BasisChanger.py

Electric Field Basis Transformation

Linear to Circular Polarization Conversion

ElecSus Library Documentation

Abstract

This document provides comprehensive documentation for `BasisChanger.py`, which implements transformations between linear (x-y-z) and circular (L-R-z) polarization bases. These transformations are essential for connecting laboratory-frame measurements to atomic transition selection rules.

Contents

1 Theoretical Foundation

1.1 Polarization Bases for Light-Atom Interaction

Axiom 1 (Selection Rules)**.** *Atomic transitions obey selection rules expressed naturally in the spherical (circular) basis:*

$$\sigma^+ : \quad \Delta m = +1 \quad \textit{(left circular)} \tag{1}$$
$$\sigma^- : \quad \Delta m = -1 \quad \textit{(right circular)} \tag{2}$$
$$\pi : \quad \Delta m = 0 \quad \textit{(linear along } z\textit{)} \tag{3}$$

Definition 1 (Spherical Basis Vectors)**.** *The spherical (circular) unit vectors in terms of Cartesian vectors:*

$$\hat{e}_{+1} = -\frac{1}{\sqrt{2}}(\hat{x} + i\hat{y}) \quad \textit{(left circular, } \sigma^+\textit{)} \tag{4}$$
$$\hat{e}_{-1} = +\frac{1}{\sqrt{2}}(\hat{x} - i\hat{y}) \quad \textit{(right circular, } \sigma^-\textit{)} \tag{5}$$
$$\hat{e}_0 = \hat{z} \quad \textit{(linear, } \pi\textit{)} \tag{6}$$

Theorem 1 (Convention Choice)**.** *Following Auzinsh, Budker, and Rochester (2010), the convention used is:*

$$\hat{L} = \frac{1}{\sqrt{2}}(\hat{x} - i\hat{y}) \tag{7}$$
$$\hat{R} = \frac{1}{\sqrt{2}}(\hat{x} + i\hat{y}) \tag{8}$$

Note: Different sign conventions exist in the literature.

1.2 Transformation Matrices

Definition 2 (Linear to Circular Transformation)**.** *The transformation matrix from* (x, y, z) *to* (L, R, z)*:*

$$U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -i & 0 \\ 1 & i & 0 \\ 0 & 0 & \sqrt{2} \end{pmatrix} \tag{9}$$

Theorem 2 (Inverse Transformation)**.** *The inverse (circular to linear):*

$$U^{-1} = U^\dagger = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 \\ i & -i & 0 \\ 0 & 0 & \sqrt{2} \end{pmatrix} \tag{10}$$

The transformation is unitary: $U^\dagger U = \mathbb{1}$.

2 Line-by-Line Code Analysis

2.1 Module Imports

```python
import numpy as np
```

NumPy for complex array operations.

2.2 Linear to Circular: xyz_to_lrz

```python
def xyz_to_lrz(E_in):
    """ Convert from linear to circular bases """
    # create output array
    E_out = np.zeros_like(E_in,dtype='complex')

    # z-component doesn't change
    E_out[2] = E_in[2]
```

Initialize complex output array. The z-component is invariant under this transformation.

```python
    ## Following sign convention in
    ## 'Optically Polarised Atoms' by Auzinsh, Budker and Rochester
    ## OUP, 2010
    # L = 1./sqrt(2) * (x - iy)
    # R = 1./sqrt(2) * (x + iy)
    E_out[0] = 1./np.sqrt(2) * (E_in[0] - 1.j*E_in[1])
    E_out[1] = 1./np.sqrt(2) * (E_in[0] + 1.j*E_in[1])

    return E_out
```

$$E_L = \frac{1}{\sqrt{2}}(E_x - iE_y), \quad E_R = \frac{1}{\sqrt{2}}(E_x + iE_y) \tag{11}$$

Left circular is the combination with $-i$, right circular with $+i$.

2.3 Circular to Linear: lrz_to_xyz

```python
def lrz_to_xyz(E_in):
    """ Convert from circular to linear bases """
    # create output array
    E_out = np.zeros_like(E_in,dtype='complex')

    # z-component doesn't change
    E_out[2] = E_in[2]
```

Same initialization; z unchanged.

```python
    # x = 1. / sqrt(2) * [L + R]
    # y = 1.j / sqrt(2) * [L - R]
    E_out[0] = 1./np.sqrt(2) * (E_in[0] + E_in[1])
    E_out[1] = 1.j/np.sqrt(2) * (E_in[0] - E_in[1])

    return E_out
```

$$E_x = \frac{1}{\sqrt{2}}(E_L + E_R), \quad E_y = \frac{i}{\sqrt{2}}(E_L - E_R) \tag{12}$$

Inverse transformation recovers Cartesian components.

3 Verification

3.1 Round-Trip Identity

Theorem 3 (Invertibility)**.** *Applying both transformations in sequence:*

$$\texttt{lrz\_to\_xyz}(\texttt{xyz\_to\_lrz}(\vec{E})) = \vec{E} \tag{13}$$

Proof. Let $\vec{E} = (E_x, E_y, E_z)$. After `xyz_to_lrz`:

$$(E_L, E_R, E_z) = \left(\frac{E_x - iE_y}{\sqrt{2}}, \frac{E_x + iE_y}{\sqrt{2}}, E_z \right) \tag{14}$$

After `lrz_to_xyz`:

$$E'_x = \frac{1}{\sqrt{2}}(E_L + E_R) = \frac{1}{\sqrt{2}} \cdot \frac{2E_x}{\sqrt{2}} = E_x \tag{15}$$

$$E'_y = \frac{i}{\sqrt{2}}(E_L - E_R) = \frac{i}{\sqrt{2}} \cdot \frac{-2iE_y}{\sqrt{2}} = E_y \tag{16}$$

□

3.2 Special Cases

Input (xyz)	Output (LRz)	Polarization
$(1,0,0)$	$(1/\sqrt{2}, 1/\sqrt{2}, 0)$	Linear x
$(0,1,0)$	$(-i/\sqrt{2}, i/\sqrt{2}, 0)$	Linear y
$(1,-i,0)/\sqrt{2}$	$(1,0,0)$	Left circular
$(1,i,0)/\sqrt{2}$	$(0,1,0)$	Right circular
$(0,0,1)$	$(0,0,1)$	Linear z

Table 1: Example transformations

4 Physical Applications

4.1 Atomic Transition Rates

The transition amplitude for polarization $\hat{e}$ is:

$$d_{eg} = \langle e | \hat{e} \cdot \vec{d} | g \rangle \tag{17}$$

In circular basis, the selection rules are simple:

$$d_{\pm 1} \propto \delta_{m_e, m_g \pm 1}, \quad d_0 \propto \delta_{m_e, m_g} \tag{18}$$

4.2 Faraday/Voigt Effects

Light propagating through an atomic medium experiences different refractive indices for L and R components:

$$n_L \neq n_R \quad \Rightarrow \quad \text{Faraday rotation} \tag{19}$$

The basis transformation allows computing these effects from atomic susceptibilities.

5 Summary

The `BasisChanger.py` module provides:

1. `xyz_to_lrz(E_in)`: Convert linear to circular basis
2. `lrz_to_xyz(E_in)`: Convert circular to linear basis

Key features:

- Input/output are 3-component complex NumPy arrays
- z-component unchanged (quantization axis)
- Follows Auzinsh-Budker-Rochester sign convention
- Unitary transformation (preserves field amplitude)
- Essential for connecting lab frame to atomic selection rules

Usage:

```python
from BasisChanger import xyz_to_lrz, lrz_to_xyz
E_circular = xyz_to_lrz(np.array([1, 0, 0]))  # x-polarized
E_linear = lrz_to_xyz(np.array([1, 0, 0]))    # L-polarized
```