Quaternion.h
Quaternion extension for
C++ vector class library

Agner Fog

Chapter 1

Introduction

Quaternions or hypercomplex numbers is a topic in theoretical algebra and quantum physics. Applications relating to 3-D geometry and electromagnetism are better served with the vector3d package to VCL.

The file quaternion.h provides classes, operators, and functions for calculations with quaternions. This is an extension to the Vector Class Library.

The classes listed below are defined. Common operators and functions are defined for these classes:

Table 1.1: Quaternion classes

Quaternion class	**Precision**	**Quaternion elements per vector**	**Corresponding real vector class**	**Total bits**	**Recommended minimum instruction set**
Quaternion1f	single	1	Vec4f	128	SSE2
Quaternion1d	double	1	Vec4d	256	AVX

1.1 Compiling

The quaternion class extension to the Vector Class Library is compiled in the same way as the Vector Class Library itself. All x86 and x86-64 platforms are supported, including Windows, Linux, and Mac OS. The following C++ compilers can be used: Gnu, Clang, Microsoft, and Intel. See the Vector class library manual for further details.

This example shows how to use the quaternion classes:

Example 1.1.

```
// Example for quaternions
#include <stdio.h>
#include "vectorclass.h"  // vector class library
#include "quaternion.h"   // quaternion extension

// function to print quaternion
template <typename Q>
void printqx (const char * text, Q a) {
    auto aa = a.to_vector(); // get elements as real vector
    printf("\n%s ", text);   // print text
    printf("(%.3G,%.3G,%.3G,%.3G)", aa[0], aa[1], aa[2], aa[3]);
}
```

```
int main() {
    // define quaternions
    Quaternion1d a(1,2,3,4);    //  1 + 2*i + 3*j + 4*k
    Quaternion1d b(2,-3,-1,0); //  2 - 3*i - 1*j + 0*k
    Quaternion1d c = a + b;     // add quaternions
    Quaternion1d d = a * b;     // multiply quaternions

    // print results
    printqx("a = ", a);         // a = (1,2,3,4)
    printqx("b = ", b);         // b = (2,-3,-1,0)
    printqx("c = ", c);         // c = (3,-1,2,4)
    printqx("d = ", d);         // d = (11,5,-7,15)
}
```

Chapter 2

Constructing quaternions and loading data

There are several ways to create quaternions and put data into them. These methods are listed here.

Method	default constructor
Defined for	all quaternion classes
Description	the quaternion is created but not initialized. The value is unpredictable
Efficiency	good

```
// Example:
quaternion1f a;     // creates a quaternion of four floats
```

Method	Construct from single real
Defined for	all quaternion classes
Description	The parameter defines the real part. The imaginary parts are zero.
Efficiency	good

```
// Example:
quaternion1d a(3);  // a = (3,0,0,0)
```

Method	Construct from one real and three imaginary parts
Defined for	all quaternion classes
Description	The parameters define the real and imaginary parts
Efficiency	good

```
// Example:
quaternion1d a(1,2,3,4);  // a = (1,2,3,4)
```

Method	Construct from two complex numbers
Defined for	all quaternion classes
Description	The second parameter is post-multiplied by j
Efficiency	good
Implementation	complexvec1.h must be included before quaternion.h

```
// Example:
```

```
Complex1d a(1,2);     // a = 1 + i*2
Complex1d b(3,4);     // b = 3 + i*4
Quaternion1d c(a,b); // c = a + b*j = 1 + i*2 + j*3 + k*4
```

Method	member function load(p)
Defined for	all quaternion classes
Description	Load data from array of same precision. Each real part must be followed by the corresponding three imaginary parts.
Efficiency	good

```
// Example:
double a[4] = {1,2,3,4};
Quaternion1d b;
b.load(a);  // b = (1,2,3,4)
```

Method	member function store(p)
Defined for	all quaternion classes
Description	Save data into array of same precision. Each real part is followed by the corresponding three imaginary parts.
Efficiency	good

```
// Example:
float a[4];
Quaternion1f b(1,2,3,4);
b.store(a);  // a = {1,2,3,4}
```

Method	member function real()
Defined for	all quaternion classes
Description	Get real part of quaternion
Efficiency	good

```
// Example:
Quaternion1d a(1,2,3,4);
double r = a.real();  // a = 1
```

Method	member function imag()
Defined for	all quaternion classes
Description	Get imaginary parts of quaternion. The real part is set to zero
Efficiency	good

```
// Example:
Quaternion1d a(1,2,3,4);
Quaternion1d im = a.imag();  // a = (0,2,3,4)
```

Method	member function get_low()
Defined for	all quaternion classes
Description	Get the real and the first imaginary part (i) as a complex vector
Efficiency	good
Implementation	complexvec1.h must be included before quaternion.h

```
// Example:
Quaternion1d a(1,2,3,4);
Complex1d b = a.get_low();  // b = (1,2)
```

Method	member function get_high()
Defined for	all quaternion classes
Description	Get the last two imaginary parts (j and k) as a complex vector
Efficiency	good
Implementation	complexvec1.h must be included before quaternion.h

```
// Example:
Quaternion1d a(1,2,3,4);
Complex1d b = a.get_low();  // b = (1,2)
Complex1d c = a.get_high(); // c = (3,4)
Quaternion1d d(b,c);        // d = (1,2,3,4)
```

Chapter 3

Operators

Operator	+
Defined for	all quaternion classes
Description	Add two quaternions, or one quaternion and one real scalar of the same precision
Efficiency	good

```
// Example:
Quaternion1d a(1,2,3,4);
Quaternion1d b(5,6,7,8);
Quaternion1d c = a + b;     // c = (6,8,10,12)
Quaternion1d d = a + 10.0; // d = (11,2,3,4)
```

Operator	-
Defined for	all quaternion classes
Description	Subtract two quaternions, or one quaternion and one real scalar of the same precision
Efficiency	good

```
// Example:
Quaternion1d a(12,11,10,9);
Quaternion1d b(5,6,7,8);
Quaternion1d c = a - b;     // c = (7,5,3,1)
Quaternion1d d = a - 10.0; // d = (2,11,10,9)
```

Operator	*
Defined for	all quaternion classes
Description	Multiply two quaternions, or one quaternion and one real scalar of the same precision. Multiplication of quaternions is not commutative, i.e. a*b and b*a are not the same.
Efficiency	medium
Accuracy	Quaternion multiplication involves the calculation of sums of products. Loss of precision may occur if the result is close to zero.

```
// Example:
Quaternion1d a(1,2,3,4);
Quaternion1d b(5,6,7,8);
```

```
Quaternion1d c = a * b;     // c = (-60,12,30,24)
Quaternion1d d = b * a;     // d = (-60,20,14,32)
Quaternion1d e = a * 10.;   // e = (10,20,30,40)
```

Operator	/
Defined for	all quaternion classes
Description	Divide two quaternions, or one quaternion and one real scalar of the same precision. Division is defined as a / b = a * reciprocal(b)
Efficiency	medium
Accuracy	Quaternion division involves the calculation of sums of products. Loss of precision may occur if the result is close to zero.

```
// Example:
Quaternion1f a(7,9,-1,7);
Quaternion1f b(1,2,3,2);
Quaternion1f c = a / b;     // c = (2,1,-1,-2)
Quaternion1f d = c * b;     // d = (7,9,-1,7)
Quaternion1f e = b / 2.0f;  // e = (0.5,1,1.5,1)
Quaternion1f f = 18.f / b;  // f = (1,-2,-3,-2)
Quaternion1f g = f * b;     // g = (18,0,0,0)
```

Operator	∼
Defined for	all quaternion classes
Description	Complex conjugate. The signs of the imaginary parts are inverted
Efficiency	good

```
// Example:
Quaternion1f a(1,2,3,4);
Quaternion1f b = ~ a;     // b = (1,-2,-3,-4)
```

Operator	==
Defined for	all quaternion classes
Description	Compare for equality. The result is a boolean scalar.
Efficiency	good

```
// Example:
Quaternion1f a(1, 2,3,4);
Quaternion1f b(1,-2,3,4);
bool         c = (a == b);  // c = false
```

Operator	!=
Defined for	all quaternion classes
Description	Compare for not equal. The result is a boolean scalar.
Efficiency	good

```
// Example:
Quaternion1f a(1, 2,3,4);
Quaternion1f b(1,-2,3,4);
bool         c = (a != b);  // c = true
```

Chapter 4

Mathematical functions

Function	abs
Defined for	all quaternion classes
Description	Gives the norm as a scalar
Efficiency	medium

```
// Example:
Quaternion1f a(2,1,0,2);
double       b = abs(a); // b = 3
```

Chapter 5

Other functions

Function	to_vector
Defined for	all quaternion classes
Description	Convert to a vector of the real part and the three imaginary parts.
Efficiency	good

```
// Example:
Quaternion1d a(1,2,3,4);
Vec4d        b = a.to_vector(); // b = (1,2,3,4)
```

Function	select
Defined for	all quaternion classes
Description	Choose between two quaternions.
Efficiency	good

```
// Example:
Quaternion1d a(1,2,3,4);
Quaternion1d b(5,6,7,8);
Quaternion1d c = select(true,a,b);  // c = (1,2,3,4)
Quaternion1d d = select(false,a,b); // d = (5,6,7,8)
```