

rotation. In the following, we call the final value, q_3 , the scalar value; note that other sources (e.g., MathWorld) may choose to call the first value the scalar value.

The conjugate of a quaternion, $q = (q_0, q_1, q_2, q_3)$, is $\bar{q} = (-q_0, -q_1, -q_2, q_3)$. Note that the vector components are negated, but not the scalar component.

3.3.2 Quaternion for a position

Given an angular position on the sky, (α, δ) , we can construct a quaternion by treating it as a unit vector in cartesian space:

$$p_0 = \cos \delta \cos \alpha \quad (81)$$

$$p_1 = \cos \delta \sin \alpha \quad (82)$$

$$p_2 = \sin \delta \quad (83)$$

and we set the scalar value to zero, $p_3 = 0$.

Given a quaternion, p , we can calculate the position using the inverse of the above equations:

$$\phi = \arctan(p_1, p_0) \quad (84)$$

$$\theta = \arcsin(p_2) \quad (85)$$

where ϕ is the longitude and θ is the latitude. Note that in this case, we neglect the scalar component of the quaternion — it should be zero.

3.3.3 Quaternion for a rotation

A rotation of angle θ about the axis defined by the unit vector (v_x, v_y, v_z) is specified by a quaternion with components:

$$r_0 = v_x \sin(\theta/2) \quad (86)$$

$$r_1 = v_y \sin(\theta/2) \quad (87)$$

$$r_2 = v_z \sin(\theta/2) \quad (88)$$

$$r_3 = \cos(\theta/2) \quad (89)$$

Note that the sine and cosine are taken of the half-angle of the rotation. Note also that this implies that the quaternion components are normalized such that $|r| \equiv r_0^2 + r_1^2 + r_2^2 + r_3^2 = 1$.

3.3.4 Multiplication of quaternions

Given two quaternions a and b , there is a third quaternion, $p = ab$. The components of p are given by:

$$p_0 = b_3 a_0 + b_2 a_1 - b_1 a_2 + b_0 a_3 \quad (90)$$

$$p_1 = -b_2 a_0 + b_3 a_1 + b_0 a_2 + b_1 a_3 \quad (91)$$

$$p_2 = b_1 a_0 - b_0 a_1 + b_3 a_2 + b_2 a_3 \quad (92)$$

$$p_3 = -b_0 a_0 - b_1 a_1 - b_2 a_2 + b_3 a_3 \quad (93)$$

Note that quaternion multiplication is associative (whether you do the left pair or the right pair first doesn't matter):

$$(ab)c = a(bc) \quad (94)$$

but not commutative (you can't switch the order of the operands):

$$abc \neq acb \quad (95)$$

3.3.5 Rotating a Vector

Rotation of a position is performed by constructing the quaternion for the position, p , and the rotation, r , according to the above equations, and calculating the product:

$$q = rp\bar{r} \quad (96)$$

q is the quaternion of the result. Note the use of the conjugate of the rotation quaternion.

A general rotation may be specified by three individual rotations about a predefined set of axes. We choose to specify rotations around the z , y and z axes, in that order. The amount of rotation around each of these axes are known as Euler angles. Given the Euler angles of a rotation, the rotation may be performed by rotating in turn about the designated axes. Euler angles are specified below for the various rotations required. To use them, the following rotation quaternions are used:

- First, about the Z axis:

$$r_0 = 0 \quad (97)$$

$$r_1 = 0 \quad (98)$$

$$r_2 = \sin(\alpha_p/2) \quad (99)$$

$$r_3 = \cos(\alpha_p/2) \quad (100)$$

- Second, about the Y axis:

$$s_0 = 0 \quad (101)$$

$$s_1 = \sin(\delta_p/2) \quad (102)$$

$$s_2 = 0 \quad (103)$$

$$s_3 = \cos(\delta_p/2) \quad (104)$$

- Finally, about the Z axis again:

$$t_0 = 0 \quad (105)$$

$$t_1 = 0 \quad (106)$$

$$t_2 = \sin(\phi_p/2) \quad (107)$$

$$t_3 = \cos(\phi_p/2) \quad (108)$$

These three quaternions may be multiplied together to yield the quaternion of the combined rotation: tsr (note the order — r is done first, so it is nearest the position quaternion, etc.).

3.3.6 Rotation Matrix

The rotation matrix representation of a rotation may be derived directly from the quaternion representation. The following formulae convert a quaternion to a rotation matrix:

$$rot_{x,x} = q_0q_0 - q_1q_1 - q_2q_2 + q_3q_3 \quad (109)$$

$$rot_{y,y} = -q_0q_0 + q_1q_1 - q_2q_2 + q_3q_3 \quad (110)$$

$$rot_{z,z} = -q_0q_0 - q_1q_1 + q_2q_2 + q_3q_3 \quad (111)$$

$$rot_{x,y} = 2(q_0q_1 + q_2q_3) \quad (112)$$

$$rot_{y,x} = 2(q_0q_1 - q_2q_3) \quad (113)$$

$$rot_{x,z} = 2(q_0q_2 - q_1q_3) \quad (114)$$

$$rot_{z,x} = 2(q_0q_2 + q_1q_3) \quad (115)$$

$$rot_{y,z} = 2(q_1q_2 + q_0q_3) \quad (116)$$

$$rot_{z,y} = 2(q_1q_2 - q_0q_3) \quad (117)$$

3.3.7 Conversion to Other Representations

You may convert a rotation matrix, *m*, to a quaternion, *p*, with the following code:

```
double diag_sum[3];
int maxi;
double recip;

diag_sum[0]=1+m[0][0]-m[1][1]-m[2][2];
diag_sum[1]=1-m[0][0]+m[1][1]-m[2][2];
diag_sum[2]=1-m[0][0]-m[1][1]+m[2][2];
diag_sum[3]=1+m[0][0]+m[1][1]+m[2][2];

maxi=0;
for(i=1;i<4;++i) {
    if(diag_sum[i]>diag_sum[maxi]) maxi=i;
}

p[maxi]=0.5*sqrt(diag_sum[maxi]);
recip=1./(4.*p[maxi]);

if(maxi==0) {
    p[1]=recip*(m[0][1]+m[1][0]);
    p[2]=recip*(m[2][0]+m[0][2]);
    p[3]=recip*(m[1][2]-m[2][1]);
} else if(maxi==1) {
```

```

    p[0]=recip*(m[0][1]+m[1][0]);
    p[2]=recip*(m[1][2]+m[2][1]);
    p[3]=recip*(m[2][0]-m[0][2]);

} else if(maxi==2) {
    p[0]=recip*(m[2][0]+m[0][2]);
    p[1]=recip*(m[1][2]+m[2][1]);
    p[3]=recip*(m[0][1]-m[1][0]);

} else if(maxi==3) {
    p[0]=recip*(m[1][2]-m[2][1]);
    p[1]=recip*(m[2][0]-m[0][2]);
    p[2]=recip*(m[0][1]-m[1][0]);
}

```

3.4 Celestial Coordinate Conversions

Changes between spherical coordinate systems (ie, Ecliptic, Galactic, and ICRS, or Celestial, coordinates) is equivalent to a rotation in 3D. Given two coordinate system, α, δ and ϕ, θ which differ by only a rotation, the transformation between these two systems are defined by the following three parameters:

- α_p : the longitude of the target system pole in the source system
- δ_p : the latitude of the target system pole in the source system.
- ϕ_p : the longitude of the ascending node in the target system

Note that θ_p , the latitude of the source system pole in the target system, is equal to δ_p by symmetry. Transformations between arbitrary systems are specified by determining the appropriate values of α_p, δ_p , and ϕ_p , and then constructing the quaternion for this transformation.

The relevant trigonometric relationships are:

$$\sin \theta = \sin \delta \cos \delta_p - \cos \delta \sin \delta_p \sin(\alpha - \alpha_p) \quad (118)$$

$$\cos \theta \sin(\phi - \phi_p) = \cos \delta \cos \delta_p \sin(\alpha - \alpha_p) + \sin \delta \sin \delta_p \quad (119)$$

$$\cos \theta \cos(\phi - \phi_p) = \cos \delta \cos(\alpha - \alpha_p) \quad (120)$$

and for the inverse transformations, the equivalent relationships are:

$$\sin \delta = \sin \theta \cos \delta_p - \cos \theta \sin \delta_p \sin(\phi - \phi_p) \quad (121)$$

$$\cos \delta \sin(\alpha - \alpha_p) = \cos \theta \cos \delta_p \sin(\phi - \phi_p) + \sin \theta \sin \delta_p \quad (122)$$

$$\cos \delta \cos(\alpha - \alpha_p) = \cos \theta \cos(\phi - \phi_p) \quad (123)$$

Since θ and δ have domains of $-\pi/2, \pi/2$, the value of these angles are found by applying the arcsin to the sine of these angles ($\theta = \arcsin \sin \theta$) which is always single-valued and defined. The value of $\alpha - \alpha_p$ may be found from $\text{atan2}(y, x)$, where $y = \cos \delta \sin(\alpha - \alpha_p)$ and $x = \cos \delta \cos(\alpha - \alpha_p)$; and similarly for $\phi - \phi_p$.

Note that the symmetry between the two sets of above equations means that inverse transformations can be made simply by switching α_p and ϕ_p , changing the sign of δ_p , and using the forward transformation.

3.4.1 Galactic to ICRS

The appropriate values, from the Hipparcos and Tycho Catalogues are:

$$\alpha_p = 180^\circ - 192.85948^\circ \quad (124)$$

$$\delta_p = 90^\circ - 62.87175^\circ \quad (125)$$

$$\phi_p = 90^\circ + 32.93192^\circ \quad (126)$$

$$(127)$$

3.4.2 Ecliptic to ICRS

The appropriate values, from Zombeck, are:

$$\alpha_p = 270^\circ \quad (128)$$

$$\delta_p = 23^\circ 27' 8''.26 - 46''.845 T - 0''.0059 T^2 + 0''.00181 T^3 \quad (129)$$

$$\phi_p = 90^\circ \quad (130)$$

where T is the time in Julian centuries since 1900.

3.4.3 Precession

Approximate precession, good for modest sub-arcsecond precision, may be rapidly calculated using the following rotation angles:

$$\alpha_p = 180^\circ + (0^\circ.6406161 T + 0^\circ.0000839 T^2 + 0^\circ.0000050 T^3) \quad (131)$$

$$\delta_p = 0^\circ.5567530 T - 0^\circ.0001185 T^2 - 0^\circ.0000116 T^3 \quad (132)$$

$$\phi_p = 180^\circ + 0^\circ.6406161 T + 0^\circ.0003041 T^2 + 0^\circ.0000051 T^3 \quad (133)$$

where T is $(\text{MJD}_{\text{out}} - \text{MJD}_{\text{in}})/36525$ is the difference between the two epochs, in Julian centuries. This precession form shall be used to implement PS_PRECESS_ROUGH.

3.4.4 Suggested test cases

$(\alpha, \delta) = (0^\circ, 0^\circ)$ transforms to Galactic coordinates $(l, b) = (96.337272^\circ, -60.188553^\circ)$, and Ecliptic coordinates $(\lambda, \beta) = (0^\circ, 0^\circ)$.

$(\alpha, \delta) = (0^\circ, 90^\circ)$ transforms to Galactic coordinates $(l, b) = (122.93192^\circ, 27.12825^\circ)$, and Ecliptic coordinates at J2000.0 (i.e., $T = 1$), $(\lambda, \beta) = (90^\circ, 66.560719^\circ)$.

$(\alpha, \delta) = (180^\circ, 30^\circ)$ transforms to Galactic coordinates $(l, b) = (195.639488^\circ, 78.353806^\circ)$, and Ecliptic coordinates at J2100.0 (i.e., $T = 2$), $(\lambda, \beta) = (167.072470^\circ, 27.308813^\circ)$.

For each of the above input coordinates, precessing from J2100 to J1900 gives the following output coordinates: $(357.437^\circ, -1.113^\circ)$, $(358.719^\circ, 88.886^\circ)$ and $(177.423^\circ, 31.113^\circ)$.