

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Presentation . . . . .	2
1.2	Context . . . . .	3
1.3	Reserved symbols . . . . .	3
<b>2</b>	<b>General functions</b>	<b>3</b>
2.1	Matrix form . . . . .	4
2.2	Kronecker product . . . . .	4
2.3	Adjoint matrix . . . . .	4
2.4	IsHermitian . . . . .	5
2.5	Scalar product of matrices . . . . .	5
2.6	Norm of matrices . . . . .	5
2.7	Normalization . . . . .	5
2.8	Random matrix generation . . . . .	6
2.9	Matrix logarithm . . . . .	6
2.10	Reading files generated by Matlab . . . . .	6
<b>3</b>	<b>Cartesian basis</b>	<b>7</b>
3.1	Parameters . . . . .	7
3.2	Different normalizations . . . . .	8
3.3	Decomposition format . . . . .	9
<b>4</b>	<b>Tensor product basis</b>	<b>10</b>
4.1	Parameters . . . . .	10
<b>5</b>	<b>Symmetrized tensor basis</b>	<b>12</b>
5.1	Parameters . . . . .	12
5.2	Decomposition format . . . . .	15
<b>6</b>	<b>Multipole tensor basis</b>	<b>16</b>
6.1	Parameters . . . . .	16
6.2	Decomposition format . . . . .	18
<b>7</b>	<b>Possible bases parameters and decomposition coefficients</b>	<b>18</b>
<b>8</b>	<b>Action of the permutation group on operators</b>	<b>20</b>
8.1	Particle permutations . . . . .	20
8.2	Symmetrizer and anti-symmetrizer . . . . .	21
8.2.1	Symmetrizer function . . . . .	21
8.2.2	Anti-symmetrizer function . . . . .	22
8.3	Projectors . . . . .	22
<b>9</b>	<b>Hamiltonians</b>	<b>24</b>
9.1	Chemical shift Hamiltonian . . . . .	24
9.2	Weak (longitudinal) coupling Hamiltonian . . . . .	25
9.3	Strong/isotropic coupling Hamiltonian . . . . .	26
9.4	Planar coupling Hamiltonian . . . . .	27
9.5	Radio frequency (control) Hamiltonian . . . . .	28
<b>10</b>	<b>Evolution and acquiring coefficients</b>	<b>29</b>

<b>11 DROPS visualization</b>	<b>31</b>
11.1 Droplets ordering	32
11.1.1 Tprod labels and grouping	32
11.1.2 Tsym labels and grouping	32
11.1.3 Tmpo labels and grouping	33
11.2 Coefficient grouping function	33
11.3 Plotting functions	34
11.3.1 Plotting a single droplet	34
11.3.2 DROPS visualization of a matrix	36
11.3.3 DROPS visualization from a list of 64 coefficients	38
11.4 Defining your own DROPS display	39
11.4.1 DropletParam option sets	39
11.4.2 DROPSPParam option sets	40
11.4.3 ConnectShape option set	41
11.4.4 Output plotting options	42
<b>12 Visualization of experiments</b>	<b>43</b>
12.1 Pulse file format	43
12.2 Reading pulse files	43
12.3 Combining the Hamiltonians generated from the pulse files	43
12.4 Creating the density matrices, Hamiltonians, effective Hamiltonian, propagators and effective propagators for an experiment	44
12.5 Creating slides	45
12.5.1 Evolution of a density matrix	45
12.5.2 Slides from a matrix of coefficients	46
12.5.3 Slides from a list of matrices	47
12.5.4 Slides from pulse files	48

# 1 Introduction

## 1.1 Presentation

The `DROPS_3B.m` package is devoted to the DROPS visualization of operators acting on three-spin-1/2 particle systems. The package gives the possibility to represent operators using three possible spherical tensor bases, each of which furnishes a different point of view to study the operators. The user will find in the present documentation the presentation of a wide variety of methods related to the construction of the tensor operators (such as projector operators and symmetrizers) and fundamental methods required in order to simulate NMR experiments. Note that deep theoretical explanations concerning the construction of irreducible spherical tensors are not presented in this documentation, but can be found in [the DROPS paper](#).

For each method, a detailed description of its function and of the taken parameters are presented. Parameters which are not necessary, i.e. for which a default value is included in the function, are presented in a white panel

**Default values : function**  
list of default values

For each function, at least one “ready to use” example is furnished and identified by a gray panel, which can be copied-pasted directly in your Mathematica notebook. Note also that

additional examples and information are directly included in the `DROPS_3B.m` package and can be accessed using the `?<functionName>` command.

## 1.2 Context

The present package is concerned with three-spin-1/2 systems. Operators acting on the system are represented by  $8 \times 8$  matrices on the ordered product of states basis:

$$\begin{pmatrix} |\alpha\alpha\alpha\rangle \\ |\alpha\alpha\beta\rangle \\ |\alpha\beta\alpha\rangle \\ |\alpha\beta\beta\rangle \\ |\beta\alpha\alpha\rangle \\ |\beta\alpha\beta\rangle \\ |\beta\beta\alpha\rangle \\ |\beta\beta\beta\rangle \end{pmatrix},$$

where  $|\alpha\rangle$  and  $|\beta\rangle$  are the two eigenfunctions for a single spin-1/2 particle corresponding to the  $J_z$ -eigenvalues  $m_{|\alpha\rangle} = +1/2$  and  $m_{|\beta\rangle} = -1/2$  respectively.

## 1.3 Reserved symbols

Some symbols used in the `DROPS_3B` package should not be reassigned once the package is loaded. To avoid mistakes, we list here these symbols and suggest the user to skim through the list at least once before getting started.

variables	Plotting options	Slides options
S1	basis	densityMatrices
S2	Output	propagators
S3	Linear	effectivePropagators
II	Bilinear	hamiltonians
Tprod	Trilinear	effectiveHamiltonians
Tsym	Id	allMatrices
Tmpo	Labels	totalTime
Basis[II]	coloring	report
Basisi[II]	GroundSphere	Extension
Basis[IInorm]	TextColor	Resolution
Basisi[IInorm]	TextRadius	file1
Basis[Tprod]	PositionsAndNames	file2
Basisi[Tprod]	SphereToPlot	file3
Basis[Tsym]	NamesToPlot	hOffset
Basisi[Tsym]	LabelPositions	scalingfactor
Basis[Tmpo]	DropletParam	timepoints
Basisi[Tmpo]	DROPSParam	directory
	ConnectShape	

In the version `DROPS_EXT`, the following symbols are also variables: `OrderBasis[II]`, `OrderBasis[IInorm]`, `OrderBasis[Tprod]`, `OrderBasis[Tsym]`, `OrderBasis[Tmpo]`.

## 2 General functions

We present here some commonly used functions. Note that throughout the document, the matrices are generally displayed using the matrix form function `MF`, even when the `MF` command

is not explicitly written in the command panels.

## 2.1 Matrix form

**MF** [**mtx**] returns the matrix form of the matrix **mtx**, and is a short-cut for the built-in Mathematica function `MatrixForm`.

### Example ( MF )

```
In:  
mtx = {{1, 2}, {3, 4}};  
MF[mtx]
```

```
Out:  

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```

## 2.2 Kronecker product

**KP** [**A, B, C, ...**] returns the Kronecker product of the matrices **A, B, C, ...**

### Example ( KP )

```
In:  
KP[{{1, 2}, {3, 4}}, {{0, 1}, {-1, 0}}]
```

```
Out:  

$$\begin{pmatrix} 0 & 1 & 0 & 2 \\ -1 & 0 & -2 & 0 \\ 0 & 3 & 0 & 4 \\ -3 & 0 & -4 & 0 \end{pmatrix}$$

```

## 2.3 Adjoint matrix

**Dagger** [**mtx**] returns the adjoint matrix of **mtx**, that is, its conjugate transpose.

### Example ( Dagger )

```
In:  
mtx = {{1, I}, {3, 4}};  
Dagger[mtx]
```

```
Out:  

$$\begin{pmatrix} 1 & 3 \\ -i & 4 \end{pmatrix}$$

```

## 2.4 IsHermitian

**IsHermitian[mtx]** returns 1 if `mtx` is Hermitian and 0 otherwise.

### Example ( IsHermitian )

```
In:
mtx = {{1, I}, {-I, 1}};
IsHermitian[mtx]
```

Out:

1

## 2.5 Scalar product of matrices

**Prod[a,b]** takes two square matrices `a` and `b` of same dimension and returns their Hilbert product  $\text{Trace}[a^\dagger b]$ .

### Example ( Prod )

```
In:
a = {{1, I}, {3, 4}};
b = {{2, 3+I}, {3+I, 8}};
Prod[a,b]
```

Out:

44

## 2.6 Norm of matrices

**MyNorm[a]** returns the norm of a square matrix as defined by the product function `Prod`, that is,  $\text{MyNorm}[a] := \sqrt{\text{Prod}[a, a]}$ .

### Example ( MyNorm )

```
In:
a = {{1, I}, {3, 4}};
MyNorm[a]
```

Out:

$3\sqrt{3}$

## 2.7 Normalization

**normalize[a]** takes a square matrix `a` and returns its normalized version  $\frac{a}{\text{MyNorm}[a]}$ .

### Example ( normalize )

```
In:  
a = {{1, I}, {3, 4}};  
MyNorm[a]  
X = normalize[a];  
MyNorm[X]
```

Out:

```
3√3  
1
```

## 2.8 Random matrix generation

**RandomMatrix** returns a random normalized  $8 \times 8$  matrix with complex entries.

### Example ( RandomMatrix )

```
In:  
a = RandomMatrix
```

Out:

Too big to be displayed

## 2.9 Matrix logarithm

**MatLog[a]** returns a matrix  $b$  such that  $\text{MatrixExp}[b]=a$ .

### Example ( MatLog )

```
In:  
a = {{1, 0}, {0, 2E+3I}}  
b = MatrixExp[a]  
MatLog[b] (*= a*)
```

Out:

```

$$\begin{pmatrix} 1 & 0 \\ 0 & 3i + 2e \end{pmatrix}$$

$$\begin{pmatrix} e & 0 \\ 0 & e^{3i+2e} \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 3i + 2e \end{pmatrix}$$

```

## 2.10 Reading files generated by Matlab

**ascii2num[x]** converts an ascii string of the form  $x="0.123\dots e+02"$  to a Mathematica floating number.

### Example

```
In:
x="1.825e+01";
ascii2num[x]

Out:
18.25
```

## 3 Cartesian basis

We present in this section the first bases of the 64-dimensional matrix space  $\text{Mat}(8, \mathbb{C})$ , which corresponds to three variations of the Cartesian basis. Three versions of them are available, namely `IInorm`, `IIone` and `II`, the elements of which are different up to a scaling factor.

### 3.1 Parameters

Cartesian operators are linear operators and are defined by the spins they are acting on as well as the type of one-particle operator among  $\{I_x, I_y, I_z\}$  acting on the spin. The possible parameters defining Cartesian operators according to their linearity are (where `II` can be replaced by `IInorm` or `IIone` at will):

- **Identity operator:** `II[E]`;
- **Linear operators:** `II[k,  $\eta$ ]` with  $k \in \{1, 2, 3\}$  and  $\eta \in \{x, y, z\}$ ;
- **Bilinear operators:** `II[k,  $\eta_1, l, \eta_2$ ]` with  $(k, l) \in \{(1, 2), (1, 3), (2, 3)\}$  and  $\eta_i \in \{x, y, z\}$ ;
- **Trilinear operators:** `II[ $\eta_1, \eta_2, \eta_3$ ]` with  $\eta_i \in \{x, y, z\}$ .

### Example (Linear, bilinear and trilinear Cartesian operators)

```
In:
II[E]
II[2,y] (*linear*)
IInorm[1,z,3,z] (*bilinear*)
IIone[x,y,z] (*trilinear*)
```

Out:

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

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

### 3.2 Different normalizations

As already mentioned, the elements in the bases `IIInorm`, `IIOne` and `II` are identical up to a scaling factor.

`IIInorm[...]` operators all have identical norm equal to  $\sqrt{2}$ .

`IIOne[...]` operators all have identical norm equal to 1.

`II[...]` operators have distinct norm depending on the linearity of the operator. Using the parameter definitions given above, the norm of `II[...]` operators satisfies:

- `II[E]` has norm  $2\sqrt{2}$ ;
- `II[k, η]` has norm  $\sqrt{2}$ ;
- `II[k, η1, l, η2]` has norm  $\frac{1}{\sqrt{2}}$ ;
- `II[η1, η2, η3]` has norm  $\frac{1}{2\sqrt{2}}$ .

Note that the basis `II` is defined according to the most common definition of Cartesian operators [1]. They are defined such that the operators  $1/2 * \text{II}[E]$ ,  $\text{II}[k, \eta]$ ,  $2 * \text{II}[k, \eta_1, l, \eta_2]$  and  $4 * \text{II}[\eta_1, \eta_2, \eta_3]$  are normalized to  $\sqrt{2}$ .

#### Example (Norm comparison of the Cartesian operators)

```
In:
MyNorm[IIInorm[E]]
MyNorm[IIOne[E]]
```

```
MyNorm[II[E]]
```

```
Out:
```

```
 $\sqrt{2}$   
1  
 $2\sqrt{2}$ 
```

### 3.3 Decomposition format

`Decomposition[mtx, II]` returns the decomposition of `mtx` in the (not-normalized) basis `II`. The output format for the `II` operators depends on their linearity and are shown in the following example.

#### Example (Output format for Cartesian operators)

```
In:  
Decomposition[II[E], II]  
Decomposition[II[3, x], II]  
Decomposition[II[1, y, 2, x], II]  
Decomposition[II[x, y, z], II]
```

```
Out:
```

```
I[e]  
I3x  
I1yI2x  
I1xI2yI3z
```

`Decomposition[mtx, IInorm]` also returns the decomposition of `mtx` in the basis `II`, but factorizes out the normalization factor required to make the `II[...]` operators normalized to  $\sqrt{2}$ .

#### Example (Output format for Cartesian operators)

```
In:  
Decomposition[IInorm[E], IInorm]  
Decomposition[IInorm[3, x], IInorm]  
Decomposition[IInorm[1, y, 2, x], IInorm]  
Decomposition[IInorm[x, y, z], IInorm]
```

```
Out:
```

```
1/2I[E]  
I3x  
2I1yI2x  
4I1xI2yI3z
```

## 4 Tensor product basis

The first tensor operator basis introduced is the one obtained recursively by taking the product of tensor operating on a  $(n - 1)$ -spin-1/2 system with the ones for a single-spin-1/2 system. These products are decomposed as a direct sum of individual tensors according to Clebsch decomposition:

$$\mathbb{T}_j^{\{(n-1)\text{- spins}\}} \otimes \mathbb{T}_{j'}^{\{1\text{ spin}\}} = \mathbb{T}_{|j-j'|}^{\{n\text{ spins}\}} \oplus \dots \oplus \mathbb{T}_{j+j'}^{\{n\text{ spins}\}}. \quad (1)$$

Some tensors are then multiplied by an additional phase factor in  $\{1, -1, i, -i\}$  such that all tensor operators of order  $m = 0$  are Hermitian. The phase factor is defined in the function `pfTprod` and takes the same parameter as the function `Tprod`.

### 4.1 Parameters

The operators in the tensor product basis are, as the Cartesian operators, linear. The linear and bilinear operators are uniquely defined by the particles they are acting on as well as the rank  $j$  and order  $m$  of the tensor. In addition to these parameters, the trilinear operators need an additional parameter in order to be uniquely defined. This additional parameter is taken to be the parents  $\mathbf{p} = \{j, j'\}$ , where  $j$  and  $j'$  are the rank of the parents on the left-hand side of Eq. (1). Explicitly, the possible parameters are:

- **Identity operator\*:** `Tprod[0, 0, 0]`;
- **Linear operators:** `Tprod[k, j, m]` with  $k \in \{1, 2, 3\}$ ,  $j = 1$  and  $m \in \{-1, 0, 1\}$ ;
- **Bilinear operators:** `Tprod[k1, k2, j, m]` with  $(k_1, k_2) \in \{(1, 2), (1, 3), (2, 3)\}$ ,  $j \in \{0, 1, 2\}$  and  $m \in \{-j, -j+1, \dots, j-1, j\}$ ;
- **Trilinear operators:** `Tprod[1, 2, 3, j, m, p]`, where the valid combinations for  $\mathbf{p}$  and  $j$  are

$\mathbf{p}$	$j$
$\{0, 1\}$	1
$\{1, 1\}$	0, 1, 2
$\{2, 1\}$	1, 2, 3

and  $m \in \{-j, -j+1, \dots, j-1, j\}$ .

\* `Tprod[0, 0, 0] =  $\frac{1}{2\sqrt{2}}$  * IdentityMatrix[8].`

#### Example

```
In:
Tprod[0, 0, 0]
Tprod[3, 1, -1] (*linear acting on spin 3, j=1, m=-1*)
Tprod[2, 3, 2, -1] (*bilinear in spins 2 and 3, j=2, m=-1*)
Tprod[1, 2, 3, 3, -2, {2, 1}] (*trilinear, j=3, m=-2, parents = {2, 1}*)
```

```
Out:
```

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2\sqrt{2}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2\sqrt{2}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & 0 & -\frac{1}{\sqrt{6}} & 0 & 0 & 0 & 0
\end{pmatrix}$$

**Example 2 (added phase factor for linear and bilinear tensors)**

In:  
pfTprod[1,1,0]  
pfTprod[2,3,0,0]  
pfTprod[2,3,1,0]  
pfTprod[2,3,2,0]

Out:  
1  
-1  
-i  
1

### Example 3 (Verification that m=0 tensor operators Hermitian)

```
In:
IsHermitian[Tprod[1,1,0]] (*linear*)
IsHermitian[Tprod[2,3,0,0]] (*bilinear*)
IsHermitian[Tprod[2,3,1,0]] (*bilinear*)
IsHermitian[Tprod[2,3,2,0]] (*bilinear*)
IsHermitian[Tprod[1,2,3,0,0,{1,1}]] (*trilinear*)
IsHermitian[Tprod[1,2,3,1,0,{0,1}]] (*trilinear*)
:

Out:
1
1
1
1
1
1
:
```

## 5 Symmetrized tensor basis

The tensor operators  $\text{Tsym}$  have a defined linearity and act on a defined particle subsystem, as do the Cartesian operators  $\text{II}$  and the tensor-product operators  $\text{Tprod}$ . They are moreover defined to have a defined symmetry specie which is used, in the case of trilinear operators, to uniquely define them. This is in contrast with the trilinear  $\text{Tprod}$  operators for which the parents are used to uniquely label the trilinear operators. We simply recall here that a tensor is said to be of symmetry specie  $\tau_i^{[k]}$  if it remains unchanged under a particular permutation action on  $k$  particles, this action being defined by the standard Young tableau  $\tau_i^{[k]}$ . The action associated to  $\tau_i^{[k]}$  is mathematically defined by the projector  $P_{\tau_i^{[k]}}$  associated to  $\tau_i^{[k]}$ . The four different symmetry species for the permutation of three particles, used to label trilinear operators, are defined according to the four standard Young tableaux:

$$\tau_1^{[3]} = \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline \end{array}, \quad \tau_2^{[3]} = \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & \\ \hline \end{array}, \quad \tau_3^{[3]} = \begin{array}{|c|c|} \hline 1 & 3 \\ \hline 2 & \\ \hline \end{array}, \quad \tau_4^{[3]} = \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array}. \quad (2)$$

As for the  $\text{Tprod}$  operators,  $\text{Tsym}$  tensors are defined such that the  $m = 0$  operators are Hermitian.

### 5.1 Parameters

The parameters characterizing the linear and bilinear  $\text{Tsym}[]$  operators and identical to those of the  $\text{Tprod}[]$  operators, *i.e.* they consist of the particle(s) involved in the operator as well as the rank  $j$  and order  $m$  of the operator. In addition to these parameters, the trilinear operators also take the symmetry specie of the operator into account. Explicitly, the possible parameters are:

- **Identity operator\***:  $\text{Tsym}[0,0,0]$ ;

- **Linear operators:**  $T_{\text{sym}}[k, j, m]$  with  $k \in \{1, 2, 3\}, j = 1$  and  $m \in \{-1, 0, 1\}$ ;
- **Bilinear operators:**  $T_{\text{sym}}[k_1, k_2, j, m]$  with  $(k_1, k_2) \in \{(1, 2), (1, 3), (2, 3)\}, j \in \{0, 1, 2\}$  and  $m \in \{-j, -j+1, \dots, j-1, j\}$ ;
- **Trilinear operators:**  $T_{\text{sym}}[1, 2, 3, j, m, \text{sym}]$  where the valid combinations of  $\text{sym}$  and  $j$  are

sym	j
1	1, 3
2	1, 2
3	1, 2
4	0

and  $m \in \{-j, -j+1, \dots, j-1, j\}$ .

\*  $T_{\text{sym}}[0, 0, 0] = \frac{1}{2\sqrt{2}} * \text{IdentityMatrix}[8]$ .

The parameter  $\text{sym}$  refers here to the symmetry type defined by the corresponding Young tableau in (2).

### Example

In:

```
Tsym[3,1,-1] (*acting on spin 3*)
Tsym[1,3,2,0] (*acting on spins 1 and 3*)
Tsym[1,2,3,2,-2,2]
Tsym[1,2,3,2,-2,3]
```

Out:

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

$$\begin{pmatrix} \frac{1}{2\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2\sqrt{3}} & 0 & 0 & -\frac{1}{2\sqrt{3}} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2\sqrt{3}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2\sqrt{3}} & 0 & 0 & -\frac{1}{2\sqrt{3}} & 0 \\ 0 & -\frac{1}{2\sqrt{3}} & 0 & 0 & -\frac{1}{2\sqrt{3}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2\sqrt{3}} & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2\sqrt{3}} & 0 & 0 & -\frac{1}{2\sqrt{3}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2\sqrt{3}} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{i}{2\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{i}{2\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{i}{\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{i}{\sqrt{3}} & \frac{i}{2\sqrt{3}} & 0 & \frac{i}{2\sqrt{3}} & 0 & 0 & 0 \end{pmatrix}$$

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

**Example 2 (added phase factor for all seven trilinear tensors)**

In:  
 pfTsym[1, 2, 3, 0, 0, 4] (\*j=0\*)  
 pfTsym[1, 2, 3, 1, 0, 1] (\*j=1\*)  
 pfTsym[1, 2, 3, 1, 0, 2]  
 pfTsym[1, 2, 3, 1, 0, 3]  
 pfTsym[1, 2, 3, 2, 0, 2] (\*j=2\*)  
 pfTsym[1, 2, 3, 2, 0, 3]  
 pfTsym[1, 2, 3, 3, 0, 1] (\*j=3\*)

Out:

-i  
 -1  
 1  
 1  
 -i  
 -i  
 1

**Example 3 (Verification that trilinear m=0 operators Hermitian)**

In:  
 IsHermitian[Tsym[1, 2, 3, 0, 0, 4]] (\*j=0\*)  
 IsHermitian[Tsym[1, 2, 3, 1, 0, 1]] (\*j=1\*)  
 IsHermitian[Tsym[1, 2, 3, 1, 0, 2]]  
 IsHermitian[Tsym[1, 2, 3, 1, 0, 3]]  
 IsHermitian[Tsym[1, 2, 3, 2, 0, 2]] (\*j=2\*)  
 IsHermitian[Tsym[1, 2, 3, 2, 0, 3]]  
 IsHermitian[Tsym[1, 2, 3, 3, 0, 1]] (\*j=3\*)

Out:

```

1
1
1
1
1
1
1
1

```

## 5.2 Decomposition format

`Decomposition[mtx, Tsym]` returns the decomposition of `mtx` in the basis `Tsym`. The output format of the `Tsym` operators depends on their linearity and are shown in the following example.

### Example (Output format for Tsym)

```

In:
Decomposition[Tsym[0,0,0],Tsym]
Decomposition[Tsym[3,1,-1],Tsym] (*linear*)
Decomposition[Tsym[1,3,2,-2],Tsym] (*bilinear*)
Decomposition[Tsym[1,2,3,3,-3,1],Tsym] (*trilinear*)

```

```

Out:
T_{s,0,0}^{0}
T_{s,1,-1}^{3}
T_{s,2,-2}^{13}
T_{s,3,-3}^{123} [1]

```

### Example 2 (Translation between II, Tprod and Tsym operators)

```

In:
Decomposition[II[3,x],Tsym]
Decomposition[Tprod[1,2,3,1,1,2,1],Tsym]
Decomposition[Sqrt[3]*Tsym[1,3,2,0],II]
Decomposition[Tsym[1,3,2,0],Tprod]

```

```

Out:
T_{s,1,-1}^{3} - T_{s,1,1}^{3}
- 2/3 (T_{s,1,1}^{123} [1]) - 1/3 sqrt(5) (T_{s,1,1}^{123} [2])
- (I_{1x}I_{3x}) - I_{1y}I_{3y} + 2(I_{1z}I_{3z})
T_{p,2,0}^{13}

```

## 6 Multipole tensor basis

The multipole tensor operators **Tmpo** [ . . . ], unlike the **II**, **Tprod** and the **Tsym** operators, are not linear operators in the sense that they combine linear, bilinear and trilinear operators. Each tensor is uniquely defined by its rank  $j$ , its order  $m$  and by a transition between the different angular momentum states set. For systems consisting of three spin-1/2 particles, three angular momentum state sets form the 8-dimensional states basis, namely:

$$\begin{aligned} S1 &:= \{|1/2, -1/2, 0\rangle, |1/2, 1/2, 0\rangle\}, \\ S2 &:= \{|1/2, -1/2, 1\rangle, |1/2, 1/2, 1\rangle\}, \\ S3 &:= \{|3/2, -3/2, 1\rangle, |3/2, -1/2, 1\rangle, |3/2, 1/2, 1\rangle, |3/2, 3/2, 1\rangle\}, \end{aligned}$$

where the last parameter  $\kappa$  in the kets  $|j, m, \kappa\rangle$  refers to the parents in the recursive construction of angular momentum states. There are nine possible state sets transitions from set  $S_i$  to set  $S_j$  with  $i, j \in \{1, 2, 3\}$ .

### 6.1 Parameters

The **Tmpo** operators are all defined as

$$\text{Tmpo}[j, m, \text{Set1}, \text{Set2}]$$

where the possible  $j$  values depend on the transition sets **Set1** and **Set2**. We list below the possible  $j$  values of each transition.

(Set1, Set2)	$j$ values
(S1, S1)	0, 1
(S1, S2)	0, 1
(S1, S3)	1, 2
(S2, S1)	0, 1
(S2, S2)	0, 1
(S2, S3)	1, 2
(S3, S1)	1, 2
(S3, S2)	1, 2
(S3, S3)	0, 1, 2, 3

and  $m \in \{-j, -j+1, \dots, j-1, j\}$ .

#### Example

In:  
**Tmpo**[2, -1, S3, S2]  
**Tmpo**[0, 0, S1, S1]  
**Tmpo**[1, 0, S2, S1]  
**Tmpo**[2, 1, S3, S3]

Out:

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{1}{2\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{2\sqrt{6}} & -\frac{1}{2\sqrt{6}} & 0 & -\frac{1}{2\sqrt{6}} & 0 & 0 & 0 \\
-\frac{1}{2\sqrt{6}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{2\sqrt{6}} & -\frac{1}{2\sqrt{6}} & 0 & -\frac{1}{2\sqrt{6}} & 0 & 0 & 0 \\
0 & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & 0 & \frac{1}{\sqrt{6}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2\sqrt{2}} & 0 & -\frac{1}{2\sqrt{2}} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2\sqrt{2}} & 0 & -\frac{1}{2\sqrt{2}} & 0 & 0 \\
0 & 0 & -\frac{1}{2\sqrt{2}} & 0 & \frac{1}{2\sqrt{2}} & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{1}{2\sqrt{2}} & 0 & \frac{1}{2\sqrt{2}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{\sqrt{6}} & -\frac{1}{2\sqrt{6}} & 0 & -\frac{1}{2\sqrt{6}} & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{1}{2\sqrt{6}} & 0 & -\frac{1}{2\sqrt{6}} & \frac{1}{\sqrt{6}} & 0 \\
0 & -\frac{1}{\sqrt{6}} & \frac{1}{2\sqrt{6}} & 0 & \frac{1}{2\sqrt{6}} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2\sqrt{6}} & 0 & \frac{1}{2\sqrt{6}} & -\frac{1}{\sqrt{6}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & 0 & -\frac{1}{\sqrt{6}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{6}} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Unlike the Tprod and Tsym operators, the Tmpo operators with  $m = 0$  components are not necessarily Hermitian and can not be made Hermitian simply by multiplying them by a  $i$  factor.

```

Example 2 (Tmpo m=0 components not necessarily Hermitian)
In:
IsHermitian[Tmpo[0,0,S1,S1]]
IsHermitian[Tmpo[1,0,S1,S1]]
IsHermitian[Tmpo[0,0,S1,S2]]
IsHermitian[Tmpo[1,0,S1,S2]]
IsHermitian[Tmpo[1,0,S1,S3]]
IsHermitian[Tmpo[2,0,S1,S3]]

Out:
1

```

```
1
0
0
0
0
0
```

## 6.2 Decomposition format

**Decomposition**[*mtx*, *Tmpo*] returns the decomposition of *mtx* in the basis *Tmpo*. The output format of the *Tmpo* operators are all similar and are shown in the following example.

### Example (Output format for *Tmpo*)

```
In:
Decomposition[Tmpo[1, -1, S2, S1], Tmpo]
Decomposition[Tmpo[3, -3, S3, S3], Tmpo]
Decomposition[Tmpo[2, -1, S3, S1], Tmpo]

Out:
Tm,1,-1[S2, S1]
Tm,3,-3[S3, S3]
Tm,2,-1[S3, S1]
```

### Example 2 (Translation between other bases)

```
In:
Decomposition[4*II[z, z, z], Tmpo]
Decomposition[3*Sqrt[3/5]*Tprod[1, 2, 3, 1, 1, 2, 1], Tmpo]
Decomposition[Tsym[1, 2, 3, 3, 0, 1], Tmpo]
Decomposition[Sqrt[2]*Tmpo[0, 0, S1, S1], II]

Out:
-  $\frac{T_{m,1,0}[S1, S1]}{\sqrt{2}} - \frac{T_{m,1,0}[S2, S2]}{\sqrt{2}} + \frac{T_{m,1,0}[S3, S3]}{\sqrt{5}} + \frac{2(T_{m,3,0}[S3, S3])}{\sqrt{5}}$ 
 $2(T_{m,1,1}[S2, S2]) - \frac{T_{m,1,1}[S2, S3]}{\sqrt{2}} + \frac{T_{m,1,1}[S3, S2]}{\sqrt{2}} - \sqrt{\frac{2}{5}}(T_{m,1,1}[S3, S3]) T_{m,3,0}[S3, S3]$ 
 $\frac{1}{4}I[e] - I_{1x}I_{2x} - I_{1y}I_{2y} - I_{1z}I_{2z}$ 
```

## 7 Possible bases parameters and decomposition coefficients

**OrderBasis** is a function with two purposes. The first purpose is to order all the elements of a given basis in a fixed order to eventually store the decomposition coefficients of a matrix in this basis. The second purpose is to remind the user which parameters are available for a given basis among *II*, *IInorm*, *Tprod*, *Tsym* and *Tmpo*.





### Example 3 ( ParticlePermute )

```
In:
mtx = Tprod[1,2,1,1];
mtx2 = ParticlePermute[mtx,{2,3}];
Decomposition[mtx,Tprod]
Decomposition[mtx2,Tprod]
```

Out:

$$T_{p,1,1}^{\{12\}}$$
$$T_{p,1,1}^{\{13\}}$$

## 8.2 Symmetrizer and anti-symmetrizer

### 8.2.1 Symmetrizer function

**symmetrize[mtx,k,l]** returns the operator obtained from `mtx` by symmetrizing over the particles `k` and `l`.

`symmetrize[mtx]` symmetrizes over the three particles.

### Example (symmetrize, two particles)

```
In:
mtx = II[1,x];
mtx2 = symmetrize[mtx,1,3];
Decomposition[mtx,II]
Decomposition[mtx2,II]
```

Out:

$$I_{1x}$$
$$\frac{I_{1x}}{2} + \frac{I_{3x}}{2}$$

### Example 2 (symmetrize, three particles)

```
In:
mtx = II[1,x];
mtx2 = symmetrize[mtx];
Decomposition[mtx,II]
Decomposition[mtx2,II]
```

Out:

$$I_{1x}$$
$$\frac{I_{1x}}{3} + \frac{I_{2x}}{3} + \frac{I_{3x}}{3}$$

### 8.2.2 Anti-symmetrizer function

**antisymmetrize[mtx, k, l]** returns the operator obtained from `mtx` by anti-symmetrizing over the particles `k` and `l`.

`antisymmetrize[mtx]` anti-symmetrizes over the three particles.

#### Example 1 ( antisymmetrize, two particles )

```
In:
mtx = II[1, x];
mtx2 = antisymmetrize[mtx, 1, 3];
Decomposition[mtx, II]
Decomposition[mtx2, II]
```

Out:

```
I1x
 $\frac{I_{1x}}{2} - \frac{I_{3x}}{2}$ 
```

#### Example 2 ( antisymmetrize, three particles )

```
In:
mtx = II[1, x];
mtx2 = antisymmetrize[mtx];
Decomposition[mtx, II]
Decomposition[mtx2, II]
```

Out:

```
I1x
0
```

## 8.3 Projectors

The projector  $P_{\tau_i^{[k]}}$  is an operator related to a symmetry specie defined by the standard Young tableau  $\tau_i^{[k]}$ . It acts on a matrix by projecting it on its component of defined symmetry specie  $\tau_i^{[k]}$ .

There is only one projector for  $k = 1$ , namely  $P_{\tau_1^{[1]}}$ , associated to the standard Young tableau

$$\tau_1^{[1]} = \boxed{1}$$

and corresponding to the identity operator. No particular function has been created for this “identity” projector.

There are two projectors for  $k = 2$ , namely  $P_{\tau_1^{[2]}}$  and  $P_{\tau_2^{[2]}}$ , associated to the standard Young tableau

$$\tau_1^{[2]} = \boxed{1 \ 2} \quad \text{and} \quad \tau_2^{[2]} = \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline \end{array}$$

respectively. The projectors  $P_{\tau_1^{[2]}}$  and  $P_{\tau_2^{[2]}}$  are respectively the symmetrizer and anti-symmetrizer on two particles introduced in Section 8.2.

Finally, there are four projectors for  $k = 3$ , namely  $P_{\tau_i^{[3]}}$  for  $i \in \{1, 2, 3, 4\}$ , associated to the standard Young tableau

$$\tau_1^{[3]} = \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline \end{array}, \quad \tau_2^{[3]} = \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & \\ \hline \end{array}, \quad \tau_3^{[3]} = \begin{array}{|c|c|} \hline 1 & 3 \\ \hline 2 & \\ \hline \end{array}, \quad \tau_4^{[3]} = \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array}$$

already encountered in the presentation of the  $T_{\text{sym}}$  basis (see Eq. 2 in Section 5).

**Projector1[mtx]**, **Projector2[mtx]**, **Projector3[mtx]** and **Projector4[mtx]** are the four corresponding projector functions. Note that the two projectors  $P_{\tau_1^{[3]}} = \text{Projector1}$  and  $P_{\tau_4^{[3]}} = \text{Projector4}$  correspond respectively to the symmetrizer and anti-symmetrizer on three particles introduced in Section 8.2.

#### Example 1 (the four projectors on 3 particles)

```
In:
mtx = II[1, x];
Decomposition[mtx, II]
Decomposition[Projector1[mtx], II]
Decomposition[Projector2[mtx], II]
Decomposition[Projector3[mtx], II]
Decomposition[Projector4[mtx], II]

Out:
I1x
 $\frac{I_{1x}}{3} + \frac{I_{2x}}{3} + \frac{I_{3x}}{3}$ 
 $\frac{I_{1x}}{3} + \frac{I_{2x}}{3} - \frac{2I_{3x}}{3}$ 
 $\frac{I_{1x}}{\sqrt{3}} - \frac{I_{2x}}{\sqrt{3}}$ 
0
```

This second example underlines the property of the symmetrized tensor operators  $T_{\text{sym}}$  to have a defined symmetry type.

#### Example 2 (the four Projector functions on 3 particles)

```
In:
mtx = Tsym[1, 2, 3, 1, 1, 2];
Decomposition[mtx, Tsym]
Decomposition[Projector1[mtx], Tsym]
Decomposition[Projector2[mtx], Tsym]
Decomposition[Projector3[mtx], Tsym]
Decomposition[Projector4[mtx], Tsym]

Out:
```

```

Ts,1,1{123}[2]
0
Ts,1,1{123}[2]
Ts,1,1{123}[3]
0

```

## 9 Hamiltonians

Methods to define some typical Hamiltonians are available in the DROPS\_3B package. We present here these different methods together with their analytical form.

### 9.1 Chemical shift Hamiltonian

**Hcshift[spin, nu]** returns the chemical shift Hamiltonian for particle spin with Larmor Frequency nu (Hz):

$$\text{Hcshift}[\text{spin}, \text{nu}] := 2\pi \text{nu } I_{\text{spin},z}.$$

**HcshiftTotal[nu1, nu2, nu3]** returns the *sum* of the three chemical shift Hamiltonian where nu1, nu2, nu3 are the chemical shift for the respective spins.

$$\begin{aligned} \text{HcshiftTotal}[\text{nu1}, \text{nu2}, \text{nu3}] := \\ \text{Hcshift}[1, \text{nu1}] + \text{Hcshift}[2, \text{nu2}] + \text{Hcshift}[3, \text{nu3}] \end{aligned}$$

#### Example ( Hcshift and HcshiftTotal )

```

In:
Hcshift[2, 30]
HcshiftTotal[0, 30, 0] (*= Hcshift[2, 30]*)
HcshiftTotal[30, 10, 10]

```

Out:

$$\begin{pmatrix} 30\pi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 30\pi & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -30\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -30\pi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 30\pi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 30\pi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -30\pi & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -30\pi \end{pmatrix}$$

$$\begin{pmatrix} 30\pi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 30\pi & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -30\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -30\pi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 30\pi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 30\pi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -30\pi & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -30\pi \end{pmatrix}$$

$$\begin{pmatrix} 50\pi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 30\pi & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10\pi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -10\pi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -30\pi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -30\pi & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -50\pi \end{pmatrix}$$

## 9.2 Weak (longitudinal) coupling Hamiltonian

**Hweak[spin1, spin2, Jc]** returns the weak coupling Hamiltonian between spin1 and spin2 with coupling constant Jc (Hz):

$$\text{Hweak}[\text{spin1}, \text{spin2}, \text{Jc}] := 2\pi \text{Jc } I_{\text{spin1},z} I_{\text{spin2},z}.$$

**HweakTotal[Jc12, Jc13, Jc23]** returns the *sum* of the three weak coupling Hamiltonians where Jc12, Jc13, Jc23 corresponds to the coupling constant between the spin pairs (1, 2), (1, 3) and (2, 3) respectively.

$$\begin{aligned} \text{HweakTotal}[\text{Jc12}, \text{Jc13}, \text{Jc23}] := \\ \text{Hweak}[1, 2, \text{Jc12}] + \text{Hweak}[1, 3, \text{Jc13}] + \text{Hweak}[2, 3, \text{Jc23}] \end{aligned}$$

### Example ( Hweak and HweakTotal )

```
In:
Hweak[2, 3, 88]
HweakTotal[0, 0, 88] (*= Hweak[2, 3, 88]*)
HweakTotal[88, 0, 88]
```

Out:

$$\begin{pmatrix} 44\pi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -44\pi & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -44\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 44\pi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 44\pi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -44\pi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -44\pi & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 44\pi \end{pmatrix}$$

$$\begin{pmatrix} 44\pi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -44\pi & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -44\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 44\pi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 44\pi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -44\pi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -44\pi & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 44\pi \end{pmatrix}$$

$$\begin{pmatrix} 88\pi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -88\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -88\pi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 88\pi \end{pmatrix}$$

### 9.3 Strong/isotropic coupling Hamiltonian

**Hstrong[spin1, spin2, Jc]** returns the strong (or isotropic) coupling Hamiltonian between spin1 and spin2 with coupling constant Jc (Hz):

$$\text{Hstrong}[\text{spin1}, \text{spin2}, \text{Jc}] := 2\pi \text{Jc} (I_{\text{spin1},x} I_{\text{spin2},x} + I_{\text{spin1},y} I_{\text{spin2},y} + I_{\text{spin1},z} I_{\text{spin2},z}).$$

**HstrongTotal[Jc12, Jc13, Jc23]** returns the *sum* of the three strong coupling Hamiltonians where Jc12, Jc13, Jc23 corresponds to the coupling constant between the spin pairs (1,2), (1,3) and (2,3) respectively.

$$\text{HstrongTotal}[\text{Jc12}, \text{Jc13}, \text{Jc23}] := \text{Hstrong}[1, 2, \text{Jc12}] + \text{Hstrong}[1, 3, \text{Jc13}] + \text{Hstrong}[2, 3, \text{Jc23}]$$

#### Example ( Hstrong and HstrongTotal )

In:  
Hstrong[1, 3, 15]  
HstrongTotal[0, 15, 0] (\*= Hstrong[1, 3, 15]\*)  
HstrongTotal[10, 0, 10]

Out:

$$\begin{pmatrix} \frac{15\pi}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{15\pi}{2} & 0 & 0 & 15\pi & 0 & 0 & 0 \\ 0 & 0 & \frac{15\pi}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{15\pi}{2} & 0 & 0 & 15\pi & 0 \\ 0 & 15\pi & 0 & 0 & -\frac{15\pi}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{15\pi}{2} & 0 & 0 \\ 0 & 0 & 0 & 15\pi & 0 & 0 & -\frac{15\pi}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{15\pi}{2} \end{pmatrix}$$

$$\begin{pmatrix} \frac{15\pi}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{15\pi}{2} & 0 & 0 & 15\pi & 0 & 0 & 0 \\ 0 & 0 & \frac{15\pi}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{15\pi}{2} & 0 & 0 & 15\pi & 0 \\ 0 & 15\pi & 0 & 0 & -\frac{15\pi}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{15\pi}{2} & 0 & 0 \\ 0 & 0 & 0 & 15\pi & 0 & 0 & -\frac{15\pi}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{15\pi}{2} \end{pmatrix}$$

$$\begin{pmatrix} 10\pi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 10\pi & -10\pi & 0 & 10\pi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10\pi & 0 & 0 \\ 0 & 0 & 10\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10\pi & 0 & -10\pi & 10\pi & 0 \\ 0 & 0 & 0 & 0 & 0 & 10\pi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10\pi \end{pmatrix}$$

## 9.4 Planar coupling Hamiltonian

**Hplan[spin1, spin2, Jc]** returns the planar coupling Hamiltonian between spin1 and spin2 with coupling constant Jc (Hz):

$$\text{Hplan}[\text{spin1}, \text{spin2}, \text{Jc}] := 2\pi \text{Jc} (I_{\text{spin1},x} I_{\text{spin2},x} + I_{\text{spin1},y} I_{\text{spin2},y}).$$

**HplanTotal[Jc12, Jc13, Jc23]** returns the *sum* of the three planar coupling Hamiltonians where Jc12, Jc13, Jc23 corresponds to the coupling constant between the spin pairs (1, 2), (1, 3) and (2, 3) respectively.

$$\begin{aligned} \text{HplanTotal}[\text{Jc12}, \text{Jc13}, \text{Jc23}] := \\ \text{Hplan}[1, 2, \text{Jc12}] + \text{Hplan}[1, 3, \text{Jc13}] + \text{Hplan}[2, 3, \text{Jc23}] \end{aligned}$$

### Example ( Hplan and HplanTotal )

```
In:
Hplan[1,2,20]
HplanTotal[20,0,0] (*= Hplan[1,2,20]*)
HplanTotal[5,10,15]
```

Out:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20\pi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20\pi & 0 & 0 \\ 0 & 0 & 20\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20\pi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20\pi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20\pi & 0 & 0 \\ 0 & 0 & 20\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20\pi & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15\pi & 0 & 10\pi & 0 & 0 & 0 \\ 0 & 15\pi & 0 & 0 & 5\pi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5\pi & 10\pi & 0 \\ 0 & 10\pi & 5\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5\pi & 0 & 0 & 15\pi & 0 \\ 0 & 0 & 0 & 10\pi & 0 & 15\pi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

## 9.5 Radio frequency (control) Hamiltonian

There are two methods to define radio-frequency (rf) Hamiltonians, depending on the parametrization of the control field.

**Hrfxy[spin, ux, uy]** returns the Hamiltonian generated by the action of the an external rf-field on spin, where the  $x$  and  $y$  components of the field are given by `ux` and `uy`:

$$\text{Hrfxy}[\text{spin}, \text{ux}, \text{uy}] := 2\pi (\text{ux } I_{\text{spin},x} + \text{uy } I_{\text{spin},y}).$$

**Hrf[spin, ampl, ph]** also returns the Hamiltonian generated by the action of the an external rf-field on spin, where the external magnetic field is parametrized by its amplitude `ampl` (Hz) and phase `ph` (Deg).

$$\begin{aligned} \text{Hrf}[\text{spin}, \text{ampl}, \text{phase}] := \\ 2\pi (\text{ampl } \cos(\text{ph}) I_{\text{spin},x} + \text{ampl } \sin(\text{ph}) I_{\text{spin},y}). \end{aligned}$$

**HrfxyTotal[u1x, u1y, u2x, u2y, u3x, u3y]** returns the *sum* of the three corresponding rf-Hamiltonians in the `(ux, uy)` parametrization:

$$\begin{aligned} \text{HrfxyTotal}[\text{u1x}, \text{u1y}, \text{u2x}, \text{u2y}, \text{u3x}, \text{u3y}] := \\ \text{Hrfxy}[1, \text{u1x}, \text{u1y}] + \text{Hrfxy}[2, \text{u2x}, \text{u2y}] + \text{Hrfxy}[3, \text{u3x}, \text{u3y}] \end{aligned}$$

**HrfTotal[ampl1, ph1, ampl2, ph2, ampl3, ph3]** also returns the *sum* of the three corresponding rf-Hamiltonians, in the amplitude and phase parametrization:

$$\begin{aligned} \text{HrfTotal}[\text{ampl1}, \text{ph1}, \text{ampl2}, \text{ph2}, \text{ampl3}, \text{ph3}] := \\ \text{Hrf}[1, \text{ampl1}, \text{ph1}] + \text{Hrf}[2, \text{ampl2}, \text{ph2}] + \text{Hrf}[3, \text{ampl3}, \text{ph3}] \end{aligned}$$

### Example ( Hrfxy, Hrf, HrfxyTotal, HrfTotal )

```
In:
Hrfxy[3, 25, 75];
HrfxyTotal[0, 0, 0, 0, 25, 75] (*= Hrfxy[3, 25, 75]*)
Hrf[2, 100, 90]; (*ampl = 100, phase = 90°*)
HrfTotal[0, 0, 100, 90, 0, 0] (*= Hrf[2, 100, 90]*)
```

Out:

Too large to be displayed

$$\begin{pmatrix} 0 & 0 & -100i\pi & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -100i\pi & 0 & 0 & 0 & 0 \\ 100i\pi & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100i\pi & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -100i\pi & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -100i\pi \\ 0 & 0 & 0 & 0 & 100i\pi & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100i\pi & 0 & 0 \end{pmatrix}$$

## 10 Evolution and acquiring coefficients

**Evolve**[**mtx**,**H**,**t**] returns the matrix resulting of the evolution of **mtx** under the constant Hamiltonian **H** for a time **t**.

### Example ( Evolve )

```
In:
rho = II[2,x];
nu = 100;
H = Hcshift[2,nu];
t = 1/(4*nu);
rho2 = Evolve[rho,H,t];
Decomposition[rho,II]
Decomposition[rho2,II]
```

Out:

```
I2x
I2y
```

**MultiEvolve**[**mtx**,{**H1**,...**Hk**},{**t1**,...**tk**}] returns the matrix resulting of the evolution of **mtx** under successive constant Hamiltonians **H1**,...**Hk** for the respective times **t1**,...**tk**.

### Example ( MultiEvolve )

```
In:
rho = II[1,z] + II[2,z]; (*initial density matrix*)
ampl = 100;
ph = 90;
H1 = HrfTotal[ampl,ph,ampl,ph,0,0]; (*rf-Hamiltonian*)
t1 = 1/(4*ampl);
Jc = 50;
Hc = Hplan[1,2,Jc]; (*coupling Hamiltonian*)
t2 = 1/(2*Jc);
rho2 = MultiEvolve[rho,{H1,Hc},{t1,t2}];
Decomposition[rho,II]
Decomposition[rho2,II]
```

Out:

$$I_{1z} + I_{2z}$$

$$-2I_{1y}I_{2z} - 2I_{1z}I_{2y}$$

**Acquire** [**mtx**, **H**, **target**, **t**, **npt**] returns the list of npt+1 coefficients for target in the decomposition of **mtx**, when **mtx** evolves under the Hamiltonian **H** for a time **t**. In other words, **Acquire** computes Trace(**target**,**mtx**(**t**)) for the given time points.

**Default values : Acquire** [**mtx**, **H**, **target**, **t**, **npt**]

```
H = IdentityMatrix[8]
target = IdentityMatrix[8]
t = 0
npt = 0
```

**Example ( Acquire, from the Evolve example )**

```
In:
rho = II[2, x];
nu = 100;
H = Hcshift[2, nu];
t = 1/(4*nu);
rho2 = Evolve[rho, H, t];
Acquire[rho, H, II[2, y], t, 5]

Out:
{0, 0.309017, 0.587785, 0.809017, 0.951057, 1.}
```

**AcquireAll** [**mtx**, **H**, **t**, **npt**, **B**] returns a  $64 \times (npt+1)$  matrix. The  $i^{\text{th}}$  line corresponds to **Acquire** [**mtx**, **H**, **B**[[**i**]], **t**, **npt**]. Each column corresponds to the coefficients of **mtx**(**t**) in the basis **B**, obtained after evolution of **mtx** for time  $dt=t/npt$ , where the dynamics is governed by the constant Hamiltonian **H**.

**Default values : Acquire** [**mtx**, **H**, **t**, **npt**, **basis**]

```
H = IdentityMatrix[8]
t = 0
npt = 0
basis = Tsym
```

**Example ( AcquireAll )**

```
In:
mtx = II[1, x];
Jc = 20;
H = Hweak[1, 2, Jc];
t = 1/(2*Jc);
npt = 4;
A = AcquireAll[mtx, H, t, npt, IIInorm];
A[[1]] (*coeff. of II[1, x]*)
A[[2]] (*coeff. of II[1, y]*)
A[[3]] (*coeff. of II[1, z]*)
```

```
A[[15]] (*coeff. of 2 II[1,y,2,z]*)
```

```
Out:
```

```
{1., 0.92388, 0.707107, 0.382683, 0.}
{0., 0., 0., 0., 0.}
{0., 0., 0., 0., 0.}
{0., 0.382683, 0.707107, 0.92388, 1.}
```

**MultiAcquireAll**[**mtx**, {**H1**, ..., **Hk**}, {**t1**, ..., **tk**}, {**n1**, ..., **nk**}, **B**] returns a matrix of dimension  $64 \times (n_1 + n_2 + \dots + n_k + 1)$  which corresponds to the coefficients of **mtx** in the basis **B** during its evolution under the Hamiltonians {**H1**, **H2**, ..., **Hk**} for times {**t1**, **t2**, ..., **tk**}, for a selection of {**n1**, ..., **nk**} points for each interval. Each column corresponds to the coefficients of **mtx**(**t**) in the basis **B** at a certain time.

#### Example ( multiAcquireAll, from the multiEvolve example)

```
In:
mtx = II[1,z];
(*From equilibrium to transverse magnetization*)
MAXRF = 10000;
H1 = Hrfxy[1,0,MAXRF];
t1 = 1/4*(1/MAXRF);
npt1 = 4;
(*Evolution under strong coupling*)
Jc = 90;
H2 = Hweak[1,2,Jc];
t2 = 1/(2*Jc);
npt2 = 4;
A = MultiAcquireAll[mtx,{H1,H2},{t1,t2},{npt1,npt2},IIInorm];
Dimensions[A]
A[[3]] (*coeff. of II[1,z]*)
A[[1]] (*coeff. of II[1,x]*)
A[[15]] (*coeff. of 2 II[1,y,2,z]*)
```

```
Out:
```

```
{1., 0.92388, 0.707107, 0.382683, 0., 0., 0., 0., 0.}
{0., 0.382683, 0.707107, 0.92388, 1., 0.92388, 0.707107, 0.382683, 0.}
{0., 0., 0., 0., 0., 0.382683, 0.707107, 0.92388, 1.}
```

## 11 DROPS visualization

Given an irreducible spherical tensor basis  $B = \{T_{j,m}\}$  (ex. Tprod, Tsym and Tmpo) where many tensors may have the same rank and order  $(j, m)$ , a DROPS representation for  $B$  consists of division of the basis into group of operators such that all tensors in a group have a distinct pair  $(j, m)$ . Each group is labelled by an element  $\ell \in L$  and the tensors in the group  $\ell$  are uniquely denoted  $T_{j,m}^{(\ell)}$ . For any operator  $A$  in the space generated by the basis,  $A^{(\ell)}$  denotes the projection of  $A$  on the sub-space generated by  $\{T_{j,m}^{(\ell)}\}$ . The DROPS visualization of  $A$  is then performed via the  $|L|$  mappings

$$A = \sum_{\ell \in L} A^{(\ell)} \iff \bigcup_{\ell \in L} f_A^{(\ell)}(\theta, \phi) \quad (3)$$

where

$$A^{(\ell)} = \sum_{j,m} c_{jm}^{(\ell)} T_{jm}^{(\ell)} \iff f_A^{(\ell)}(\theta, \phi) = \sum_{j,m} c_{jm}^{(\ell)} Y_{jm}(\theta, \phi). \quad (4)$$

The functions  $Y_{jm}(\theta, \phi)$  are the well-known spherical harmonics and the functions  $f_A^{(\ell)}$  are called *droplets*.

## 11.1 Droplets ordering

To each of the three tensor bases `Tprod`, `Tsym` and `Tmpo` corresponds a specific partition of the basis used to define the associated DROPS representation. For each set, we present the grouping labels  $\ell$  and the tensors tagged by these labels.

### 11.1.1 Tprod labels and grouping

The `Tprod` tensor basis is divided into 10 droplets. The labels  $\ell \in L$  are chosen to represent a specific spin sub-system for the linear and bilinear operators, and a specific parent couple for the trilinear operators. Each operator in the `Tprod` basis then belongs to one of the 10 following sets:

labels $\ell \in L$	Basis operators $T_{jm}^{(\ell)}$
1	{Tprod[1, 1, m]}
2	{Tprod[2, 1, m]}
3	{Tprod[3, 1, m]}
(1, 2)	{Tprod[1, 2, j, m]}
(1, 3)	{Tprod[1, 3, j, m]}
(2, 3)	{Tprod[2, 3, j, m]}
{0, 1}	{Tprod[1, 2, 3, j, m, {0, 1}]}
{1, 1}	{Tprod[1, 2, 3, j, m, {1, 1}]}
{2, 1}	{Tprod[1, 2, 3, j, m, {2, 1}]}
$\emptyset$	Tprod[0, 0, 0]

### 11.1.2 Tsym labels and grouping

The `Tsym` tensor basis is divided into 11 droplets. The labels  $\ell \in L$  are chosen to represent a specific spin sub-system for the linear and bilinear operators, and a specific symmetry specie for the trilinear operators. Each operator in the `Tsym` basis then belongs to one of the 11 following

sets:

labels $\ell \in L$	Basis operators $T_{jm}^{(\ell)}$
1	{Tsym[1, 1, m]}
2	{Tsym[2, 1, m]}
3	{Tsym[3, 1, m]}
(1, 2)	{Tsym[1, 2, j, m]}
(1, 3)	{Tsym[1, 3, j, m]}
(2, 3)	{Tsym[2, 3, j, m]}
$\tau_1^{[3]}$	{Tsym[1, 2, 3, j, m, 1]}
$\tau_2^{[3]}$	{Tsym[1, 2, 3, j, m, 2]}
$\tau_3^{[3]}$	{Tsym[1, 2, 3, j, m, 3]}
$\tau_4^{[3]}$	{Tsym[1, 2, 3, j, m, 4]}
$\emptyset$	Tsym[0, 0, 0]

### 11.1.3 Tmpo labels and grouping

The Tmpo tensor basis is divided in 9 droplets. This time, the labels  $\ell \in L$  characterizing the tensors are not related to any spin sub-system but to the sets between which operator is making a transition. Any operator in the Tmpo basis belongs to one of the following set:

labels $\ell \in L$	Basis operators $T_{jm}^{(\ell)}$
(S1, S1)	{Tmpo[j, m, S1, S1]}
(S1, S2)	{Tmpo[j, m, S1, S2]}
(S1, S3)	{Tmpo[j, m, S1, S3]}
(S2, S1)	{Tmpo[j, m, S2, S1]}
(S2, S2)	{Tmpo[j, m, S2, S2]}
(S2, S3)	{Tmpo[j, m, S2, S3]}
(S3, S1)	{Tmpo[j, m, S3, S1]}
(S3, S2)	{Tmpo[j, m, S3, S2]}
(S3, S3)	{Tmpo[j, m, S3, S3]}

## 11.2 Coefficient grouping function

**PartitionCoeffList[listCoeff, B]** takes a list listCoeff of 64 coefficients and converts it into a list of droplet coefficients {D1, D2, ..., Dk}. The droplet coefficients  $D_k = \{a_1, a_2, \dots\}$  assigns to each coefficient the corresponding spherical harmonics {Y<sub>0,0</sub>, Y<sub>1,-1</sub>, Y<sub>1,0</sub>, Y<sub>1,1</sub>, Y<sub>2,-2</sub>, ...}, in this order. If there is no tensor of rank  $j$  and order  $m$  in the composition of a droplet, the corresponding position (assigned to  $Y_{j,m}$ ) is filled with a zero. The droplet coefficients in D1 are used to construct the first droplet associated to the basis B, the order of the droplets being given by their order of appearance in Section 11.1. Similarly, D2 defines the second droplet, and so on.

The list of coefficient listCoeff can be generated via the function CoeffList[mtx, basis] defined in Section 7.

#### Example ( PartitionCoeffList )

```
In:
CList = CoeffList[Tsym[1, 1, -1], Tsym]
PartitionCoeffList[CList, Tsym]
```



section 11.4). The options corresponding to the sets can be seen using the commands `Options[DropletColorQuick]`, `Options[DropletColor]`, `Options[DropletGrayQuick]` and `Options[DropletGray]`.

- `label` corresponds to the label of the droplet.
- `PlotLabel` is a boolean variable specifying if the label should be displayed (1) or not (0).
- `PlotSphere` is a boolean variable specifying if a gray sphere or radius 0.5 should be displayed (1) or not (0).
- `LabelPosition` specifies the position of the label with respect to the center of the droplet.

#### Example 1 ( MyPlotCoeff - Options )

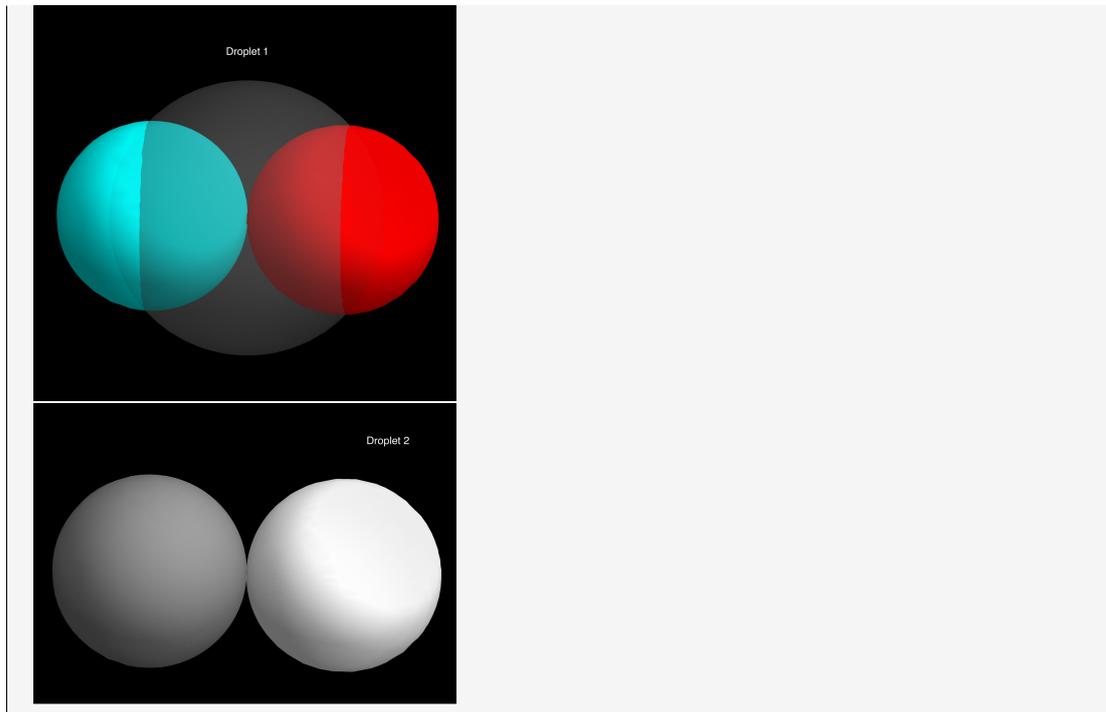
```
In:
Options[MyPlotCoeff] (*Available options and default values*)
(*****)
Options[DropletColorQuick] (*Options in the dropletOutput set*)

Out:
{position -> {0,0,0}, dropletOutput -> DropletColorQuick, label ->
"", PlotLabel -> 0, PlotSphere -> 1, LabelPosition -> {0, 0, 5/8}}
(*****)
{PlotPoints -> 10, Mesh -> None, PlotRange -> All,
ColorFunctionScaling -> False, coloring -> 1, PlotStyle ->
Directive[RGBColor[1, 0, 0], Opacity[0.9],
Specularity[GrayLevel[1], 10]], ViewPoint -> 2, -25, 7, Axes ->
False, AxesOrigin -> {0, 0, 0}, AxesLabel -> Automatic, BoxRatios
-> Automatic, Boxed -> False, GroundSphere -> GrayLevel[0.4,
0.4], TextColor -> GrayLevel[1], TextRadius -> 0, -0.72,
0.609375, Background -> GrayLevel[0]}
```

#### Example 2 ( MyPlotCoeff )

```
In:
MyPlotCoeff[{0,1,0,-1},dropletOutput->DropletColor,label->"Droplet
1",PlotLabel->1]
(*****)
MyPlotCoeff[{0,1,0,-1},dropletOutput->DropletGray,label->"Droplet
2",PlotLabel->1,PlotSphere->0,LabelPosition->{0.5,0,0.5}]

Out:
```



### 11.3.2 DROPS visualization of a matrix

**PlotMatrix**[*mtx*, <options>] plot the DROPS representation of *mtx*. The possible plotting options with their default values are:

**Default values : PlotMatrix**[*mtx*, <options>]

```
basis -> Tsym
Output -> ColorQuick
Linear -> 1
Bilinear -> 1
Trilinear -> 1
Id -> 1
Labels -> 1
```

where

- *basis* specifies in which tensor basis the DROPS visualization must be performed. The possible values are *Tprod*, *Tsym* and *Tmpo*.
- *Output* specifies the type of plotting to be performed. The actual built-in *Output* options are *ColorQuick*, *Color*, *GrayQuick*, *Gray*. It is also possible to defined new *Output* options (see section 11.4)
- *Linear*, *Bilinear*, *Trilinear* and *Id* are boolean variables which precise if the linear, bilinear, trilinear and identity droplets must be plotted. The possible values are 0 or 1. These options only make sense for the bases *Tprod* and *Tsym*\*
- *Labels* is a boolean variable which precise if the labels must be displayed. The possible values are 0 or 1.

\* In fact, setting `Linear->0` with the basis `Tmpo` would set to zero the three droplets corresponding to the transitions  $(S_1, S_i)$ . Similarly, setting `Bilinear->0` and/or `Trilinear->0` with the basis `Tmpo` would set to zero the three droplets corresponding to the transitions  $(S_2, S_i)$  and/or  $(S_3, S_i)$  respectively.

### Example 1 (Options for PlotMatrix)

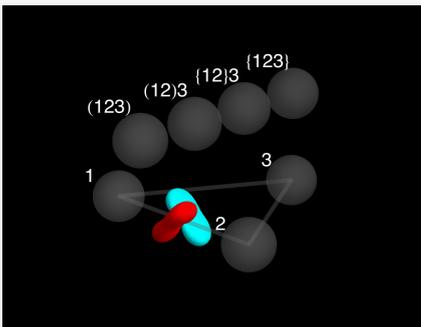
```
In:
Options[PlotMatrix]
(*****)
Options[ColorQuickTsym]
Options[ColorQuickTprod]
Options[ColorQuickTmpo]

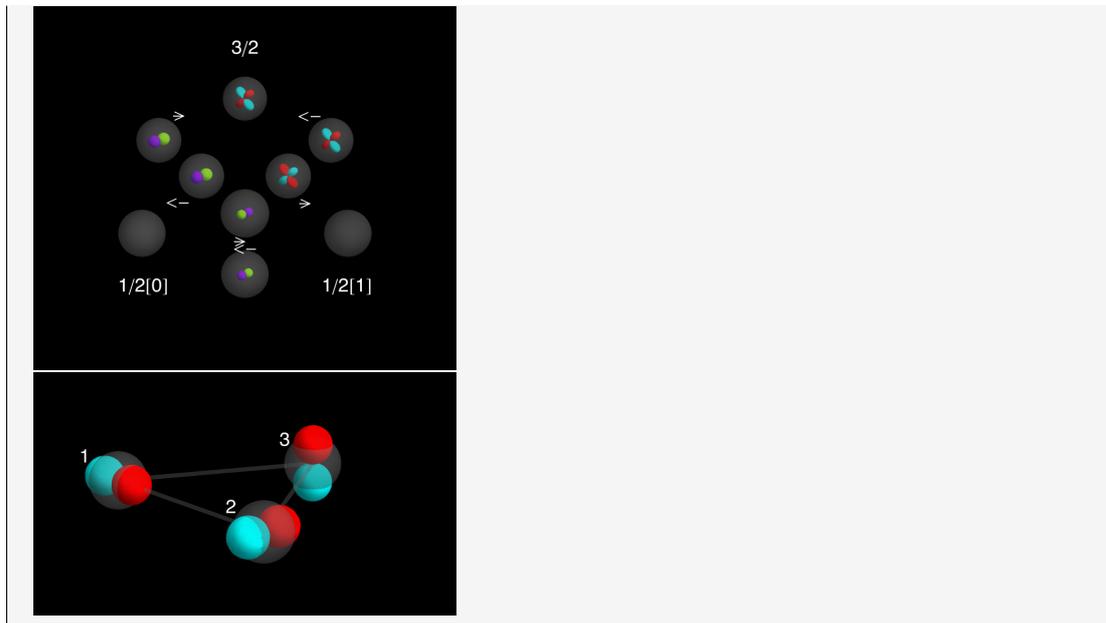
Out:
{basis -> Tsym, Output -> ColorQuick, Linear -> 1, Bilinear -> 1,
Trilinear -> 1, Id -> 1, Labels -> 1}
(*****)
{DropletParam -> DropletColorQuick, DROPSParam -> DROPSTsym,
ConnectShape -> LinGray}
{DropletParam -> DropletColorQuick, DROPSParam -> DROPSTprod,
ConnectShape -> LinGray}
{DropletParam -> DropletColorQuick, DROPSParam -> DROPSTmpo,
ConnectShape -> LinGrayTmpo}
```

### Example 2 ( PlotMatrix )

```
In:
PlotMatrix[2 II[1,x,2,z]]
PlotMatrix[2 II[1,x,2,z],basis->Tmpo]
PlotMatrix[II[1,x]+II[2,y]+II[3,z],Trilinear->0]
```

Out:





### 11.3.3 DROPS visualization from a list of 64 coefficients

**PlotFromCoeff[CoeffList, B, <options>]** plots the DROPS representation of the operator which has decomposition coefficients *CoeffList* in the basis *B*. The possible plotting options with their default values are:

#### Default values : **PlotFromCoeff[CoeffList, options]**

```
Output -> ColorQuick
Linear -> 1
Bilinear -> 1
Trilinear -> 1
Id -> 1
Labels -> 1
```

The description of these different options can be found in the section 11.3.2 dedicated to the function `PlotMatrix`. Note that unlike `PlotMatrix`, the function `PlotFromCoeff` takes the basis as a mandatory parameter.

#### Example ( **PlotFromCoeff** )

```
In:
Options[PlotFromCoeff]
mtx = Tprod[1, 2, 3, 3, 1, {2, 1}] + II[1, x] + 2 II[2, z, 3, z];
Dec = CoeffList[mtx, Tsym]
PlotFromCoeff[Dec, Tsym, Output->Color]
```

```
Out:
{Output -> ColorQuick, Linear -> 1, Bilinear -> 1, Trilinear ->
1, Id -> 1, Labels -> 1}
```



```
ViewPoint -> {2,-25,7},
Axes -> False,
AxesOrigin -> {0,0,0},
AxesLabel -> Automatic,
BoxRatios -> Automatic,
Boxed -> False,
GroundSphere -> GrayLevel[0.4,0.4],
TextColor -> White,
TextRadius -> {0,-0.72,0.975*15/24},
Background -> Transparent
};
```

Out:

Most of these options are inherited from the Mathematica built-in function `Plot` options. The four options unique to the `DropletParam` sets are presented below.

- `coloring` is a boolean variable specifying if the plot should be colored (1) or be black and white (0).
- `GroundSphere` defines the coloring property of the sphere (of radius 0.5) proper to a droplet.
- `TextColor` specifies the color of the text.
- `TextRadius` specifies the position of the droplet label *with respect to the center* of the droplet.

#### 11.4.2 DROPSParam option sets

An option set of type `DROPSParam` contains the display information for the global `DROPS` picture: the relative position of each droplet, the spheres to be plotted (or not), the labels, etc.

**Name setting:** The **name** of a set of type `DROPSParam` is again arbitrary, but we advise the use of the prefix `DROPS` when defining such an option set, to remember that these options concern the global `DROPS` representation (in contrast with the droplet options). Moreover, it is strongly advised to also use the basis name as a suffix to remember that such a set is defined differently for different bases (the number and order of droplets differ). For instance, the `DROPSParam` option sets available with this package are named `DROPSTsym`, `DROPSTprod` and `DROPSTmpo`.

The declaration of a `DROPSParam` option set has the following structure.

#### Example ( `DROPSParam` option set declaration for `ColorQuickTsym` )

```
In:
Options[DROPSTsym] = {
PositionsAndNames -> { {2 Cos[7 Pi/6],2 Sin[7 Pi/6],0,"1"}, {2
Cos[11 Pi/6],2 Sin[11 Pi/6],0,"2"}, {2 Cos[Pi/2],2
Sin[Pi/2],0,"3"}, {Cos[9 Pi/6],Sin[9 Pi/6],0,"12"}, {Cos[5
Pi/6],Sin[5 Pi/6],0,"13"}, {Cos[Pi/6],Sin[Pi/6],0,"23"},
{0,-2.0,1.8, "(123)"}, {0,-0.65,1.8, "(12)3"}, {0,0.65,1.8, "{12}3"},
{0,2.0,1.8, "{123}"}, {0,0,0,"Id"}},
SpheresToPlot -> {1,1,1,0,0,0,1,1,1,1,0},
```

```

NamesToPlot -> {1,1,1,0,0,0,1,1,1,1,0},
LabelPositions -> {{0,-0.72,0.61}, {0,-0.72,0.61}, {0,-0.72,0.61},
{0,-0.72,0.61}, {0,-0.72,0.61}, {0,-0.72,0.61}, {0,-0.72,0.81},
{0,-0.72,0.81}, {0,-0.72,0.81}, {0,-0.72,0.81}, {0,-0.72,0.61}},
ImageSize -> 200,
PlotRange -> All,
BoxRatios -> Automatic,
Boxed -> False,
Axes -> False,
AxesOrigin -> {0,0,0},
AxesLabel -> Automatic,
Ticks -> False,
ViewPoint -> {4,-4,2},
Background -> Black
};

Out:

```

Here again, most of these options are inherited from the Mathematica built-in function `Plot`. The four options unique to the `DROPSParam` sets are presented now.

- `PositionsAndNames` is a list of 4-tuples, each of which corresponds to the position of the corresponding droplet and its label. There are as much 4-tuples as there are droplets in the basis to which it applies. The droplet ordering in each basis is found in Section 11.1.
- `SphereToPlot` is a list of boolean values. The *i*th value specifies if a ground sphere should be displayed for the *i*th droplet (1) or not (0). The length of this list of boolean values equals the number of droplets in the basis.
- `NamesToPlot` is a list of boolean values. The *i*th value specifies if the name of the *i*th droplet should be displayed (1) or not (0). The length of this list equals the number of droplets in the basis.
- `LabelPositions` is a list of the positions for the labels. The *i*th 3-tuple specifies the position of the label of the *i*th droplet *with respect to the center* of the droplet. There are as many 3-tuples as there are droplets in the basis.

### 11.4.3 ConnectShape option set

The option set of type `ConnectShape` contains the display information for any additional shape/line to appear in the DROPS visualization.

```

Example ( ConnectShape option set declaration for ColorQuickTsym )
In:
Options[LinGray] = {
ToPlot -> 1,
Shape -> {{2 Cos[7 Pi/6],2 Sin[7 Pi/6],0}, {2 Cos[11 Pi/6],2
Sin[11 Pi/6],0}, {2 Cos[Pi/2],2 Sin[Pi/2],0}, {2 Cos[7 Pi/6],2
Sin[7 Pi/6],0}},
Color -> GrayLevel[0.4,0.4],
Line -> Thick
};

```

```
Out:
```

The options of type `ConnectShape` are detailed below.

- `ToPlot` precise if a shape should be plotted (1) or not (0). If its is set to zero, no other option need to be defined.
- `Shape` is a list of points to be connected together.
- `Color` precise the color of the shape to be displayed.
- `Line` precise the thickness of the shape.

#### 11.4.4 Output plotting options

Once the three option sets (of type `DropletParam`, `DROPSPParam` and `ConnectShape`) are defined, for the basis `Basis`, the final the Output plotting options can finally be defined by the declaration

```
Options[<Output><Basis>] =  
{DropletParam->..., DROPSPParam->..., Connectshape->... }
```

**IMPORTANT!!!** The Output name **must have the basis as a suffix, the Output name as a prefix, and nothing in-between**. Indeed, all the plotting functions and the experiment visualization functions (except for `MyPlotCoeff` which plots a single droplet) call the output name `<Output>` and the `<Basis>` separately.

Proceeding with the Output option set `ColorQuickTsym`, where here `<Output>=Colorquick` and `<Basis>=Tsym`, let us recall what have been declared so far:

option set	Declaration
<code>DropletParam</code>	<code>Options[DropletColorQuick]=...</code>
<code>DROPSPParam</code>	<code>Options[DROPSTsym]=...</code>
<code>ConnectShape</code>	<code>Options[Lingray]=...</code>

Then the declaration of the option set `ColorQuickTsym` has the form as given in the following example.

#### Example ( Output option set declaration for `ColorQuickTsym` )

```
In:  
Options[ColorQuickTsym] = {  
DropletParam -> DropletColorQuick,  
DROPSPParam -> DROPSTsym,  
ConnectShape -> LinGray  
};  
  
Out:
```

Once an Output option set is defined, under the name `<Output><Basis>`, it can be called by any plotting functions under its output name `<Output>`, with the proper basis. The Output names available with this package are `ColorQuick`, `Color`, `GrayQuick` and `Gray`, defined for all three tensor bases.

## 12 Visualization of experiments

### 12.1 Pulse file format

A pulse file is a file containing three columns with the duration ( $\mu\text{s}$ ), the relative amplitude (in %) and the phase (rank) of each pulse slice.

One line of the file (generally the first one) must contain the information of the maximal (corresponding to 100%) radio frequency (Hz) in the following form:

```
#MAX_RF 15000
```

for instance. The other lines beginning with # are considered as comment lines and are ignored in the reading process.

For example, the file `pulse1.txt` corresponding to the pulse sequence acting on the first particle in the INEPT experiment has the form

```
#Max_RF 10000
25 100 0
1180 0 0
50 100 0
1180 0 0
25 100 90
```

### 12.2 Reading pulse files

`ImportPulseSequence[filename]` reads a pulse file and returns  $\{L, \text{MAX\_RF}\}$  where  $L$  is a  $n \times 3$  matrix the lines of which are:

$$\{\text{duration}(i), \text{amplitude}(i), \text{phase}(i)\}, 1 \leq i \leq n.$$

If the file is not in the current directory (type `Directory[]` to see the current directory), then filename should have the form of the full path to find the directory.

**Example ( `ImportPulseSequence` ) for the file `pulse1.txt` which is in the directory `BasicINEPT` itself in the notebook directory**

```
In:
File1 = FileNameJoin[{NotebookDirectory[] <> "/BasicINEPT/"
<> "pulse1.txt"}];
ps1 = ImportPulseSequence[File1]
```

```
Out:
{{{25, 100, 0}, {1205, 0, 0}, {1255, 100, 0}, {2435, 0, 0},
{2460, 100, 90}}, 10000}
```

### 12.3 Combining the Hamiltonians generated from the pulse files

`CombinePulseSequences[ps1, ps2, ps3, Hoffset, scalingFactor]` combine the three pulse sequences `ps1`, `ps2`, `ps3` generated by the `ImportPulseSequence` method, where each pulse acts on the corresponding particle. An empty pulse sequence can be set by hand and has the form

```
emptypulse:={{}, 0}.
```

The pulses are merged together with the offset Hamiltonian `Hoffset`. The output is a matrix with lines

```
{time, Hamiltonian, {amp11, phase1}, {amp12, phase2}, {amp13, phase3}}
```

for the consecutive time durations with distinct Hamiltonians. The scaling factor `scalingfactor` converts the application of a radio-frequency Hamiltonian  $H$  for a time  $t$  to a Hamiltonian  $\frac{H}{\text{scalingFactor}}$  for a duration `scalingFactor*t`. The scaling factor must satisfy `scalingfactor > 1`.

**Default values :**

**CombinePulseSequences [ps1, ps2, ps3, Hoffset, scalingFactor]**

`scalingfactor = 1`

**Example ( CombinePulseSequences ) for the files `pulse1.txt` and `pulse2.txt` which are in the directory `BasicINEPT` itself in the notebook directory**

In:

```
File1 = FileNameJoin[{NotebookDirectory[] <> "/BasicINEPT/" <> "pulse1.txt"}];
File2 = FileNameJoin[{NotebookDirectory[] <> "/BasicINEPT/" <> "pulse2.txt"}];
ps1 = ImportPulseSequence[File1];
ps2 = ImportPulseSequence[File2];
ps3 = {{}, 0};
Hoffset = HweakTotal[212, 0, 0] + HcshiftTotal[30, 10, 0];
CombinePulseSequences[ps1, ps2, ps3, Hoffset, 2]
```

Out:

Too large to be displayed

## 12.4 Creating the density matrices, Hamiltonians, effective Hamiltonian, propagators and effective propagators for an experiment

**CreateAllMatrices [InitialDensity, CombinedPulseSequence, timepoints, <options>]** uses the output of the `CombinePulseSequences` method to evolve the initial density operator `InitialDensity` in time.

`timepoints` specifies the sampling points on the time scale (possibly adjusted by the scaling factor). It is either a positive value (for the given number of evenly distributed time points over the whole pulse sequence) or a list of time values ( $\mu\text{s}$ ), at which the matrices are calculated each. The output is a list of five matrices lists corresponding to the density matrices, propagators, effective propagators, Hamiltonians and effective Hamiltonians in this order. The possible options are

- `totalTime` which specifies the overall time of the evolution. If set to `Automatic`, this time is the minimum time to include all the pulses if there is and is set to 1000 if there is just evolution (no pulses).
- `report` prints the progress status. The possible values are `True` or `False`.

```
Default values : CreateAllMatrices[InitialDensity,  
CombinedPulseSequences, timepoints, <options>
```

```
report -> False  
totalTime -> Automatic
```

```
Example ( CreateAllMatrices ) continuing the example for  
CombinePulseSequences
```

```
In:  
cps = CombinePulseSequences[ps1,ps2,ps3,Offset,2];  
timepoints = 4;  
AllMatrices = CreateAllMatrices[II[1,z],cps,timepoints];  
  
(*** Assignment of different matrices lists ***)  
Densities = AllMatrices[[1]];  
Propagators = AllMatrices[[2]];  
EffPropagators = AllMatrices[[3]];  
Hamiltonians = AllMatrices[[4]];  
EffHamiltonians = AllMatrices[[5]];
```

```
Out:
```

## 12.5 Creating slides

When many matrices has to be plotted, it is advantageous to create files with the corresponding DROPS visualizations instead of plotting them in the Notebook as it can be long and also to use them for doing animations. The last methods presented all have as a function to create “slides” from matrices, coefficients array or from pulse files.

### 12.5.1 Evolution of a density matrix

**SlidesEvolve[mtx, H, t, npt, slidesName, <options>]** returns npt+1 picture files corresponding to the evolution of  $mtx$  under the action of the Hamiltonian  $H$  for a time  $t$ . The slides are stored in the current directory (type `Directory[]` to see what the current directory is and use `SetDirectory` to change it). The possible options are presented below.

- `basis` precises the basis in which to generate the DROPS pictures. It can be `Tprod`, `Tsym` or `Tmpo`.
- `Extension` precises the type of file to generate. It can be for instance `".png"`, `".eps"`, `".pdf"`, etc.
- `Resolution` sets the quality of the pictures.
- Inherits of all the options from the function `PlotCoefficients` (see Section 11.3.3).

```
Default values : SlidesEvolve[mtx, H, t, npt, slidesName, <options>]
```

```
H = Sqrt[2] (II[1,z]+II[2,z]+II[3,z])  
t = 2 Pi  
npt = 16
```

```

slidesName = "slides"
(*For options*)
basis -> Tsym
Extension -> ".png"
Resolution -> 200
(*From PlotFromCoeff*)
Output -> ColorQuick
Linear -> 1
Bilinear -> 1
Trilinear -> 1
Id -> 1
Labels -> 1

```

#### Example ( SlidesEvolve )

```

In:
Options[SlidesEvolve] (*remind default options*)
SetDirectory[NotebookDirectory[]]; (*place to store*)
H = HcshiftTotal[10,10,10];
t = 2*Pi;
npt = 1;
SlidesEvolve[4*II[x,y,x],H,t,npt,"SlidesA",basis -> Tsym,
Resolution -> 200]

```

Out:

```

{basis -> Tsym, Extension -> ".png", Resolution -> 200, Output ->
ColorQuick, Linear -> 1, Bilinear -> 1, Trilinear -> 1, Id -> 1,
Labels -> 1}

```

( Creation of two slides named "SlidesA.1.png" and "SlidesA.2.png" in the Notebook directory )

### 12.5.2 Slides from a matrix of coefficients

**SlidesCoeff**[**CoeffArray**, **slidesName**, **<options>**] take a  $64 \times N$  matrix of coefficients (as returned by the `Acquire`, `AcquireAll`, `MultiAcquireAll` methods (see section 10) for instance) and creates  $N$  slides out of it. The slides are stored in the current directory (type `Directory[]` to see what the current directory is and use `SetDirectory` to change it). The possible options values are the same as those for the `SlidesEvolve` method (see section 12.5.1).

#### Default values : **SlidesCoeff**[**CoeffArray**, **slidesName**, **<options>**]

```

slidesName = "slides"
(*For options*)
Identical to the SlidesEvolve method options.

```

#### Example ( SlidesCoeff )

```

In:
Options[SlidesCoeff] (*remind default options*)

```

```

SetDirectory[NotebookDirectory[]]; (*place to store*)
H = HcshiftTotal[10,10,10];
t = 2*Pi;
npt = 1;
B = Tsym;
CoeffArray = AcquireAll[4*II[x,y,x],H,t,npt,B];
SlidesCoeff[CoeffArray,"SlidesB", Output -> ColorQuick,
Resolution -> 250];

```

Out:

```

{basis -> Tsym, Extension -> ".png", Resolution -> 200, Output ->
ColorQuick, Linear -> 1, Bilinear -> 1, Trilinear -> 1, Id -> 1,
Labels -> 1}

```

( Creation of two slides named "SlidesB.1.png" and "SlidesB.2.png" in the Notebook directory )

### 12.5.3 Slides from a list of matrices

**SlidesMatrices[Listmat, slidesName, <options>]** takes a list of N matrices (as returned by the `CreateAllMatrices` method for instance) and creates N slides out of it. The slides are stored in the current directory (type `Directory[]` to see what the current directory is and use `SetDirectory` to change it). The possible options values are the same as those for the `SlidesEvolve` method (see section 12.5.1).

**Default values : SlidesMatrices[ListMat, slidesName, <options>]**

`slidesName = "slides"`

(\*For options\*)

Identical to the `SlidesEvolve` method options.

#### Example ( **SlidesMatrices** ) using **CreateAllMatrices**

In:

```

File1 = FileNameJoin[{NotebookDirectory[] <> "/BasicINEPT/"
<> "pulse1.txt"}];
File2 = FileNameJoin[{NotebookDirectory[] <> "/BasicINEPT/"
<> "pulse2.txt"}];
ps1 = ImportPulseSequence[File1];
ps2 = ImportPulseSequence[File2];
ps3 = {{}, 0};
cps = CombinePulseSequences[ps1, ps2, ps3, Hoffset, 2];
timepoints = 4;
AllMatrices = CreateAllMatrices[II[1, z], cps, timepoints];
Hamiltonians = AllMatrices[[4]];
SlidesMatrices[Hamiltonians, "TheHamiltonians"];

```

Out:

( Creation of two slides named "TheHamiltonians.1.png" and "TheHamiltonians.2.png" in the Notebook directory )

#### 12.5.4 Slides from pulse files

**SlidesPulseFile**[**ExperimentName**, **InitialDensity**, **<options>**] reads pulse files and produces slides from the matrices created by the method `CreateAllMatrices`. This method has many options, in particular the user can decide for which matrices should the slides be generated. By default, only slides visualizing the density matrices are generated. Different folders (named “hamiltonians”, “effectiveHamiltonians”, “propagators”, “effectivePropagators” and “densityMatrices”) are created, and placed in a single folder labeled “ExperimentName”. If the folder called `ExperimentName` already exists, a new one called `ExperimentName1` will be created instead. If `ExperimentName1` also already exists then the folder `ExperimentName2` will be created, and so on. The options unique to this method are presented below.

- `file1`, `file2` and `file3` are the options to which are assigned the pulse files.
- 

##### Default values :

##### **SlidesPulseFile**[**ExperimentName**, **InitialDensity**, **<options>**]

```
file1-> ""
file2-> ""
file3-> ""
hOffset -> Table[0,{i,8},{j,8}]
scalingfactor -> 1
timepoints -> 1
basis -> Tsym
densityMatrices -> 1
propagators -> 0
effectivePropagators -> 0
hamiltonians -> 0
effectiveHamiltonians -> 0
allMatrices -> 0
directory -> Directory[]
All the options for the SlidesEvolve method.
```

##### Example ( **SlidesPulseFile** )

```
In:
Options[SlidesPulseFile] (*remind default options*)
F1 = FileNameJoin[{NotebookDirectory[] <> "/BasicINEPT/" <>
"pulse1.txt"}];
F2 = FileNameJoin[{NotebookDirectory[] <> "/BasicINEPT/" <>
"pulse2.txt"}];
InitialDensity = II[1,z];
SlidesPulseFile["BasicINEPT", InitialDensity, file1 ->F1, file2
->F2, densityMatrices ->0, propagators ->1,basis->Tsym]
```

Out:

( list of all the options )

( Creation of the directory BasicInept1, in which there is the directory "propagators" containing "slide\_1.png" and "slide\_2.png" )

## References

- [1] Ernst, R. R.: *Principles of Nuclear Magnetic Resonance in one and two dimensions*, International Series of Monographs on Chemistry, Oxford University Press, Oxford (1990).