# Uso de Cuaterniones para Representar Rotaciones

**Por Dr. Anatoli Markelov**
**anatoli@cwpanama.net**

El articulo actual presenta una breve descripción de objetos matemáticos inventados por Hamilton hace casi un siglo y medio y llamados Cuaterniones (en inglés, Quaternions). Por su complejidad, estos objetos no han logrado una propagación amplia en matemáticas aplicadas, pero ya llegó tiempo de usarles en la práctica ingenieril por haber implementado en gran escala computadoras personales. Una visión sobre el uso de Cuaterniones para representar rotaciones en el espacio de tres dimensiones (3D) se propone en el artículo.

Es bien conocido que en el plano, la operación de multiplicación de dos números complejos representa una rotación con escalamiento. Con esto el conjunto de números complejos forma una estructura algebraica llamada campo lo que significa, en particular, que el producto de dos números complejos es una operación conmutativa, además de otras propiedades. Un Cuaternión se define como un objeto de cuatro dimensiones: $q = w + x*\mathbf{i} + y*\mathbf{j} + z*\mathbf{k}$, donde $w, x, y, z$ son números reales (elementos del espacio $\mathbf{R}^4$=RXRXRXR), **i**, **j**, **k** son unidades imaginarias definidas por medio de un sistema de igualidades:

$\mathbf{i}*\mathbf{i} = -1$; $\mathbf{j}*\mathbf{j} = -1$; $\mathbf{k}*\mathbf{k} = -1$;

$\mathbf{i}*\mathbf{j} = -\mathbf{j}*\mathbf{j} = \mathbf{k}$; $\mathbf{j}*\mathbf{k} = -\mathbf{k}*\mathbf{j} = \mathbf{i}$;

$\mathbf{k}*\mathbf{i} = -\mathbf{i}*\mathbf{k} = \mathbf{j}$; $\mathbf{i}*\mathbf{j}*\mathbf{k} = -1$.

La parte real de un Caternión, $w$, se llama parte escalar y se denota frecuentemente por $s$, la suma de los tres números complejos forma parte vectorial y se denota por **v**, con esto un Caternión suele estar escrito como $q = (s,\mathbf{v})$. Sobre el conjunto de Cuaterniones están definidas todas las operaciones aritméticas que operan sobre los números complejos. Así, si

$q1 = w1 + x1*\mathbf{i} + y1*\mathbf{j} + z1*\mathbf{k}$ y

$q2 = w1 + x2*\mathbf{i} + y2*\mathbf{j} + z2*\mathbf{k}$,

entonces la suma de $q1$ y $q2$ se define como

$q = q1 + q2 = (s,\mathbf{v})$,

donde $s = w1 + w2$,

$\mathbf{v} = (x1+x2)*\mathbf{i} + (y1+y2)*\mathbf{j} + (z1+z2)*\mathbf{k}$.

Análogamente se definen otras operaciones aritméticas. Pero a diferencia respecto a números complejos, la operación de multiplicación para Caterniones **no es conmutativa**, es decir, $q1 * q2$ no es lo mismo que $q2 * q1$ (en caso general). Se puede verificar muy fácilmente que

$q1 * q2 = (s1,\mathbf{v1}) * (s2,\mathbf{v2}) = (s1*s2 - \mathbf{v1}*\mathbf{v2}, s1*\mathbf{v2} + s2*\mathbf{v1} + \mathbf{v1} X \mathbf{v2})$,

donde la operación * para vectores representa el producto escalar, y X representa el producto vectorial.

Si $q = w + x*\mathbf{i} + y*\mathbf{j} + z*\mathbf{k}$,

entonces un Cuaternión conjugado con $q$ es

$q' = w - x*\mathbf{i} - y*\mathbf{j} - z*\mathbf{k}$,

es decir, $q' = (s, -\mathbf{v})$. La norma (longitud) de un Cuaternión $q$ se define como

$|q| = sqrt(w*w + x*x + y*y + z*z)$,

donde *sqrt()* es la operación de extracción de raíz cuadrada. Un Cuaternión con la norma igual a la unidad se llama **unitario**.

Un Cuaternión **inverso** con respecto al Caternión q, se define por la expresión
$inv(q) = q' / (q * q')$. Para un Cuaternión unitario su inverso coincide con su conjugado: $inv(q) = q'$ si $|q| = 1$. Esta propiedad juega un papel muy importante para representar rotaciones en forma más simple.

Sea **p** un punto en el espacio de 3 dimensiones: $\mathbf{p} = \mathbf{p}(x,y,z)$. Para poder rotarle en el espacio alrededor de un eje pasando por el origen del sistema de coordenadas el punto se representa en forma de un Cuaternión: $\mathbf{P} = (0,\mathbf{p})$. El eje de rotación se determina por un **Cuaternión unitario** $c = (0,\mathbf{u})$, donde el vector **u** coincide con la dirección del eje de rotación. **La operación de rotación** en un ángulo **fi** alrededor del eje definido por el Cuaternión unitario $c = (0,\mathbf{u})$ **se representa por un Cuaternión unitario**

$q = (s,\mathbf{v})$, donde

$s = \cos(\mathbf{fi}/2)$ y $\mathbf{v} = \mathbf{u}*\text{sen}(\mathbf{fi}/2)$, es decir, $q = (cos(\mathbf{fi}/2),$

$\mathbf{u}*\text{sen}(\mathbf{fi}/2))$. **El resultado de la rotación**, un punto **r**, se obtiene en forma de un Cuaternión $\mathbf{R} = (0,\mathbf{r})$ por medio de la siguiente **operación de multiplicación de tres Caterniones**:

$\mathbf{R} = q*\mathbf{P}*q'$. (1)

Sean dos Cuaterniones unitarios, $q1$ y $q2$, que representan dos rotaciones. Si primero se realiza la

rotación *q1* y después la rotación *q2*, entonces la posición final de un punto determinado por un Cuaternión **P** se obtiene por medio de las siguientes operaciones:

$q2*(q1*\mathbf{P}*q1')*q2' = (q2*q1)*\mathbf{P}*(q1'*q2') = (q2*q1)*\mathbf{P}*(q2*q1)' = q*\mathbf{P}*q'$,

es decir, la realización de dos rotaciones en serie primero *q1* y después *q2* es equivalente a una sola rotación *q* igual al producto de dos rotaciones *q2*q1*. Este resultado es muy práctico, pues se puede siempre sustituir una serie de rotaciones determinadas por Cuaterniones
*q1, q2,...,qn*, con una rotación resultante determinada por un solo Cuaternión.

Existen varios métodos de formalizar el proceso de rotación en el espacio 3D por Cuaterniones según la expresión (1). Lo siguiente representa un resultado de programar este procedimiento en el lenguaje de programación VisualC++. En el programa se construye una clase que encapsula el algoritmo de rotación. Los datos de entrada son siguientes:

- un punto a rotar,
- un vector de cualquier longitud el cual representa el eje de rotación pasado por el origen de coordenadas,
- un ángulo de rotación en grados.

Un objeto de la clase CQRot convierte los datos de entrada en Cuaterniones pertinentes, sí es necesario, normaliza el Cuaternión de rotación y por medio de la función miembro Rotar () realiza rotación, con esto las coordenadas del punto a rotar se sustituyen por las coordenadas del punto resultante. Se ve claramente que el trabajo con Cuaterniones es ocultado completamente del usuario de este programa: la interfaz de usuario consta solo de entregar al programa valores entendidos bien por el usuario: coordenadas de un punto a rotar, componentes de un vector y un valor de ángulo de rotación y en la salida recibir del programa las coordenadas del punto rotado.

```
                                        /* Qrot.c */

#include "stdio.h"
#include "math.h"
#define PI 3.1415926

class CQRot
{
        double s,v[3];
        void Norm(double* u)
        {
                double
        r=sqrt(u[0]*u[0]+u[1]*u[1]+u[2]*u[2]);
                if(r != 0.0)
                { u[0]/=r; u[1]/=r; u[2]/=r;  }
                else
                {       printf("\nVector=NULL!\n");
        getch(); exit(1);  }
        }
public:
        void Rotar(double*, double*, double);
};

void CQRot::Rotar(double r[3], double n[3], double fi)
{
        double a,b,rr[3],w=sin(fi/2);
        Norm(n);
        s=cos(fi/2);    v[0]=n[0]*w;    v[1]=n[1]*w;
v[2]=n[2]*w;
        a=s*s-v[0]*v[0]-v[1]*v[1]-v[2]*v[2];
        b=2*(v[0]*r[0]+v[1]*r[1]+v[2]*r[2]);
        rr[0]=a*r[0]+b*v[0]+2*s*(v[1]*r[2]-
v[2]*r[1]);
        rr[1]=a*r[1]+b*v[1]+2*s*(v[2]*r[0]-
v[0]*r[2]);
        rr[2]=a*r[2]+b*v[2]+2*s*(v[0]*r[1]-
v[1]*r[0]);
        r[0]=rr[0]; r[1]=rr[1]; r[2]=rr[2];
}
```

Se puede verificar el programa en el siguiente ejemplo:

```
void main(void)
{
        double T[3]={5.0, 5.0, 0.0}, W[3]={2.0, 0.0,
0.0}, fi=90;      // Punto, Vector, Angulo
        fi*=(PI/180);
        CQRot q;
        q.Rotar(T,W,fi);
        printf("\nPunto      rotado:      %f      %f
%f\n",T[0],T[1],T[2]);
}
```

El resultado es: Punto rotado: 5.000000, 0.000000, 5.000000.
Como se puede ver, el programa de rotación del punto en el espacio 3D es extremadamente simple y funciona muy rápido al comparar con procedimientos de rotación basados en el uso de ángulos de Euler, el único método ampliamente usado en la práctica ingenieril hasta hace poco. En base de la clase CQRot se puede desarrollar un conjunto de programas para simular movimientos en el espacio 3D, en particular, para investigaciones en el área de robótica. Los últimos años el uso de Caterniones para representar rotaciones en el espacio 3D se aumenta más y más especialmente para visualizar en modo gráfico la dinámica de diferentes sistemas mecánicos.