加藤 忠

クオータニオン（四元数，Quaternion）とは，複素数を拡張した数体系で，実数部と，虚数単位$i$，$j$，$k$の虚数部で表現されます．虚数単位$i$に，$j$と$k$が追加されています．合計4つの次元を持つため，日本語では四元数とも呼ばれます．三元数や五元数では数体系として成立せず，四元数である理由は大変興味深く，文献(1)に深い考察があります．

本稿ではクオータニオンの数学的な特徴を紹介し，ベクトルとの親和性や，回転変換との関係性について説明します．最後に，角速度センサから姿勢角を計算する技術を紹介します．

なお，本章では数式を通し番号で記します．

### ● Python環境設定

クオータニオン演算は，numpy-quaternionライブラリを使います．pipコマンドで，インストール可能です．

```
pip install numpy-quaternion
```

依存ライブラリは，NumPy，SciPy，Numbaで，別途インストールが必要です．Anacondaに同梱されているため，AnacondaからのPython導入がお勧めです．類似ライブラリではpyquaternionライブラリも同等の機能を有しています．回転変換にだけ特化するならば，SciPyライブラリのscipy.spatial.transform.Rotationも使われています．

◆参考文献◆

(1) 矢野 忠；四元数の発見，2014年9月，海鳴社．

# 5-1 クオータニオンの基本(1)

### ● 概要

クオータニオンの基本事項として，クオータニオンの表記の仕方，虚数単位の取り扱い，結合則など演算方法について解説します．

### ● 仕組み

クオータニオンは，$a$，$b$，$c$，$d$をスカラ値とした場合，次のように表記されます．

$$\tilde{q} = a + bi + cj + dk \quad \cdots\cdots(1)$$

**▶虚数単位の演算**

次のように定義されます．虚数の次元を拡張した代償として，積の交換則は成立しません．

$$i^2 = j^2 = k^2 = ijk = -1 \quad \cdots\cdots(2)$$
$$ij = k,\ jk = i,\ ki = j \quad \cdots\cdots(3)$$
$$ji = -k,\ kj = -i,\ ik = -j \quad \cdots\cdots(4)$$

この不思議なクオータニオンは3次元の回転変換を記述できる性質を持っています．飛翔体の姿勢角や，3D CG計算などで多用されています．

**▶結合則，分配則，交換則**

和と積の結合則［式(5)，式(6)］，分配則［式(7)，式(8)］，和の交換則［式(9)］は成立します．積の交換則［式(10)］だけ成立しません．

$$\tilde{q}_1 + (\tilde{q}_2 + \tilde{q}_3) = (\tilde{q}_1 + \tilde{q}_2) + \tilde{q}_3 \quad \cdots\cdots(5)$$
$$\tilde{q}_1(\tilde{q}_2\tilde{q}_3) = (\tilde{q}_1\tilde{q}_2)\tilde{q}_3 \quad \cdots\cdots(6)$$
$$\tilde{q}_1(\tilde{q}_2 + \tilde{q}_3) = \tilde{q}_1\tilde{q}_2 + \tilde{q}_1\tilde{q}_3 \quad \cdots\cdots(7)$$
$$(\tilde{q}_1 + \tilde{q}_2)\tilde{q}_3 = \tilde{q}_1\tilde{q}_3 + \tilde{q}_2\tilde{q}_3 \quad \cdots\cdots(8)$$
$$\tilde{q}_1 + \tilde{q}_2 = \tilde{q}_2 + \tilde{q}_1 \quad \cdots\cdots(9)$$
$$\tilde{q}_1\tilde{q}_2 \neq \tilde{q}_2\tilde{q}_1 \quad \cdots\cdots(10)$$

**▶和差積**

次の2つのクオータニオン$\tilde{q}$，$\tilde{r}$を使って表したとき，

$$\tilde{q} = q_w + q_x i + q_y j + q_z k$$
$$\tilde{r} = r_w + r_x i + r_y j + r_z k$$

$\tilde{q}$，$\tilde{r}$の和と差を式(11)と式(12)に，積を式(13)に示します．積は分配則と式(2)，式(3)，式(4)から導かれます．

$$\tilde{q} + \tilde{r} = (q_w + r_w) + (q_x + r_x)i + (q_y + r_y)j + (q_z + r_z)k \quad \cdots(11)$$
$$\tilde{q} - \tilde{r} = (q_w - r_w) + (q_x - r_x)i + (q_y - r_y)j + (q_z - r_z)k \quad \cdots(12)$$
$$\begin{aligned}\tilde{q}\tilde{r} = {} & (q_w r_w - q_x r_x - q_y r_y - q_z r_z) + \\ & (q_w r_x + q_x r_w + q_y r_z - q_z r_y)i + \\ & (q_w r_y - q_x r_z + q_y r_w + q_z r_x)j + \\ & (q_w r_z + q_x r_y - q_y r_x + q_z r_w)k \end{aligned} \quad \cdots\cdots(13)$$

### ● コード

クオータニオン演算には，numpy-quaternionライブラリを使います．式(1)と式(11)～式(13)の基本演

**リスト1 クオータニオンの基本演算**（その1）

```
import numpy as np
import quaternion

q1 = np.quaternion(1,2,3,4)   # 1+2i+3j+4k …式(1)
q2 = np.quaternion(5,6,7,8)    # 5+6i+7j+8k
# .w : 実数部 , .x /.y /.z : 虚数部 i / j / k
print("q1 = {}+{}i+{}j+{}k".format(
          q1.w, q1.x, q1.y, q1.z) )
print( q1 + q2 )    # 和 …式(11)
print( q1 - q2 )    # 差 …式(12)
print( q1 * q2 )    # 積 …式(13)
# スカラ値と混合した四則演算も可能 （除算は後述）
q3 = q1 * 2.1 + q1 * q2 + 4.2 + 3.6 / q1
```