AY120A,B CHEAT-SHEET FOR SPHERICAL COORDINATE TRANSFORMATION

Carl Heiles

In our never-ending attempt to make your life easier, we present you with the quickest of quick summaries of spherical coordinate transformation with matrix techniques. For example, you might need to point the telescope at some particular position in the Galaxy. In other words, you need to convert from Galactic coordinates to altitude and azimuth. This involves three separate coordinate transformations: Galactic longitude and latitude to equatorial right ascension and declination $[(\ell, b) \rightarrow (\alpha, \delta)]$; to hour angle and declination $[(\alpha, \delta) \rightarrow (ha, \delta)]$; to azimuth and altitude $[(ha, \delta) \rightarrow (az, alt)]$. Or maybe you need to go all or partway in the the other direction, i.e. $[(az, alt) \rightarrow [(\alpha, \delta)]$ or maybe just $[(az, alt) \rightarrow [(ha, \delta)]$.

These are conversions among four spherical coordinate systems. Such conversions involve all those complicated combinations of trig functions (sigh). You can write down all these trig functions for the various possible transformation—twelve possibilities in all if you include going both directions.

But there is a much easier, more elegant, and more politically correct way: using *rotation matrices*. In this method, you generate a vector in the original coordinate system; convert the vector to another coordinate system by rotating the coordinates using matrix multiplication; and convert the vector to the angles of the new coordinate system.

There are two big advantages with this method. First, you can apply several transformations in succession by multiplying the rotation matrices in succession, so you break the process down into single transformations, each with its own rotation matrix. Second, it's easy to go "backwards"—you just use the inverse of the matrix.

The method is *general* and can be applied to *any* coordinate transformation. Spherical coordinates are characterized by two angles. One "goes around the z-axis"—it is like longitude on the earth. The other "goes up and down" and is like latitude on the earth. These angles are "longitude-like" and "latitude-like", and we'll denote them by *long* and *lat*. Thus, for Galactic coordinates, ℓ is the "longitude-like" *long* and b is the "latitude-like" *lat*; for equatorial coordinates, it's α (or ha) and δ; for terrestrial coordinates, it's az and alt.

One more thing before we get into details. Our discussion is oriented towards astronomy, but the method works for any type of spherical coordinate transformation. There is an excellent, short discussion of the general situation in Goldstein's *Classical Mechanics*, §4.4; we include a copy of these 3 pages as an attachment at the end.

1. ROTATION MATRICES: THE METHOD

To restate the problem: we start with $(long, lat)$ in one coordinate system and want to convert to $(long', lat')$ in some other coordinate system. Here's the prescription:

Step 1. First, convert the angles to rectangular coordinates. One would usually call these (x, y, z); here, to emphasize the vector/matrix flavor, we call them (x_0, x_1, x_2) and denote the 3-element vector $\mathbf{x}$. To accomplish this conversion:

$$x_0 = \cos(lat)\cos(long) \ , \tag{1a}$$

$$x_1 = \cos(lat)\sin(long) \ , \tag{1b}$$

$$x_2 = \sin(lat) \ . \tag{1c}$$

The IDL commands to accomplish this should be obvious, so we won't state them here; but remember to convert the arguments to radians (in IDL, converting degrees to radians is most easily done by multiplying by $!dtor$).

Step 2. Apply the rotation matrix $\mathbf{R}$ (we'll discuss its definitions below):

$$\mathbf{x}' = \mathbf{R} \cdot \mathbf{x} \ . \tag{2}$$

In IDL, we'll use the IDL variable **xp** to represent **x'**, and you do matrix multiplication with the operator ##...

$$\mathbf{xp} = \mathbf{R\#\#x} \ . \tag{3}$$

Step 3. Convert the primed rectangular coordinates to the new set of spherical coordinates $(long', lat')$:

$$long' = \tan^{-1}\left(\frac{\mathbf{x'}_1}{\mathbf{x'}_0}\right) \ , \tag{4a}$$

$$lat' = \sin^{-1}(\mathbf{x'}_2) \ . \tag{4b}$$

To do these in IDL, where we'll write **longp, latp**:

$$\mathbf{longp} = \mathbf{atan}(\mathbf{xp}(\mathbf{1}), \mathbf{xp}(\mathbf{0})) \tag{5a}$$

$$\mathbf{latp} = \mathbf{asin}(\mathbf{xp}(\mathbf{2})) \ ; \tag{5b}$$

if you want to convert their outputs to degrees, multiply by *!radeg*. *IMPORTANT:* writing **atan(xp(1), xp(0))** instead of **atan(xp(1)/xp(0))** ensures that the angle is given in the correct quadrant. See the IDL documentation.

That's it! If you want to "go backwards", you just apply the matrix multiplications using the inverse matrices. For rotation matrices, the inverse is always equal to the transpose(!)[1]—symbolically for the matrix $\mathbf{R}$, $\mathbf{R^{-1}} = \mathbf{R^T}$. So to go from the primed system to the unprimed, you switch the primed and unprimed in equation (2) and use the inverse rotation matrix

$$\mathbf{x} = \mathbf{R^{-1}} \cdot \mathbf{x'} = \mathbf{R^T} \cdot \mathbf{x'} \ . \tag{6}$$

and in IDL...

$$\mathbf{x} = \mathbf{inverse(R)\#\#xp} = \mathbf{transpose(R)\#\#xp} \ . \tag{7}$$

2. ROTATION MATRICES: SPECIFICS FOR OUR PROBLEM

OK, what are these rotation matrices? We'll do you a big favor and tell you.

2.1. (RA, DEC) to (HA, DEC)—any epoch.

Converting from $(\alpha, \delta) \rightarrow (ha, \delta)$ keeps the declination the same and uses the relationship $ha = LST - \alpha$. It is easiest to think of this in two steps, so we express

$$\mathbf{R}_{(\alpha,\delta)\rightarrow(ha,\delta)} = \mathbf{R}_{(\alpha,\delta)\rightarrow(ha,\delta),2} \cdot \mathbf{R}_{(\alpha,\delta)\rightarrow(ha,\delta),1} \ . \tag{8}$$

First, we rotate around the equatorial pole by an angle equal to the Local Sidereal Time (LST), which does $\alpha \rightarrow (\alpha - LST)$:

$$\mathbf{R}_{(\alpha,\delta)\rightarrow(ha,\delta),1} = \begin{bmatrix} \cos(LST) & \sin(LST) & 0 \\ -\sin(LST) & \cos(LST) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{9a}$$

[1]If you don't believe this, check on the $\mathbf{R}$'s below in §2!

Next, the ha and α go in opposite directions, which is equivalent to converting from the original left-handed to a right-handed coordinate system, so the second step is just to perform this reversal:

$$\mathbf{R}_{(\alpha,\delta)\rightarrow(ha,\delta),2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{9b}$$

The full rotation matrix is the matrix product $\mathbf{R}_{(\alpha,\delta)\rightarrow(ha,\delta),2} \cdot \mathbf{R}_{(\alpha,\delta)\rightarrow(ha,\delta),1}$. *Note the order!* Applying $\mathbf{R}_{(\alpha,\delta)\rightarrow(ha,\delta),2}$ at the *beginning* in the matrix product means that it operates *last* on the vector $\mathbf{x}$, which is what we want. So we have as the product...

$$\mathbf{R}_{(\alpha,\delta)\rightarrow(ha,\delta)} = \begin{bmatrix} \cos(LST) & \sin(LST) & 0 \\ \sin(LST) & -\cos(LST) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{10}$$

2.2. (HA, DEC) to (AZIMUTH, ALTITUDE).

Because (ha, δ) are Earth-based coordinates, this conversion depends only on your terrestrial latitude ϕ:

$$\mathbf{R}_{(ha,\delta)\rightarrow(az,alt)} = \begin{bmatrix} -\sin\phi & 0 & \cos\phi \\ 0 & -1 & 0 \\ \cos\phi & 0 & \sin\phi \end{bmatrix} . \tag{11}$$

2.3. EQUATORIAL to GALACTIC.

$$\mathbf{R}_{(\alpha,\delta)_{1950}\rightarrow(\ell,b)} = \begin{bmatrix} -0.066989 & -0.872756 & -0.483539 \\ 0.492728 & -0.450347 & 0.744585 \\ -0.867601 & -0.188375 & 0.460200 \end{bmatrix} . \tag{12a}$$

This is from Green's *Spherical Astronomy*, chapter 14.6, problem 14.6 (answers in back of book). We should've made you derive this, but we're softies. You *really should* at least *glance* at Green's chapter 2.7, which defines Galactic coordinates. In truth, the precession of the equatorial coordinate system makes this matrix a function of time: the equatorial coordinates move around the sky, but the Galactic ones do not. Precession amounts to nearly an arcminute per year! In principle, you should derive the matrix for the current epoch. In practice, you may not need such high accuracy. Better than the 1950 version are the numbers for epoch 2000, which are in Green's equation (14.55); epoch 2000 is lots closer to the present than is epoch 1950:

$$\mathbf{R}_{(\alpha,\delta)_{2000}\rightarrow(\ell,b)} = \left[\begin{array}{rrr} -0.054876 & -0.873437 & -0.483835 \\ 0.494109 & -0.444830 & 0.746982 \\ -0.867666 & -0.198076 & 0.455984 \end{array} \right] . \tag{12b}$$

2.4. Precession—converting equatorial between epochs.

We won't need these for the lab course, but we give you the info for the sake of completeness. Generating the rotation matrix for precession is a bit tedious and we won't give the explicit formulae here. They are in Green's book. The elements of the matrix are in equation (9.31). These elements contain angles, which depend on time as in equation (9.23) if you are converting from epoch 2000 to some other epoch. Precession isn't all there is; for precision exceeding $\sim 10''$ you also need to account for nutation of the Earth, which has a random component and is not completely predictable. For the complete story, see Green's chapter 9 and *The Astronomical Almanac 1998*, pages B39-B43—for interested parties only!

3. DOING ALL THIS IN IDL

Obviously, all this stuff is simple in IDL, which deals easily with matrices. Before beginning, though, a cautionary note about 2-D arrays in IDL:

3.1. A CRUCIAL PRELIMINARY: 2-D arrays in IDL.

In a computer, a multidimensional data set can be indexed in two ways, the *column-major* and *row-major* formats. IDL uses the row-major format, as does Fortran; the other major language, C, uses column-major. Suppose you have a 2×2 matrix called $\mathbf{A}$. In IDL's row-major format, when you type [*print, A*] IDL prints

$$\left[\begin{array}{cc} A_{0,0} & A_{1,0} \\ A_{0,1} & A_{1,1} \end{array} \right] , \tag{13a}$$

which is different from what you are used to seeing in standard matrix notation which is the column-major format

$$\begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix} . \tag{13b}$$

In this writeup, we are defining matrices such that, when displayed in a standard IDL *print* statement, they look correct. For example, in equation (12b), the upper right-hand element -0.483835 is $R_{2,0}$.

If you want to be a purist and define the matrices in the standard manner, that is with the lower left-hand element -0.483835 being $R_{0,2}$ instead of $R_{2,0}$, go ahead and do so. You then need to do two things. First, if you want to see the matrix displayed in the usual way, then print its transpose by typing [*print, transpose(A)*]. Second, in all our IDL matrix equations, replace ## by #.

Why does IDL do this nonstandard thing? It's because it's more straightforward for image processing, in which traditionally the images are scanned row-by-row (as in a TV set) instead of column-by-column. And IDL's origins are image processing, not matrix math.

3.2. Try the following examples in IDL.

Test $\mathbf{R}_{(ha,\delta)\rightarrow(az,alt)}$ using Green's example in chapter 2.3. In this, note that azimuth is measured positive from north towards the east; and that Green's example (b), done without rotation matrices, has a sign ambiguity in the azimuth—but when done with rotation matrices there is *no* sign ambiguity.

Test $\mathbf{R}_{(\alpha,\delta)\rightarrow(ha,\delta)}$ by making up your own example, using the fact that $ha = LST - \alpha$.

Test $\mathbf{R}_{(\alpha,\delta)_{1950}\rightarrow(\ell,b)}$ using Green's Crab Nebula example in chapter 2.7.

Finally, put them all together and make sure that works, too. For *all* of these, make sure you know how to go backwards! For example, suppose you want to convert $(az, alt) \rightarrow (\alpha, \delta)$. You need to first apply $\mathbf{R}^{-1}_{(ha,\delta)\rightarrow(az,alt)}$ and then $\mathbf{R}^{-1}_{(\alpha,\delta)\rightarrow(ha,\delta)}$. So the full rotation matrix in this case is. . .

$$\mathbf{R}_{(az,alt)\rightarrow(\alpha,\delta)} = \mathbf{R}^{-1}_{(\alpha,\delta)\rightarrow(ha,\delta)} \cdot \mathbf{R}^{-1}_{(ha,\delta)\rightarrow(az,alt)} \tag{14}$$

Again, *note the order!* Applying $\mathbf{R}^{-1}_{(\alpha,\delta)\rightarrow(ha,\delta)}$ at the *beginning* in the matrix product means that it operates *last* on the vector $\mathbf{x}$.