Introduction to Group Theory and Rotations in Microstructural Analysis

Lecture 2 - Quaternions

J. Gruber
Department of Materials Science and Engineering
Carnegie Mellon University

Outline

Lecture 2 - Quaternions

Complex numbers
- The complex plane
- Unit complex numbers
- Complex numbers and 2D rotations
- Reparameterizations

Quaternions
- Historical notes
- Definition and properties
- Unit quaternions
- Quaternions and 3D rotations
- Computations with quaternions

Summary

Complex numbers (1)

The ***complex numbers*** $\mathbb{C}$ are can be thought of as elements in $\mathbb{R}^2$ with special definitions for addition and multiplication, i.e.

$$(a, b) \leftrightarrow a + ib = c \quad \text{with} \quad a, b \in \mathbb{R} \quad \text{and} \quad c \in \mathbb{C}.$$

The use of i is really for notational convenience only. The algebraic structure $(\mathbb{C}, +, \times)$ is an example of a ***field***. The binary operators + and × are defined so that

$$c_1 + c_2 = (a_1 + a_2) + i(b_1 + b_2) \quad \text{and}$$

$$c_1 \times c_2 = (a_1 a_2 - b_1 b_2) + i(a_1 b_2 + a_2 b_1)$$

for $c_1, c_2 \in \mathbb{C}$. We could, of course, use this to define the same operators for elements (vectors) in $\mathbb{R}^2$.

Complex numbers (2)

Unit complex numbers $\mathbb{C}^*$ are those complex numbers with unit norm, norm being defined as

$$\| c \|^2 = a^2 + b^2 \quad \text{with} \quad c = a + ib$$

Visualizing this in $\mathbb{R}^2$, the unit complex numbers are all points that are unit distance from the origin, i.e. the unit circle.

Recall that we can parameterize $\mathbb{R}^2$ using the polar coordinates r and θ, so that now (with $r = 1$ for unit length)

$$(cos(\theta), sin(\theta)) = (a, b) \leftrightarrow \quad a + ib = c$$

or, using Euler's formula,

$$c = e^{i\theta} \quad \text{for } c \in \mathbb{C}^*$$

Complex numbers (3)

There's something quite special about multiplication using the unit complex numbers ...

$$c_1 = e^{i\theta_1} \text{ and } c_2 = e^{i\theta_2} \text{ with } c_1, c_2 \in \mathbb{C}^*$$

$$c_1 \times c_2 = e^{i\theta_1} \times e^{i\theta_2} = e^{i(\theta_1+\theta_2)}$$

The product of two unit complex numbers can be computed by simply adding their polar angles. It's fairly simple to show that the algebraic structure $(\mathbb{C}^*, \times)$ is a group.

Noting that a polar angle sum is equivalent modulo 2π, we find that the $(\mathbb{C}^*, \times)$ is isomorphic to the real numbers under addition modulo 2π, the group $(\mathbb{R}, +_{2\pi})$, which in turn is isomorphic to the group of rotations in 2 dimensions, $SO(2)$.

Complex numbers (4)

We can represent 2D rotations by unit complex numbers, and the combination of two rotations with complex multiplication.

It's also easy to determine how a rotation represented by a unit complex number transforms a vector $\boldsymbol{x}$:

Matrix form is $\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$ so we have

$$\boldsymbol{x}' = c\,\boldsymbol{x} = (ax_1 + bx_2,\ ax_2 - bx_1) \quad \text{with} \quad c = a + ib$$

Complex numbers require storing half as many variables as matrices. Also note that complex multiplication requires 4 floating point multiplications and 2 floating point additions, whereas 2×2 matrix multiplication requires twice as many of each operation.

Complex numbers (5)

One final comment: it's not absolutely necessary to use a bijection (isomorphism) from the complex numbers to SO(2) to parameterize 2D rotations. We could just as well use an *n*-to-one mapping, as long as it preserves the binary operation, i.e. is a ***homomorphism***.

As an example, let $f : \mathbb{C}^* \rightarrow SO(2)$ be defined by

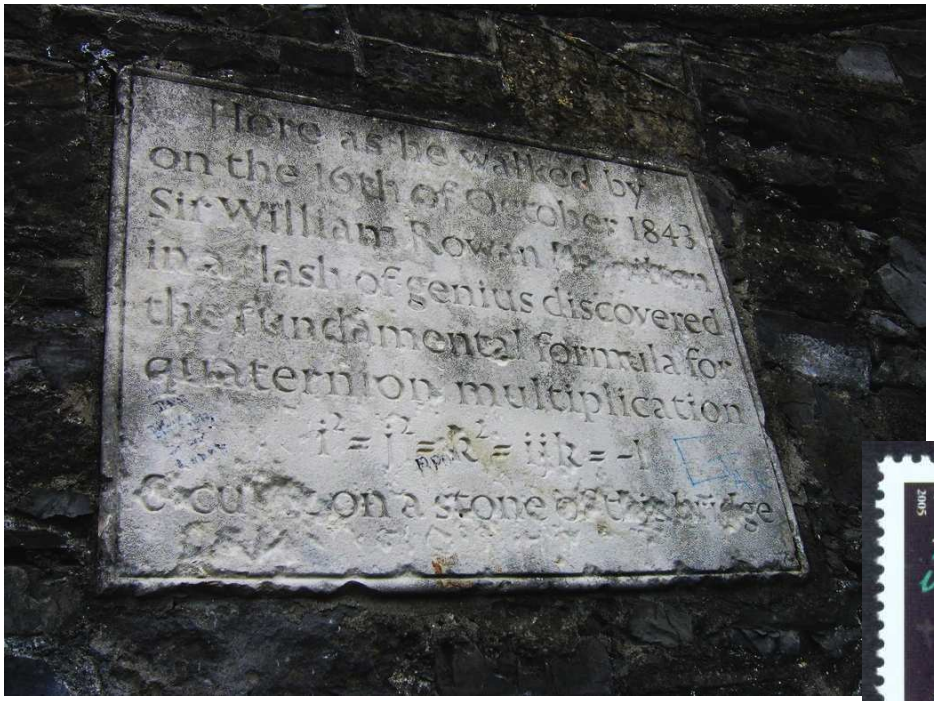
$$f : cos(\theta/2) + i\ sin(\theta/2) \rightarrow \theta$$

then f is a homomorphism that maps two unit complex numbers to each element in $SO(2)$.

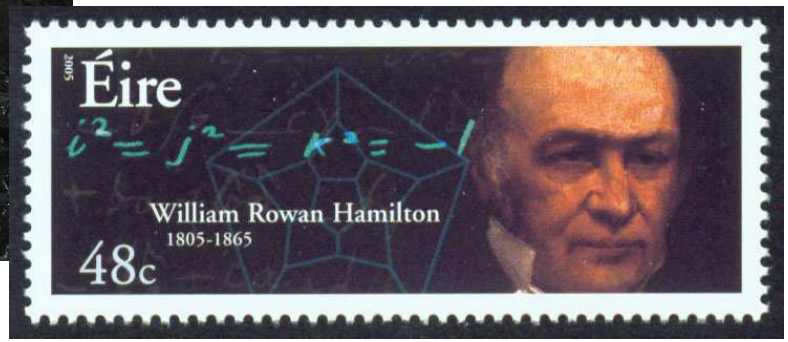
We could define a similar mapping for any $n > 0$.

Quaternions (1)

The ***quaternions*** were discovered in 1843 by Irish mathematician Sir William Rowan Hamilton (1805-1865). According to his own account, the discovery occured as a flash of insight while walking near a Dublin bridge.



Here as he walked by
on the 16th of October 1843
Sir William Rowan Hamilton
in a flash of genius discovered
the fundamental formula for
quaternion multiplication
$i^2 = j^2 = k^2 = ijk = -1$
& cut it on a stone of this bridge.



Quaternions (2)

Just as complex numbers could be interpreted as vectors in $\mathbb{R}^2$, the ***quaternions*** Q can be thought of as vectors in $\mathbb{R}^4$, usually with basis $\{1, i, j, k\}$ such that each $q \in Q$ is of the form

$$q = (a, b, c, d) = a + ib + jc + kd$$

with $a, b, c, d \in \mathbb{R}$

Like the complex numbers, $(Q, +, \times)$ uses vector addition, but uses a special definition for the associative (not commutative) multiplication operator. We compute the product of two quaternions using Hamilton's formula

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

Thus, for example, $ij = k$ and $ji = -k$, etc.

Quaternions (3)

The ***unit quaternions*** are defined in a similar way to unit complex numbers. By analogy with the unit complex numbers, might we use the unit quaternions to represent rotations in three dimensions?

Hopf (1940) proved that it is impossible to construct an isomorphism between the unit quaternions and $SO(3)$. We can, however, construct a two-to-one homomorphism $f : Q \rightarrow SO(3)$ exactly like we did for the complex numbers

$$f : cos(\theta/2) + i\, x\, sin(\theta/2) + j\, y\, sin(\theta/2) + k\, z\, sin(\theta/2) \rightarrow (\mathbf{x}, \theta)$$

where $\mathbf{x} = (x, y, z)$ is a normalized rotation axis and θ is a rotation angle. The proof in this case is, of course, much more difficult.

Quaternions (4)

With the two-to-one mapping between Q^* and $SO(3)$, it's always the case that q and $-q$ represent the same rotation.

Note also that the multiplicative inverse of any $q = (a, b, c, d)$ is simply $q^{-1} = (-a, b, c, d)$. Computing inverses with quaternions is more efficient than with matrices.

Now, more on multiplication: we perform the multiplication using the distributive property and Hamilton's formula, i.e.

$$\begin{aligned} q_1 \times q_2 = {} & (a_1 + ib_1 + jc_1 + kd_1)\,(a_2 + ib_2 + jc_2 + kd_2) \\ = {} & (a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2) \\ & + i\,(a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2) \\ & + j\,(a_1c_2 + c_1a_2 - b_1d_2 + d_1b_2) \\ & + k\,(a_1d_2 + d_1a_2 + b_1c_2 - c_1b_2) \end{aligned}$$

Quaternions (5)

A warning: the formula given for quaternion multiplication implies that ***the rotation q2 follows q1***. Multiplication of quaternions is ***not commutative***, and there is nothing in their definition or the definition of multiplication that suggestions in what order we should ***write*** the product $q_1 \times q_2$.

Also, note that with the given two-to-one homomorphism, unit quaternions represent ***coordinate (passive) transformations***.

Finally, recall that ***the rotation angle is counter-clockwise***.

How do we keep all this straight? Whether using your own computations or code from another source, test everything! With so many different conventions implicit in using quaternions, make no assumptions about what others have used.

Quaternions (6)

Why should we use quaternions (as opposed to matrices)?

(1) Efficiency - Quaternion multiplication requires far fewer computations than matrix multiplication.

(2) Simplicity - We ususally think of rotations as being composed of a rotation axis and a rotation angle. Converting an axis/angle pair to a quaternion is trivial and has fewer chances for error than converting to a matrix.

(3) Other nice properties - easy to extract rotation angle from a quaternion, parameter space is singularity free, etc.

Quaternions (7)

Now let's look at conversions.

Converting Euler angles (Bunge, Kocks, Roe, etc.) to a unit quaternion is easy. First write each of the three rotations as quaternions, perform quaternion multiplication, then simplify. Computing the inverse transformation is then trivial.

Example: Convert Bunge angles to quaternion.

$$q(\varphi_1) = cos(\varphi_1/2) + i\ sin(\varphi_1/2)$$
$$q(\Phi) = cos(\Phi/2) + k\ sin(\Phi/2)$$
$$q(\varphi_2) = cos(\varphi_2/2) + i\ sin(\varphi_2/2)$$

Compute $q(\varphi_1) \times q(\Phi) \times q(\varphi_2)$

Simplify using trionometric identities

Quaternions (8)

More on conversions.

Converting to axis/angle representation or to Rodrigues vectors is trivial.

Converting to a matrix could be accomplished by first performing a simpler conversion, then converting that representation to matrix. The answer is

$$q \leftrightarrow \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2(ad + bc) & -2(ac - bd) \\ -2(ad - bc) & a^2 - b^2 + c^2 - d^2 & 2(ab + cd) \\ 2(ac + bd) & -2(ab - cd) & a^2 - b^2 - c^2 + d^2 \end{bmatrix}$$

Quaternions (9)

Now an example application.

Example: Reproduce the Mackenzie distribution.

The Mackenzie distribution is a histogram that describes the probability of finding a pair of grains related by a given disorientation angle in cubic materials.

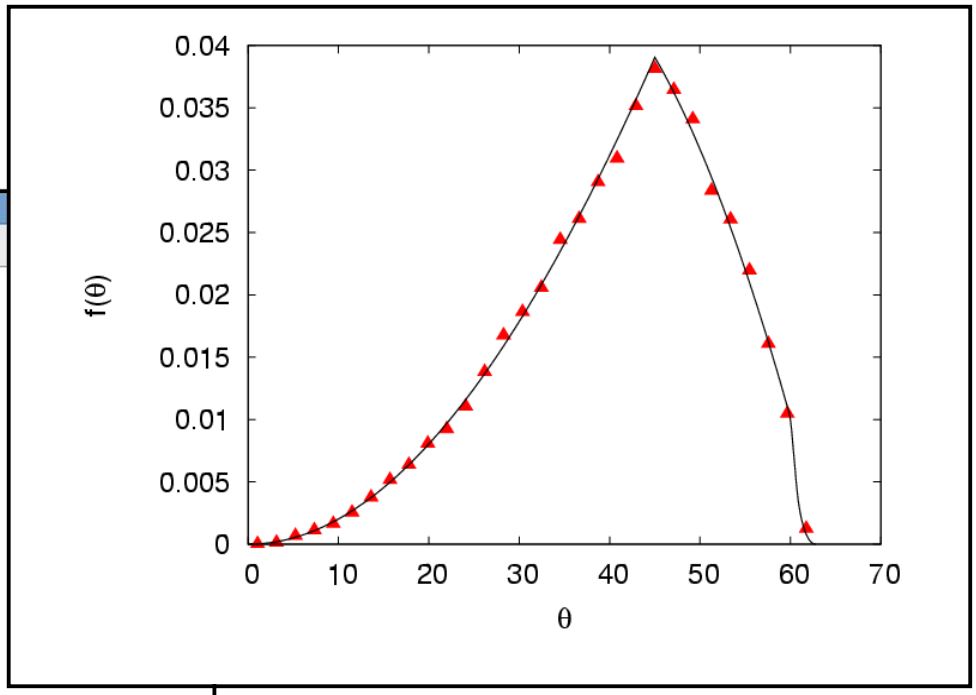
We need to generate random unit quaternion orientations for two imaginary grains, compute their disorientation, and repeat. To generate random orientations $q(a, b, c, d)$, use

$$a = cos(2\pi r_1)\sqrt{r_3}, \qquad b = sin(2\pi r_2)\sqrt{(1-r_3)}$$
$$c = cos(2\pi r_2)\sqrt{(1-r_3)}, \qquad d = sin(2\pi r_1)\sqrt{r_3}$$

where r_1, r_2, r_3 are random numbers in [0,1).

Quaternions (10)

Result : 80 K "grain boundaries"



disorientation.h (~/src/texture) - VIM

File Edit View Terminal Tabs Help

```
#include"symmetry.h"


double disorientation(const quaternion& q)
{
    // return cubic symmetry disorientation
    using namespace symmetry::O;

    double disorientation = 2.0*M_PI;
    for (int i=0; i<operators; i++) {
        quaternion a = O[i]*q;
        for (int j=0; j<operators; j++) {
            quaternion b = a*O[j];
            double angle1 = 2.0*acos(b[0]);
            double angle2 = 2.0*M_PI-angle1;
            if (angle1<disorientation) disorientation = angle1;
            if (angle2<disorientation) disorientation = angle2;
        }
    }

    return disorientation;
}
```

Summary

Lecture 2 - Quaternions

In this lecture, we've introduced the use of complex numbers and quaternions for parameterizing rotations in two and three dimensions.

Quaternions are incredibly useful for parameterizing three dimensional rotations, however, great care must be taken to perform computations in a self-consistent manner.