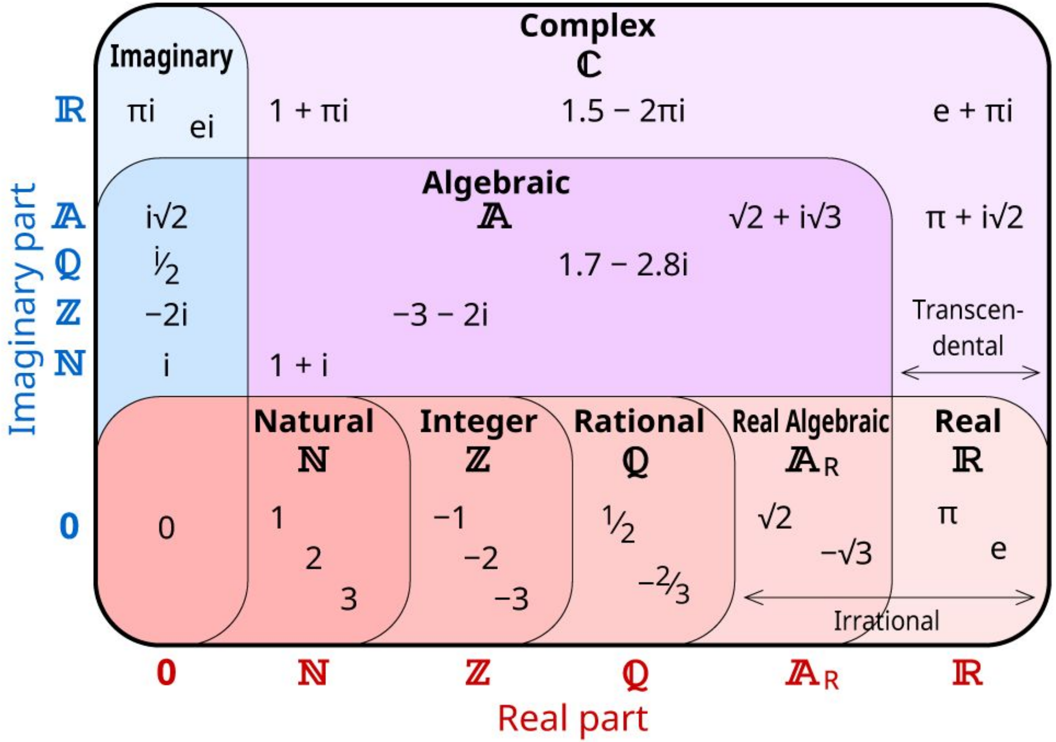
VGP334 - Quaternions

Instructor: Peter Chan

Number Sets



Complex
ℂ
Imaginary
πi
ei
1 + πi
1.5 − 2πi
e + πi
Algebraic
𝔸
i√2
√2 + i√3
π + i√2
i/2
1.7 − 2.8i
−2i
−3 − 2i
Transcen-
dental
i
1 + i
Natural
ℕ
Integer
ℤ
Rational
ℚ
Real Algebraic
𝔸R
Real
ℝ
0
1
2
3
−1
−2
−3
1/2
−2/3
√2
−√3
π
e
Irrational
Imaginary part
ℝ
𝔸
ℚ
ℤ
ℕ
0
Real part
0
ℕ
ℤ
ℚ
𝔸R
ℝ

Complex Numbers

An ordered pair of real numbers z = (a, b) is a complex number, where a is the *real* component and b is the *imaginary* component. Moreover, the arithmetic operations are defined as follows:

$$(a+bi)+(c+di) = (a+c)+(b+d)i$$

$$(a+bi)-(c+di) = (a-c)+(b-d)i$$

$$(a+bi)*(c+di) = (ac-bd)+(ad+bc)i$$

$$(a+bi)\div(c+di) = \frac{ac+bd}{c^2+d^2}+\frac{bc-ad}{c^2+d^2}i$$

Imaginary Unit

We define the *imaginary unit*, i = (0, 1). Using the definition of complex multiplication, observe that:

$$i^2 = (0, 1)(0, 1) = (-1, 0) = -1$$

Or

$$i = \sqrt{-1}$$

Complex Conjugate

The *complex conjugate* of a complex number z = (a, b) is denoted by:

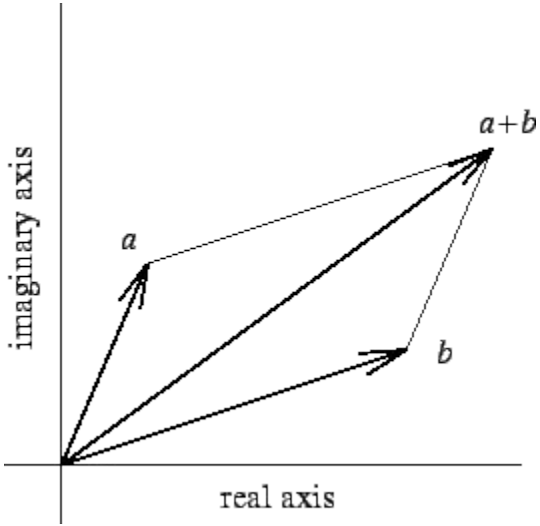
$$\begin{aligned} z &= (a + bi) \\ z^* &= (a - bi) \end{aligned}$$

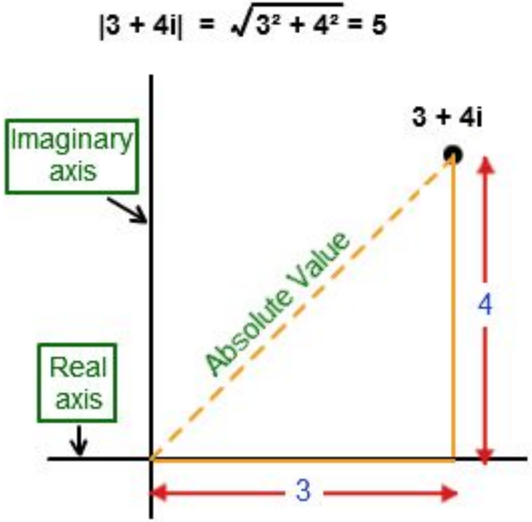
The product of a complex number and its conjugate gives a real number:

$$\begin{aligned} z &= (a + bi) \\ z^* &= (a - bi) \\ zz^* &= (a + bi)(a - bi) \\ &= a^2 - abi + abi + b^2 \\ &= a^2 + b^2 \end{aligned}$$

Geometric Interpretation

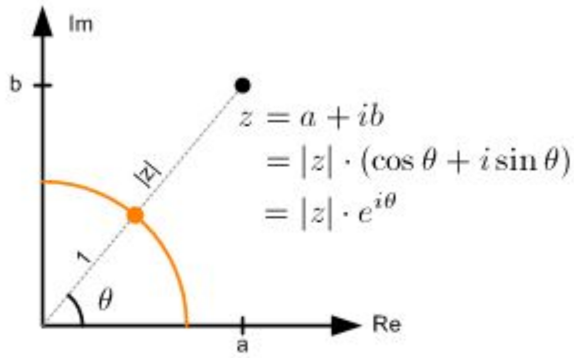
We can think of a complex number geometrically as a 2D vector. In fact, addition, scalar multiplication, and the absolute value (magnitude) works the same:





Polar Representation

Because complex numbers can be viewed as 2D vectors, we can just as well express their components using polar coordinates:



Rotation

What happens when you multiply two complex numbers in polar form?

$$z_1 = r_1(\cos\theta_1 + i\sin\theta_1) \quad \text{and} \quad z_2 = r_2(\cos\theta_2 + i\sin\theta_2)$$

Product: $z_1 z_2 = r_1 r_2\left[\cos(\theta_1 + \theta_2) + i\sin(\theta_1 + \theta_2)\right]$

Quotient: $\dfrac{z_1}{z_2} = \dfrac{r_1}{r_2}\left[\cos(\theta_1 - \theta_2) + i\sin(\theta_1 - \theta_2)\right]$

You have ROTATION!

Quaternions

An ordered 4-tuple of real numbers q = (s, x, y, z) is a quaternion. This is commonly abbreviated as q = (s, v), where v = (x, y, z) is the imaginary vector part and s is the real part. Quaternions addition is similar to vector addition:

$$\text{If } p = [s, v], \quad q = [s', v'], \quad \text{then } p + q = [s + s', v + v']$$

$$\begin{aligned} p + q &= [s, v] + [s', v'] \\ &= (s + ix + jy + kz) + (s' + ix' + jy' + kz') \\ &= (s + s') + i(x + x') + j(y + y') + k(z + z') \\ &= [s + s', v + v'] \end{aligned}$$

Special Products

By definition, the basis i, j, k have the following multiplicative property:

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

which leads to the following results:

$$\begin{aligned} ij &= k, \qquad & ji &= -k, \\ jk &= i, \qquad & kj &= -i, \\ ki &= j, \qquad & ik &= -j, \end{aligned}$$

Hamilton Product

We can now define the multiplication of two quaternions as follows using the special products and the distributive law:

$$
\begin{aligned}
&(p_0, p_1, p_2, p_3)(q_0, q_1, q_2, q_3) \\
&\quad = (p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3,\ p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2, \\
&\qquad p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1,\ p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0) \qquad (5.10)
\end{aligned}
$$

Hamilton Product

Conveniently, this can be represented as a matrix. Note that this implies that quaternion multiplication is not commutative:

$$PQ = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

Relation with Scalar and Vector

We can relate real numbers, vectors, and quaternions in the following way. Let s be a real number and let u = (x, y, z) be a vector, then:

$$s = (s, 0, 0, 0) \text{ and } u = (0, x, y, z)$$

In other words, any real number can be thought of as a quaternion with a zero vector part, and any vector can be thought of as a quaternion with a zero real part. Furthermore, the *identity* quaternion is simply:

$$1 = (1, 0, 0, 0)$$

Conjugate and Norm

The conjugate and norm (magnitude) of quaternions are similar to complex number as follows:

$$Q^* = (q_0, -q_1, -q_2, -q_3)$$

$$|Q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$$

Inverse

Similar to matrices, quaternion division can be carried out by simply multiplying by its inverse. We can compute the inverse as follows:

$$qq^{-1} = [1, \mathbf{0}] = 1$$

$$q^* q q^{-1} = q^*$$

$$|q|^2 q^{-1} = q^*$$

$$q^{-1} = \frac{q^*}{|q|^2}$$

For unit quaternions whose norm is 1, we can write:

$$q^{-1} = q^*$$

Axis-Angle Rotation

$$\begin{pmatrix} x^2(1-c)+c & xy(1-c)-zs & xz(1-c)+ys & 0 \\ xy(1-c)+zs & y^2(1-c)+c & yz(1-c)-xs & 0 \\ xz(1-c)-ys & yz(1-c)+xs & z^2(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where $c = \cos\theta, \quad s = \sin\theta$

Axis-Angle Quaternion (for Quaternion::RotationAxis)

$$q_w = \cos\frac{\theta}{2.0}$$

$$q_x = v_x \sin\frac{\theta}{2.0}$$

$$q_y = v_y \sin\frac{\theta}{2.0}$$

$$q_z = v_z \sin\frac{\theta}{2.0}$$

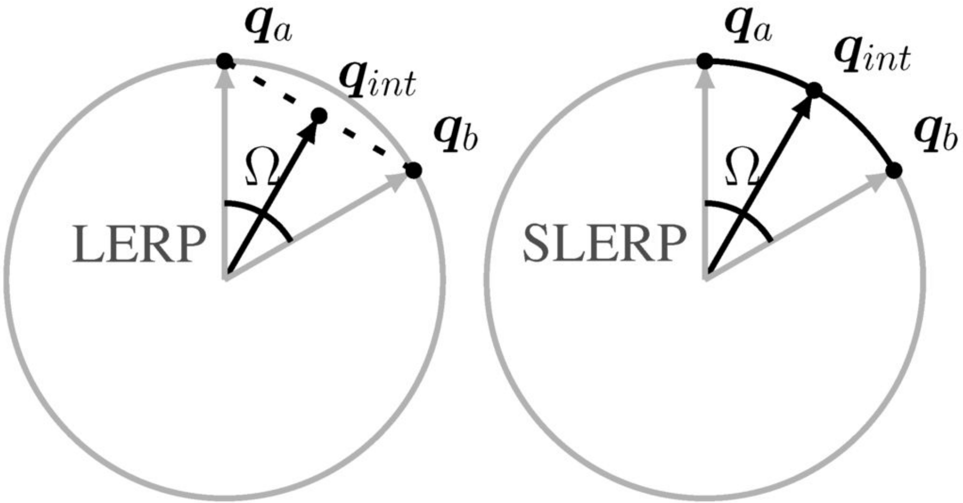
Quaternion Rotation Matrix (for Matrix4::RotationQuaternion)

$$\mathbf{q} = (\cos(\theta/2), \sin(\theta/2)\vec{a}) = (w, (x, y, z))$$

$$R_{\mathbf{q}} = \begin{pmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2wz & 2xz + 2wy & 0 \\ 2xy + 2wz & 1 - 2x^2 - 2z^2 & 2yz - 2wx & 0 \\ 2xz - 2wy & 2yz + 2wx & 1 - 2x^2 - 2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This is for right handed system, i.e. we need to transpose this for our math library!

Quaternion Slerp



Quaternion Slerp

$$Lerp(v_0, v_1, t) = v_0 + t(v_1 - v_0) \quad \text{(points)}$$

$$Slerp(q_i, q_f, t) = q_i (q_i^{-1} q_f)^t \quad \text{(quaternions)}$$

$$Slerp(p_0, p_1, t) = \frac{\sin(1-t)\Omega}{\sin\Omega} p_0 + \frac{\sin t\Omega}{\sin\Omega} p_1$$

$$\text{where } \Omega = \arccos(p_0 \cdot p_1)$$

** Note that you may need to determine the Slerp direction (by checking the Dot(p_o, p_1)), see code

Quaternion Pros and Cons

Pros

- Does not suffer from Gimbal Lock (https://en.wikipedia.org/wiki/Gimbal_lock)
- Memory efficient (4 floats for any arbitrary rotation compared to 9 for a rotation matrix)
- Can smoothly interpolate via SLERP

Cons

- Harder to comprehend and debug