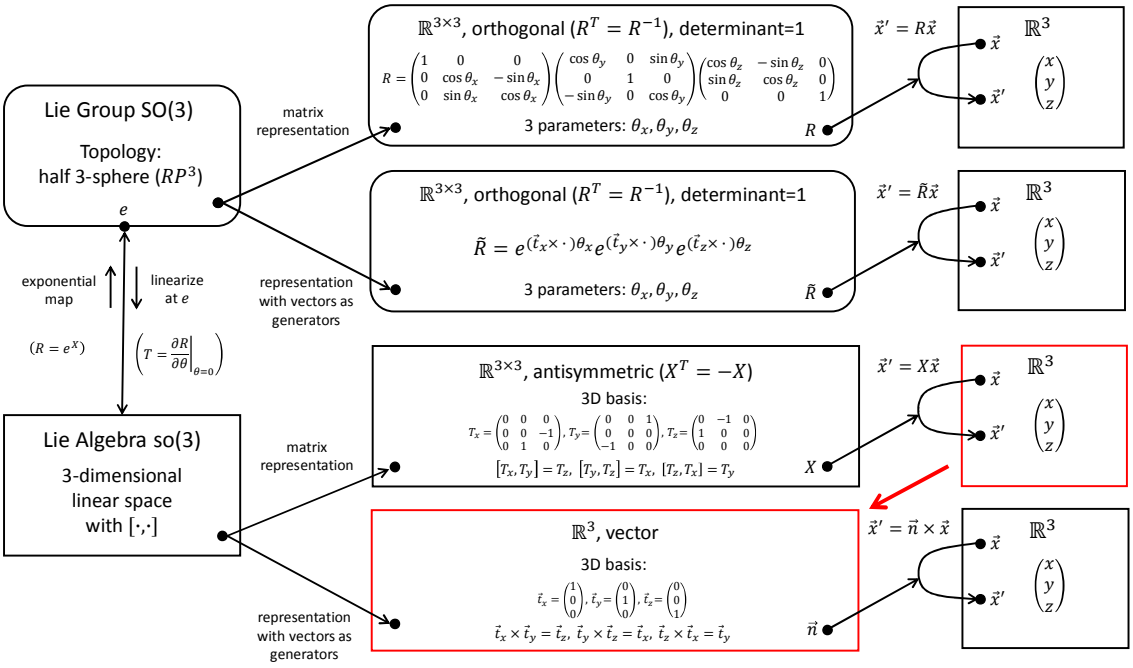
3.25 SO(3): Defining Representation with Vectors as Generators; Cross Product



Instead of representing the elements of so(3) by the usual 3×3 matrices (upper branch of the diagram), we can also represent them by 3-component column vectors (lower branch of the diagram). Remember, so(3) is just a 3-dimensional vector space plus a Lie-bracket operation. Now, the Lie-algebra elements do not only act *on* vectors but they *are* vectors themselves (red arrow in the diagram). But how do these "Lie-algebra vectors" act on the vectors in the representation space?

To represent a Lie-algebra matrix by a column vector, we expand the matrix X into a linear combination of the basis generators, $X = n_x T_x + n_y T_y + n_z T_z$, and put the three coefficients into a column vector $\vec{n} = (n_x, n_y, n_z)^T$. These vectors are now our new Lie-algebra elements. Next, we need to find a vector-vector operation that reproduces the results of the matrix-vector multiplication $\vec{x}' = X\vec{x}$. Amazingly, the good old *cross product* fits the bill: $\vec{x}' = \vec{n} \times \vec{x}$. More explicitly, we can write

$$\begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} n_y z - n_z y \\ n_z x - n_x z \\ n_x y - n_y x \end{pmatrix}.$$

But we are not done yet, we also need to find the new Lie-bracket operation. In the case of matrix elements, it was given by the commutator $[X, Y] = XY - YX$. What is the corresponding operation for our vector elements? Surprisingly, the cross product does this job too: $\vec{a} \times \vec{b} = \vec{c}$ corresponds to $[X, Y] = Z$, where $\vec{a}, \vec{b}, \vec{c}$ are the vectors corresponding to the matrices X, Y, Z, respectively. More explicitly, we can write the commutator $[X, Y]$ in terms of $\vec{a}$ and $\vec{b}$ as

$$\left[\begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix}, \begin{pmatrix} 0 & -b_z & b_y \\ b_z & 0 & -b_x \\ -b_y & b_x & 0 \end{pmatrix}\right] = \begin{pmatrix} 0 & a_y b_x - a_x b_y & a_z b_x - a_x b_z \\ a_x b_y - a_y b_x & 0 & a_z b_y - a_y b_z \\ a_x b_z - a_z b_x & a_y b_z - a_z b_y & 0 \end{pmatrix}$$

and identify the components of the resulting matrix Z with the components of $\vec{c}$ as $c_x = a_y b_z - a_z b_y$, $c_y = a_z b_x - a_x b_z$, and $c_z = a_x b_y - a_y b_x$, which is exactly the cross product $\vec{c} = \vec{a} \times \vec{b}$.

What is the meaning of the vectors $\vec{n}$ in our reformulated Lie algebra? Remember that the elements of a Lie algebra (= generators) correspond to infinitesimal transformations, in this case, infinitesimal 3D rotations. To describe such a rotation, all we need to specify is the axis of rotation, and this is exactly what the vector $\vec{n}$ does! (Note that this works only in 3D space. In 2D space, we rotate about a point and in 4D space, we rotate about a 2D plane, not about an axis.)

Why do the vectors $\vec{n}$ in the Lie algebra act on the representation space by means of the cross product? If we pick an element of the Lie algebra and let it act on all points (vectors) in the representation space, it produces a vector field. The flow of this vector field describes the transformations generated by the Lie-algebra element. Now, let's pick an axis vector $\vec{n}$ and then take the cross product of $\vec{n}$ with all vectors $\vec{x}$. What do we get? For points $\vec{x}$ on the axis, the product vectors are zero; for points $\vec{x}$ away from the axis, the product vectors point in the direction orthogonal to the plane spanned by the axis and the point; as the points $\vec{x}$ get farther away from the axis, the product vectors get larger. We can see now that the vector field, $\vec{n} \times \vec{x}$, describes rotation about the axis $\vec{n}$! The operator $\vec{n} \times \cdot$, where the dot is a place holder for the acted-upon vector, is the generator of rotation about the axis $\vec{n}$.

Why is the Lie bracket in this reformulation realized by the cross product? The Lie bracket describes how infinitesimal transformations in the Lie group fail to commute. In the case of 3D rotations, this means that if we "commute" two infinitesimal rotations (e.g., we make an infinitesimal rotation about the x axis followed by one about the y axis and then undo them in the opposite order), we end up with an infinitesimal rotation about the orthogonal direction (e.g., about the z axis). The cross product is tailor-made for this: taking the cross product of two vectors yields a vector in the orthogonal direction!

Can we write the elements of the Lie *group* in terms of the axis-vector generators introduced above? Yes, we exponentiate the generators, as usual! Let's first review the familiar case of a Lie algebra with matrix elements. Picking a rotation by θ_z about the z axis, we get the transformation $R = \exp(T_z \theta_z)$. To interpret this, we expand the matrix exponential into the power series $R = I + T_z \theta_z + \frac{1}{2} T_z^2 \theta_z^2 + \cdots$. Thus, a vector transforms like $\vec{x}' = R\vec{x} = \vec{x} + (T_z \vec{x})\theta_z + \frac{1}{2}(T_z^2 \vec{x})\theta_z^2 + \cdots$. Now, we follow the same logic for the Lie algebra with vector elements. Picking a rotation by θ_z about the z axis, we get the transformation $\tilde{R} = \exp[(\vec{t}_z \times \cdot)\theta_z]$. But how are we to interpret the incomplete cross product in the exponent? We expand the exponential function into a power series $\tilde{R} = I + (\vec{t}_z \times \cdot)\theta_z + \frac{1}{2}(\vec{t}_z \times (\vec{t}_z \times \cdot))\theta_z^2 + \cdots$. Thus, a vector transforms like $\vec{x}' = \tilde{R}\vec{x} = \vec{x} + (\vec{t}_z \times \vec{x})\theta_z + \frac{1}{2}(\vec{t}_z \times (\vec{t}_z \times \vec{x}))\theta_z^2 + \cdots$. Now, all the cross products have two operands and make sense!

Let's take a step back and think about how the cross product emerged. We started with a 3D vector space and asked: "what transformations preserve the dot product?" Answer: 3D rotations. Then, we asked: "what generates the vector fields that characterize these 3D rotations?" Answer: the cross product with the axis vector!