SPHERICAL/ASTRONOMICAL COORDINATE TRANSFORMATION

Carl Heiles and Deepthi Gorthi*

February 6, 2018

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.

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 $\vec{x}$. To accomplish this conversion:

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

$$x_1 = \cos(lat)\sin(long)\ , \tag{2}$$

$$x_2 = \sin(lat)\ . \tag{3}$$

*Only translation to Python commands

The Python/NumPy commands to accomplish this should be obvious, so we won't state them here; but remember to convert the arguments to radians (converting degrees to radians is most easily done by using `np.radians(angle)`).

Step 2. Apply the rotation matrix R (we'll discuss its definitions below):

$$\vec{x}' = R \cdot x \ . \tag{4}$$

In Python, we'll use the Python variable $\vec{x}p$ to represent $\vec{x}'$, and you do matrix multiplication with the function `np.dot(mat1,mat2)`...

$$\vec{x}p = np.dot(R, \vec{x}) \ . \tag{5}$$

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

$$long' = \tan^{-1}\left(\frac{\vec{x}'_1}{\vec{x}'_0}\right) \ , \tag{6}$$

$$lat' = \sin^{-1}(\vec{x}'_2) \ . \tag{7}$$

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

$$longp = np.arctan2(xp(1), xp(0)) \tag{8}$$

$$latp = np.arcsin(xp(2)) \ ; \tag{9}$$

if you want to convert their outputs to degrees, use the function `np.degrees(angle)`. *IMPORTANT:* writing `np.arctan2(xp(1), xp(0))` instead of `np.arctan(xp(1)/xp(0))` ensures that the angle is given in the correct quadrant. See the Python 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 R, $R^{-1} = R^T$. 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

$$\vec{x} = R^{-1} \cdot x' = R^T \cdot x' \ . \tag{10}$$

and in Python...

$$\vec{x} = np.dot(\texttt{np.transpose(R)}, \vec{x}p) \tag{11}$$

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

$$R_{(\alpha,\delta)\rightarrow(ha,\delta)} = R_{(\alpha,\delta)\rightarrow(ha,\delta),2} \cdot R_{(\alpha,\delta)\rightarrow(ha,\delta),1} \ . \tag{12}$$

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

$$R_{(\alpha,\delta)\rightarrow(ha,\delta),1} = \begin{bmatrix} \cos(LST) & \sin(LST) & 0 \\ -\sin(LST) & \cos(LST) & 0 \\ 0 & 0 & 1 \end{bmatrix} \ . \tag{13}$$

[1] If you don't believe this, check on the 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:

$$R_{(\alpha,\delta)\to(ha,\delta),2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{14}$$

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

$$R_{(\alpha,\delta)\to(ha,\delta)} = \begin{bmatrix} \cos(LST) & \sin(LST) & 0 \\ \sin(LST) & -\cos(LST) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{15}$$

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

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

$$R_{(ha,\delta)\to(az,alt)} = \begin{bmatrix} -\sin\phi & 0 & \cos\phi \\ 0 & -1 & 0 \\ \cos\phi & 0 & \sin\phi \end{bmatrix} . \tag{16}$$

2.3 EQUATORIAL to GALACTIC.

$$R_{(\alpha,\delta)_{1950}\to(\ell,b)} = \begin{bmatrix} -0.066989 & -0.872756 & -0.483539 \\ 0.492728 & -0.450347 & 0.744585 \\ -0.867601 & -0.188375 & 0.460200 \end{bmatrix} . \tag{17}$$

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:

$$R_{(\alpha,\delta)_{2000}\to(\ell,b)} = \begin{bmatrix} -0.054876 & -0.873437 & -0.483835 \\ 0.494109 & -0.444830 & 0.746982 \\ -0.867666 & -0.198076 & 0.455984 \end{bmatrix} . \tag{18}$$

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!

Note on Angle Representations Angles are usually specified in degrees or radians with decimal places representing the fractional angle. In astronomy, you will often encounter angles represented in degrees, minutes and seconds. You can convert this into decimal degrees (dd) using the fact that there are 60 minutes in a degree and 60 seconds in a minute.

$$dd = \text{degree} + \text{minutes}/60 + \text{seconds}/3600 \tag{19}$$

You will also often encounter the sexagesimal notation where angles are represented in hours, minutes and seconds. This is especially useful to represent angles that are dependent on longitude of the location since 15° is equivalent to an hour. You can convert a sexagesimal angle to decimal degrees using equation (19) and the additional multiplying factor of 15°.

3 DOING ALL THIS IN Python

Obviously, all this stuff is simple in Python, which deals easily with matrices:

3.1 Try the following examples in Python.

Test $R_{(ha,\delta)\to(az,alt)}$, going both forwards and backwards. For an observatory at latitude 41.36°, $(az, alt) = (137.60°, 32.43°)$ transforms to $(ha, \delta) = (325.05s°, -6.52°)$. Hour angle is usually given in hours using sexagesimal notation: $ha = 21^h40^m12^s$.)

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

Test $R_{(\alpha,\delta)_{1950}\to(\ell,b)}$ for the Crab Nebula. The Crab has 1950 equatorial coordinates $(\alpha, \delta) = (05^h31^m.5, 21°59')$ and Galactic coordinates $(\ell, b) = (184°33', -5°47')$.

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 $(az, alt) \to (\alpha, \delta)$. You need to first apply $R^{-1}_{(ha,\delta)\to(az,alt)}$ and then $R^{-1}_{(\alpha,\delta)\to(ha,\delta)}$. So the full rotation matrix in this case is...

$$R_{(az,alt)\to(\alpha,\delta)} = R^{-1}_{(\alpha,\delta)\to(ha,\delta)} \cdot R^{-1}_{(ha,\delta)\to(az,alt)} \tag{20}$$

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