1 Rapid Structural Analysis Methods

Emergence of large structural databases which do not allow manual (visual) analysis and require efficient 3-D search and classification methods. Structure is much better preserved than sequence – proteins may have similar structures but dissimilar sequences. Structural motifs may predict similar biological function Getting insight into protein folding. Recovering the limited (?) number of protein folds. Comparing proteins of not necessarily the same family.

Implementing structural algorithms (folding, docking, alignment) requires geometric manipulation of protein structures. A 3-D protein structure is represented as a set of x, y, z coordinates (vectors). Structural manipulation of protein structures is done via *geometric transformations* (translation, rotation) of some or all the coordinates. Transformations can be represented using matrices applied on the coordinate vectors. Matrix transformations are performed through matrix multiplication of a coordinate vector by a transformation matrix.

1.1 Spherical Coordinates

To understand how to represent a point in 3D, we can first extend the polar coordinate representation discussed earlier into *spherical coordinates*. In a cartesian coordinate system we represent a 3D point p by its x, y, z coordinates: $p = (p_x, p_y, p_z)$. In spherical coordinates we have three magnitudes: ρ, θ, ϕ. Similar to polar coordinates, ρ is the magnitude of the vector, or $\rho = \sqrt{p_x^2 + p_y^2 + p_z^2}$. ϕ is the angle of the vector $p = (p_x, p_y, p_z)$ with the z axis. It is restricted to $[0, \pi]$. We denote by r the projection of p on the xy plane, so that $r = \rho \sin \phi$. Notice that $z = \rho \cos \phi$. θ is the same as in polar coordinates, the CCW rotation of r on the xy plane. See Figure 1. Therefore, substituting in the polar coordinate formula, we have the following:

$$\begin{aligned} x &= \rho \sin \phi \cos \theta \\ y &= \rho \sin \phi \sin \theta \\ z &= \rho \cos \phi \end{aligned}$$

1.2 Internal coordinates

There are other ways to represent a protein structure, for example using internal coordinates. Internal coordinates representation depicts the protein as a set of internal properties of the molecules that do not depend on the absolute position and orientation of the molecule in space: bonds, angles and dihedral angles. It is often a better way to represent a structure, especially when we want to manipulate the structure and calculate its energy. An example of internal coordinate representation can be seen below. Table 1 shows the internal coordinates of Methane (CH4). The four hydrogens are symmetric around the carbon. Every three hydrogen atoms generate a tetrahedral structure with the carbon. The internal coordinates are read as follows: The first row represents an arbitrarily selected first atom, in this case C. The next four lines describe the four hydrogens that are covalently bonded to the carbon. The format in the table is: *Element, atom 1, bond-length, atom 2, bond-angle, atom 3, dihedral-angle* . The position of the current atom is then specified by giving the length of the bond joining it to atom1, the angle formed by this bond and the bond joining atom1 and atom2, and the dihedral (torsion) angle formed by the plane containing atom1, atom2 and atom3 with the plane containing the current atom, atom1 and atom2. Note that bond angles must

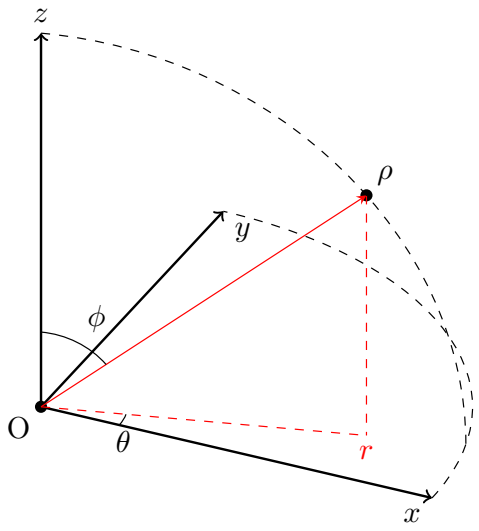


Figure 1: A representation of spherical coordinates

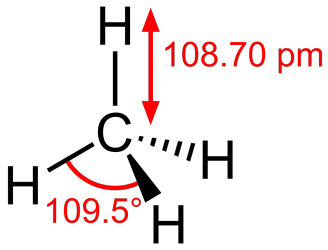


Figure 2: A methane (CH4) molecule

be in the range $0 - 180°$. The last line in the table, then, describes atom 5, a Hydrogen atom bonded to atom 1 (the Carbon), with a bond length of 1.089Å, has a 109.471° tetrahedral angle formed by this bond and the C bond with the second hydrogen, and a $-120°$ dihedral angle formed with atoms 1, 2, 3. Notice that redundant information, like the dihedral angle between atoms 1, 3 and 4 is omitted from the table, since this angle is already imposed by the other constraints given in the table.

We can switch back and forth between different representations, up to an arbitrary rigid transformation (absolute position and orientation in space). To move from internal to cartesian coordinates we need the first three atoms, a, b, c.

The first atom, a, represents the origin of the coordinate systems. Set its three cartesian coordinates to $(0, 0, 0)$.
The second atom, b, is at a fixed distance from a, which is their bond distance. Fix the z axis as the axis lying on the bond between the two atoms. b's z value is the distance and its x and y

Table 1: An example of a Z-matrix representing the internal coordinates of methane (CH4)

Atom	Bonded	Dist	Angle	Value	Dihe	Value
C						
H	1	1.089				
H	1	1.089	2	109.471		
H	1	1.089	2	109.471	3	120.0
H	1	1.089	2	109.471	3	-120.0

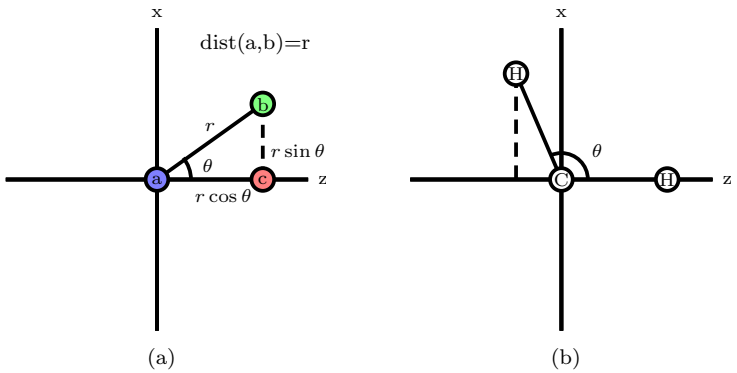


Figure 3: (a) Obtaining the cartesian coordinates based on distance and angle with the xz axis. (b) When aligning the bond between C-H with the z axis, the coordinates of the other H can be obtained using the formula above.

coordinates are set to zero $(0, 0, 1.089)$.

The third atom, c, makes a bond with atom 1 and an angle with atoms 1 and 2. We can define the $x - z$ plane by atoms 1, 2, 3, since every three ordered, non-collinear points uniquely define a plane. We can infer the x and z coordinates from these two constraints and set the y coordinate to zero. Let us denote the three atoms a, b, c in the order they appear in the angle (see Figure 3. Let r_{ac} be the distance between atoms a and c. The x, z coordinates can be inferred by converting from polar to cartesian coordinates using the following formula:

$$z = r_{ac} * \cos(\theta) = -r_{ac} \cos(180 - \theta)$$
$$x = \sin(\theta) = r_{ac} \sin(180 - \theta)$$

The point c in the figure is the projection of b on the z axis. The resulting triangle defines the angle θ. (see Figure 3).

Hence, to obtain the cartesian coordinates of H3, we see that due to the symmetry of the methane molecule, it has the same distance from C and the same angle with H1-C as the other hydrogens, but according to the internal coordinate table, the H2-C-H3 forms a 120° dihedral angle with the H1-C-H2 plane, which we defined as the $x - z$ plane. Therefore, we can rotate the position

Table 2: The Cartesian coordinate representation of Methane (CH4). Right: Rotated to show symmetry

Atom	X	Y	Z	X(rot)	Y(rot)	Z(rot)
C	0.000	0.000	0.000	0.000	0.000	0.000
H	0.000	0.000	1.089	0.629	0.629	0.629
H	1.027	0.000	-0.363	-0.629	-0.629	0.629
H	-0.513	-0.889	-0.363	-0.629	0.629	-0.629
H	-0.513	0.889	-0.363	0.629	-0.629	-0.629

of H2, $\{1.027, 0.000, -0.363\}$, by 60° around the z axis using the following formula:

$$x = -1.027 * \cos(60) = -0.513$$
$$y = -1.027 * \sin(60) = -0.889$$

So we get $\{-0.513, -0.889, -0.363\}$. Notice that since we are rotating around the z axis, the z coordinate does not change.
Similarly, H2-C-H4 creates a $-120°$ dihedral angle with the H1-C-H2 plane. Therefore, we can rotate the position of H2, $\{1.027, 0.000, -0.363\}$, by $-60\circ$ around the z axis to get:

$$x = -1.027 * \cos(-60) = -0.513$$
$$y = -1.027 * \sin(-60) = 0.889$$

with the coordinates being $\{-0.513, 0.889, -0.363\}$.

Table 2 shows the cartesian coordinates following the conversion formula above.

To further emphasize the symmetry of the molecule, we can reorient it as in Table 2 on the right.

2 Rotation Matrices and Translation Vectors

Let us formalize the notion of rotation and translation through matrices and vectors. In order to manipulate the dihedral angles of a molecule we have to be able to apply a *transformation* to some or all of the atoms of the molecule. Here is the mathematical background needed to carry this out:

Definition 1 (dot product) *The* dot product *of two vectors* $v = (v_1 \dots v_n)$ *and* $k = (k_1, \dots k_n)$ *is defined as:*

$$v \cdot k = \sum_{i=1}^{n} v_i * k_i$$

Notice that $v \cdot k$ is a scalar (number) and not a vector. It is the projection of v on k (and vice versa) and is also $|k| * |v| \cos\alpha$ where $|k|$ and $|v|$ are the magnitudes of the vectors k and v

respectively, and α is the angle between k and v. A 2-D example of a dot product can be seen in Figure 3, where the vector (a,c) is the projection of the vector (a,b) on the (unit) x axis.

Definition 2 (cross product) *The* cross product *of two vectors* $k = (k_x, k_y, k_z)$ *and* $v = (v_x, v_y, v_z)$ *is a vector defined as:*

$$w = k \times v = (k_y * v_z - k_z * v_y, k_z * v_x - k_x * v_z, k_x * v_y - k_y * v_x)$$

The result is a vector that is perpendicular to both k and v. Notice that $v \times k = -(k \times v)$. Also notice that for every two vectors k and v, $|k \times v| = |k||v| \sin \alpha$ where α is the angle between a and b.

Definition 3 (Rotation Matrix) *An* $N \times N$ *matrix* R *is a* rotation matrix *in dimension* N *if it is orthonormal (its columns are pairwise orthogonal and normalized) and* $det(R) = 1$. *Such matrix has the property* $R^T = R^{-1}$.

Definition 4 (Translation Vector) *A vector* $t = \{t_1, t_2, ..., t_N\}$ *is a translation vector in dimension* N.

Definition 5 (Rigid Transformation) *A* rigid transformation *on a vector* v *is a combination of a rotation and a translation. It has the form* $T = Rv + t$. *The transformation is called rigid since it is applied to the entire vector and does not change the distances between pairs of points.*

It is important to remember that T represents a rotation followed by a translation (not the other way around).

The rotation and translation in N dimensions can be combined into a *transformation matrix*. The problem is that translation is not a *linear transformation*, since it does not keep the origin fixed. However, it is an *affine transformation*. An affine transformation is a transformation that does not bend or twist its input. In other words – lines have to stay linear, parallel lines have to stay parallel and planes have to stay planar. Therefore, a rotation matrix and a translation vector can be combined into rigid affine transformation using *homogeneous coordinates*. This is done by adding a "dummy" zero vector to the rotation component and 1 to the translation component. In other words, we add a dimension to the matrix, so now it is of the form:

$$T = \left[\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right]$$

By doing this we transform the system from the Euclidean space to the Projective space. The new translation vector, $\{t_1, t_2, \ldots, t_N, 1\}$ is the representation of t in the projective plane rather than the Euclidean plane. One can think of the added component as a scaling factor which in principle can be any non-zero number since homogeneous coordinates are scale invariant, so for convenience we will use 1. A zero number would be a "point at infinity". To transform a vector $v = \{v_1, v_2, \ldots, v_n\}$ using homogenous coordinates we simply use $v = \{v_1, v_2, \ldots, v_n, 1\}$ so that we can apply the transformation in $N + 1$ dimensions. To transform a point back from the projective plane into Euclidean coordinates we simply ignore the 1.

2.1 Transformations as Groups

Let us first define some basic mathematical terms.

Definition 6 (Group) *A group is a set G with an operation that combines any two elements to form a third element. Let us denote the operation $\circ$. The operation satisfies four conditions called the group axioms:*

1. *Closure:* $\forall a, b \in G, a \circ b \in G$
2. *Associativity:* $\forall a, b, c \in G, (a \circ b) \circ c = a \circ (b \circ c)$
3. *Identity:* $\exists e \in G \ s.t \ \forall a \in G, a \circ e = e \circ a = a$
4. *Inverse:* $\forall a \in G \exists a^{-1} \in G \ s.t \ \forall a \in G, a \circ a^{-1} = e$

If these axioms are satisfied, then G is a group under $\circ$

Rotation matrices are groups under matrix multiplication. Notice that they satisfy the four axioms:

1. Closure: The multiplication of every two rotation matrices is a rotation matrix.
2. Associativity: This is true for every matrix.
3. Identity: The identity matrix, which is a rotation by 0 degrees.
4. Inverse: For every rotation matrix, its inverse is the rotation matrix in the inverse direction. Multiplying both gives the identity rotation.

In linear algebra, rotation can be performed in any N dimensional space, but physical objects are limited, obviously, to two or three dimensions.

2.2 Transformations in Two Dimensions

A rotation in 2D is defined as follows:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

θ is the counter-clockwise (CCW) rotation angle in the 2D plane, as shown in Figure 4. Given a vector $v = (x, y)$ multiplication by the above rotation matrix results in:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos(\theta) - y\sin(\theta) \\ y\cos(\theta) + x\sin(\theta) \end{bmatrix}$$

Hence, the rotation matrix takes (x, y) to a new location $(x', y') = (x\cos(\theta) - y\sin(\theta), y\cos(\theta) + x\sin(\theta))$.

Combining two rotations in 2D (performing two rotations one after the other) is simply multiplying two rotation matrices, $R(\theta_1)$ and $R(\theta_2)$. Remember that, since rotation matrices are a group under matrix multiplication, $R(\theta_1) * R(\theta_2)$ is also a rotation matrix due to closure.

A simple way to combine two rotations is to use Euler's formula, which establishes the relationship between trigonometric functions and complex numbers. According to the formula, for any real number θ,

$$e^{i\theta} = \cos(\theta) + i\sin(\theta)$$

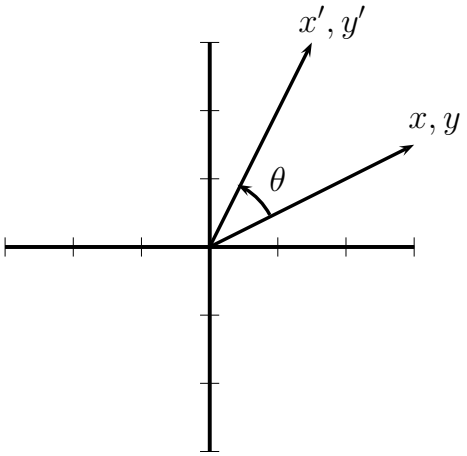


Figure 4: An example of a 2D rotation

where e is the base of the natural logarithm and i is the imaginary unit, and θ is given in radians. The meaning is that, when the graph of complex exponential function, e^{ix} is projected to the complex plane, the complex exponential function, e^{ix} is tracing the unit circle. It is a periodic function with the period 2π. See Figure 4.

First let us think of the complex plane where every point is represented as $(x + iy)$. We can use this to represent 2D rotations using the polar form of the complex exponential function, e^{ix} because it can represent a point in the complex plane with only single term instead of two terms, $(x + iy)$. It is only logical, since a rotation in 2D essentially has only one degree of freedom! It also makes the math easier when combining two or more rotations. First, let us calculate what happens when we rotate a point $x + iy$ by an angle θ:

$$\begin{aligned}(x + iy) * e^{i\theta} &= (x + iy) * (\cos(\theta) + i\sin(\theta)) \\ &= x((\cos(\theta) + i\sin(\theta)) + y(i\cos(\theta) - \sin(\theta)) \\ &= x\cos(\theta) - y\sin(\theta) + i(x\sin(\theta) + y\cos(\theta))\end{aligned}$$

which is equivalent to the matrix form $R(\theta)$ displayed above:

$$R(\theta) = e^{i\theta}$$

Therefore, we can use Euler's formula to combine two rotations easily:

$$R(\theta_1) * R(\theta_2) = e^{i\theta_1} * e^{i\theta_2} = e^{i(\theta_1+\theta_2)} = R(\theta_1 + \theta_2)$$

In other words, combining two rotations θ_1 and θ_2 results in a rotation by $\theta_1 + \theta_2$! Several things are worth mentioning about rotation in two dimensions: First – it has only one degree of freedom –one independent parameter needed to specify the transformation. This degree of freedom is the rotation angle θ. This is all we need to know in order to perform a rotation. Second – rotation in

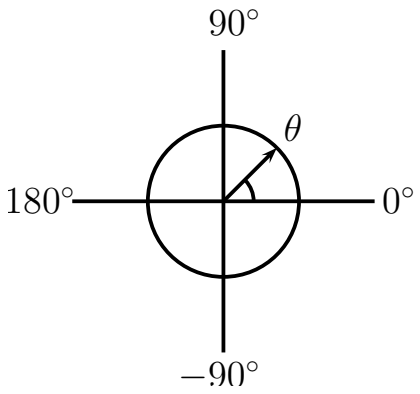


Figure 5: The SO(2) group

two dimensions is done around the z axis. In other words – the rotation axis is out of the plane! There is only one stationary point at the center of the rotation. This is one fundamental difference between rotations in two and three dimensions, as we will point out later.

2.3 Representing Rotations in 2D – SO(2)

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$a + bi \leftrightarrow \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \simeq S^1$$

Rotations and Translations

Combining rigid body translations and rotations results in a rigid transformation SE(n) – special Euclidean groups:

$$SE(n) = \begin{bmatrix} R & V \\ 0 & 1 \end{bmatrix}$$

Where R is a rotation matrix and V is a translation vector.

3 Representing Rotations in 3D

A point in 3D is To understand how rotations in 3D look like, let us first define the orthogonal group O(3) as the group of all 3X3 orthonormal matrices:

(1) $$O(3) \rightarrow AA^T = 1$$

Matrices in O(3) may have a determinant of either 1 or -1. The special orthogonal group, SO(3), is those matrices of O(3) whose determinant is 1:

$$SO(3) \rightarrow det(A) = 1 \tag{2}$$

This group represents rotations in 3D. The matrices whose determinant is -1 represent reflections. Remember that just like in the 2D case (or every dimension), if R is a rotation matrix, then $R^T = R^{-1}$. So R^T represents the rotation in the inverse direction.

Just like SO(2), SO(3) is the special orthogonal group in 3D, which represents rigid body rotation in 3D. Unfortunately, it is not a simple extension of 2D rotation. Matrices in SO(2) and hence rotations in 2D have one degree of freedom. rotations in SO(3) can be represented by three independent parameters. Remember that in 2D the rotation axis is outside the plane, whereas in 3D the axis is inside the space. Just imagine a door rotating around its axis! The axis is part of the door, and it is the only part which does not move when the door opens and closes. Indeed, one way to represent rotations in 3D, and perhaps the most convenient one to apply to protein structural manipulation, is by an angle around an axis. Also unfortunately, SO(3), despite being a 3D space, cannot be smoothly mapped into $\mathbb{R}^3$. As a matter of fact it has the topology of the projective space, $\mathbb{P}^3$. More on that later.

Here are several ways to represent rotations in 3D:

- 3x3 matrix representing an arbitrary rotation
- Euler angles (phi,theta,psi or Yaw, pitch, roll angles)
- Axis-angle representation
- Quaternions

3.1 3X3 Matrix

The simplest, most straight-forward way to represent rotations in 3D is a 3X3 matrix where each column vector represents a rotation around one of the principal axes:

$$R = \begin{bmatrix} \tilde{x}_1 & \tilde{y}_1 & \tilde{z}_1 \\ \tilde{x}_2 & \tilde{y}_2 & \tilde{z}_2 \\ \tilde{x}_3 & \tilde{y}_3 & \tilde{z}_3 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \in SO(3)$$

Rotating a Point in 3D: To perform a 3D rotation of a point $p =< p_x, p_y, p_z >$, you simply multiply the matrix and the point. The result is a counter-clockwise (CCW) rotated vector according to the rules of matrix multiplication:

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} R_{11}p_x + R_{12}p_y + R_{13}p_z \\ R_{21}p_x + R_{22}p_y + R_{23}p_z \\ R_{31}p_x + R_{32}p_y + R_{33}p_z \end{bmatrix}$$

One may imagine a rotation matrix as an alternative axis system, where rotating a vector is actually transforming the vector into the alternative system (see Figure **??**). Therefore, we can thing of a rotation matrix as a combination of individual rotations around the three principal axes. As a reminder, a rotation around a principal axis takes the form:

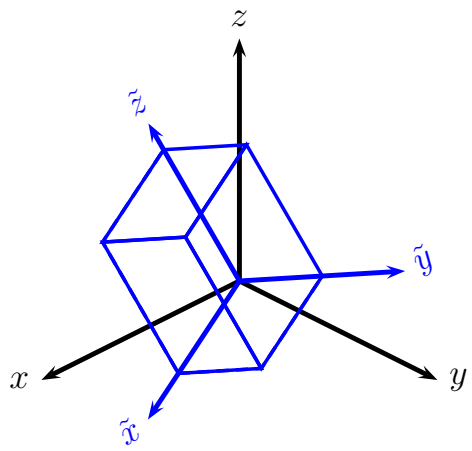


Figure 6: A 3D rotation matrix can be seen as an alternative axis system which is rotated with respect to the global axis system.

1. CCW rotation around the x-axis by an angle γ:

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

2. CCW rotation around the y-axis by an angle β:

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}$$

3. CCW rotation around the z-axis by an angle α:

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Notice that when each one of the above angles is zero we simply get the identity matrix, which does nothing!

Example: Let us rotate the vector $v = \{1, 2, 3\}$ around the z axis by $60°$ and then around the y axis by $-60°$:

$$\begin{bmatrix} \cos 60 & -\sin 60 & 0 \\ \sin 60 & \cos 60 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1*0.5 - 2*0.866 + 3*0 \\ 1*0.866 + 2*0.5 + 3*0 \\ 0 + 0 + 3*1 \end{bmatrix} = \begin{bmatrix} -1.232 \\ 1.866 \\ 3 \end{bmatrix}$$

Then:

$$\begin{bmatrix} \cos 60 & 0 & -\sin 60 \\ 0 & 1 & 0 \\ \sin 60 & 0 & \cos 60 \end{bmatrix} \begin{bmatrix} -1.232 \\ 1.866 \\ 3 \end{bmatrix} = \begin{bmatrix} -1.232 * 0.5 + 1.866 * 0 - 3 * 0.866 \\ -1.232 * 0 + 1.866 * 1 + 3 * 0 \\ -1.232 * 0.866 + 1.866 * 0 + 3 * 0.5 \end{bmatrix} = \begin{bmatrix} -3.214 \\ 1.866 \\ 0.433 \end{bmatrix}$$

Roll Pitch Yaw and Euler Angles

The above rotation matrices around the x, y and z axes are often referred to as *roll*, *pitch* and *yaw*, respectively. These terms are borrowed from aviation technology.

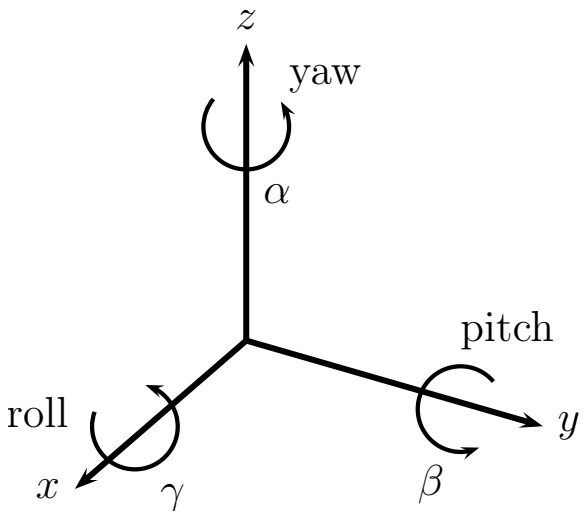


Figure 7: Roll (x), Pitch (y) and Yaw (z) rotation angles

To combine rotations, one simply has to multiply their rotation matrices. Since $SO3$ is a group, the resulting matrix is also a rotation matrix. However, remember that matrix multiplication is not commutative (order matters!) To figure out what a combined rotation may look like, you should look at order from right to left. For example – $R_x(\gamma)R_y(\beta)$ rotates by *beta* around y and then rotates the result around x by γ See Figure **??** to see how the order influences the end result.

Here is an example of combining rotations around the three axes:

$$R(\alpha, \beta, \gamma) = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma\cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{3}$$

Euler Angles

The representation of rotations in 3D by combining three individual rotations was presented by Leonhard Euler. The three parameters representing rotations around the principal axes are often referred to as α, β, γ or ϕ, θ, ψ. We can achieve any rotation by combining rotations. There are several combinations known as proper Euler angles –

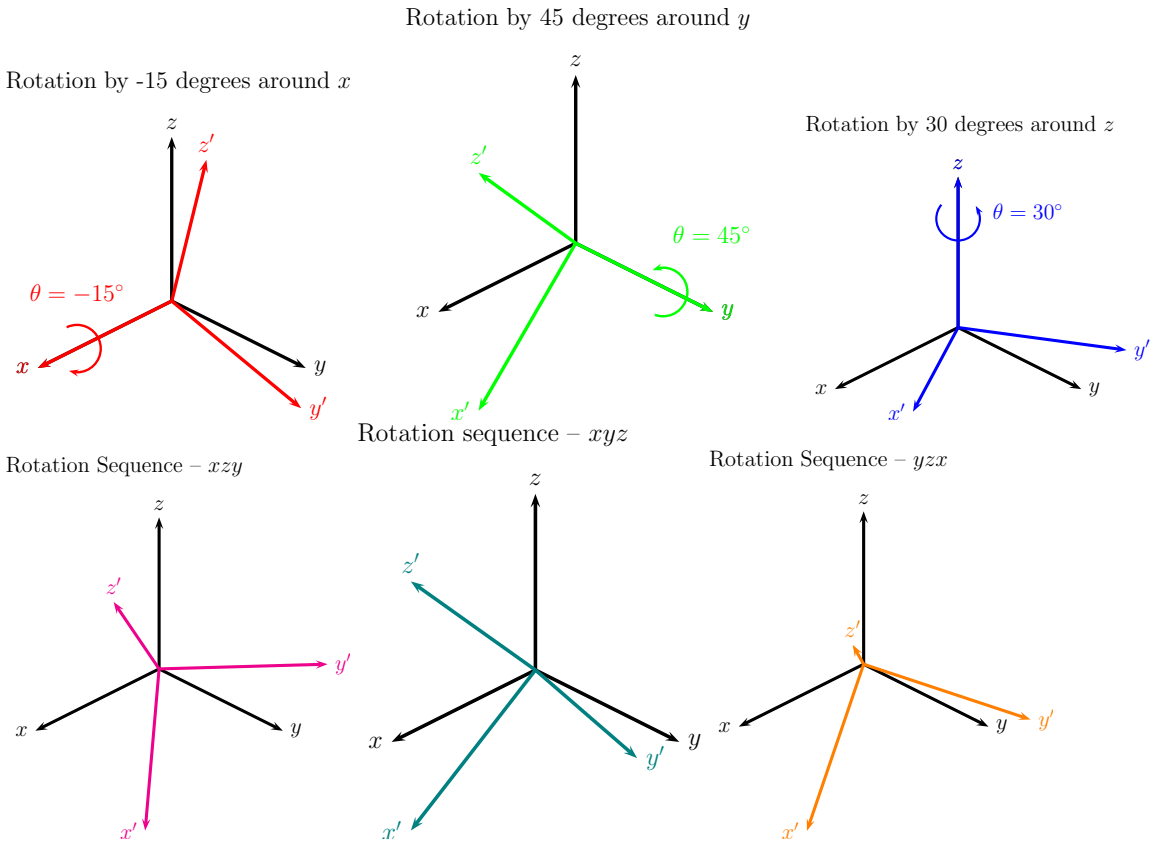


Figure 8: Examples of different counter-clockwise rotations.

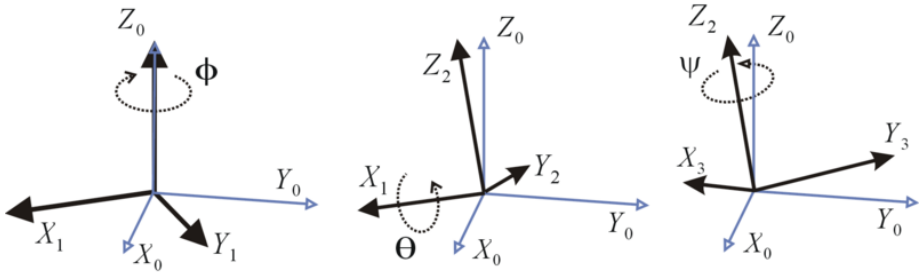
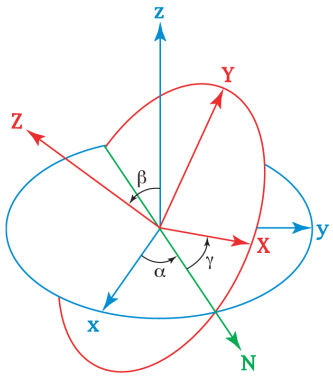


Figure 9: Euler Angles – Example: $-60°, 30°, 45°$



$$R = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

There are two major problems with yaw, pitch, roll and Euler angles:

1. There are cases where a continuum of values yield the same rotation matrix (no unique solution in certain cases).
2. There are cases where non-zero angles yield the identity rotation matrix which is equivalent to zero angles

For example – when $\beta = 0$, Euler angle representation becomes:

$$\begin{aligned} R &= \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos(\alpha+\gamma) & -\sin(\alpha+\gamma) & 0 \\ \sin(\alpha+\gamma) & \cos(\alpha+\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

This situation is called a "Gimbal lock" (see Figure **??**).

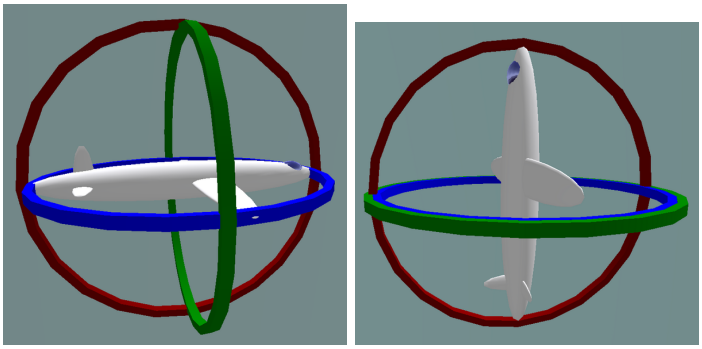


Figure 10: Example of a gimbal. Left: No lock. Right: Lock

3.2 Axis-Angle Representations

Given a rotation axis represented as a unit vector $\hat{k}$, rotating a vector v by an angle θ, a nice way to model the rotation is through Rodrigues' formula:

$$v_{rot} = v\cos\theta + (\hat{k} \times v)\sin\theta + \hat{k}(\hat{k} \cdot v)(1 - \cos\theta)$$

Explanation: The plane of rotation is perpendicular to $\hat{k}$. Using the dot and cross products, the vector v can be decomposed into components parallel and perpendicular to the axis k as follows:

$$v = v_{\parallel} + v_{\perp}$$

Where $v_{\parallel} = (v \cdot \hat{k})\hat{k}$ is the vector projection of v on the rotation axis $\hat{k}$. This part is parallel to the rotation axis and hence is not affected by the rotation. As mentioned above, the cross product of two vectors creates a vector perpendicular to both vectors. Hence, the three vectors $\hat{k}$, $\hat{k} \times v$ and $\hat{k} \times (\hat{k} \times v)$ are three mutually perpendicular unit vectors, so $\hat{k} \times v$ is perpendicular to the plane defined by $\hat{k}$ and v, and $\hat{k} \times (\hat{k} \times v)$ is perpendicular to both $\hat{k}$ and $\hat{k} \times v$. The perpendicular component $v_{\perp}$ is the projection of v on the plane perpendicular to $\hat{k}$ and $k \times v$, in the direction facing away from $\hat{k} \times (\hat{k} \times v)$. Therefore $v_{\perp} = v - v_{\parallel} = v - (\hat{k} \cdot v)\hat{k} = -\hat{k} \times (\hat{k} \times v)$. Figure **??** shows an illustration of the above.

So, as seen in the figure, the only part that rotates is $v_{\perp} = v - (\hat{k} \cdot v)\hat{k}$, and it is a 2D rotation by θ around the plane defined by $\hat{k} \times v$ and $\hat{k} \times (\hat{k} \times v)$. Using the formula for 2D rotation, the rotated component is:

$$v'_{\perp} = (\hat{k} \times v)\sin\theta - (\hat{k} \times (\hat{k} \times v))\cos\theta = (\hat{k} \times v)\sin\theta + v\cos\theta - (\hat{k} \cdot v)\hat{k}\cos\theta$$

Add to it the $v_{\parallel}$ component that did not change and get:

$$v' = (\hat{k} \times v)\sin\theta + v\cos\theta + (\hat{k} \cdot v)\hat{k}(1 - \cos\theta)$$

Obviously, you can convert the representation into a matrix, but it is ugly and cumbersome, so let us leave it at that.

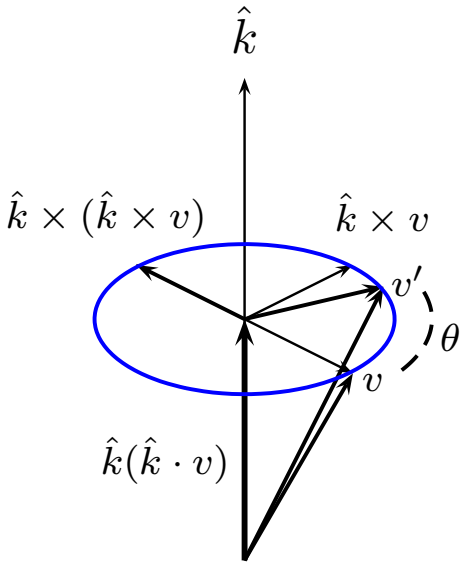


Figure 11: Dividing v into $v_{parallel}$ and $v_{\perp}$ with respect to the rotation axis $\hat{k}$. Only $v_{\perp}$ is rotated

Rotating a Molecule Around a Dihedral Angle

Often, when we manipulate a protein structure we manipulate its backbone (sometimes side-chain) dihedral angles by a magnitude of, say, θ. Then, when we think of rotating a backbone dihedral angle, it is more convenient to think of rotating atoms by an angle θ around an axis defined by the two middle atoms of the dihedral angles, as seen in Figure **??**. If we denote the atoms defining the dihedral angle as $(i, i+1, i+2, i+3)$, the rotation axis is defined as the bond between $(i, i+1)$. When rotating around an axis, the axis itself does not change as well as what precedes the axis. This means that the only atoms whose positions change are $i+3$ and above. These atoms should be rotated by angle θ around the axis. There is one caveat though. The above rotation matrices rotate the point as if its distance from the origin is $d = \sqrt{p_x^2 + p_y^2 + p_z^2}$. However, in most PDB files the atoms are located in an arbitrary coordinate system. The axis defined by $(i, i+1)$ also resides in an arbitrary place in space. For the rotation to be correct, the axis should be centered at the origin.

We should be cautious about the outcome and first translate the axis to the origin. Otherwise, the atoms will be rotated around an axis that resides in an arbitrary location in space, at an arbitrary distance from the origin. Therefore, a simple protocol for modifying a dihedral angle defined by atoms $(i, i+1, i+2, i+3)$ is as follows:

1. Translate the molecule so that atom $i+1$ is at the origin (just subtract its coordinates from every atom).

2. Define the rotation axis as $||i+2-i+1||$ (the unit vector on the bond between i+2 and i+1).

3. Apply a rotation by θ to every atom starting at i+3 around the axis.

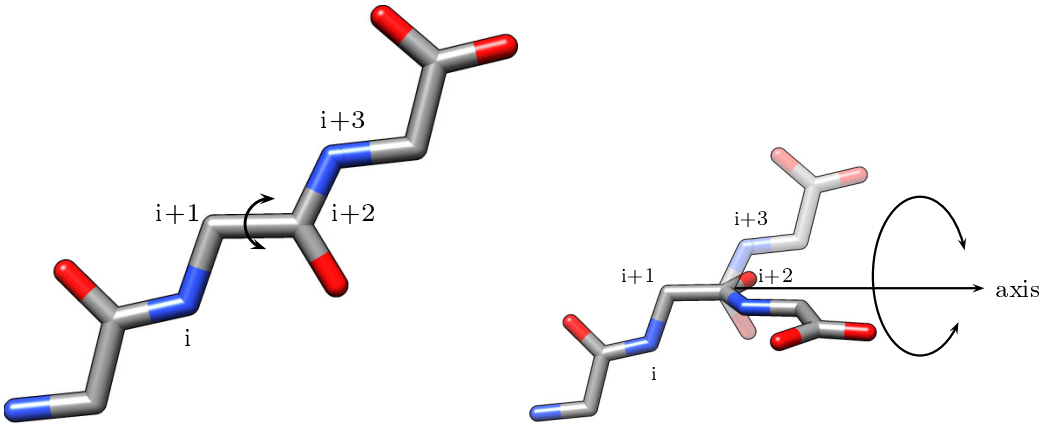


Figure 12: Modification of a dihedral angle is like rotating around the axis defined by the bond between the two middle atoms.

3.3 Axis-Angle Representations Through Quaternions

Quaternions are an extension of complex numbers. As the name suggests, they have four components: $h = a + bi + cj + dk$, where a, b, c, d are real numbers.

i,j,k are imaginary numbers such that:

- $i^2 = j^2 = k^2 = -1$
- $ij = k$, $jk = i$, $ki = j$
- $ij = -ji$, $jk = -kj$, $ki = -ik$

In other words, i, j, and k represent three distinct roots of -1.

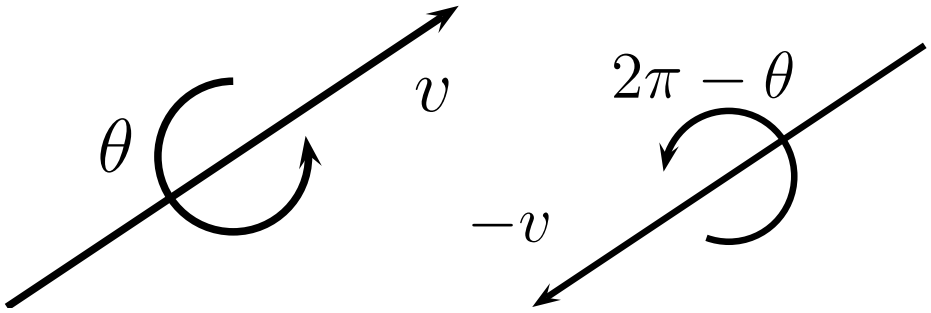
Just like a vector, the Magnitude of a quaternion is $||h|| = \sqrt{a^2 + b^2 + c^2 + d^2}$ A unit quaternion is a quaternion such that $||h|| = 1$

We can normalize a quaternion by dividing it by its magnitude: $h/||h||$.

Quaternions are a very convenient way to represent a rotation around an axis by an angle. The real component, a, represents the angle. The axis is represented by the three numbers making up the imaginary component, $v = [b, c, d]$

$$h = \cos\frac{\theta}{2} + (v_1 \sin\frac{\theta}{2})i + (v_2 \sin\frac{\theta}{2})j + (v_3 \sin\frac{\theta}{2})k$$
$$h = \cos\frac{\theta}{2} + v \sin\frac{\theta}{2}$$
$$-h = -\cos\frac{\theta}{2} - v \sin\frac{\theta}{2}$$

We assume that v is a unit vector!



(a) (b)

Figure 13: A rotation around an axis. Notice that the left and right rotations are equivalent!

3.4 Operations on Quaternions – Multiplication

One of the most important operations with a quaternion is multiplication. Given two unit quaternions representing rotations, we can think of multiplying them as combining rotations.

Given two quaternions – $h_1 = a_1 + ib_1 + jc_1 + kd_1$, $h_2 = a_2 + ib_2 + jc_2 + kd_2$ Assume $v = [b, c, d]$, like a 3-D vector. $h1 \cdot h2 = (a_1 * a_2 - v_1 \cdot v_2, a_1 v_2 + a_2 v_1 + v_1 \times v_2)$ $v_1 \cdot v_2$ is the dot product of v_1 and v_2, $v_1 \times v_2$ is the cross product. $h_1 \cdot h_2 = a_3 + ib_3 + jc_3 + kd_3$ Where:

$$
\begin{aligned}
a_3 &= a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\
b_3 &= a_1b_2 + a_2b_1 + c_1d_2 - c_2d_1 \\
c_3 &= a_1c_2 + a_2c_1 + b_2d_1 - b_1d_2 \\
d_3 &= a_1d_2 + a_2d_1 + b_1c_2 - b_2c_1
\end{aligned}
$$

Rotations Using Quaternions: Given a unit quaternion $h = a+bi+cj+dk$, define its conjugate quaternion $h* = a - bi - cj - dk$:

1. Transform point $p(x, y, z)$ by sandwiching: $h \cdot p \cdot h*$
2. Treat p as a quaternion with no real component (a=0).
3. The rotated point $p'(x', y', z')$ is obtained by the i,j,k components of the result
4. To multiply a vector and a quaternion, see matrix representation above.
5. Don't forget to translate the vector to the origin and translate back.

Combining Two Rotations

- Lemma: $(pq)^* = q^*p^*$.
- Sandwiching: $S_h(v) = h \cdot v \cdot h^*$
- $(S_{h_1} \circ S_{h_2})(v) = S_{h_1}(S_{h_2}(v)) = S_{h_1}(h_2 v h_2^*) = h_1(h_2 v h_2^*)h1^* = (h_1h_2)v(h_2^*h_1^*) = S_{h_1h_2}(v)$

Here are some useful examples:
Here are some examples:

Table 3: Quaternions representing some useful rotations. Data taken from http://www.ogre3d.org/ .

a	b	c	d	Description
1	0	0	0	Identity, no rotation
0	1	0	0	180° turn around X axis
0	0	1	0	180° turn around Y axis
0	0	0	1	180° turn around Z axis
$\sqrt{0}.5$	$\sqrt{0}.5$	0	0	90° rotation around X axis
$\sqrt{0}.5$	0	$\sqrt{0}.5$	0	90° rotation around Y axis
$\sqrt{0}.5$	0	0	$\sqrt{0}.5$	90° rotation around Z axis
$\sqrt{0}.5$	-$\sqrt{0}.5$	0	0	-90° rotation around X axis
$\sqrt{0}.5$	0	-$\sqrt{0}.5$	0	-90° rotation around Y axis
$\sqrt{0}.5$	0	0	-$\sqrt{0}.5$	-90° rotation around Z axis

Example – Rotation by 90° around Y axis: $v = [0, 1, 0]$ (the rotation axis, which is the Y axis). $\theta = 90°$. $h = \cos\frac{\theta}{2} + (v_1 \sin\frac{\theta}{2})i + (v_2 \sin\frac{\theta}{2})j + (v_3 \sin\frac{\theta}{2})k = \sqrt{0}.5 + 0 * i + \sqrt{0}.5 * j + 0 * k$ $h = \sqrt{0}.5 + \sqrt{0}.5 * j$ $h* = \sqrt{0}.5 - \sqrt{0}.5 * j$ Say $p = [1, 2, 3] = 1 * i + 2 * j + 3 * k$ Transforming p: $p' = h \cdot p \cdot h* = (\sqrt{0}.5 + \sqrt{0}.5 * j) \cdot (i + 2 * j + 3 * k) \cdot (\sqrt{0}.5 - \sqrt{0}.5 * j)$

Example – Rotation by 90° around Y axis

- Transforming p: $p' = h \cdot p \cdot h* = (\sqrt{0}.5 + \sqrt{0}.5 * j) \cdot (i + 2 * j + 3 * k) \cdot (\sqrt{0}.5 - \sqrt{0}.5 * j)$
- $h_1 \cdot h_2 = (a_1 * a_2 - v_1 \cdot v_2, a_1 v_2 + a_2 v_1 + v_1 \times v_2)$
- $p \cdot h* = -[1, 2, 3] \cdot [0, -\sqrt{0.5}, 0], \sqrt{0}.5 * [1, 2, 3] + [1, 2, 3] \times [0, -\sqrt{0.5}, 0]...$
- $h \cdot p \cdot h* = 0, 3, 2, -1$

Summary – Quaternions Vs. Matrices

- A quaternion needs 4 doubles instead of 9
- Sandwiching takes 28 multiplications while matrices need 9
- Composing rotations takes 16 multiplications with quaternions and 27 for matrices
- When composing matrices, numerical inaccuracies lead to distortions. Vectors are no longer orthonormal and angles are distorted.
- Quaternions do not distort angles and renormalization is just a division by the quaternion magnitude : $q = q/|q|$
- In interpolation with matrices $R(t) = (1 - t)R0 + tR1, R(t)$ does not represent a rotation.
- With $q(t) = (1 - t)q0 + tq1, q(t)/|q(t)|$ is a valid rotation

Some Resources

- http://mathworld.wolfram.com/RotationMatrix.html
- http://mathworld.wolfram.com/EulerAngles.html
- http://mathworld.wolfram.com/Quaternion.html

4 Problems in Structural Bioinformatics

Here – what do we do with these representations