

Quaternions and SLERP

Verena Elisabeth Kremer

University of Saarbrücken, Department for Computer Science
Seminar Character Animation
Supervisor: Michael Kipp
SS 2008

Abstract. Computing rotations is a common problem in both computer graphics and character animation. In his paper about animating rotations with quaternion curves, Ken Shoemake introduced an algorithm using Quaternions, SLERP, and Bézier Curves to solve this. The present paper discusses Shoemake's algorithm and the underlying concepts of quaternions, SLERP and Bézier Curves.

Key words: Character Animation, Quaternions, SLERP, Bézier Curves

1 Introduction

To smoothly simulate the motion of an animated character's joint (and the attached limb), one needs to compute the rotation and transition of the joint and also the way in between the starting and the ending point of the movement. The latter computation is called the in-between, or in-betweening a movement.

In the present paper, we will focus on an algorithm to compute rotations and the in-between presented by Ken Shoemake in his paper "Animating Rotation with Quaternion Curves" [1]. Shoemake suggests the use of quaternions to compute the rotation and spherical linear interpolation (SLERP) on parametrized Bézier Curves to compute the in-between. We will discuss all and their use as well as Shoemake's grand scheme in this paper.

Quaternions are hyper complex, four-dimensional numbers used to replace rotation matrices. Shoemake uses them due to the fact that a rotation can be smoothly computed by multiplying quaternions. We will have a more detailed look at this in section 2.

In section 3 we will discuss the computation of the in-between. Shoemake uses spherical linear interpolation (SLERP), a simple, straight forward formula for the interpolation between two points on a sphere. We will present some critique and discuss alternatives like normalized linear interpolation.

SLERP only yields good results for the interpolation between two points, i.e. the starting and ending points of a single rotation. To compute a smooth movement along several rotations and several corners, Shoemake used parametrized Bézier Curves which we will discuss in sections 4 and 5.

2 Geometry of Rotations

We know that movements from one point to another in three dimensional space are linear transitions of vectors, and we are able to decompose one complex transition into several simpler ones: a translation along a straight line and a rotation by an angle about an axis.

For further use we assume as given a time t_0 , a vector $\vec{u} \in \mathbb{R}^3$, and a translation $f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$. Our aim is to compute not only a vector $\vec{u}' \in \mathbb{R}^3$ at time t_1 and $\vec{u}' = f(\vec{u})$, but also corresponding vectors for time t , $t_0 \leq t \leq t_1$.

The vectors \vec{u} and \vec{u}' might represent for instance a camera or joint location at time t_0 and t_1 respectively. A particular time t would thus refer to a frame in a picture sequence. It is then not difficult to picture the sought-after vectors as the data needed to in-between frames t_0 and t_1 : this is exactly what we are after.

Rotations of a vector \vec{x} in a vector space are defined as linear transformations R that fulfill the following properties:

1. $\forall R(\vec{x}) \exists R^{-1} : R^{-1}(R(\vec{x})) = \vec{x}$
2. $R(\vec{x}) \cdot R(\vec{y}) = \vec{x} \cdot \vec{y}$
3. $R(\vec{x}) \times R(\vec{y}) = R(\vec{x} \times \vec{y})$

Rotations defined like this form with addition, scalar-, and vector product the space of rotations. The unit element is a rotation by 0° . The inverse of a rotation by α° is the rotation by $(-\alpha)^\circ$. We will discuss the space of rotations again later on.

A real square matrix whose transpose is its inverse and whose determinant is +1 fulfills the requirements listed above and is therefore a valid representation of a rotation. Matrices like this are also called rotation matrices. They are a very common way to represent rotations.

Rotation matrices conveniently correspond to geometric rotations about a fixed origin in an n-dimensional Euclidean space equipped with an Euclidean inner product (the matrix R being $\in \mathbb{R}^{n \times n}$). Thus multiplying the matrix with a n-dimensional real vector results in rotating the vector around an axis in this space. The Euclidean inner product of a real vector and a rotation matrix results

in a vector of same size, shape, and handedness. In the three-dimensional space \mathbb{R}^3 we have

$$R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \psi & -\sin \psi \\ 0 & \sin \psi & \cos \psi \end{bmatrix}, R_y(\psi) = \begin{bmatrix} \cos \psi & 0 & \sin \psi \\ 0 & 1 & 0 \\ -\sin \psi & 0 & \cos \psi \end{bmatrix}, \text{ and}$$

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

For example, to perform a 90° - rotation of a vector $\vec{v} \in \mathbb{R}^3$ around the x-axis, we compute $Rot_x(\psi) \times \vec{v}$ with $\psi = 90^\circ$.

3 Euler Angles

In order to precisely define the orientation of a rigid object in space Euler Angles can be used. Even though they may be used to compute rotations, they are originally designed to provide a clear and graphic description of the spatial orientation of a rigid object in space. Euler Angles are a set of angles between a rotated system – the one to be described – and a static system – the space the object is situated in. Euler angles are of great importance to physics, especially classical mechanics. In this context, the static system might be referred to as space coordinates while the moving system is called the body coordinates.

The fixed system consists of three axis, x , y , and z . It is associated with the Euclidean space. The rotated system consists of the axis X , Y , and Z . When no rotation has taken place, both systems are equal; $x = X$ and so on. As soon as the object is being moved, the XYZ -system rotates accordingly while the xyz system remains fixed.

The orientation of the object may now be described through three angles α , β and γ while

$$\alpha \stackrel{\text{def}}{=} \sphericalangle(x\text{-axis, line of nodes}) \pmod{2\pi} \quad (1)$$

$$\beta \stackrel{\text{def}}{=} \sphericalangle(z\text{-axis, } Z\text{-axis}); 0 \leq \beta \leq \pi \quad (2)$$

$$\gamma \stackrel{\text{def}}{=} \sphericalangle(\text{line of nodes, } X\text{-axis}) \pmod{2\pi} \quad (3)$$

There are several feasible definitions; this definition is used quite often and referred to in literature as the z-x-z convention.

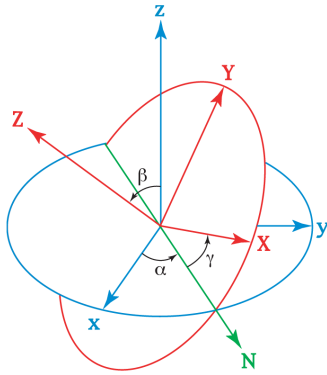


Fig. 1. The z-x-z convention; N denotes the line of nodes.

4 Quaternions

Shoemaker suggests one should use hyper complex four-dimensional numbers called quaternions to compute rotations and to perform SLERP with. Quaternions were invented by the Irish mathematician, philosopher, and astronomer Sir William Rowan Hamilton [5]. Quaternions form a group under multiplication called the Hamilton algebra [2].

The Hamilton algebra is defined as $\mathbb{H} = \{t + xi + yj + zk | t, x, y, z \in \mathbb{R}\}$ with the three imaginary units i, j, k :

$$i^2 = j^2 = k^2 = ijk = -1$$

For the sake of convenience the following notations will be used:

- a quaternion is written in **bold** text and might have an index:
 $\mathbf{q}_n \in \mathbb{H}; n \in \mathbb{Q}$
- the three imaginary units are written as $i, j,$ and k
- we write the imaginary part of a quaternion \mathbf{q} as $\vec{q} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3$
- a real number is called a scalar and is written as t_n ;
 $n \in \mathbb{Q}$ is an index number

\Rightarrow a quaternion \mathbf{q} might therefore also be written as

$$\mathbf{q} = t + (i, j, k) \vec{q}; \quad t \in \mathbb{R}$$

The multiplication table of the three imaginary units i , j , and k :

$$\begin{array}{lll} ij = k & jk = i & ki = j \\ ik = -j & ji = -k & kj = -i \end{array}$$

Quaternion are multiplied by the so called Graham Product. It is defined as

$$\begin{aligned} \vec{q}_1 \vec{q}_2 &= -1(\vec{q}_1 \cdot \vec{q}_2) + \vec{q}_1 \times \vec{q}_2 \\ \mathbf{q}_1 \mathbf{q}_2 &= t_1 t_2 (-1)(\vec{q}_1 \cdot \vec{q}_2) + (i, j, k)(t_1 \vec{q}_2 + t_2 \vec{q}_1 + (\vec{q}_1 \times \vec{q}_2)) \end{aligned}$$

Note that quaternion multiplication is not commutative up to the case of $|\mathbf{q}| \in \{1, 0\}$. This can be a significant error-source, especially in code which computes several rotations at once. To avoid this, normalized quaternions can be used.

These formulas can be derived from vector multiplication using the standard definitions of inner and outer product as in vector space on \mathbb{R} [4]. The identity quaternion number is 1 and it corresponds to the identity rotation in the space of rotations.

An inverse element can be constructed for every quaternion number:

$$\forall \mathbf{q} \in \mathbb{H} : \mathbf{q}^{-1} = t - (i, j, k) \vec{q}$$

Furthermore, the norm on \mathbb{H} is given by $|\mathbf{q}| = \sqrt{\mathbf{q} \mathbf{q}^{-1}}$.

5 Quaternions Used as Rotation Representation

Quaternions provide a straight forward way to represent, illustrate and compute rotations. This is due to the fact that quaternions can be used as a different and easier representation then rotation matrices; and a rotation can be smoothly computed by multiplying a quaternion with another. Since describing a single smooth rotation motion in three dimensional space is a principal task in computer animation, quaternions are of particular use to computer animation.

To illustrate this, we need to take a deeper look on how the space of rotation is constructed. Every rotation in a three-dimensional space is a rotation by some angle about some axis. The identity rotation is a rotation about any axis by the angle 0° . The set of possible rotations by any angle $\alpha \neq 0^\circ$ can be described as a unit sphere $S(\alpha)$; and to describe three-dimensional rotations we need a hypersphere. A hypersphere is defined as a four-dimensional sphere with a three-dimensional surface and can be described by quaternions, since unit quaternions represent by themselves a unit hypersphere:

$$S^3 = \{\mathbf{q} \in \mathbb{H} : |\mathbf{q}| = 1\}$$

Computing the rotation of a quaternion is thus very easy: we only need to multiply it with the rotation matrix' hyper complex representation, i. e. with

another quaternion [1]: a vector $\vec{v} \in \mathbb{R}^3$ can be rotated by a quaternion \mathbf{q} by computation of $\mathbf{v} = (i, j, k) \vec{v}$ and

$$\vec{v}' = Rot(\vec{v}) = \mathbf{q}\mathbf{v}\mathbf{q}^{-1}$$

The proof of a unit quaternion qualifying as rotation can be found in the appendix.

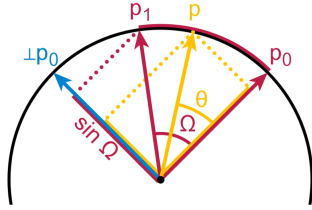
6 The In-Between – LERP, SLERP, and NLERP

Having computed a rotation successfully, whether using quaternions, rotation matrices, or Euler angles, one needs to think about the in-between. When a joint goes from one place to another, what does it look like? How can we compute the path this joint will take, and the points situated alongside?

The easiest way to interpolate between two points is the use of linear interpolation (LERP). LERP is a basic geometric formula: given the starting- and ending-points p_0 and p_1 and the interpolation parameter $t \in [0, 1]$, $LERP(p_0, p_1, t)$ yields for each t a point along the straight line connecting them.

$$LERP(p_0, p_1; t) = (1 - t)p_0 + tp_1.$$

A straight line is however not useful to animations since a rotating joint is supposed to move along a smooth curve. For this, one can use spherical linear interpolation (SLERP). SLERP is LERP, performed on the surface of a unit sphere:



$$SLERP(p_0, p_1; t) = \frac{\sin(1-t)\Omega}{\sin\Omega} p_0 + \frac{\sin t\Omega}{\sin\Omega} p_1 \quad (4)$$

$$SLERP(\mathbf{q}_1, \mathbf{q}_2, t) = \mathbf{q}_1(\mathbf{q}_1^{-1}\mathbf{q}_2)^t \quad (5)$$

Using SLERP we get $p_t = SLERP(p_0, p_1; t)$, a point along the great circle arc on the surface of the unit sphere SLERP was performed upon.

NLERP finally stands for normalized (quaternion) SLERP:

$$NLERP(|\mathbf{q}_1|, |\mathbf{q}_2|, t) = |\mathbf{q}_1|(|\mathbf{q}_1|^{-1}|\mathbf{q}_2|)^t.$$

Since the quaternions are normalized NLERP is commutative, which SLERP is not [3]. It is therefore not as error-prone as SLERP is and it is also slightly faster. But it has some disadvantages as well: since the vectors are all of the same length, the computed points in between are not as evenly arranged. A joint following a NLERP computed curve is bound to move at the beginning and at the end faster than in the middle of the movement, while SLERP ensures constant velocity. Thus NLERP can be seen as the “quick ’n’ dirty” alternative; easy, but not always reliable enough.

7 Bézier Curves

If there are only two points between to interpolate, we simply calculate $\text{SLERP}(\mathbf{q}_1, \mathbf{q}_2, t)$ – interpreting t as a discrete time parameter (the number of the current frame). But what if we want to interpolate not two, but several points on a sphere, i.e. three (\mathbf{q}_{n-1} , \mathbf{q}_n , and \mathbf{q}_{n+1})?

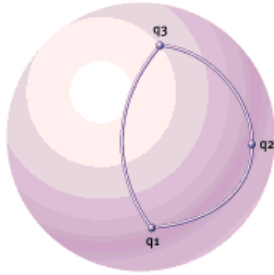


Fig. 2. Three points given[7]

We could interpolate first \mathbf{q}_{n-1} and \mathbf{q}_n and afterwards (separately) \mathbf{q}_n and \mathbf{q}_{n+1} . We would get valid results, the algorithm would work. Unfortunately we would also remain with a non-differentiable curve. To avoid this, Shoemaker suggests the use of the parametrized Bézier curves.

Bézier curves are parametric and always differentiable curves named after Pierre Bézier, who used them to design automobile bodies[6]. Nevertheless not Bézier but Paul de Casteljau was the first to construct them. The curves are dependent on a number of parameters, three in a two-dimensional space and four in a three-dimensional environment.

We assume the three successive key quaternions \mathbf{q}_{n-1} , \mathbf{q}_n , and \mathbf{q}_{n+1} as given. To build a Bézier curve with them, we have to

- compute two additional quaternion numbers, \mathbf{a}_n , and \mathbf{b}_n ,

- draw a Bézier curve with the parameters \mathbf{q}_n , \mathbf{a}_n , \mathbf{b}_n , and \mathbf{q}_{n+1} ,
- apply SLERP to the quaternions on this curve, computing the desired number of in-between points,
- and convert the data back into the original format.

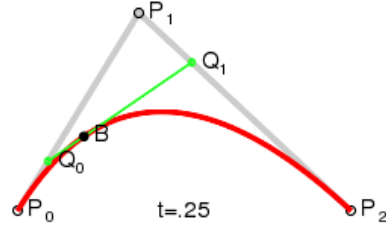


Fig. 3. An example for a two-dimensional Bézier curve [8].

To construct the quaternions \mathbf{a} and \mathbf{b} we approximate the derivative of a sampled function (with \mathbf{q}_{n-1} , \mathbf{q}_n , and \mathbf{q}_{n+1} as samples) by averaging the central differences between \mathbf{q}_{n-1} , \mathbf{q}_n , and \mathbf{q}_{n+1} . We do so using the formula Shoemaker presents in [1]:

$$\text{Biscet}(\mathbf{p}, \mathbf{q}) = \frac{\mathbf{p} + \mathbf{q}}{|\mathbf{p} + \mathbf{q}|}$$

$$\text{Double}(\mathbf{p}, \mathbf{q}) = 2 * (\mathbf{p} * \mathbf{q}) * \mathbf{q} - \mathbf{p}$$

$$\mathbf{a}_n = \text{Biscet}(\text{Double}(\mathbf{q}_{n-1}, \mathbf{q}_n), \mathbf{q}_{n+1})$$

$$\mathbf{b}_n = \text{Double}(\mathbf{a}_n, \mathbf{q}_n)$$

To construct the spherical Bézier curve with the quaternion parameters \mathbf{q}_n , \mathbf{a}_n , \mathbf{b}_n , and \mathbf{q}_{n+1} , we compute

$$\begin{aligned} \mathbf{p}_{0,0} &= \mathbf{q}_n, & \mathbf{p}_{1,0} &= \mathbf{a}_n, & \mathbf{p}_{0,1} &= \text{SLERP}(\mathbf{p}_{0,0}, \mathbf{p}_{1,0}, t), \\ \mathbf{p}_{2,0} &= \mathbf{b}_{n+1}, & \mathbf{p}_{3,0} &= \mathbf{q}_{n+1}, & \mathbf{p}_{1,1} &= \text{SLERP}(\mathbf{p}_{1,0}, \mathbf{p}_{2,0}, t), \\ & & & & \mathbf{p}_{2,1} &= \text{SLERP}(\mathbf{p}_{2,0}, \mathbf{p}_{3,0}, t), \\ & & & & \mathbf{p}_{0,2} &= \text{SLERP}(\mathbf{p}_{0,1}, \mathbf{p}_{1,1}, t), \\ & & & & \mathbf{p}_{1,2} &= \text{SLERP}(\mathbf{p}_{1,1}, \mathbf{p}_{2,1}, t), \end{aligned}$$

and finally

$$\mathbf{p}_{0,3} = \text{SLERP}(\mathbf{p}_{0,2}, \mathbf{p}_{1,2}, t) =: \mathbf{q}_{n+t}$$

while t goes from 0 to 1 in steps as big (or small) as we desire.[1] The smaller the steps, the more calculations are needed and the smoother the result will be.

8 Conclusion

Lets assume we want to perform a rotation on a joint described through vectors or Euler Angles in three-dimensional space; and we want to use Shoemake's suggestions. How does his algorithm looks like if everything is put together?

Shoemake's Grand Scheme

1. Since we want to work exclusively in \mathbb{H} , we need to convert the original data into a quaternion \mathbf{q}_n .
2. We design a unit quaternion \mathbf{r}_n or convert a rotation matrix into \mathbf{r}_n .
3. To perform the rotation itself, we compute $\mathbf{q}_{n+1} = \mathbf{r}_n \mathbf{q}_n (\mathbf{r}_n)^{-1}$
4. We repeat for any other rotations \mathbf{r}_{n+1} , \mathbf{r}_{n+2} , etc.
5. If we stopped after the first rotation, we compute the in-between via $\text{SLERP}(\mathbf{q}_n, \mathbf{q}_{n+1}, t_m) = \mathbf{q}_1 (\mathbf{q}_1^{-1} \mathbf{q}_2)^t$ with $t \in [0, 1]$, yielding the numbers $\mathbf{q}_n, \mathbf{q}_{n+t_1}, \mathbf{q}_{n+t_2}, \dots, \mathbf{q}_{n+1}$ along a smooth curve.
6. If we performed several rotations, we construct a Bézier curve to compute SLERP upon (see section 7), yielding, again, the numbers $\mathbf{q}_n, \mathbf{q}_{n+t_1}, \mathbf{q}_{n+t_2}, \dots, \mathbf{q}_{n+1}$ along a smooth curve, but also the numbers $\mathbf{q}_{n+1+t_1}, \mathbf{q}_{n+1+t_2}, \dots, \mathbf{q}_{n+2}, \dots, \mathbf{q}_{n+3}, \dots$
7. Afterwards we have a set of quaternion numbers along the path the joint takes while rotating. We only need to convert them back.

Since Shoemake introduced his idea about using quaternion SLERP on Bézier curves in 1985, the world of computation and programing has changed a lot. Thus it is even more remarkable that his algorithm is still considered important and up-to-date.

References

1. Ken Shoemake: Animating Rotation with Quaternion Curves. SIGGRAPH, Volume 19, Number 3 (1985)
2. Michael E. Mortenson: Mathematics for Computer Graphics Applications. Industrial Press, New York (1999)
3. Falk Reinhardt, Soeder: Atlas der Mathematik, Band 1. Deutscher Taschenbuch Verlag, Mnchen (2001)
4. Sir William Hamilton: On quaternions, or on a new system of imaginaries in algebra. Philosophical Magazine xxv, Dublin (1844)
5. Thomas Hankins: Sir William Rowan Hamilton. John Hopkins University Press, Baltimore (1980)
6. Pierre Bézier: Numerical Control – Mathematics and Applications. John Wiley and Sons, London (1972)
7. Image by Nick Bobick, wiki commons (2004)
8. Image by Phil Tregoning, wiki commons (2007)

9 Appendix: Unit Quaternions Represent Rotations

Assuming the computation of a quaternion rotation to be $R(\mathbf{x}) = \mathbf{r}\mathbf{x}\mathbf{r}^{-1}$, we have to show $\forall \mathbf{r}, |\mathbf{r}| = 1$ and $\forall \mathbf{x}, \mathbf{y}; \Re \mathbf{x} = \Re \mathbf{y} = 0$ that the following equations hold:

1. $\exists R^{-1} : R^{-1}(R(\mathbf{x})) = \mathbf{x}$
2. $R(\mathbf{x}) \cdot R(\mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$
3. $R(\mathbf{x}) \times R(\mathbf{y}) = R(\mathbf{x} \times \mathbf{y})$

To prove $\exists R^{-1} : R^{-1}(\mathbf{x}) = \mathbf{x}$, we can derive from $R^{-1} = \mathbf{r}^{-1}$:

$$R^{-1}(R(\mathbf{x})) = (\mathbf{r}^{-1})(R(\mathbf{x}))(\mathbf{r}^{-1})^{-1} \quad (6)$$

$$= (\mathbf{r}^{-1})((\mathbf{r})(\mathbf{x})(\mathbf{r}^{-1}))(\mathbf{r}) \quad (7)$$

$$= (\mathbf{r}^{-1}(\mathbf{r}))(\mathbf{x})(\mathbf{r}^{-1}(\mathbf{r})) \quad (8)$$

$$= (\mathbf{x}) \quad (9)$$

$$(10)$$

Prove of $R(\mathbf{x}) \cdot R(\mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$:

$$R(\mathbf{x}) \cdot R(\mathbf{y}) = \frac{1}{2}(R(\mathbf{x})(R(\mathbf{y}))^{-1} + R(\mathbf{y})(R(\mathbf{x}))^{-1}) \quad (11)$$

$$= \frac{1}{2}((\mathbf{r}\mathbf{x}\mathbf{r}^{-1})(\mathbf{r}\mathbf{y}\mathbf{r}^{-1})^{-1} + (\mathbf{r}\mathbf{y}\mathbf{r}^{-1})(\mathbf{r}\mathbf{x}\mathbf{r}^{-1})^{-1}) \quad (12)$$

$$= \frac{1}{2}((\mathbf{r}\mathbf{x}\mathbf{r}^{-1})(\mathbf{r}\mathbf{y}^{-1}\mathbf{r}^{-1}) + (\mathbf{r}\mathbf{y}\mathbf{r}^{-1})(\mathbf{r}\mathbf{x}^{-1}\mathbf{r}^{-1})) \quad (13)$$

$$= \frac{1}{2}((\mathbf{r}\mathbf{x})(\mathbf{r}^{-1}\mathbf{r})(\mathbf{y}^{-1}\mathbf{r}^{-1}) + (\mathbf{r}\mathbf{y})(\mathbf{r}^{-1}\mathbf{r})(\mathbf{x}^{-1}\mathbf{r}^{-1})) \quad (14)$$

$$= \frac{1}{2}((\mathbf{r}\mathbf{x}\mathbf{y}^{-1}\mathbf{r}^{-1}) + (\mathbf{r}\mathbf{y}\mathbf{x}^{-1}\mathbf{r}^{-1})) \quad (15)$$

$$= \mathbf{r}\left(\frac{\mathbf{x}\mathbf{y}^{-1} + \mathbf{y}\mathbf{x}^{-1}}{2}\right)\mathbf{r}^{-1} \quad (16)$$

$$= \mathbf{x} \cdot \mathbf{y}. \quad (17)$$

$$(18)$$

Prove of $R(\mathbf{x}) \times R(\mathbf{y}) = R(\mathbf{x} \times \mathbf{y})$:

$$R(\mathbf{x}) \times R(\mathbf{y}) = \frac{1}{2}(R(\mathbf{x})R(\mathbf{y}) - R(\mathbf{y})R(\mathbf{x})) \quad (19)$$

$$= \frac{1}{2}((\mathbf{rxr}^{-1})(\mathbf{ryr}^{-1}) - (\mathbf{ryr}^{-1})(\mathbf{rxr}^{-1})) \quad (20)$$

$$= \frac{1}{2}((\mathbf{rx})(\mathbf{r}^{-1}\mathbf{r})(\mathbf{yr}^{-1}) - (\mathbf{ry})(\mathbf{r}^{-1}\mathbf{r})(\mathbf{xr}^{-1})) \quad (21)$$

$$= \frac{1}{2}((\mathbf{rx})(\mathbf{yr}^{-1}) - (\mathbf{ry})(\mathbf{xr}^{-1})) \quad (22)$$

$$= \frac{1}{2}((\mathbf{r}(\mathbf{xy})\mathbf{r}^{-1}) - (\mathbf{r}(\mathbf{yx})\mathbf{r}^{-1})) \quad (23)$$

$$= \mathbf{r}\left(\frac{\mathbf{xy} - \mathbf{yx}}{2}\right)\mathbf{r}^{-1} \quad (24)$$

$$= R(\mathbf{x} \times \mathbf{y}) \quad (25)$$

$$(26)$$