Table B-1. Rotation: A C++ Class for 3-Dimensional Rotations—Reference Sheet

Operation	Mathematical notation	Rotation class
Definition[a]	Let R be an unspecified rotation.	Rotation R;
	Let R be a rotation specified by yaw, pitch, and roll.[b]	Rotation R(y,p,r,ZYX);
	Let R be the rotation specified by three angles, ϕ_1, ϕ_2, ϕ_3 applied in the order x-y-z.	Rotation b(ph1,ph2,ph3,XYZ);
	Let $R_{\hat{\boldsymbol{a}}}(\alpha)$ be the rotation about the vector $\boldsymbol{a}$ through the angle α.	Vector a; Rotation R(a,alpha);
	Let R be the rotation about the direction specified by the angles (θ, ϕ) through the angle α.	pair<double,double> p(th,ph); Rotation R(p,alpha);
	Let R be the rotation specified by the vector cross product $\boldsymbol{a} \times \boldsymbol{b}$.	Vector a, b; Rotation R(a,b);
	Let R be the rotation that maps the set of linearly independent vectors $\boldsymbol{a}_i$ to the set $\boldsymbol{b}_i$, where $i = 1, 2, 3$.	Vector a1,a2,a3,b1,b2,b3; Rotation R(a1,a2,a3,b1,b2,b3);
	Let R be the rotation specified by the (unit) quaternion q.[c]	quaternion q; Rotation R(q);
	Let R be the rotation specified by the 3×3 rotation matrix A_{ij}.[d]	matrix A; Rotation R(A);
	Let R be a random rotation, designed to randomly orient any vector uniformly over the unit sphere.	rng::Random rng; R(rng);
Input a rotation R	*n/a*	cin >> R;
Output the rotation R	*n/a*	cout << R;
Assign one rotation to another	Let $R_2 = R_1$ *or* $R_2 \Leftarrow R_1$	R2 = R1; *or* R2(R1);
Product of two successive rotations[e]	$R_2 R_1$	R2 * R1;
Rotation of a vector $\boldsymbol{a}$	$R\,\boldsymbol{a}$	R * a;
Inverse rotation	R^{-1}	inverse(R); *or* -R;
Convert a rotation to a quaternion	If $R_{\hat{\boldsymbol{u}}}(\theta)$ is the rotation, then $q = \cos(\theta/2) + \hat{\boldsymbol{u}} \sin(\theta/2)$.	to_quaternion(R);
Convert a rotation to a 3×3 matrix	*See description on next page.*	to_matrix(R);
Factor a rotation into a rotation sequence	*See description on next page.*	sequence s = factor(R,ZYX);[f]
Unit vector along the axis of rotation	Unit vector $\hat{\boldsymbol{u}}$ in the rotation $R_{\hat{\boldsymbol{u}}}(\theta)$	Vector(R); *or* vec(R);
Rotation angle	Angle θ in the rotation $R_{\hat{\boldsymbol{u}}}(\theta)$	double(R); *or* ang(R);

[a] A rotation is represented in the Rotation class by the pair $(\hat{\boldsymbol{u}}, \theta)$, where $\hat{\boldsymbol{u}}$ is the unit vector along the axis of rotation, and θ is the counterclockwise rotation angle.

[b] The order is significant: first yaw is applied as a counterclockwise (CCW) rotation about the z-axis, then pitch is applied as a CCW rotation about the y'-axis, and finally, roll is applied as a CCW rotation about the x''-axis. The coordinate system is constructed from the local tangent plane in which the z-axis points toward earth center, the x-axis points along the direction of travel, and the y-axis points to the right, to form a right-handed coordinate system. The particular order is specified by using ZYX. There are a total of 12 possible orderings available to the user, 6 of them have distinct principal rotation axes: XYZ, XZY, YXZ, YZX, ZXY, ZYX; and 6 have repeated principal rotation axes: XYX, XZX, YXY, YZY, ZXZ, ZYZ.

Convert rotation to a 3×3 matrix: First we convert the rotation $R_{\hat{\boldsymbol{u}}}(\theta)$ into the unit quaternion, via $q = \cos(\theta/2) + \hat{\boldsymbol{u}} \sin(\theta/2)$, and set $w = \cos(\theta/2)$, the scalar part, and $\boldsymbol{v} = \hat{\boldsymbol{u}} \sin(\theta/2)$, the vector part. Then the rotation matrix is

$$\begin{bmatrix} 2w^2 - 1 + 2v_1^2 & 2v_1v_2 - 2wv_3 & 2v_1v_3 + 2wv_2 \\ 2v_1v_2 + 2wv_3 & 2w^2 - 1 + 2v_2^2 & 2v_2v_3 - 2wv_1 \\ 2v_1v_3 - 2wv_2 & 2v_2v_3 + 2wv_1 & 2w^2 - 1 + 2v_3^2 \end{bmatrix}.$$

Factor a rotation into an (aerospace) rotation sequence: First we convert the rotation $R_{\hat{\boldsymbol{u}}}(\theta)$ into the unit quaternion, via $q = \cos(\theta/2) + \hat{\boldsymbol{u}} \sin(\theta/2)$, and set $w = \cos(\theta/2)$, the scalar part, and $\boldsymbol{v} = \hat{\boldsymbol{u}} \sin(\theta/2)$, the vector part. Next, let $p_0 = w$, $p_1 = v_1$, $p_2 = v_2$, $p_3 = v_3$ and set $A = p_0p_1 + p_2p_3$, $B = p_2^2 - p_0^2$, $D = p_1^2 - p_3^2$. Then $\phi_3 = \tan^{-1}(-2A/(B + D))$ is the third angle, which is *roll* about the x-axis in this case. Now set $c_0 = \cos(\phi_3/2)$, $c_1 = \sin(\phi_3/2)$, $q_0 = p_0c_0 + p_1c_1$, $q_2 = p_2c_0 - p_3c_1$, and $q_3 = p_3c_0 + p_2c_1$. Then $\phi_1 = 2\tan^{-1}(q_3/q_0)$ is the first angle, which is *yaw* about the z-axis, and $\phi_2 = 2\tan^{-1}(q_2/q_0)$ is the second angle, which is *pitch* about the y-axis.

---

[c] A quaternion is defined in the Rotation class as follows:

```
struct quaternion {
double w; // scalar part
Vector v; // vector part
};
```

A *unit* quaternion requires that $w^2 + \|\boldsymbol{v}\|^2 = 1$.

[d] A matrix is defined in the Rotation class as follows:

```
struct matrix {
double a11, a12, a13; // 1st row
double a21, a22, a23; // 2nd row
double a31, a32, a33; // 3rd row
};
```

To qualify as a rotation, the 3 matrix A must satisfy the 2 conditions: $A^\dagger = A^{-1}$ and $\det A = 1$.

[e] In general, rotations do not commute, i.e. $R_1R_2 \neq R_2R_1$, so the order is significant and goes from right to left.

[f] A (rotation) sequence is defined in the Rotation class as follows:

```
struct sequence {
double first; // 1st rotation (rad) to apply to body axis
double second; // 2nd rotation (rad) to apply to body axis
double third; // 3rd rotation (rad) to apply to body axis
};
```

The order these are applied is always left to right: `first, second, third`. How they get applied is specified by using one of the following, which is applied left to right: `ZYX, XYZ, XZY, YZX, YXZ, ZYX, ZYZ, ZXZ, YZY, YXY, XYX, XZX`. For example, the order `XYZ` would apply `first` to rotation about the x-axis, `second` to rotation about the y-axis, and `third` to rotation about the z-axis.