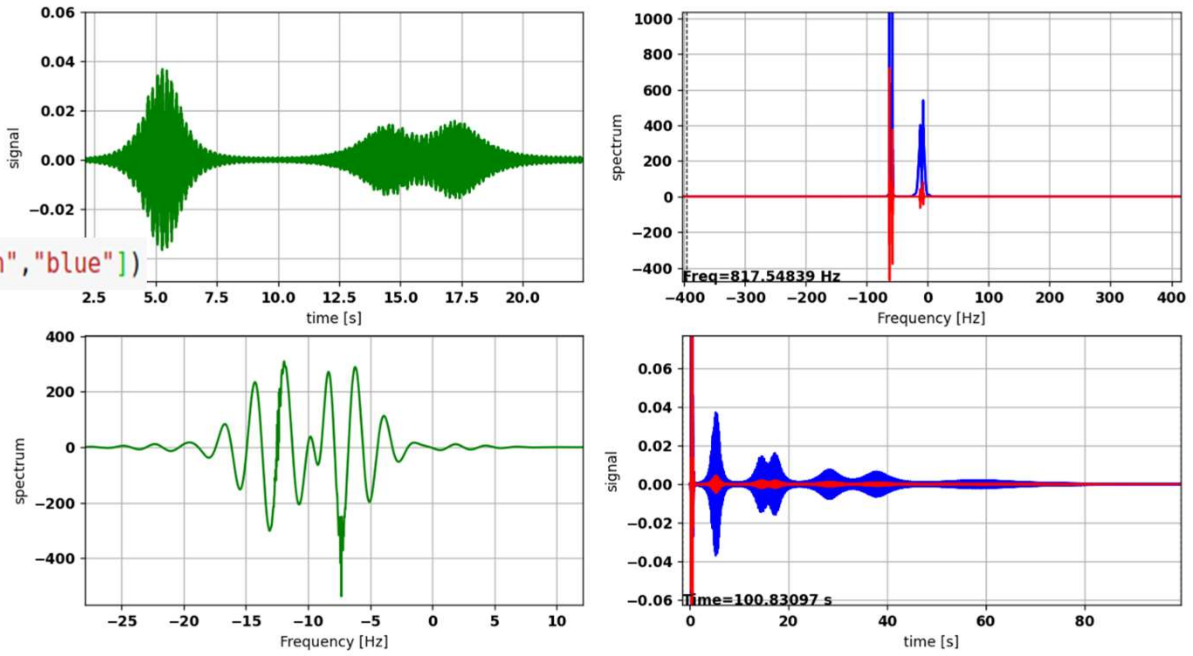
```python
dt = 0.0001
AQ = 100.0
Npoints = int(AQ/dt)

method = "ODE Solver"
ode_solver = 'DOP853'
System.ODE_Method(ode_solver)

start_time = time.time()
t, rho_t = System.Evolution_H(rhoeq,rho,Sx,Sy,Sz,Sp,Sm,Hz+Hj,dt,Npoints,method,Rprocess)
end_time = time.time()
timetaken = end_time - start_time
print("Total time = %s seconds " % (timetaken))
```

Maser from 1H (thermal) due to cross-relaxation and J coupling

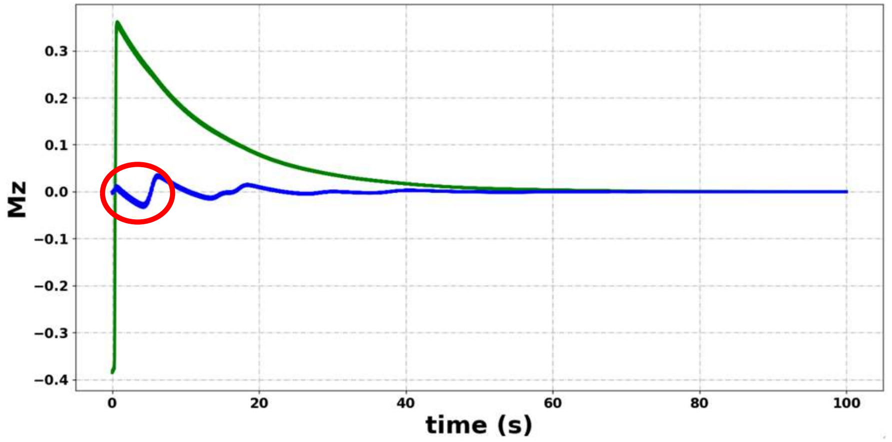
```
System.PlottingTwin(8,t,signal,signal1,'time','Mt','Mz',"red","blue")
```



```
fig, fourier = System.PlottingMultimodeAnalyzer(t,freq,signal,spectrum)
```



```
System.PlottingMulti(6,[t,t],[signal1a,signal1b],"time (s)","Mz",["green","blue"])
```



Multi-mode Analyzer

Acknowledgement

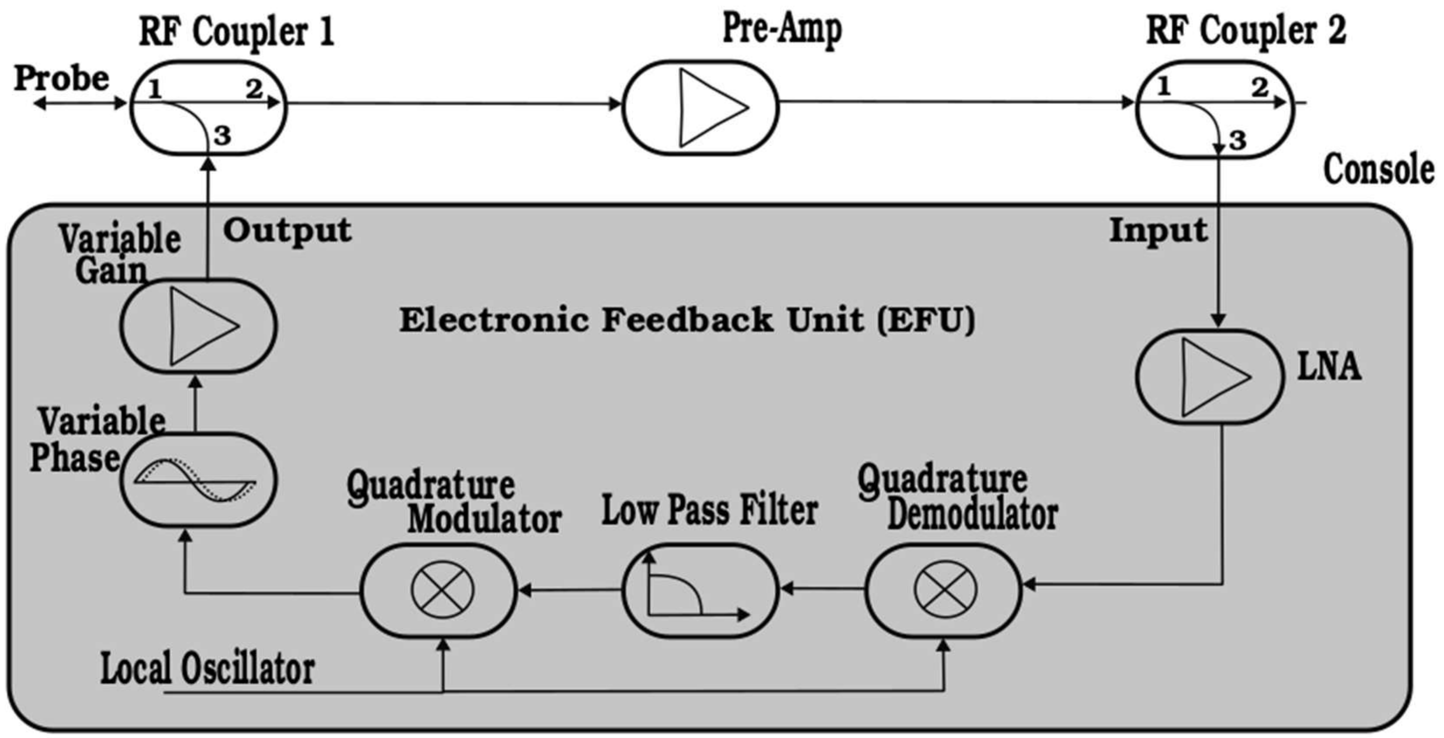
Daniel Abergel and ANR (ANR-22-CE29-0006-01–DynNonlinPol)

PyOR is all yours

"Let what is created surpass the creator"

Thank You

Electronic Feedback Unit


RF Coupler 1
Pre-Amp
RF Coupler 2
Probe
1
2
3
1
2
3
Console
Variable Gain
Output
Input
Electronic Feedback Unit (EFU)
LNA
Variable Phase
Quadrature Modulator
Low Pass Filter
Quadrature Demodulator
Local Oscillator