NMR Maser: Long Coherent RF signal

PHYSICAL REVIEW LETTERS **133,** 158001 (2024)

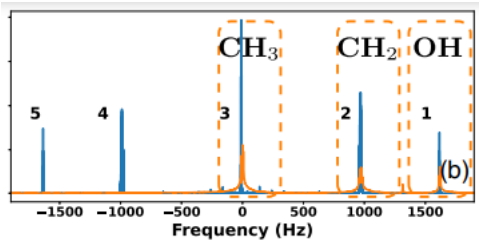
Multimode Masers of Thermally Polarized Nuclear Spins in Solution NMR

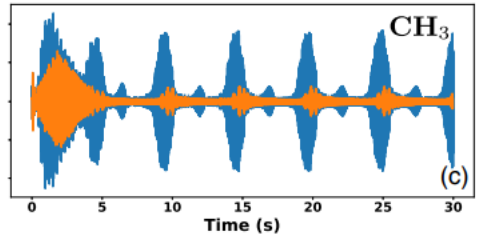
Vineeth Francis Thalakottoor Jose Chacko,[1] Alain Louis-Joseph,[2] and Daniel Abergel[1,*]
[1]*Laboratoire des Biomolécules, LBM, Département de Chimie, Ecole Normale Supérieure, PSL University, Sorbonne Université, CNRS, 75005 Paris, France*
[2]*Laboratoire de Physique de la Matière Condensée, UMR 7643, CNRS, École Polytechnique, IPP 91120 Palaiseau, France*

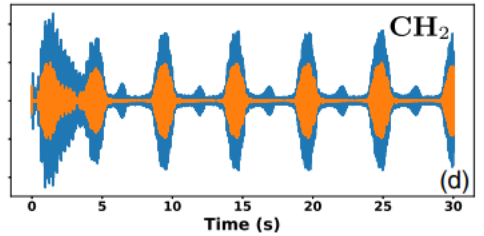
(Received 16 April 2024; revised 5 August 2024; accepted 5 September 2024; published 10 October 2024)

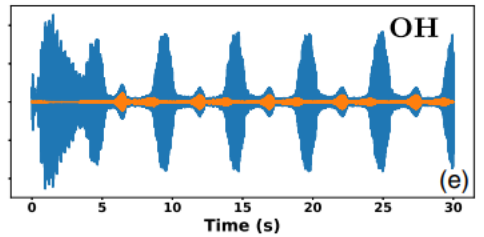
We present experimental single and multimode sustained ^{1}H NMR masers in solution on thermally polarized spins at room temperature and 9.4 T achieved through the electronic control of radiation feedback (radiation damping). Our observations illustrate the breakdown of the usual three-dimensional Maxwell-Bloch equations for radiation feedback and a simple toy model of few coupled classical moments is used to interpret these experiments. This Letter represents a significant step to bring the spontaneous radiation damping based NMR masers in various contexts to the next stage of feedback-controlled and reproducible experiments.

DOI: 10.1103/PhysRevLett.133.158001









PyOR: A versatile NMR Simulator

Vineeth Francis Thalakottoor Jose Chacko
LBM, ENS, Paris

PyOR (Python On Resonance)

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

PyOR *is made for learning NMR easy for beginners even from chemistry or biology background.*

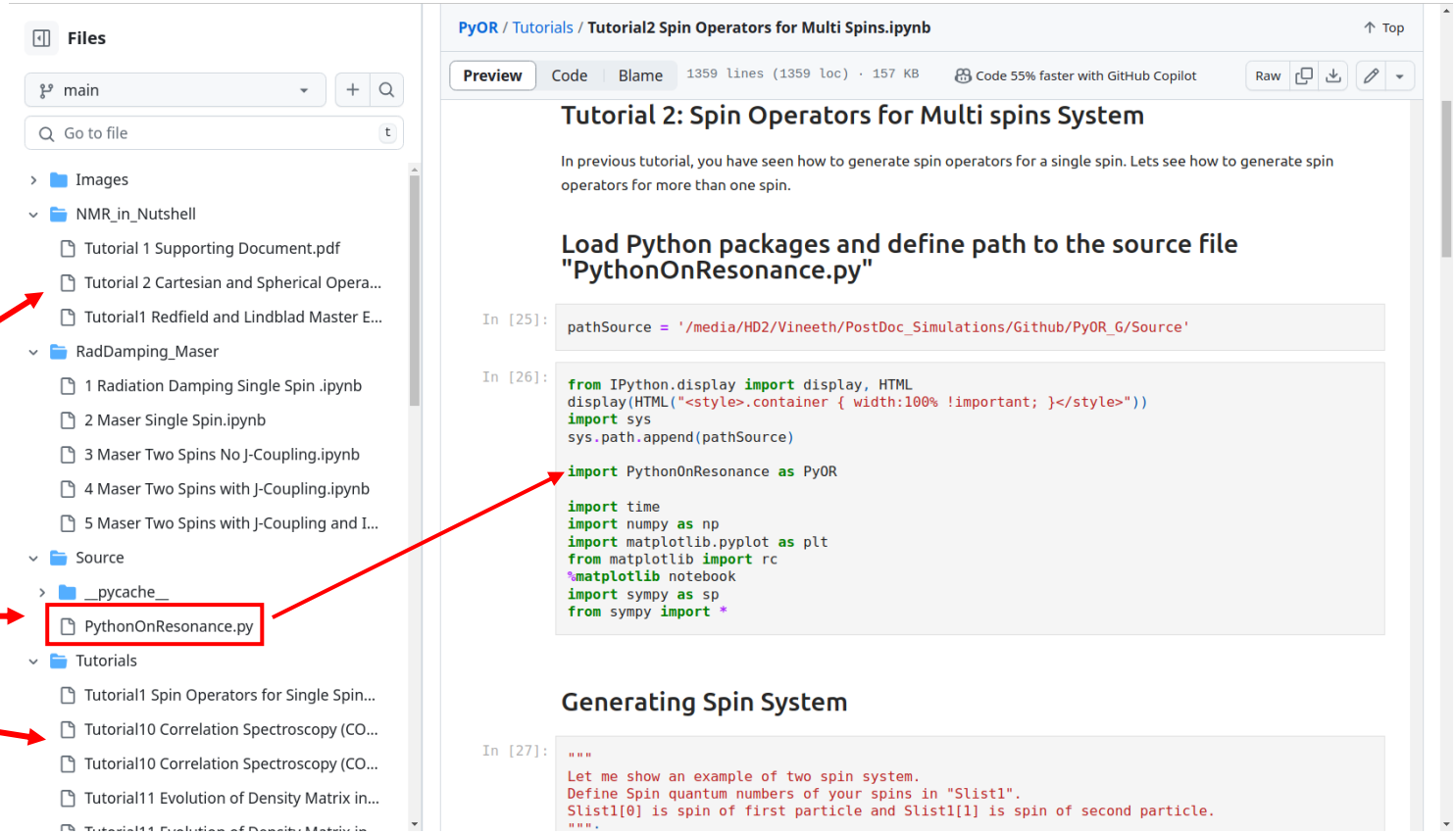
- **Python based NMR Simulator**
- **Why Python**? It's free, easy to learn, a lot packages, great online support.
- **What makes PyOR unique?** Great readability of the source code unlike other popular simulators available.
- **Versatile**: From **basic to specialized** NMR experiments
- https://github.com/VThalakottoor/PyOR-Jeener-Beta

Main Features

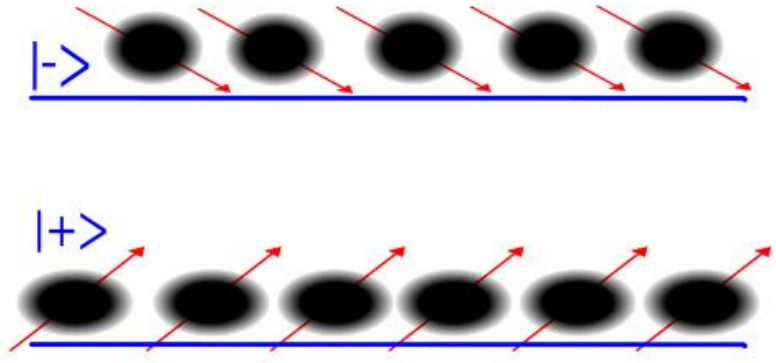
- Generate **Spin Operators**, (S_x, S_y S_z, S_+ and S_-) for a system with any number of particles with any spin quantum number
- **Hamiltonians**: Zeeman (Lab and Rotating Frame), B1, J coupling and Dipolar Coupling
- Solve **Liouville Equation** in ***Hilbert Space or Liouville Space***
  - **Unitary 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
- Many other functions to learn NMR Spin Physics

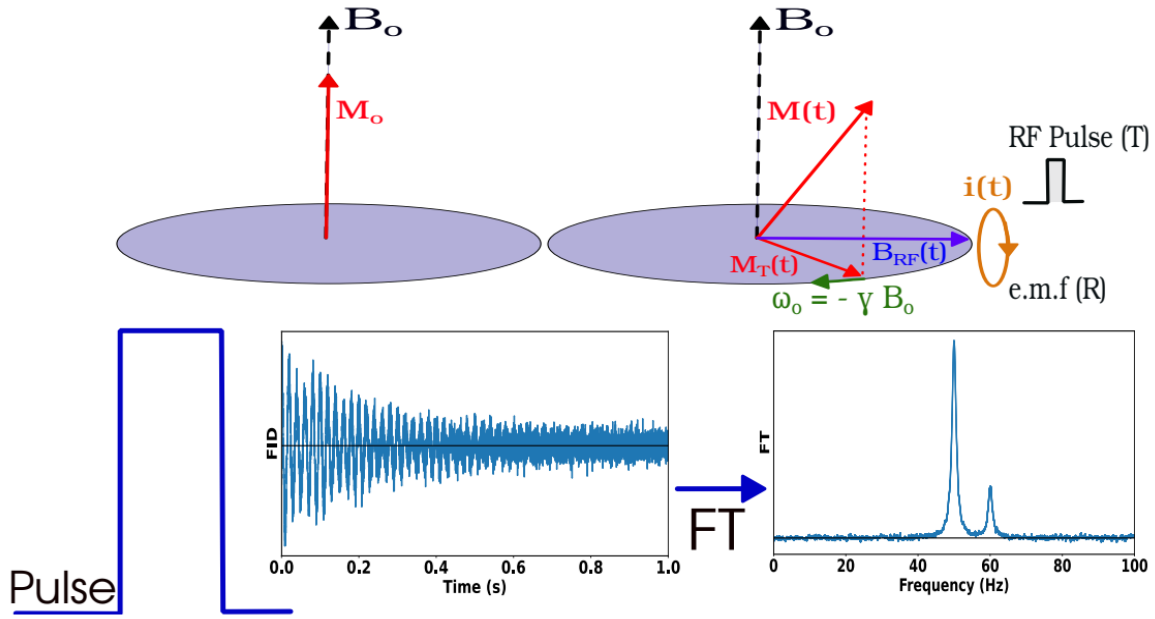
How to use PyOR?

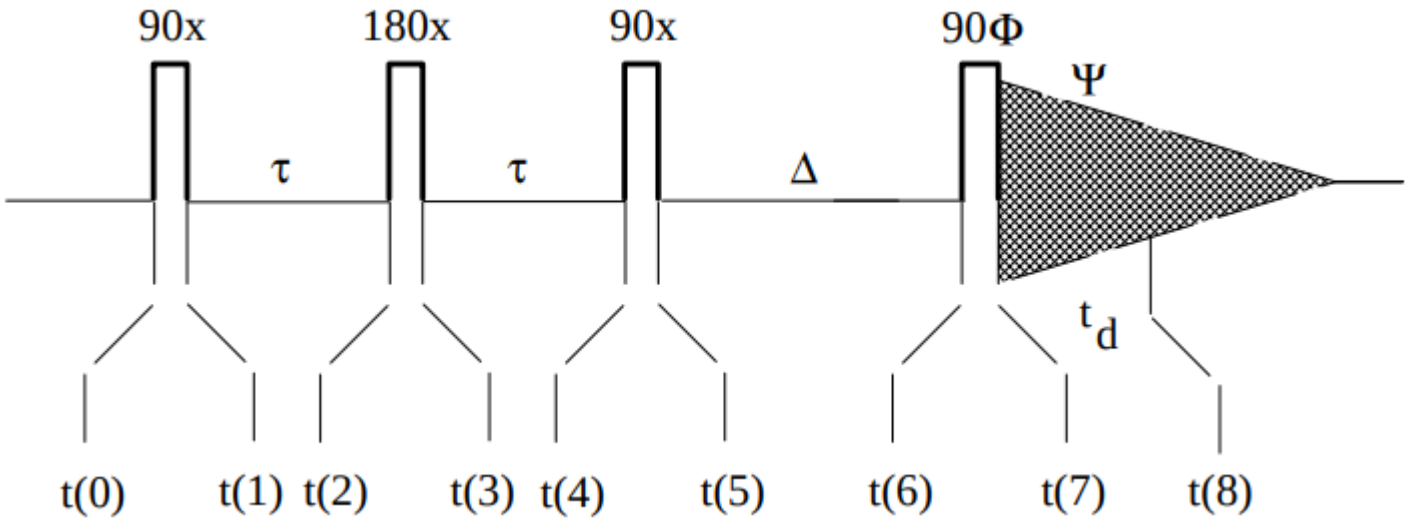
- Install Python in your computer (I prefer **Anaconda Python Distribution**)
- PyOR **Source code**: PythonOnResonance.py
- **Jupyter Notebook**
- PyOR and tutorials are in the **Github**
  - version: **Jeener-Beta**
  - https://github.com/VThalakottoor/PyOR-Jeener-Beta
  - **Descriptive Tutorials**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



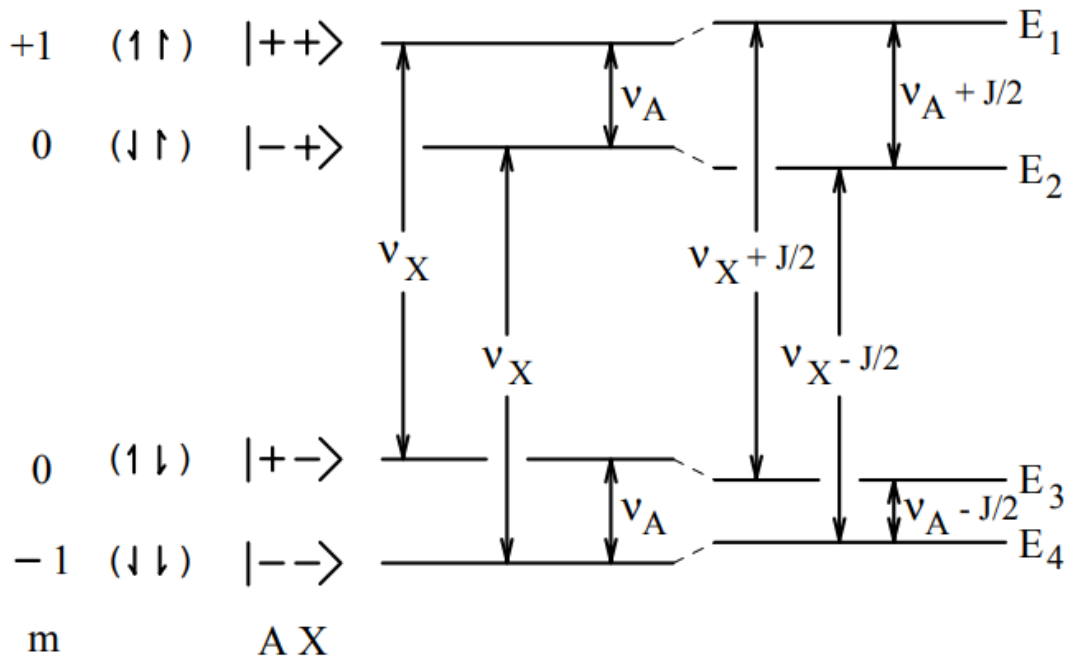
NMR Spectroscopy







Density Matrix (ρ): State of system



$\|AX\rangle$	$\|++\rangle$	$\|-+\rangle$	$\|+-\rangle$	$\|--\rangle$
$\|++\rangle$	P_1	$1Q_A$	$1Q_X$	$2Q_{AX}$
$\|-+\rangle$		P_2	ZQ_{AX}	$1Q_X$
$\|+-\rangle$			P_3	$1Q_A$
$\|--\rangle$				P_4

Expectation value of an observable, ***A*:** (***A* = Sx** or **Sy** or **Sz or ...**)

$$\langle A \rangle_\rho = \mathrm{Tr}(\rho A)$$

Liouville-von Neumann Equation: *Evolution of Density Matrix*

Hilbert Space

Liouville Space

Density Matrix

$$\rho = \begin{bmatrix} \rho_{11} & \rho_{12} & \rho_{13} & \rho_{14} \\ \rho_{21} & \rho_{22} & \rho_{23} & \rho_{24} \\ \rho_{31} & \rho_{32} & \rho_{33} & \rho_{34} \\ \rho_{41} & \rho_{42} & \rho_{43} & \rho_{44} \end{bmatrix} \qquad \tilde{\rho} = \begin{bmatrix} \rho_{11} \\ \rho_{12} \\ \rho_{13} \\ \rho_{14} \\ \rho_{21} \\ \vdots \\ \rho_{44} \end{bmatrix}$$

$$\frac{d}{dt}\tilde{\rho} = \frac{-i}{\hbar}\hat{H}_0\tilde{\rho}$$

$$\frac{d}{dt}\rho = \frac{-i}{\hbar}[H_0, \rho]$$

Solve ODEs

$$\frac{d}{dt}\tilde{\rho} = \frac{-i}{\hbar}(H_0 \otimes \mathbb{1} - \mathbb{1} \otimes H_0^T)\tilde{\rho}$$

$$\rho(t) = e^{iH_0t}\rho(0)e^{-iH_0t}$$

Exponential Propagation

$$\tilde{\rho}(t) = e^{-i\hat{H}_0t}\tilde{\rho}(0)$$

Define Spin System and Generate Spin Operators

- For single spin half system
  - Spin_list = [1/2]
- For single spin one system
  - Spin_list = [1]
- For two spin half system
  - Spin_list = [1/2, 1/2]
- For spin half and spin one system
  - Spin_list = [1/2, 1]

```
System = PyOR.Numerical_MR(Spin_list,hbarEQ1)

Sx,Sy,Sz = System.SpinOperator()
```

In [3]:
```
Spin_list = [1/2, 1/2]
```

In [6]:
```
Matrix(Sx[0])
```

Out[6]:

$$\begin{bmatrix} 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \end{bmatrix}$$

Documentation

Any PyOR functions

```
1 help(System.SpinOperator)
```

```
Help on method SpinOperator in module PythonOnResonance:

SpinOperator() method of PythonOnResonance.Numerical_MR instance
    Generate spin operators for all spins: Sx, Sy and Sz
    INPUT
    -----
    nill

    OUTPUT
    ------
    Sx : array [Sx of spin 1, Sx of spin 2, Sx of spin 3, ...]
    Sy : array [Sy of spin 1, Sy of spin 2, Sy of spin 3, ...]
    Sz : array [Sz of spin 1, Sz of spin 2, Sz of spin 3, ...]
```

Hamiltonian (Zeeman)

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

```
# B0 Field in Tesla, Static Magnetic field (B0) along Z
B0 = 9.4
```

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

```
# Offset Frequency in rotating frame (Hz)
Offset = [10.0,20.0]
```

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

```
Larmor Frequency in MHz:  [-400.22802765 -400.22803765]
```

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

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

Initialize Density Matrix

```
Thermal_DensMatrix = True

if Thermal_DensMatrix:
    # Spin temperature of individual spins (initial) Kelvin
    Tin = [300.0,300.0]

    # Spin temperature of individual spins (equlibrium) Kelvin
    Tfi = [300.0,300.0]

    # High Temperature
    HT_approx = False

    # Initial Density Matrix
    rho_in = System.EqulibriumDensityMatrix_Advance(LarmorF,Sz,Tin,HT_approx)

    # Equlibrium Density Matrix
    rhoeq = System.EqulibriumDensityMatrix_Advance(LarmorF,Sz,Tfi,HT_approx)
else:
    rho_in = np.sum(Sz,axis=0)
    rhoeq = np.sum(Sz,axis=0)
```

Generate Product Operators Basis (Hilbert Space)

$$\rho(t) = \sum_{k=1}^{K} b_k(t)\mathbf{B}_k$$

```
sort = 'negative to positive'
Index = False
Normal = True
Basis_PMZ, coh_PMZ, dic_PMZ = System.ProductOperators_SpinHalf_PMZ(sort,Index,Normal)
```

Call Product Operator with string index

```
OpB_H = System.String_to_Matrix(dic_PMZ, Basis_PMZ)
```

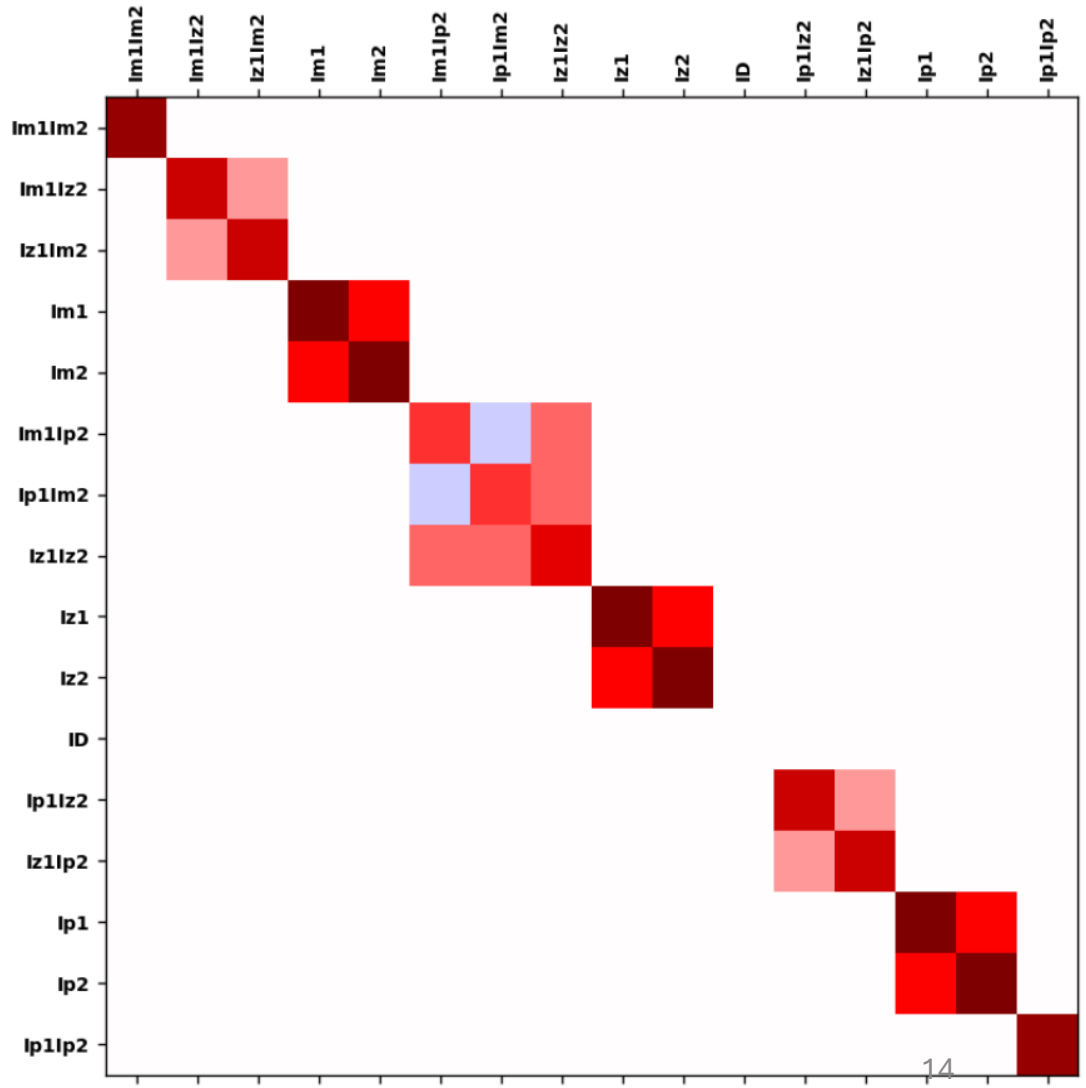
```
['Im1Im2', 'Im1Iz2', 'Im1', 'Iz1Im2', 'Im2', 'Im1Ip2', 'Iz1Iz2', 'Iz1', 'Iz2', '', 'Ip1Im
2', 'Iz1Ip2', 'Ip2', 'Ip1Iz2', 'Ip1', 'Ip1Ip2']
```

```
Matrix(OpB_H["Im1Im2"])
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 \end{bmatrix}$$

Relaxation (Liouville)

```
R = None
Rprocess = "Auto-correlated Dipolar Homonuclear"
tau = [10.0e-12]
bIS = [30.0e3]
System.Relaxation_Parameters(LarmorF, OmegaRF, tau, bIS)
R_L = System.Relaxation_L(Rprocess,R,Sx,Sy,Sz,Sp,Sm)
```



Evolve and Expectation Value

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

Define method to solve

```
start_time = time.time()
t, rho_t = System.Evolution_L(rhoeq_L,rho_L,Sx,Sy,Hz_L,R_L,dt,Npoints,method)
end_time = time.time()
timetaken = end_time - start_time
print("Total time = %s seconds " % (timetaken))
```

Evolution

```
(16, 500000)
Total time = 3.5411036014556885 seconds
```

```
LEXP_Z1 = System.Detection_L(det_Z1)
LEXP_Z2 = System.Detection_L(det_Z2)
```

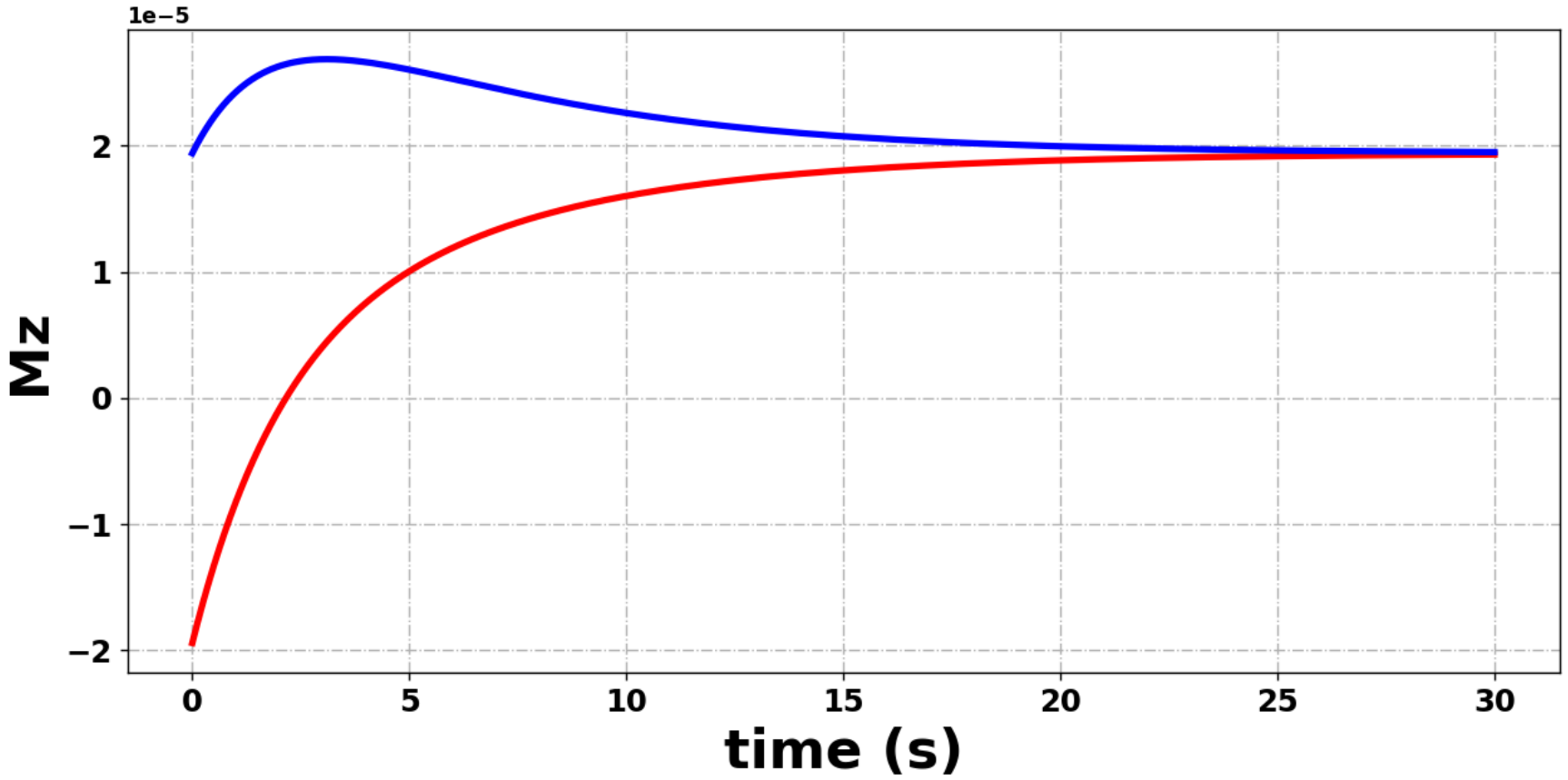
Observable

```
t, MZ_1 = System.Expectation_L(rho_t,LEXP_Z1,dt,Npoints)
t, MZ_2 = System.Expectation_L(rho_t,LEXP_Z2,dt,Npoints)
```

Expectation Value

Simulation Results Visualization:
Multiple Plots (NOE)

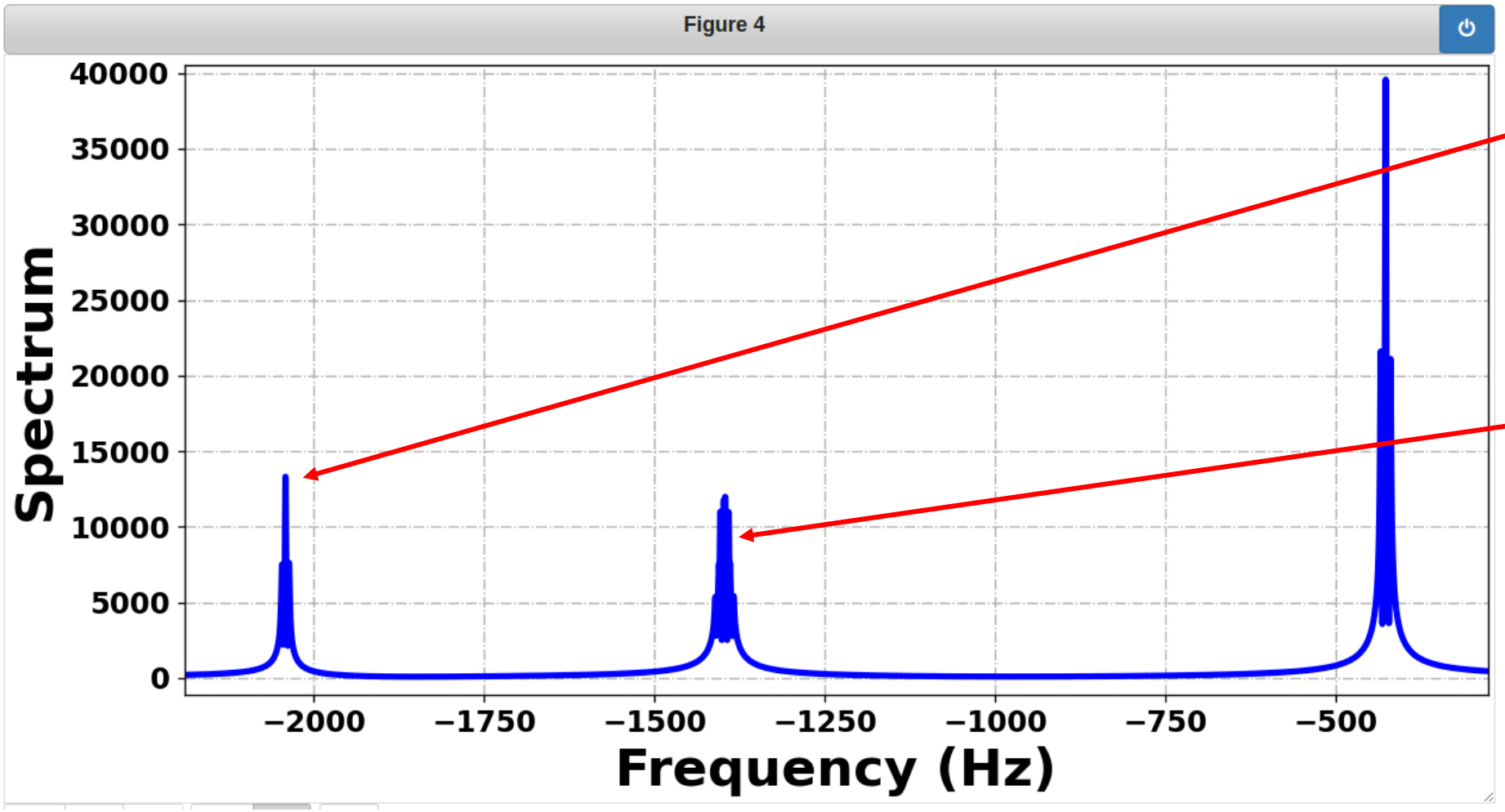
```
"""
Mz1: Red
Mz2: Blue
"""
System.PlottingMulti(4,[t,t],[Mz1,Mz2],"time (s)","Mz",["red","blue"])
```

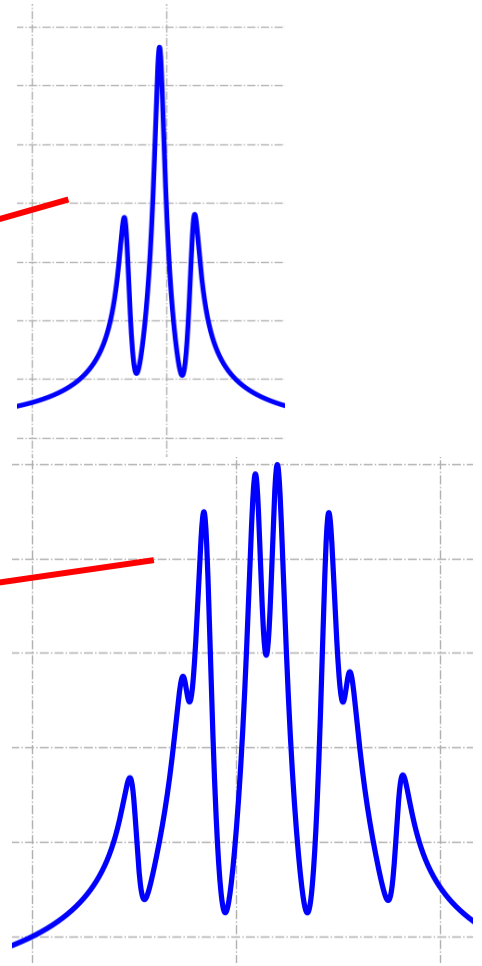


Simulation Results Visualization:

Plotting (Ethanol Spectra)

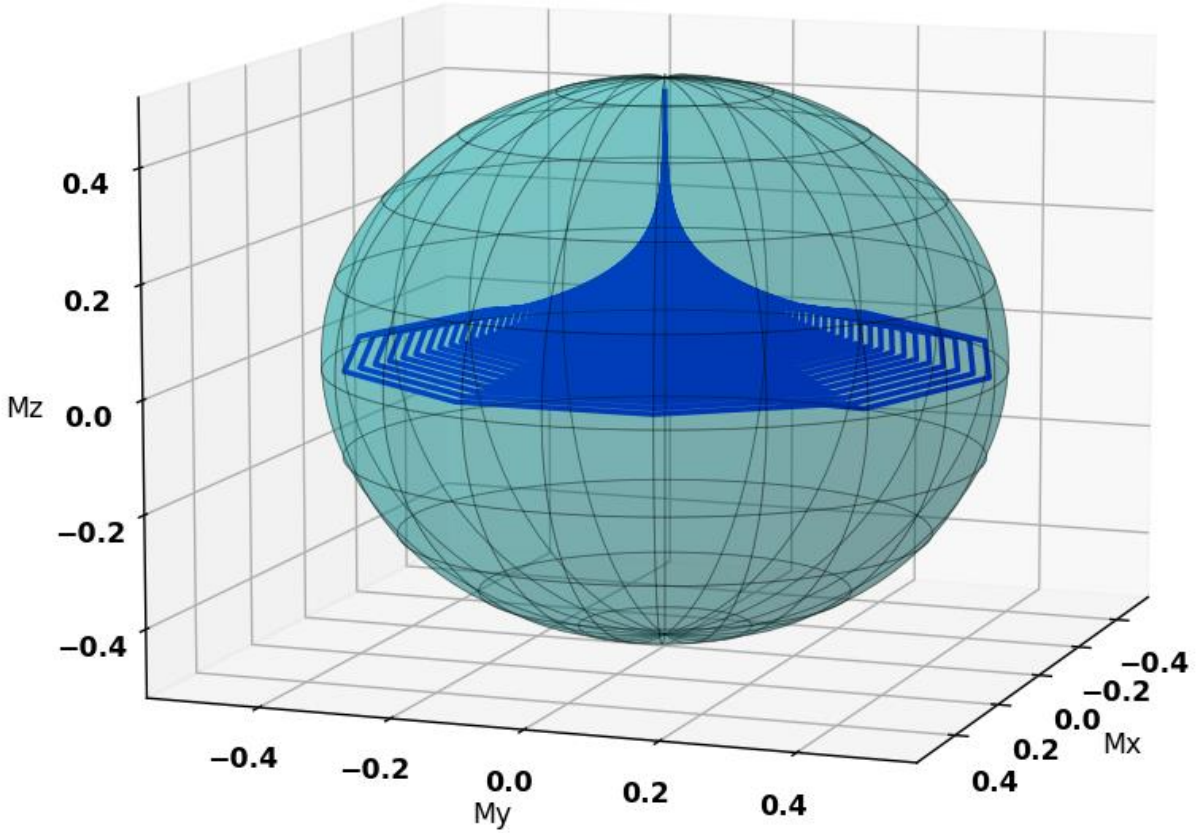
```
1 System.Plotting(4,freq,np.absolute(spectrum),"Frequency (Hz)","Spectrum","blue")
```





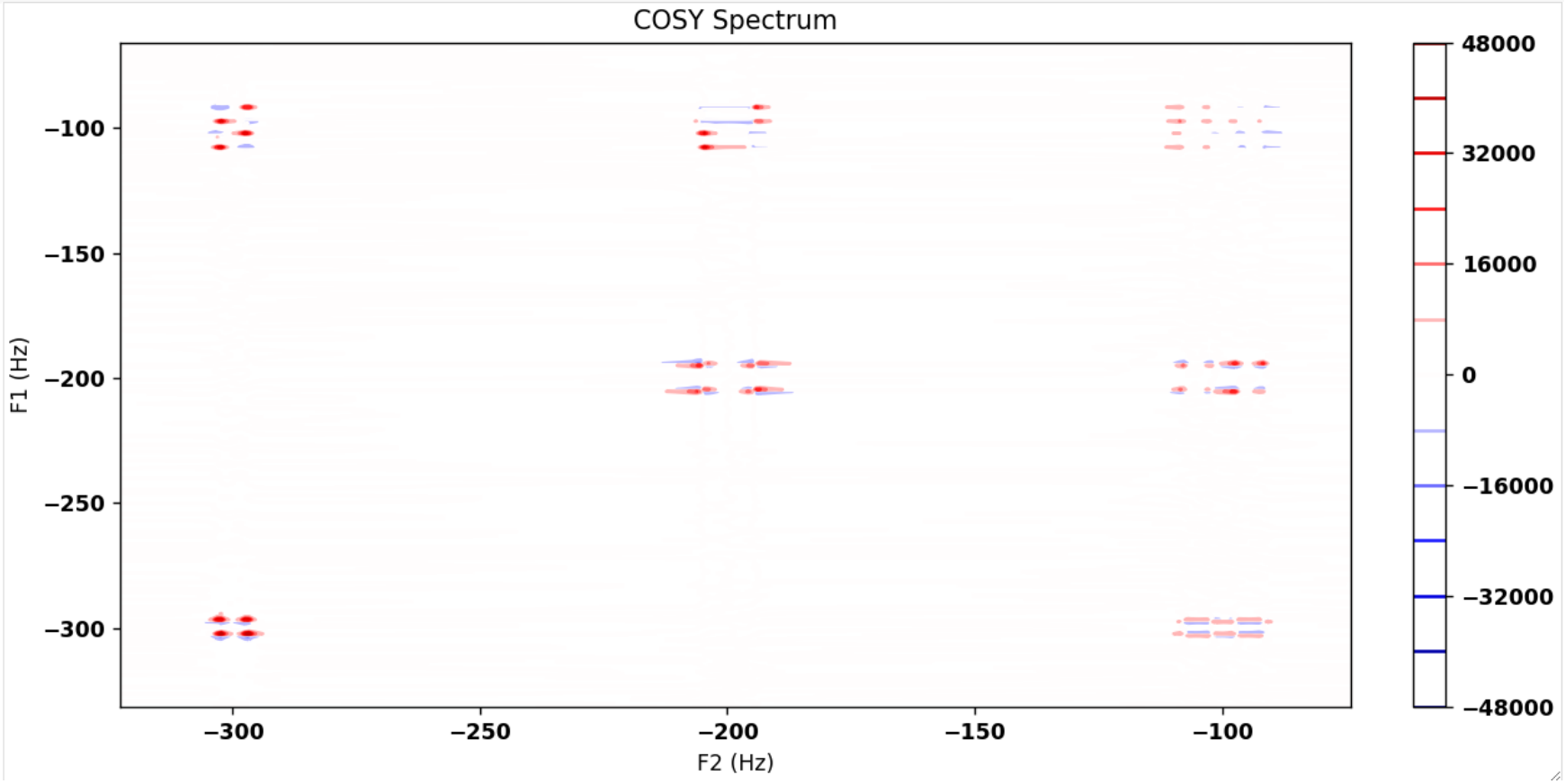
Simulation Results Visualization:
Sphere

```
plot_vector = False
scale_datapoints = 2
System.PlottingSphere(8,Mp.real,Mp.imag,Mz,rhoeq,np.sum(Sz,axis=0),plot_vector,scale_datapoints)
```



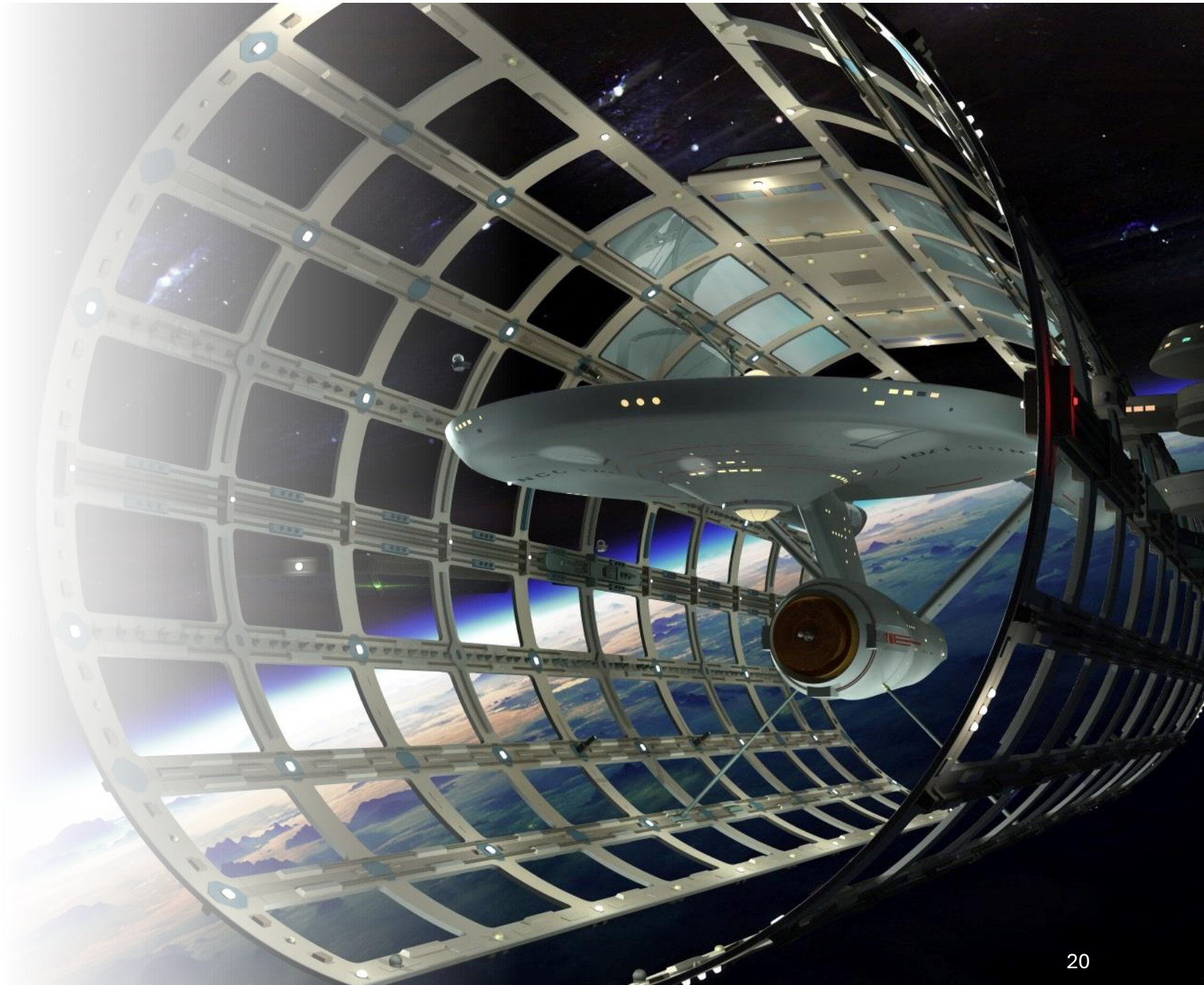
Simulation Results Visualization:
Contour Plotting (COSY)

```
# Contour Plot
PH0 = 45
spectrum_PH0_2D = System.PhaseAdjust_PH0(spectrum,PH0)
System.PlottingContour(4,F2,F1,spectrum_PH0_2D,"F2 (Hz)","F1 (Hz)","COSY Spectrum")
```



Future

- Maser Simulations
- EPR
- Solid State NMR
- Shaped Pulses
- DNP
- Anything I see interesting
- Anything you see interesting



Acknowledgement

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

PyOR is all yours

"Let what is created surpass the creator"

Thank You