On CROTAi

Introduction

Almost since the creation of FITS and the first elementary implementation of World Coordinate Systems, there has been a discussion of the “correct” implementation of coordinate rotations in more than two dimensions. After brooding on this issue for several decades, I have produced some insights into the problem. In this document I explain the technical nature of rotations in many dimensions, and offer some speculations on how these might be implemented in FITS. On the other hand I do not see any urgency in doing this; over the more than thirty years that CROTn and CD_xy have represented the possibilities, no one has complained that another implementation was necessary, so I conclude that this is probably the case. I motivate the discussion below from a an esthetic desire to find an elegant solution, and a psychological desire for closure.

I use various criteria to rank the quality of a representation. Besides logical consistency, these include symmetry (e.g. no preference for particular axes), succintness (no redundant parameters), and geometric intuitiveness. None of my suggestions is ideal in all these areas. It is self-evident that if an ideal representation existed, it would be long known.

Mathematical nature of rotations in many dimensions

I have derived most of the results here by myself, only to learn later that they were all well known among mathematicians, and most can be found in the Wikipedia. For the most part, then, I only state various properties, without any proofs.

The rotation group in *n* dimensions is known as *SO(n)* and consists of linear transformations that preserve distances and angles. Any such transformation can be represented as a $n*n$ matrix with the constraint that the rows and columns are *orthonormal*, i.e. that if a_i and a_j are two rows (or columns) of the matrix, then $a_i . a_j = \delta_{ij}$. Since a $n*n$ matrix has n^2 elements, and the orthonomality requirements represents $n(n+1)/2$ conditions, an *SO(n)* matrix has $n(n-1)/2$ independent parameters. We will consider later how these might best be represented.

Geometrical interpretation

The following arguments follow those of Goldstein’s **Classical Mechanics** and are similar to those used in discussing Hermetian matrices in quantum mechanics.

The SO(n) matrices can be largely described in terms of their eigenvalues and eigenvectors. The **characteristic equation** for the eigenvalues is a **real** algebraic equation of degree *n*, so it has *n* roots, which must be either **real** or occur in complex conjugate pairs. The *normal* part of *orthonormality* then requires than the absolute value of any eigenvalue be **unity**, so that in fact the only possible eigenvalues are **1, -1,** or $\mathbf{e^{\pm i\varphi}}$. The *ortho* part requires that all eigenvectors with distinct eigenvalues be orthogonal. Note that the components of the eigenvectors are usually not real. Now an eigenvector associated with an eigenvalue of **1** is unchanged by the *SO(n)* rotation, while one with **-1** is reversed. If we only consider rotations than can be realized continuously without reflections, we can and will ignore this last case. If there are **multiple** eigenvectors with eigenvalues of **1**, these represent a invariant **subspace** of the original *n*-dimensional space unaffected by the rotation. We will also ignore this possibility as trivial. Thus the remaining cases are:

- if *n* is **even**, there are *n/2* pairs of complex conjugate eigenvectors, all orthogonal
- if *n* is **odd** there are *(n-1)/2* such pairs, and there is a single **invariant real** eigenvector with eigenvalue of **1** (in 3 dimensions this is the rotation axis).

We also assume that none of the complex eigenvalues are duplicate. Suppose we take one of the conjugate pairs. Since the eigenvalues are complex the eigenvectors must also be complex and they can be taken to be complex conjugates of each other: $\vec{\mathbf{e}}$ and $\vec{\mathbf{e}}^*$. From these two complex vectors, *two* real orthogonal vectors can be formed by linear combinations: $\vec{c} = \vec{e} + \vec{e}^*$ and $\vec{s} = (\vec{e} - \vec{e}^*)/i$. These real vectors are not eigenvectors, but any linear combination of them, after the specified rotation, can still be represented as a linear combination of the same two vectors. Put another way, these two real vectors span a two dimensional subspace, otherwise known as a plane, that is invariant under the rotation. Any vector in this plane is rotated into another vector in the plane through an angle $\boldsymbol{\varphi}$ which is the logarithm of the eigenvalue $\mathbf{e}^{\pm i\boldsymbol{\varphi}}$. It turns out that the eigenvectors are independent of the angle $\boldsymbol{\varphi}$**.** Since we have assumed no duplicate eigenvalues, each of the invariant planes is orthogonal to the other ones.

Summarized: for *n* even, the *SO(n)* can be represented by *n/2* independent **primitive** rotations in *n/2* orthogonal invariant planes. By primitive rotation I mean the original simple rotation defined by sines and cosines of a rotation angle in a plane. For *n* odd, the result is (*n-1)/2* primitive rotations plus one invariant vector. For convenience I let $n_e = n$ for *n* even and *(n-1)* for *n* odd.

So suggestion #**1** for a representation of *SO(n)*, is to specify the $n_e/2$ invariant planes, and the $n_e/2$ rotation angles, one for each plane. I call this the **IP** (Invariant Plane) representation. It has the advantages of being reasonably geometrically intuitive and symmetric. In a **FITS** environment one could specify with keywords the components of the *c* and *s* vectors, and the rotation angles φ. This suggestion fails rather badly in the succinctness criterion; the number of parameters is approximately $1.5{*}n^2$. The reason is that there are many inexplicit constraints, or redundant degrees of freedom (*rdf's)* on the eigenvectors. Consider defining the first invariant plane by specifying any two real vectors that span the plane. Naively this requires *2n* parameters:the vector coordinates. But these are more than are necessary to define the plane. We can arbitrarily add 3 constraints:that the vectors be of unit length and that they be orthogonal. There is a 4th *rdf*, namely that a pair of such unit vectors, rotated in the invariant plane, represents the same plane. This is equivalent to the fact that the original eigenvector $\vec{\mathbf{e}}$ can be multiplied by any arbitrary phase factor without changing its validity. So in fact the number of parameters necessary to specify the first plane is *2n-4*. If we want to specify a second plane we can only do this in the *n-2* dimensional subspace that is orthogonal to the first plane, so the number of new parameters is *2(n-2)-4=2n-8*, and so on in more dimensions.

We can check this arithmetic in 3 and 4 dimensions. We know that the invariant plane for a 3-d rotation can be specified with 2 parameters, the polar coordinates of the rotation pole. Reassuringly, this equals *2n-4* for *n=3*. The complete specification of the rotation requires additionally the rotation angle, for a total of 3 parameters, equal to *n*(n-1)/2.* In 4 dimensions we need 4 parameters to specify the first plane and *2n-8=0* to specify the 2nd plane. This makes sense because in 4-d, once we have specified the first plane, there is only one plane that is orthogonal to it. The full specification then requires a rotation angle in each plane, for a total of 6 parameters, again equal to *n*(n-1)/2*. In 2-dimensions the numbers check trivially:there are *2n-4=0* free parameters necessary to specify the invariant plane, and an additional 1 rotation angle in that plane.

So this is all logically consistent, but if we tried to implement a non-redundant version of this in **FITS** most of the symmetry and geometric clarity would be lost. I note that this **IP** representation is closely analogous to the Euler angle representation in 3-dimensions, which also lacks symmetry, although it is reasonably clear geometrically.

A matrix logarithm (ML) representation

An elegant but more abstract approach to the *SO(n)* representation is to realize that the matrix *S* for such a general rotation can always be represented as a **matrix exponential,** $\boldsymbol{S} = \boldsymbol{exp}(\boldsymbol{L})$ where ***L*** is the **matrix logarithm** of *S* and the exponential is the obvious generalization of the real exponential: $\mathbf{exp(L)} \equiv \mathbf{I} + \mathbf{L} + \frac{\mathbf{L}^2}{2!}$ ... and **I** is the identity matrix.

It can be shown that if ***S*** is orthonormal, then ***L*** exists, is unique modulo 2π in each component, and is **anti-symmetric**, $\boldsymbol{L_{ij}=-L_{ji}}$ **,** $\boldsymbol{L_{ii}=0}$. I will try to make some of these statements plausible below. Since ***L*** commutes with itself and its transpose, the exponential is well defined, and $SS^* = \exp(L) * \exp(L^*) = \exp(L + L^*) = exp(0) = \boldsymbol{I}$**,** i.e. the exponential of any anti-symmetric matrix is orthonormal.

There are many algorithms in the literature for calculating matrix exponents and logarithms, so my suggestion #**2** for a representation is to use either the upper or lower triangle of ***L*** to represent the general rotation. This I call the **ML** representation, and evidently requires *n*(n-1)/2* parameters, so it is succinct.

A non-rigorous justification of the above

Consider our invariant plane description of the general rotation. In each plane *i* there is a rotation φ_i . Given some large number *N*, we can break the total rotation into *N* infinitesimal rotations through angles φ_i/N. This defines an infinitesimal rotation matrix S_N . Since the planes and the rotation angles are all independent of each other, if we apply S_N *N* times, we get back the original transformation:$S=(S_N)^N$. But since S_N is an infinitesimal transformation, it can be written as $S_N = \boldsymbol{I} + \varepsilon R$ where ε is of order *1/N* say *ε=β/N*. Since S_N is itself a legitimate orthonormal transformation, the requirement that $S_N S_N^* = \boldsymbol{I}$ (to first order) immediately implies that $R= -R^*$. Finally then $S = (\boldsymbol{I} + \beta R/N)^N$. Then following elementary calculus we can regard *L=βR* as the logarithm of *S*. The rest of the properties follow easily.

Is this a good idea?

So we could implement this in **FITS** by specifying as keywords the *(n-1)*n/2* components of the top-right or bottom-left triangles of ***L***. Then the full transformation matrix *S* is the exponential function of ***L***, which is relatively easy to compute. This approach is logically consistent, symmetric, and succinct, but has lost a bit of geometric clarity. Note that in 2-dimensions this clarity is present. For a rotation matrix with the usual cos(φ)/sin(φ) components, the triangular matrix logarithm is very simple with the off-axis component φ.

A minimalist suggestion.

What about the status quo? The CROTAi convention is poorly defined and not particularly logically consistent and is now deprecated. The CD_xy convention is well defined, consistent, and well understood but usually redundant in two ways: there are more parameters than usually necessary, and some of them, referring to scale factors, are redundant with the CDELTi. Are the small changes that could improve the situation?

We note that true multi-dimensional rotations seldom make physical sense; the dimensions have to be "equivalent" in some sense for this to be the case, and there are seldom more than 3 equivalent dimensions, although there might be some relativistic models involving 4. We also note that by far the most common need for rotation is around one of the existing defined axes, e.g. rotating the "face plane" of a 3-d array around the Z-axis. So a simple upgrade to the current convention that is fairly clear-cut, logically consistent, and eliminates the need for CD_xy in many cases would be a CROTA_ab parameter that specifies a rotation angle mixing the a- and b-axes, but leaving all others unchanged. So CROTA_12 would specify a rotation of the "face plane".

You could allow several of these, leading to an Euler-like definition of a more general rotation, but this is a bad idea.

Recommendations

I wrote this for closure and to clarify some misconceptions about higher-dimensional rotations, but as I said above, the urgency to change the status quo is low. My current opinions about doing anything concrete are:

1. **Do not** implement anything like the **IP** (Invariant Plane) representation. This is too confusing for just about everybody.
2. **Consider** implementing a **ML** (Matrix logarithm) representation. Combined with CDELTi this provides a clean separation between rotations and scale transformations. CD_xy is then largely unnecessary but should be kept for backwards compatibility and to represent non-orthogonal transformations.
3. If step (2.) is too distasteful, **consider** a CROTA_xy keyword, which will cover the most common rotations in a simple way.