

# UniDyn--Demo-04.nb

John A. Marohn  
jam99@cornell.edu  
Cornell University

**Abstract:** Use the **UniDyn** Evolver function to calculate the evolution of the magnetization of a two coupled spin = 1/2 particles.

---

## Set the path to the package

Tell *Mathematica* the path to the directory containing the package.

EDIT THE FOLLOWING PATH STRING:

```
In[1]:= $NCPATH = "/Users/jam99/Dropbox";  
$UniDynPath =  
    "/Users/jam99/Dropbox/MarohnGroup__Software_Library/UniDyn/  
    unidyn";
```

YOU SHOULD NOT NEED TO EDIT ANYTHING FROM HERE ONWARDS.

---

## Load the package

---

## Evolver with simplification

```
In[9]:= SimplifyingEvolver[H_, t_, ρ$0_] :=  
    Evolver[H, t, ρ$0] // Simplify // ExpToTrig // FullSimplify
```

---

## Unitary evolution of a single spin 1/2

### Create a single spin

The assumptions define below are required for *Mathematica* to recognize  $\sqrt{-\Delta^2 - \omega^2} = i \sqrt{\Delta^2 + \omega^2}$  inside an exponential. One of the variables has to be defined to be  $> 0$  and not just  $\geq 0$ .

```
In[10]:= Clear[
  Δω,          (* resonance offset frequency *)
  ω$1,         (* Rabi frequency of the applied irradiation *)
  t,           (* time *)
  Ix, Iy, Iz,  (* spin angular momentum operators *)
  ρ$0,         (* initial spin density operator *)
  H            (* spin Hamiltonian *)]
```

```
CreateScalar[Δω, ω$1, t];
SpinSingle$CreateOperators[Ix, Iy, Iz, L = 1/2];
```

```
$Assumptions = {Element[Δω, Reals], Δω ≥ 0,
  Element[ω$1, Reals], ω$1 > 0, Element[t, Reals], t ≥ 0};
```

```
... SpinSingle$CreateOperators : Creating spin operators.
```

```
... SpinSingle$CreateOperators : Adding spin commutations relations.
```

```
... SpinSingle$CreateOperators : Angular momentum L = 1/2. Adding operator simplification rules.
```

## On-resonance nutation

On-resonance irradiation Hamiltonian written in the interaction representation. The initial density operator is parallel to  $I_x$ .

```
In[14]:= H = ω$1 Ix;
ρ$0 = Iz;
```

Calculate the time-dependent density operator.

```
In[16]:= SimplifyingEvolver[H, t, ρ$0] /. {ω$1 → Subscript[ω, 1]}
```

```
Out[16]= Iz Cos[t ω1] - Iy Sin[t ω1]
```

## Free evolution

Zeeman-interaction Hamiltonian written in the interaction representation. The initial density operator is parallel to  $I_x$ .

```
In[17]:= H = Δω Iz;
ρ$0 = Ix;
```

Calculate the time-dependent density operator.

```
In[19]:= SimplifyingEvolver[H, t, ρ$0]
```

```
Out[19]= Ix Cos[t Δω] + Iy Sin[t Δω]
```

---

## Unitary evolution of two coupled spins

### Create a two spins

The assumptions define below are required for *Mathematica* to recognize

$\sqrt{-\Delta^2 - \omega^2} = i \sqrt{\Delta^2 + \omega^2}$  inside an exponential. One of the variables has to be defined to be  $> 0$  and not just  $\geq 0$ .

In[20]:= **Clear[**

```

 $\Delta I$ ,      (* resonance offset frequency *)
 $\Delta S$ ,      (* resonance offset frequency *)
J,          (* spin-spin coupling *)
Ix, Iy, Iz, (* spin angular momentum operators *)
Sx, Sy, Sz, (* spin angular momentum operators *)
 $\rho$ ,        (* spin density operator *)
t,          (* time *)
 $\rho_0$ ,       (* initial spin density operator *)
H           (* spin Hamiltonian *)]

```

```
CreateScalar[ $\Delta I$ ,  $\Delta S$ , J, t];
```

```
CreateOperator[{{Ix, Iy, Iz}, {Sx, Sy, Sz}}];
```

```
SpinSingle$CreateOperators[Ix, Iy, Iz, L = 1/2];
```

```
SpinSingle$CreateOperators[Sx, Sy, Sz, L = 1/2];
```

```

$Assumptions = {Element[ $\Delta I$ , Reals],  $\Delta I \geq 0$ ,
  Element[ $\Delta S$ , Reals],  $\Delta S \geq 0$ , Element[J, Reals], J > 0};

```

```
... SpinSingle$CreateOperators : Spin operators already exist.
```

```
... SpinSingle$CreateOperators : Adding spin commutations relations.
```

```
... SpinSingle$CreateOperators : Angular momentum L = 1/2. Adding operator simplification rules.
```

```
... SpinSingle$CreateOperators : Spin operators already exist.
```

```
... SpinSingle$CreateOperators : Adding spin commutations relations.
```

```
... SpinSingle$CreateOperators : Angular momentum L = 1/2. Adding operator simplification rules.
```

## Evolution under J coupling

On-resonance irradiation Hamiltonian written in the interaction representation. The initial density operator is parallel to  $I_x$ .

In[26]:= **H\$0 =  $\Delta I$  Iz +  $\Delta S$  Sz;**

```
H$J = J Iz ** Sz;
```

```
 $\rho_0$  = Ix;
```

Try to calculate the density operator in a single step.

```
In[29]:= SimplifyingEvolver[H$0 + H$J, t, ρ$0]
```

... Evolver : Unrecognized evolution

$$\text{Out[29]} = \left\{ I_x, I_y \Delta I + J I_y ** Sz, -\frac{1}{4} I_x (J^2 + 4 \Delta I^2) - 2 J \Delta I I_x ** Sz, \right. \\ \left. -\frac{1}{4} I_y \Delta I (3 J^2 + 4 \Delta I^2) - \frac{1}{4} (J^3 + 12 J \Delta I^2) I_y ** Sz, \right. \\ \left. \frac{1}{16} I_x (J^4 + 24 J^2 \Delta I^2 + 16 \Delta I^4) + J \Delta I (J^2 + 4 \Delta I^2) I_x ** Sz \right\}$$

This single-step evolution fails. Instead, let us calculate the density operator in two steps. We can do this because the H\$J and H\$0 Hamiltonians commute. Evolve under the J-coupling first and under the chemical shift second.

```
In[30]:= ρ = ρ$0 // SimplifyingEvolver[H$J, t, #] & //  
SimplifyingEvolver[H$0, t, #] &
```

$$\text{Out[30]} = \cos\left[\frac{J t}{2}\right] (I_x \cos[t \Delta I] + I_y \sin[t \Delta I]) + \\ 2 \sin\left[\frac{J t}{2}\right] (\cos[t \Delta I] I_y ** Sz - I_x ** Sz \sin[t \Delta I])$$

Try another way -- evolve under the chemical shift first and under the J-coupling second.

```
In[31]:= ρ = ρ$0 // SimplifyingEvolver[H$0, t, #] & //  
SimplifyingEvolver[H$J, t, #] &
```

$$\text{Out[31]} = \cos\left[\frac{J t}{2}\right] (I_x \cos[t \Delta I] + I_y \sin[t \Delta I]) + \\ 2 \sin\left[\frac{J t}{2}\right] (\cos[t \Delta I] I_y ** Sz - I_x ** Sz \sin[t \Delta I])$$

We see by inspection that we get the same answer either way.

### Load spin 1/2 operator matrices

Load matrices representing two J=1/2 spins and give them a test drive.

```
In[32]:= << Matrices--two-spin-half.m
```

Loaded spin one-half matrices for mIx, mIy, mIz, mSx, mSy, mSz

Let's look at one of the matrices, the matrix for Iz.

```
In[33]:= mIz // MatrixForm
mSz // MatrixForm
```

Out[33]//MatrixForm=

$$\begin{pmatrix} -\frac{1}{2} & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Out[34]//MatrixForm=

$$\begin{pmatrix} -\frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

For a single spin 1/2 particle,  $I_z$  is a 2 x 2 matrix. In the product space of two spin 1/2 particles, however, the  $I_z$  is now a 4 x 4 matrix. All the operators are 4 x 4 matrices.

Check that the  $[I_x, I_y]/I = I_z$  and  $[S_x, S_y]/I = S_z$  commutation relations hold with the matrices.

```
In[35]:= m$1 = 
$$\frac{mI_x \cdot mI_y - mI_y \cdot mI_x}{I};$$

m$2 = 
$$\frac{mS_x \cdot mS_y - mS_y \cdot mS_x}{I};$$

```

```
In[37]:= m$1 // MatrixForm (* should be Iz *)
m$2 // MatrixForm (* should be Sz *)
```

Out[37]//MatrixForm=

$$\begin{pmatrix} -\frac{1}{2} & 0 & 0 & 0 \\ 0 & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Out[38]//MatrixForm=

$$\begin{pmatrix} -\frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{pmatrix}$$

Make the (matrix) operator  $I_{\text{total}} = I_z + S_z$ . We can see by inspection that this matrix

looks correct. The  $I_{\text{total}}$  operator should have eigenvalues  $\{-1, 0, 0, +1\}$ . The matrix  $I_{\text{total}}$  is diagonal and we can read the eigenvalues off diagonal elements.

```
In[39]:= m$1 + m$2 // MatrixForm
```

```
Out[39]//MatrixForm=
```

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

## Calculate the analytical signal using the matrices

Write the matrix representation of  $I_y S_z$  using the matrices loaded above.

```
In[40]:= mIy . mSz // MatrixForm
```

```
Out[40]//MatrixForm=
```

$$\begin{pmatrix} 0 & 0 & -\frac{i}{4} & 0 \\ 0 & 0 & 0 & \frac{i}{4} \\ \frac{i}{4} & 0 & 0 & 0 \\ 0 & -\frac{i}{4} & 0 & 0 \end{pmatrix}$$

Show that we can obtain the same matrix from the symbolic product  $I_y ** S_z$  by (1) substituting the spin operators with matrices and (2) replacing the NonCommutativeMultiply operator with the Dot (i.e., matrix multiplication) operator.

```
In[41]:= Iy ** Sz /. {Ix -> mIx, Iy -> mIy, Iz -> mIz, Sx -> mSx, Sy -> mSy, Sz -> mSz} /.  
NonCommutativeMultiply -> Dot // MatrixForm
```

```
Out[41]//MatrixForm=
```

$$\begin{pmatrix} 0 & 0 & -\frac{i}{4} & 0 \\ 0 & 0 & 0 & \frac{i}{4} \\ \frac{i}{4} & 0 & 0 & 0 \\ 0 & -\frac{i}{4} & 0 & 0 \end{pmatrix}$$

```
In[42]:= rho // Expand
```

```
Out[42]= Ix Cos[ $\frac{J t}{2}$ ] Cos[t Δ$I] + 2 Cos[t Δ$I] Iy ** Sz Sin[ $\frac{J t}{2}$ ] +  
Iy Cos[ $\frac{J t}{2}$ ] Sin[t Δ$I] - 2 Ix ** Sz Sin[ $\frac{J t}{2}$ ] Sin[t Δ$I]
```

Calculate the density operator matrix.

```
In[44]:= Clear[ρ$matrix, ρ$temp, t];
ρ$temp = Expand[ρ] /. NonCommutativeMultiply -> Dot;
ρ$matrix[t_] = ρ$temp /.
  {Ix -> mIx, Iy -> mIy, Iz -> mIz, Sx -> mSx, Sy -> mSy, Sz -> mSz};
```

Check that the resulting object is a 4 x 4 matrix as expected.

```
In[47]:= Dimensions[ρ$matrix[t]]
```

```
Out[47]= {4, 4}
```

From this matrix we can calculate the I-spin signals as  $\text{Trace}[\rho I_x]$  and  $\text{Trace}[\rho I_y]$

```
In[48]:= Tr[ρ$matrix[t] . mIx] // Simplify
Tr[ρ$matrix[t] . mIy] // Simplify
```

```
Out[48]= Cos[ $\frac{J t}{2}$ ] Cos[t Δ$ I]
```

```
Out[49]= Cos[ $\frac{J t}{2}$ ] Sin[t Δ$ I]
```

We can mimic the complex signal collected by the NMR spectrometer by calculating the expectation value of the  $I_x + I_y$  operator:  $\text{Trace}[\rho (I_x + I_y)]$ .

```
In[50]:= Clear[S];
S[t_] := Tr[ρ$matrix[t] . (mIx - I mIy)] // TrigToExp
```

### Plot the signal as a function of time

Create a more realistic experimental signal by multiplying the above-calculated signal by a decaying exponential.

```
In[52]:= Clear[S$expt];
S$expt[t_] := S[t] Exp[-t / T2]
```

To plot, set the total number of points (NN) and the total time (T). We set NN equal to a power of 2 in anticipation of taking a digital Fourier transform.

```
In[54]:= NN = 2 ^ 10;
T$final = 10.0;
```

From NN and T\$final we derive the time step (dt) and the frequency step (df). Now generate a list of data points based on the above function. At the same time, generate a list of time points (t) and frequency points (f) for plotting.



```
In[56]:= dt = T$final / (NN - 1);
df = 1 / dt;
```

```
f$table = Table[jj / T$final, {jj, -NN / 2, NN / 2 - 1}];
t$table = Table[ii * dt, {ii, 0, NN - 1}];
```

```
Print["The time step is ", dt, " s"]
Print["The Nyquist frequency is ", 1 / (2 * dt), " Hz"]
```

The time step is 0.00977517 s

The Nyquist frequency is 51.15 Hz

Give numbers for the chemical shift and the J-coupling.

```
In[62]:= Δ$I$value = 2 π × 10.0; (* chemical shift [rad/s] *)
J$value = 2 π × 8.0;
(* heteronuclear scalar coupling constant [rad/s] *)
T2$value = 1.0; (* spin dephasing time [s] *)
```

Create an array of complex signal.

```
In[65]:= S$table = S$expt[t] /. {t → t$table,
    Δ$I → Δ$I$value, J → J$value, T2 → T2$value};
```

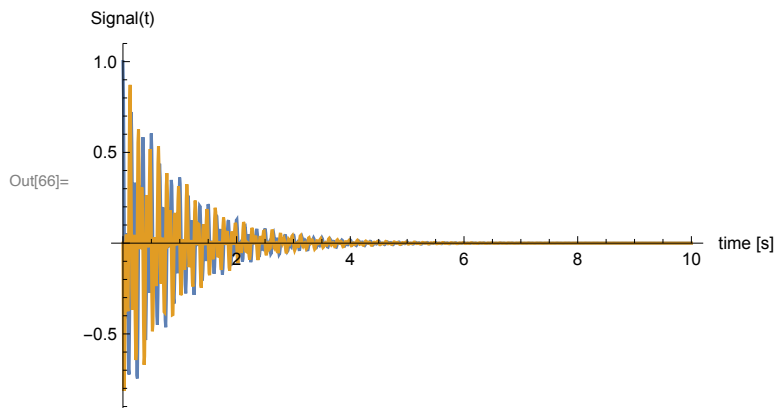
Plot the real and imaginary part of the signal

```

In[66]:= ListLinePlot[
  {
    {t$table, Re[S$table]} // Transpose,
    {t$table, Im[S$table]} // Transpose},
  Joined → True,
  PlotRange → All,
  PlotLabel →
    "Free induction decay of 1H coupled to 13C\n",
  AxesLabel → {"time [s]", "Signal(t)"}]

```

Free induction decay of 1H coupled to 13C



A function to calculate the digital Fourier transform of signal.

```

In[67]:= DFFT[signal$table_, time$table_, query_ : True] :=
  (* True to plot both real and imaginary FT parts *)
  Module[{NN, T$final},

    NN = Dimensions[signal$table][[1]];
    FFTS$table = RotateRight[Fourier[signal$table], NN/2];

    T$final = time$table[[-1]];
    f$table = Table[jj/T$final, {jj, -NN/2, NN/2 - 1}];

    p1 = ListLinePlot[
      {Transpose[{time$table, Re[signal$table]}],

```

```

Transpose[{time$table, Im[signal$table]}]},
PlotRange → {-1, 1},
AxesLabel → {"time [s]", "signal"}];

```

```
If[query == True,
```

```

p2 = ListLinePlot[
  {Transpose[{f$table, Re[FFTS$table]}],
    Transpose[{f$table, Im[FFTS$table]}]},
  PlotRange → All,
  AxesLabel → {"freq [Hz]", "DFT{signal}"},

```

```

p2 = ListLinePlot[
  Transpose[{f$table, Re[FFTS$table]}],
  PlotRange → All,
  AxesLabel → {"freq [Hz]", "DFT{signal}"}]
];

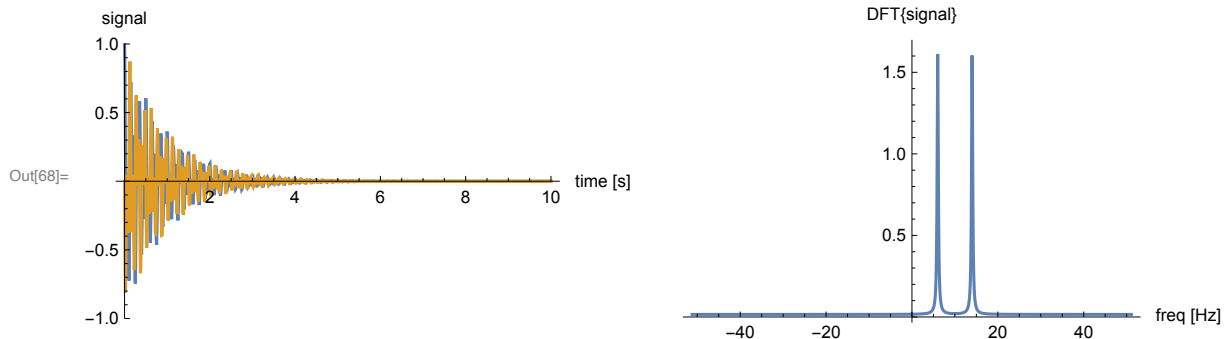
```

```
Show[GraphicsGrid[{{p1, p2}}]]
```

```
]
```

Fourier transform the signal to obtain the spectrum.

```
In[68]:= DFFT[S$table, t$table, False]
```



Find peaks in the FT spectrum. Require the peaks to be larger than 1.0. See the Find-Peaks function documentation here.

```
In[69]:= peaks = FindPeaks[Re[FFTS$table], 0, 0, 1.]
```

```
Out[69]= {{573, 1.61249}, {653, 1.60297}}
```

Read out the frequencies at which the peaks are located. We see peaks at the expected frequencies of  $10-8/2 = 6$  Hz and  $10+8/2 = 14$  Hz.

```
In[70]:= Part[f$table, Transpose[peaks][[1]]]
```

```
Out[70]= {6., 14.}
```

---

## Clean up