Algorithms for manipulating quaternions in floating-point arithmetic

Mioara Joldeş, Jean-Michel Muller

Algorithms for Manipulating Quaternions in Floating-Point Arithmetic

Mioara Joldeş
CNRS, LAAS
Toulouse, France
joldes@laas.fr

Jean-Michel Muller
CNRS, LIP, Université de Lyon
Lyon, France
jean-michel.muller@ens-lyon.fr

***Abstract*—Quaternions form a set of four global but not unique parameters, which can represent three-dimensional rotations in a non-singular way. They are frequently used in computer graphics, drone and aerospace vehicle control. Floating-point quaternion operations (addition, multiplication, reciprocal, norm) are often implemented "by the book". Although all usual implementations are algebraically equivalent, their numerical behavior can be quite different. For instance, the arithmetic operations on quaternions as well as conversion algorithms to/from rotation matrices are subject to spurious under/overflow (an intermediate calculation underflows or overflows, making the computed final result irrelevant, although the exact result is in the domain of the representable numbers). The goal of this paper is to analyze and then propose workarounds and better accuracy alternatives for such algorithms.**

***Index Terms*—Floating-point arithmetic, quaternions, rounding error analysis.**

INTRODUCTION

The quaternions were invented by W.R. Hamilton in 1853 [9], [14], in his attempt to extend the algebra of complex numbers in a three dimensional space. They are "numbers" of the form $q = q_0 + q_1 i + q_2 j + q_3 k$, where q_0, q_1, q_2 and q_3 are real numbers called the *components* of q, and i, j and k are symbols that follow the non-commutative multiplication rules given in Table I. The number q_0 is the *scalar part* of q, while $q_1 i + q_2 j + q_3 k$ is its *vector part*.

TABLE I
THE QUATERNION MULTIPLICATION TABLE

$\times$	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

Almost at the same time and motivated by the geometrical solution of the Euler problem of the product of two rotations, O. Rodrigues provided a set of four global (not affected by singularities), non minimal and non unique parameters representing a rotation and known as the Euler-Rodrigues parameters [20]. If these four parameters are substituted for the q_i components of the quaternion, then the multiplication rules of Table I are exactly the ones of the multiplication of two rotations. Indeed, the components of a normalized quaternion are defined by their relations to the Euler axis-angle representation of a rotation (the scalar part is directly related to the main angle of the rotation, while the vector part is proportional to the axis rotation vector). Relying on the powerful algebra defined by Hamilton, this parametrization of the rotation group $SO(3)$ is particularly appealing from a computational point of view, compared to the main alternatives such as Euler angles, Euler axis-angle representation or Gibbs vector. In particular, quaternions represent 3D-rotations in a more compact (4 real numbers instead of 9) and numerically efficient manner than rotation matrices [14]. Due to this property, quaternions are frequently used in computer graphics [24], computer vision and robotics [4], and drone and aerospace vehicle control [25]. They have also been used for processing bivariate signals [5], and several applications to modern physics are related in [8].

Let us now examine the arithmetic operations that can be performed on quaternions.

A quaternion can be multiplied (resp. divided) by a real number λ: this is done by multiplying (resp. dividing) its components by λ.

The *sum* of $q = q_0 + q_1 i + q_2 j + q_3 k$ and $r = r_0 + r_1 i + r_2 j + r_3 k$ is

$$q + r = (q_0 + r_0) + (q_1 + r_1)\cdot i + (q_2 + r_2)\cdot j + (q_3 + r_3)\cdot j, \quad (1)$$

and the *product* of $q \cdot r$ of q and r is $\pi_0 + \pi_1 i + \pi_2 j + \pi_3 k$, with

$$\begin{cases} \pi_0 &= q_0 r_0 - q_1 r_1 - q_2 r_2 - q_3 r_3, \\ \pi_1 &= q_0 r_1 + q_1 r_0 + q_2 r_3 - q_3 r_2, \\ \pi_2 &= q_0 r_2 - q_1 r_3 + q_2 r_0 + q_3 r_1, \\ \pi_3 &= q_0 r_3 + q_1 r_2 - q_2 r_1 + q_3 r_0. \end{cases} \quad (2)$$

Note that (2) is easily deduced from Table I and beware that quaternion multiplication is *not* a commutative operation: in general, $q \cdot r \neq r \cdot q$.

The norm of $q = q_0 + q_1 i + q_2 j + q_3 k$ is

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}. \quad (3)$$

The *conjugate* of the quaternion $q = q_0 + q_1 i + q_2 j + q_3 k$ is defined as $\overline{q} = q_0 - q_1 i - q_2 j - q_3 k$. It satisfies $q\overline{q} = |q|^2$.

This allows one to define the *reciprocal* of q as

$$q^{-1} = \frac{\overline{q}}{|q|^2}. \quad (4)$$

Since multiplication is not commutative, there is no unambiguous notion of division, and notation q/r (unless q is real) should be avoided, since it would not be clear whether it would mean $q \cdot r^{-1}$ or $r^{-1} \cdot q$.

While it is *commonly known* that arithmetic operations on quaternions are numerically stable, few articles recently tackled their error analysis [22], [23], focusing mainly on overflows/underflows and statistical tests.

The goal of this paper is to give and analyze algorithms for manipulating quaternions in floating-point arithmetic.

The setting is the following: we assume an underlying radix-2, precision-p floating-point arithmetic, with subnormal numbers available, and correctly-rounded (to nearest) floating-point operations, similar to default mode of the binary arithmetic specified by the IEEE 754 Standard [1]. We also define $e_{\min}$ and $e_{\max}$ as the extremal floating-point exponents, so that the largest finite floating-point number is

$$\Omega = 2^{e_{\max}+1} - 2^{e_{\max}-p+1},$$

the smallest positive nonzero number is

$$\alpha = 2^{e_{\min}-p+1},$$

and the smallest positive normal number is $2^{e_{\min}}$.

We denote $u = 2^{-p}$ the *rounding unit*, $v = u/(1+u)$, and RN the round-to-nearest, ties-to-even function. We remind the reader (see for instance [16]) that for any t between $2^{e_{\min}}$ and Ω, we have

$$|\mathrm{RN}(t) - t| \leq v \cdot |t| = \left(\frac{u}{1+u}\right) \cdot |t| < u \cdot |t|.$$

We are interested in obtaining bounds on the componentwise and normwise relative errors of the arithmetic operations on quaternions. If $\hat{q} = \hat{q}_0 + \hat{q}_1 i + \hat{q}_2 j + \hat{q}_3 k$ approximates $q = q_0 + q_1 i + q_2 j + q_3 k$, then the componentwise relative error of that approximation is

$$\max_{n=0,\ldots,3} \left| \frac{\hat{q}_n - q_n}{q_n} \right|,$$

(with the convention that if $q_n = 0$ then $|(\hat{q}_n - q_n)/q_n|$ is replaced by 0 if $\hat{q}_n = 0$, and by $+\infty$ otherwise), and the normwise relative error is

$$\left| \frac{\hat{q} - q}{q} \right|,$$

(with the convention that if $q = 0$ then the normwise error is 0 if $\hat{q} = 0$, and $+\infty$ otherwise).

In the following, we will also use other norms than the "standard" one (3): the "infinite norm" defined as

$$\|q\|_\infty = \max\{|q_0|, |q_1|, |q_2|, |q_3|\},$$

and the "1-norm" defined as

$$\|q\|_1 = |q_0| + |q_1| + |q_2| + |q_3|.$$

These norms satisfy:

$$\left\{ \begin{array}{lcccl} \|q\|_\infty & \leq & |q| & \leq & 2 \cdot \|q\|_\infty, \\ \|q\|_\infty & \leq & \|q\|_1 & \leq & 4 \cdot \|q\|_\infty, \\ |q| & \leq & \|q\|_1 & \leq & 2 \cdot |q|. \end{array} \right. \tag{5}$$

Adding quaternions is straightforward and very accurately done using (1). Straightforward use of (2), (3) or (4) for multiplying two quaternions, or computing the norm or the reciprocal of a quaternion is more problematic because the intermediate operations that appear in these formulas can easily lead to *spurious overflows or underflows*, i.e., an intermediate calculation underflows or overflows whereas the exact final result (or its components if it is a quaternion) is well within the domain of normal representable numbers. In such a case, instead of a reasonably good approximation to that exact result, we can obtain NaNs, infinities, or very inaccurate results. Some libraries [2], [3], [7] essentially implement formulas (2), (3) or (4). This is not necessarily a problem if the input operands are known to lie in a domain in which overflow and underflow are impossible or harmless: for example, a quaternion associated to a 3D rotation has norm 1. For building a "general" library, however, this issue must be addressed (furthermore, some papers mention applications that use nonunit quaternions [8], [21]). We will do this by using *scaling* techniques similar to the ones that are sometimes used in complex arithmetic [6], [11], [19].

I. Scaling a quaternion

Let $q = q_0 + q_1 i + q_2 j + q_3 k$ be a quaternion, where q_0, q_1, q_2, and q_3 are binary floating-point numbers. The goal is to compute a (real) *scaling factor* F such that

- F is a power of 2 (so that a multiplication by F is errorless);
- $\|q/F\|_\infty$ is not far from, and below, 1 (typically it will be between $1/16$ and 1).

We can use two functions specified by the IEEE 754 Standard [1, p. 32]:

- **scaleB**(x, k), which returns (in a binary format, which is the case considered in this paper) $x \cdot 2^k$ (where x is a FP number and k is an integer). In the C language, that function is called `scalbn`;
- **logB**(x), which returns (in a binary format) $\lfloor \log_2 |x| \rfloor$ (where x is a FP number). In the C language, that function is called `logb`.

A natural solution is to choose, as a scaling factor, the power of 2 immediately larger than the largest of $|q_0|$, $|q_1|$, $|q_2|$, and $|q_3|$, i.e.,

$$F_\infty(q) = 2^{\lfloor \log_2 \|q\|_\infty \rfloor + 1}.$$

However, on many recent architectures, $\|q\|_1 = |q_0| + |q_1| + |q_2| + |q_3|$ will be computed more quickly than $\|q\|_\infty = \max\{|q_0|, |q_1|, |q_2|, |q_3|\}$. Hence it may be preferable to use

$$F_1(q) = 2^{\lfloor \log_2 \|q\|_1 \rfloor + 1}.$$

The definition of F_∞ implies that

$$\frac{1}{2} \leq \max_{i=1,\ldots,4} \frac{|q_i|}{F_\infty(q)} < 1, \tag{6}$$

and, using (5), we obtain

$$\frac{1}{8} \leq \max_{i=1,\ldots,4} \frac{|q_i|}{F_1(q)} < 1. \tag{7}$$

II. Computing the norm of a quaternion

Let $q = q_0 + q_1 i + q_2 j + q_3 k$ be a quaternion, where q_0, q_1, q_2, and q_3 are binary floating-point numbers. Algorithm 1 presents the classical method for computing its norm defined by Eq. (3).

ALGORITHM 1: Naive algorithm for computing the norm of $q = q_0 + q_1 i + q_2 j + q_3 k$.

1: $\hat{s_0} \leftarrow \mathrm{RN}(q_0^2)$
2: $\hat{s_1} \leftarrow \mathrm{RN}(q_1^2)$
3: $\hat{s_2} \leftarrow \mathrm{RN}(q_2^2)$
4: $\hat{s_3} \leftarrow \mathrm{RN}(q_3^2)$
5: $\hat{\sigma_0} \leftarrow \mathrm{RN}(\hat{s_0} + \hat{s_1})$
6: $\hat{\sigma_1} \leftarrow \mathrm{RN}(\hat{s_2} + \hat{s_3})$
7: $\hat{\sigma} \leftarrow \mathrm{RN}(\hat{\sigma_0} + \hat{\sigma_1})$
8: $\hat{N} \leftarrow \mathrm{RN}(\sqrt{\hat{\sigma}})$
9: **return** $\hat{N}$

Unless we have some preliminary information on q (e.g., we know in advance lower and upper bounds on the values $|q_i|$) that allows us to be sure that the terms q_i^2 will not be too large or too tiny, Algorithm 1 should not be used without a preliminary test and/or a preliminary scaling of the terms q_is, because:

– spurious overflow may occur, leading to a returned result equal to $+\infty$ even in cases where the exact result is far below the overflow threshold. Just consider, in binary32 arithmetic, the case $q_0 = 2^{65}$, $q_1 = q_2 = q_3 = 0$, for which $|q| = 2^{65}$ and $\hat{N} = +\infty$;

– spurious underflow may occur, leading to poor accuracy (this is an issue only when *all* $|q_i|$s are small: when this is not the case, the underflowing terms will be negligible in front of the largest one, so that a large relative error on these terms will not undermine the computation). An example of a poor result due to spurious underflow in binary32 arithmetic is $q_0 = (3/2) \times 2^{-75}$ and $q_1 = q_2 = q_3 = 0$, for which $|q| = q_0 \approx 3.97 \times 10^{-23}$, and the computed value is $11863283/2^{98} \approx 3.74 \times 10^{-23}$.

For now, let us assume that no underflow or overflow occurs, and let us bound the error of Algorithm 1. By convention, we associate to an exact mathematical value say s, the corresponding computed value (after rounding) written with a "hat", say $\hat{s}$. Without any difficulty, we obtain

$$\forall i, s_i(1-v) \le \hat{s_i} \le s_i(1+v),$$

hence

$$\forall i, \sigma_i(1-v)^2 \le \hat{\sigma_i} \le \sigma_i(1+v)^2,$$

and

$$\sigma(1-v)^3 \le \hat{\sigma} \le \sigma(1+v)^3.$$

This gives

$$\sqrt{\sigma}(1-v)^{3/2} \le \sqrt{\hat{\sigma}} \le \sqrt{\sigma}(1+v)^{3/2},$$

and

$$N(1-v)^{5/2} \le \hat{N} = \mathrm{RN}(\sqrt{\hat{\sigma}}) \le N(1+v)^{5/2}.$$

Therefore when no underflow or overflow occurs, the relative error of Algorithm 1 is bounded by $(1+v)^{5/2} - 1$, which is less than $(5/2)u$.

Now, assume that we "scale" the input values, i.e., we divide q_0, q_1, q_2 and q_3 by $F = F_1(q)$ or $F_\infty(q)$ (whichever is the fastest to compute on the system being used), to get new input values q'_0, q'_1, q'_2 and q'_3. Note that in practice we do not actually perform a division by F, but rather use functions **scaleB** and **logB** (unless they are poorly implemented). That is, we compute

$$c = \mathbf{logB}(\|q\|_1) + 1 \text{ or } \mathbf{logB}(|q|) + 1,$$

and

$$q'_n = \mathbf{scaleB}(q_n, -c).$$

From (6) and (7), we find

$$\frac{1}{8} \le \max\{|q'_0|, |q'_1|, |q'_2|, |q'_3|\} \le 1. \tag{8}$$

Then we apply Algorithm 1 to the scaled inputs, and perform a final multiplication of the obtained result by F. Due to the scaling, spurious overflow can no longer happen. If no underflow occurs, then, since multiplying or dividing by F is errorless, the previously computed error bound still applies.

If an underflow occurs (either in one of the divisions by F or later on), one can show using (8) that the error on the corresponding terms has very little influence on the sum $\hat{\sigma}$, so that the error bound remains valid.

III. Computing the product of two quaternions

A. The naive multiplication algorithm

The "naive" way of implementing quaternion multiplication consists in directly translating (2), i.e., in computing

$$\begin{cases} \hat{\pi}_0 &= \mathrm{RN}\Big(\mathrm{RN}\big(\mathrm{RN}(q_0 r_0) - \mathrm{RN}(q_1 r_1)\big) \\ &\quad -\mathrm{RN}\big(\mathrm{RN}(q_2 r_2) + \mathrm{RN}(q_3 r_3)\big)\Big) \\ \hat{\pi}_1 &= \mathrm{RN}\Big(\mathrm{RN}\big(\mathrm{RN}(q_0 r_1) + \mathrm{RN}(q_1 r_0)\big) \\ &\quad +\mathrm{RN}\big(\mathrm{RN}(q_2 r_3) - \mathrm{RN}(q_3 r_2)\big)\Big) \\ \hat{\pi}_2 &= \mathrm{RN}\Big(\mathrm{RN}\big(\mathrm{RN}(q_0 r_2) - \mathrm{RN}(q_1 r_3)\big) \\ &\quad +\mathrm{RN}\big(\mathrm{RN}(q_2 r_0) + \mathrm{RN}(q_3 r_1)\big)\Big) \\ \hat{\pi}_3 &= \mathrm{RN}\Big(\mathrm{RN}\big(\mathrm{RN}(q_0 r_3) + \mathrm{RN}(q_1 r_2)\big) \\ &\quad -\mathrm{RN}\big(\mathrm{RN}(q_2 r_1) - \mathrm{RN}(q_3 r_0)\big)\Big) \end{cases} \tag{9}$$

Similarly to what we did for the calculation of the norm of a quaternion, let us first analyze the error committed by using (9) when no underflow or overflow occurs. Then, we will see how scaling the inputs can allow one to avoid spurious overflows.

1) Accuracy of the naive multiplication algorithm: Consider the calculation of $\hat{\pi}_0$ (i.e., Line 1 of (9)) — the reasoning with the calculation of $\hat{\pi}_1$, $\hat{\pi}_2$, and $\hat{\pi}_3$ is similar. We have $\mathrm{RN}(q_0 r_0) = q_0 r_0 \cdot (1+\epsilon)$, with $|\epsilon| \le v$, and a similar relation holds for the other products $q_1 r_1$, $q_2 r_2$, and $q_3 r_3$. Therefore,

$$\mathrm{RN}(q_0 r_0) - \mathrm{RN}(q_1 r_1) = q_0 r_0 - q_1 r_1 + (|q_0 r_0| + |q_1 r_1|) \cdot \epsilon_1$$

with $|\epsilon_1| \leq v$, therefore,

$$\begin{array}{c}\mathrm{RN}\big(\mathrm{RN}(q_0r_0) - \mathrm{RN}(q_1r_1)\big)\\ = q_0r_0 - q_1r_1 + (q_0r_0 - q_1r_1)\cdot\epsilon_2\\ +(|q_0r_0| + |q_1r_1|)\cdot\epsilon_1\cdot(1+\epsilon_2),\end{array}$$

with $|\epsilon_2| \leq v$. Similarly, for $q_2r_2 + q_3r_3$, we obtain

$$\begin{array}{c}\mathrm{RN}\big(\mathrm{RN}(q_2r_2) + \mathrm{RN}(q_3r_3)\big)\\ = q_2r_2 + q_3r_3 + (q_2r_2 + q_3r_3)\cdot\hat{\epsilon}_2\\ +(|q_2r_2| + |q_3r_3|)\cdot\hat{\epsilon}_1\cdot(1+\hat{\epsilon}_2),\end{array}$$

with $|\hat{\epsilon}_1|,\ |\hat{\epsilon}_2| \leq v$.

The number $(q_0r_0 - q_1r_1)\cdot\epsilon_2 + (q_2r_2 + q_3r_3)\cdot\hat{\epsilon}_2$ has an absolute value less than or equal to

$$|q_0r_0 - q_1r_1| + |q_2r_2 + q_3r_3|.$$

The number

$$(|q_0r_0| + |q_1r_1|)\cdot\epsilon_1\cdot(1+\epsilon_2) + (|q_2r_2| + |q_3r_3|)\cdot\hat{\epsilon}_1\cdot(1+\hat{\epsilon}_2)$$

has an absolute value less than or equal to

$$(|q_0r_0| + |q_1r_1| + |q_2r_2| + |q_3r_3|)\cdot v\cdot(1+v).$$

We therefore obtain

$$\begin{array}{c}\mathrm{RN}\big(\mathrm{RN}(q_0r_0) - \mathrm{RN}(q_1r_1)\big) - \mathrm{RN}\big(\mathrm{RN}(q_2r_2) + \mathrm{RN}(q_3r_3)\big)\\ = q_0r_0 - q_1r_1 - q_2r_2 - q_3r_3\\ +\big(|q_0r_0 - q_1r_1| + |q_2r_2 + q_3r_3|\big)\cdot\epsilon_2'\\ +\big(|q_0r_0| + |q_1r_1| + |q_2r_2| + |q_3r_3|\big)\cdot\xi,\end{array}$$

with $|\epsilon_2'| \leq v$ and $|\xi| \leq v + v^2$, therefore

$$\begin{array}{rcl}\hat{\pi}_0 &=& \mathrm{RN}\Big(\mathrm{RN}\big(\mathrm{RN}(q_0r_0) - \mathrm{RN}(q_1r_1)\big)\\ && -\mathrm{RN}\big(\mathrm{RN}(q_2r_2) + \mathrm{RN}(q_3r_3)\big)\Big)\\ &=& q_0r_0 - q_1r_1 - q_2r_2 - q_3r_3\\ && +(q_0r_0 - q_1r_1 - q_2r_2 - q_3r_3)\cdot\epsilon_3\\ && +\big(|q_0r_0 - q_1r_1| + |q_2r_2 + q_3r_3|\big)\cdot\epsilon_2'\cdot(1+\epsilon_3)\\ && +\big(|q_0r_0| + |q_1r_1| + |q_2r_2| + |q_3r_3|\big)\cdot\xi\cdot(1+\epsilon_3),\end{array}$$

with $|\epsilon_3| \leq v$. This gives,

$$\hat{\pi}_0 = \pi_0 + \pi_0\epsilon_3 + \big(|q_0r_0| + |q_1r_1| + |q_2r_2| + |q_3r_3|\big)\epsilon_4, \quad (10)$$

with $|\epsilon_4| \leq 2v + 3v^2 + v^3$. There is a similar equation (deduced through symmetries) for π_1, π_2, and π_3. We therefore obtain

Lemma 1 (Componentwise absolute error of the "naive" quaternion multiplication algorithm)**.** *When no underflow or overflow occurs, the values $\hat{\pi}_0$, $\hat{\pi}_1$, $\hat{\pi}_2$, and $\hat{\pi}_3$ computed as indicated in (9) satisfy*

$$\begin{array}{rcl}|\pi_n - \hat{\pi}_n| &\leq& u\cdot|\pi_n| + \left(\frac{2u}{1+u} + \frac{3u^2}{(1+u)^2} + \frac{u^3}{(1+u)^3}\right)\cdot M_n\\ &\leq& u\cdot|\pi_n| + \big(2u + u^2\big)\cdot M_n, \quad n = 0,1,2,3\end{array}$$

with

$$\left\{\begin{array}{rcl}M_0 &=& |q_0r_0| + |q_1r_1| + |q_2r_2| + |q_3r_3|\\ M_1 &=& |q_0r_1| + |q_1r_0| + |q_2r_3| + |q_3r_2|\\ M_2 &=& |q_0r_2| + |q_1r_3| + |q_2r_0| + |q_3r_1|\\ M_3 &=& |q_0r_3| + |q_1r_2| + |q_2r_1| + |q_3r_0|.\end{array}\right.$$

Lemma 1 does not allow one to bound the *componentwise relative* error of the quaternion product, because $|\pi_n|/M_n$ can be very large. An example (also given in [10]) is $q_0 = 2^p - 2$, $q_1 = 2^p - 1$, $q_2 = q_3 = 0$, $r_0 = 2^p$, $r_1 = 2^p - 1$, $r_2 = r_3 = 0$, for which $\pi_0 = -1$ and $\hat{\pi}_0 = 0$: the relative error on π_0 is equal to 1. Now, let us try to bound the *normwise* relative error of the quaternion product. From (10) and the similar equations for π_1, π_2, and π_3, one obtains

$$\frac{|\pi - \hat{\pi}|^2}{|\pi|^2} \leq \frac{\sum_{n=0}^{3}(|\pi_n|\cdot v + M_n w)^2}{\pi_0^2 + \pi_1^2 + \pi_2^2 + \pi_3^2}, \quad (11)$$

with $w = 2v + 3v^2 + v^3$. From (11), we obtain

$$\begin{array}{rcl}&&\frac{|\pi - \hat{\pi}|^2}{|\pi|^2}\\ &\leq& v^2 + \frac{2vw\cdot\sum_{n=0}^{3}M_n\cdot|\pi_n| + w^2\cdot\sum_{n=0}^{3}M_n^2}{\pi_0^2 + \pi_1^2 + \pi_2^2 + \pi_3^2}\\ &\leq& v^2 + \frac{(2vw + w^2)(M_0^2 + M_1^2 + M_2^2 + M_3^2)}{\pi_0^2 + \pi_1^2 + \pi_2^2 + \pi_3^2}\\ &\leq& v^2 + \frac{(2vw + w^2)(M_0^2 + M_1^2 + M_2^2 + M_3^2)}{|q|^2\cdot|r|^2}\end{array} \quad (12)$$

(by noting that $\pi_0^2 + \pi_1^2 + \pi_2^2 + \pi_3^2 = |q\cdot r|^2 = |q|^2\cdot|r|^2$). In (12), the term $M_0^2/(|q|^2\cdot|r|^2)$ is of the form

$$\frac{\langle a|b\rangle^2}{|a|^2\cdot|b|^2},$$

where a (resp. b) is made up with the absolute values of the components of q (resp. r). The term $M_1^2/(|q|^2\cdot|r|^2)$ is of the same form with the same a and $b = (|r_1|, |r_0|, |r_3|, |r_2|)$, the term $M_2^2/(|q|^2\cdot|r|^2)$ is of the same form with $b = (|r_2|, |r_3|, |r_0|, |r_1|)$, and the term $M_3^2/(|q|^2\cdot|r|^2)$ is of the same form with $b = (|r_3|, |r_2|, |r_1|, |r_0|)$. The Cauchy-Schwarz inequality implies that all these terms are less than or equal to 1. Therefore, we obtain

$$\begin{array}{rcl}\frac{|\pi - \hat{\pi}|^2}{|\pi|^2} &\leq& v^2 + 4\cdot(2v + w^2)\\ &\leq& 33v^2 + 72v^3 + 60v^4 + 24v^5 + 4v^6.\end{array} \quad (13)$$

Hence, the normwise relative error of the quaternion product implemented as indicated in (9) is bounded by

$$\sqrt{33v^2 + 72v^3 + 60v^4 + 24v^5 + 4v^6},$$

which is less than $\sqrt{33}\cdot u + u^2 \approx 5.75u + u^2$.

If an FMA instruction is available, one can replace (9) by

$$\left\{\begin{array}{rcl}\hat{\pi}_0 &=& \mathrm{RN}\Big(\mathrm{RN}\big(q_0r_0 - \mathrm{RN}(q_1r_1)\big) \\ && -\mathrm{RN}\big(q_2r_2 + \mathrm{RN}(q_3r_3)\big)\Big) \\ \hat{\pi}_1 &=& \mathrm{RN}\Big(\mathrm{RN}\big(q_0r_1 + \mathrm{RN}(q_1r_0)\big) \\ && +\mathrm{RN}\big(q_2r_3 - \mathrm{RN}(q_3r_2)\big)\Big) \\ \hat{\pi}_2 &=& \mathrm{RN}\Big(\mathrm{RN}\big(q_0r_2 - \mathrm{RN}(q_1r_3)\big) \\ && +\mathrm{RN}\big(q_2r_0 + \mathrm{RN}(q_3r_1)\big)\Big) \\ \hat{\pi}_3 &=& \mathrm{RN}\Big(\mathrm{RN}\big(q_0r_3 + \mathrm{RN}(q_1r_2)\big) \\ && -\mathrm{RN}\big(q_2r_1 - \mathrm{RN}(q_3r_0)\big)\Big)\end{array}\right. \tag{14}$$

By doing this, the calculations will be faster, and frequently more accurate. However, the error bound will remain unchanged.

2) Scaling the naive multiplication algorithm to avoid spurious overflow: Still consider the product of $q = q_0 + q_1 i + q_2 j + q_3 k$ and $r = r_0 + r_1 i + r_2 j + r_3 k$. Let us scale q and r by $F_1(q)$ and $F_1(r)$, respectively, i.e., we compute "scaled quaternions" q' and r' defined by $q'_i = q_i/F_1(q)$ and $r'_i = r_i/F_1(r)$. We then apply the naive algorithm to the scaled operands in order to compute $q' \cdot r'$. We finally obtain the desired product $q \cdot r$ by multiplying each component of $q' \cdot r'$ by the product of the scaled factors. Again, the scaling as well as the final multiplication can be implemented using functions **logB** and **scaleB**.The scaling ensures

$$\frac{1}{2} \leq \|q'\|_1; \|r'\|_1 < 1.$$

Therefore, using (5),

$$\frac{1}{4} \leq |q'|; |r'| < 1, \tag{15}$$

and

$$\frac{1}{8} \leq \|q'\|_\infty; \|r'\|_\infty < 1. \tag{16}$$

Eq. (16) implies that none of the intermediate or final results in (9) can overflow. Underflow is more difficult to handle. Eq (15) implies that $1/16 \leq |q' \cdot r'| < 1$, from which we deduce, using (5) again,

$$\frac{1}{32} \leq \|q' \cdot r'\|_\infty < 1.$$

Which implies that, even if underflow can occur during the intermediate calculations cannot have a significant impact on the *normwise* error. It can, however, significantly worsen the *componentwise* error.

B. A more accurate algorithm

Since the calculations that appear in (2) are dot products, one can use accurate dot-product algorithms published in the literature [18] to perform them. These algorithms use, as basic building blocks, Algorithms 2 and 3 (for proofs and explanations, see for instance [16]).

Using Alg. 5.3 in [18], π_0 is computed as shown in Algorithm 4 (the calculation of π_1, π_2, and π_3 is similar). Using Proposition 5.5 of [18], we obtain

ALGORITHM 2: 2Sum(x, y). The 2Sum algorithm [13], [15]. The returned results satisfy $s + t = x + y$.

$s \leftarrow \mathrm{RN}(x + y)$
$x' \leftarrow \mathrm{RN}(s - y)$
$y' \leftarrow \mathrm{RN}(s - x')$
$\delta_x \leftarrow \mathrm{RN}(x - x')$
$\delta_y \leftarrow \mathrm{RN}(y - y')$
$t \leftarrow \mathrm{RN}(\delta_x + \delta_y)$
return (s, t)

ALGORITHM 3: Fast2Mult(x, y)**.** The Fast2Mult algorithm (see for instance [12], [16], [17]). It requires the availability of a fused multiply-add (FMA) instruction for computing $\mathrm{RN}(xy - w)$. The returned results satisfy $w + e = xy$.

$w \leftarrow \mathrm{RN}(xy)$
$e \leftarrow \mathrm{RN}(xy - w)$
return (w, e)

Lemma 2 (Componentwise error of the quaternion multiplication algorithm derived from Ogita, Rump and Oishi's dot-product algorithm [18])**.** *When no underflow or overflow occurs, the values $\hat{\pi}_0$, $\hat{\pi}_1$, $\hat{\pi}_2$, and $\hat{\pi}_3$ computed as indicated in Algorithm 4 satisfy*

$$|\pi_n - \hat{\pi}_n| \leq u \cdot |\pi_n| + \frac{1}{2}\left(\frac{4u}{1-4u}\right)^2 \cdot M_n.$$

When $M_n/|\pi_n|$ is large, this is a much better bound than the one given by Lemma 1. Let us now give a bound on the normwise relative error. This will be very similar to what we did in Section III-A1, so we refer to that section for the

ALGORITHM 4: Calculation of π_0 using Ogita, Rump and Oishi's algorithm (Alg. 5.3 in [18]).

1: $(s_0, e_0) \leftarrow \mathrm{Fast2Mult}(q_0, r_0)$
2: $(s_1, e_1) \leftarrow \mathrm{Fast2Mult}(-q_1, r_1)$
3: $(s_2, e_2) \leftarrow \mathrm{Fast2Mult}(-q_2, r_2)$
4: $(s_3, e_3) \leftarrow \mathrm{Fast2Mult}(-q_3, r_3)$
5: $\sigma \leftarrow e_0$
6: $S \leftarrow s_0$
7: **for** $i = 1$ **to** 3 **do**
8: $\quad (S, \rho) \leftarrow \mathrm{2Sum}(S, s_i)$
9: $\quad \sigma \leftarrow \mathrm{RN}(\sigma + \mathrm{RN}(\rho + e_i))$
10: **end for**
11: $\hat{\pi}_0 \leftarrow \mathrm{RN}(S + \sigma)$
12: **return** $\hat{\pi}_0$

detailed explanations. We have,

$$\begin{aligned}
&\frac{|\pi-\hat{\pi}|^2}{|\pi|^2}\\
&\leq \frac{\sum_{n=0}^{3}\left(\pi_n u+\frac{1}{2}\left(\frac{4u}{1-4u}\right)^2\cdot M_n\right)^2}{\pi_0^2+\pi_1^2+\pi_2^2+\pi_3^2}\\
&\leq u^2+\frac{\left(u\cdot\left(\frac{4u}{1-4u}\right)^2+\frac{1}{4}\cdot\left(\frac{4u}{1-4u}\right)^4\right)\sum_{n=0}^{3}M_n^2}{|q|^2\cdot|r|^2}\\
&\leq u^2+4u\cdot\left(\frac{4u}{1-4u}\right)^2+\left(\frac{4u}{1-4u}\right)^4.
\end{aligned} \tag{17}$$

The obtained quantity is less than $(u+32u^2)^2$ as soon as $p\geq 4$ (i.e., as soon as $u\leq 1/16$), which always holds in practice. Therefore, we obtain,

Lemma 3 (Nomwise error of the quaternion multiplication algorithm derived from the dot-product algorithm in [18])**.** *If $p\geq 4$ and if no underflow or overflow occurs, the normwise error obtained when using Algorithm 4 for performing a quaternion multiplication is bounded by $u+32u^2$.*

The scaling method suggested in Section III-A2 can be used as well with Algorithm 4.

IV. Computing the reciprocal of a quaternion

Let us first analyze the error due to computing the reciprocal of q using Formula (4), and assuming that underflow and overflow do not occur. The term $|q|^2$ is obtained as variable $\hat{\sigma}$ at Line 7 of Algorithm 1. As shown in the analysis of that algorithm, it satisfies

$$|q|^2\cdot(1-v)^3\leq\hat{\sigma}\leq|q|^2\cdot(1+v)^3,$$

therefore, for $n=0,\ldots,3$, and with $\overline{q}_n=q_n$ if $n=0$, $-q_n$ otherwise, we have

$$\left|\frac{\overline{q}_n}{|q|^2}\right|\cdot\frac{(1-v)}{(1+v)^3}\leq\left|\mathrm{RN}\left(\frac{\overline{q}_n}{\hat{\sigma}}\right)\right|\leq\left|\frac{\overline{q}_n}{|q|^2}\right|\cdot\frac{(1+v)}{(1-v)^3}.$$

Hence, the componentwise and normwise relative errors of computing the reciprocal using (4) are bounded by

$$\frac{(1+v)}{(1-v)^3}-1=4u+5u^2+2u^3.$$

To avoid spurious overflow (and make underflow harmless) we can "scale" q by $F_1(q)$ (i.e., we obtain a quaternion $q'=q/F_1(q)$). This gives, using (5), $1/4\leq|q'|<1$. Therefore, in the calculation of $|q'|^2$, no overflow occurs, and a possible underflow has no incidence on the accuracy of the result. Then, since

$$\left|\frac{\overline{q'}}{|q'|^2}\right|=\frac{1}{|q'|},$$

we deduce, using (5) again, that

$$\frac{1}{2}<\left|\left|\frac{\overline{q'}}{|q'|^2}\right|\right|_\infty\leq 4. \tag{18}$$

Hence no overflow can occur during the division of $\overline{q}$ by $|q'|^2$. An underflow is possible on some (but not all) of the components of the result, leading to a possibly poor componentwise relative error. However, Eq. (18) implies that a possible underflowing component will be negligible in front of the largest component, so that the underflow will have no significant impact on the normwise relative error.

V. Conversion to/from a rotation matrix

Using quaternions for efficiently performing intermediate calculations involving rotations requires being able to efficiently convert rotations matrices to and from quaternions.

Converting from a quaternion to a rotation matrix is not difficult. Assume that the rotation matrix is

$$\mathcal{R}=\begin{pmatrix} r_{11} & r_{12} & r_{13}\\ r_{21} & r_{22} & r_{23}\\ r_{31} & r_{32} & r_{33}\end{pmatrix},$$

and let $q_0+q_1i+q_2j+q_3k$ be a (unitary) quaternion associated to the same rotation. We have [23]:

$$\mathcal{R}=2\times\begin{pmatrix} (q_0^2+q_1^2)-\frac{1}{2} & q_1q_2-q_0q_3 & q_1q_3+q_0q_2\\ q_1q_2+q_0q_3 & (q_0^2+q_2^2)-\frac{1}{2} & q_2q_3-q_0q_1\\ q_1q_3-q_0q_2 & q_2q_3+q_0q_1 & (q_0^2+q_3^2)-\frac{1}{2}\end{pmatrix}. \tag{19}$$

Applying (19) naively can lead to a large componentwise relative error. Consider for example the computation of the first diagonal element of $\mathcal{R}$ in the case $q_0=1/2-u$ and $q_1=1/2+u$. Computing r_{11} as $\mathrm{RN}(\mathrm{RN}(\mathrm{RN}(q_0^2)+\mathrm{RN}(q_1^2))-1/2)$ leads to a computed result equal to 0 whereas the exact result is $2u^2$. However, the normwise relative error remains small. To define the normwise error we have to consider a norm for $\mathcal{R}$. Let us choose $\|\mathcal{R}\|_\infty=\max_{i,j}|\mathcal{R}_{ij}|$. One easily shows that evaluating the r_{ij} naively leads to an absolute error on each component less than $\frac{7}{2}u$. Hence, if $\hat{\mathcal{R}}$ is the computed value of $\mathcal{R}$,

$$\frac{\|\hat{\mathcal{R}}-\mathcal{R}\|_\infty}{\|\mathcal{R}\|_\infty}\leq\frac{\frac{7}{2}u}{\|\mathcal{R}\|_\infty}.$$

Since $\mathcal{R}$ is a rotation matrix, it is orthogonal: the sum of the squares of the elements of any column in $\mathcal{R}$ is 1. As a consequence, at least one element has absolute value larger than $1/\sqrt{3}$. Hence, $\|\mathcal{R}\|_\infty\geq\sqrt{3}/3$. Therefore, the normwise relative error is bounded by $\frac{7}{2}\sqrt{3}u\leq 6.063u$.

Converting from a rotation matrix to a quaternion is significantly more difficult. Several solutions have been suggested and we refer to [23] for a recent presentation of them. Let us here analyse one possible solution, which is also employed in the Patrius Library of CNES (French Space Agency) [7]. First, one can observe from Eq. (19) that

$$\begin{aligned}
|q_0| &= \tfrac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}},\\
|q_1| &= \tfrac{1}{2}\sqrt{1+r_{11}-r_{22}-r_{33}},\\
|q_2| &= \tfrac{1}{2}\sqrt{1-r_{11}+r_{22}-r_{33}},\\
|q_3| &= \tfrac{1}{2}\sqrt{1-r_{11}-r_{22}+r_{33}}.
\end{aligned} \tag{20}$$

One can choose $q_0>0$, and to be consistent with that choice q_1 must have the sign of $r_{32}-r_{23}$, q_2 must have the

sign of $r_{13} - r_{31}$, and q_3 must have the sign of $r_{21} - r_{12}$. Straightforward use of Eq. (20) leads to possible large inaccuracies. However, since the norm of the obtained quaternion will be 1, at least one of its components is of absolute value larger than $1/2$. For that component, the corresponding value of $\pm r_{11} \pm r_{22} \pm r_{33}$ in Eq. (20) will be larger than 0. We can then compute that component using Eq. (20), and then deduce the other components using the following relations [23]:

$$\begin{array}{rcl} 4q_2q_3 &=& r_{23}+r_{32},\\ 4q_1q_3 &=& r_{31}+r_{13},\\ 4q_1q_2 &=& r_{21}+r_{12},\\ 4q_0q_1 &=& r_{32}-r_{23},\\ 4q_0q_2 &=& r_{13}-r_{31},\\ 4q_0q_3 &=& r_{21}-r_{12}. \end{array} \tag{21}$$

Hence, we will do the following: we start by successively computing the terms $\pm r_{11} \pm r_{22} \pm r_{33}$ that appear in Eq. (20) as $\mathrm{RN}(\pm r_{11} \pm \mathrm{RN}(r_{22} \pm r_{33}))$. As soon as we have found a term strictly larger than a certain fixed threshold η, we compute the component corresponding to that term using Eq. (20), and then we deduce the other components using Eq. (21) (i.e., for each term, we need a floating-point addition/subtraction followed by a division). The threshold is selected based on statistical trials in [22]: for instance it has the value $\eta = -0.25$ in [22] or it is fixed to $\eta = -0.19$ in the Patrius Library. Let us evaluate the maximum relative error of this algorithm for a generic threshold $\eta = -2^{-e}$, for $e \in \mathbb{N}$, $0 < e < p$. For the sake of simplicity, we assume that the component that is directly computed (i.e., the one for which $\mathrm{RN}(\pm r_{11} \pm \mathrm{RN}(r_{22} \pm r_{33})) > -2^{-e}$) is q_0.

We have successively computed $s_1 = \mathrm{RN}(r_{22} + r_{33})$ and $s_2 = \mathrm{RN}(r_{11} + s_1)$. Since $s_2 > -2^{-e}$, we have $r_{11} + s_1 \geq -2^{-e} + 2^{-e}u/2$. Also, since the coefficients of a rotation matrix are all of absolute value ≤ 1, we have $|r_{22}+r_{33}| \leq 2$, so that $|s_1| \leq 2$ (which implies $|s_2| \leq 3$) and $|s_1 - (r_{22}+r_{33})| \leq u$. All this gives $r_{11} + r_{22} + r_{33} \geq -2^{-e} - u + 2^{-e}u/2$, so that

$$4q_0^2 = 1 + r_{11} + r_{22} + r_{33} \geq 1 - 2^{-e} - u + 2^{-e}u/2. \tag{22}$$

Denoting the righthand side of (22) by

$$f(e,u) = (1 - 2^{-e}) - u(1 - 2^{-e-1}),$$

we have $q_0^2 \geq \frac{f(e,u)}{4}$.

Also, since $|s_2| \leq 3$, $|s_2 - (r_{11} + s_1)| \leq 2u$, hence we always have

$$|s_2 - (r_{11} + r_{22} + r_{33})| \leq 3u. \tag{23}$$

Now consider the addition

$$s_3 = \mathrm{RN}(1 + s_2),$$

we have $|s_3| \leq 4$ and $|s_3 - (1+s_2)| \leq 2u$, therefore, combined with (23),

$$|s_3 - (1 + r_{11} + r_{22} + r_{33})| \leq 5u. \tag{24}$$

Using this result and (22), we have

$$\begin{array}{c} (1 + r_{11} + r_{22} + r_{33}) \left[1 - \frac{5u}{f(e,u)}\right] \leq s_3 \\ \leq (1 + r_{11} + r_{22} + r_{33}) \left[1 + \frac{5u}{f(e,u)}\right], \end{array}$$

which implies

$$q_0 \left[1 - \frac{5u}{f(e,u)}\right]^{\frac{1}{2}} \leq \frac{1}{2}\sqrt{s_3} \leq q_0 \left[1 + \frac{5u}{f(e,u)}\right]^{\frac{1}{2}},$$

therefore, the computed value of q_0, namely $\hat{q}_0 = \frac{1}{2}\mathrm{RN}\sqrt{s_3}$, satisfies

$$q_0 \left[1 - \frac{5u}{f(e,u)}\right]^{\frac{1}{2}} (1-u) \leq \hat{q}_0 \leq q_0 \left[1 + \frac{5u}{f(e,u)}\right]^{\frac{1}{2}} (1+u).$$

Hence, the relative error on $\hat{q}_0$ satisfies

$$\mathrm{lb}(e,u) \leq \frac{\hat{q}_0 - q_0}{q_0} \leq \mathrm{ub}(e,u),$$

where the functions to be studied are:

$$\mathrm{lb}(e,u) := \left[1 - \frac{5u}{f(e,u)}\right]^{\frac{1}{2}} (1-u) - 1,$$

and

$$\mathrm{ub}(e,u) := \left[1 + \frac{5u}{f(e,u)}\right]^{\frac{1}{2}} (1+u) - 1.$$

Since the Taylor series at $u = 0$ of $\mathrm{lb}(e,u)$ and $\mathrm{ub}(e,u)$ is convergent as soon as $|u| \leq \frac{2^{e+1}-2}{12 \cdot 2^e - 1}$ (with $e > 0$), one obtains after a classical but tedious analysis:

$$\mathrm{lb}(e,u) = -\left(1 + \frac{5}{2\left(1 - 2^{-e}\right)}\right) u - \frac{5\left(5 + 2 \cdot 2^{-e}\right)}{8\left(1 - 2^{-e}\right)^2} u^2 - R_l(u^3),$$

$$\mathrm{ub}(e,u) = \left(1 + \frac{5}{2\left(1 - 2^{-e}\right)}\right) u + \frac{15\left(1 - 2 \cdot 2^{-e}\right)}{8\left(1 - 2^{-e}\right)^2} u^2 + R_u(u^3),$$

where both $|R_l|$ and $|R_u|$ are bounded by $280u^3$ as soon as $u \leq 1/32$ and $e < 1$.

For instance, for $\eta = -1/8$ (i.e. $e = 3$), the relative error on $\hat{q}_0$ is bounded by

$$\frac{27}{7}u + 7u^2,$$

as soon as $u \leq 1/128$ (i.e., as soon as $p \geq 7$, which always holds in practice). Note that similar bounds can be obtained for other values of η, with the same condition $u \leq 1/128$. For instance, for $\eta = -0.25$ (i.e. $e = 2$) the relative error on $\hat{q}_0$ is bounded by $\frac{13}{3}u + 9u^2$, while for $\eta = -0.19$, it becomes $4.1u + 7.4u^2$. For simplicity, we continue our analysis with $e = 3$.

Now, consider the calculation of

$$q_1 = \frac{r_{32} - r_{23}}{4q_0},$$

which is approximated by

$$\hat{q}_1 = \frac{1}{4}\mathrm{RN}\left(\frac{\mathrm{RN}(r_{32} - r_{23})}{\hat{q}_0}\right),$$

(the analysis is the same for q_2 and q_3). Without difficulty, we obtain

$$|q_1| \cdot \frac{(1-u)^2}{(1+\mathrm{ub}(e,u))} \leq \hat{q}_1 \leq |q_1| \cdot \frac{(1+u)^2}{(1+\mathrm{lb}(e,u))}.$$

For instance, by fixing $\eta = -1/8$ as above, this error becomes:

$$|q_1| \cdot \frac{(1-u)^2}{\left(1+\frac{27}{7}u+7u^2\right)} \leq \hat{q}_1 \leq |q_1| \cdot \frac{(1+u)^2}{\left(1-\frac{27}{7}u-7u^2\right)},$$

hence the relative error on q_1 is bounded by

$$\frac{(1+u)^2}{\left(1-\frac{27}{7}u-7u^2\right)} - 1,$$

which is less than

$$\frac{41}{7}u + 40u^2$$

as soon as $u \leq 1/128$ (i.e., $p \geq 7$), which always holds in practice. Therefore, we conclude

Lemma 4. *When the threshold η is $-1/8$ and as soon as $p \geq 7$, the componentwise relative error of computing the quaternion coefficients from the rotation matrix coefficients using the method presented in this section is bounded by $\frac{41}{7}u + 40u^2$.*

The bound of Lemma 4 also holds for the nomwise relative error.

Remark 1. *When the threshold η is set higher in magnitude, slightly worse componentwise relative error bounds are obtained, with the same setting as above:*

- *for $\eta = -0.19$, one has a bound of $6.1u + 40u^2$;*
- *for $\eta = -1/4$, one has a bound of $\frac{19}{3}u + 40u^2$.*

Hence, it is slightly more interesting to take a threshold which is lower in magnitude. In theory, $\eta \to 0$ provides the best error bound (which is greater than $3.5u$), but it more drastically restricts the component that can be chosen in Equation (20).

Conclusion

We have given relative error bounds for the major operations required for manipulating quaternions in floating-point arithmetic. These bounds are small, which confirms the fact that quaternions are easy to manipulate.

Acknowledgement

We thank a lot Denis Arzelier for his advice.

References

[1] IEEE standard for floating-point arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pages 1–84, July 2019.

[2] J. Burkardt. Quaternions: Quaternion arithmetic. Available in C and Fortran90 at http://people.math.sc.edu/Burkardt/f_src/quaternions/quaternions.html, 2018.

[3] R. Eisele. Quaternion.js, javascript library of quaternions. Available at https://github.com/infusion/Quaternion.js/blob/master/quaternion.js, 2016.

[4] O.D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-d objects. *The International Journal of Robotics Research*, 5(3):27–52, 1986.

[5] J. Flamant, P. Chainais, and N. Le Bihan. A complete framework for linear filtering of bivariate signals. *IEEE Transactions on Signal Processing*, 66(17):4541–4552, September 2018.

[6] B. P. Flannery, W. H. Press, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C,* 2nd edition. Cambridge University Press, New York, NY, 1992.

[7] CNES (French National Centre for Space Studies). CNES Patrius Java Library. Available at https://patrius.cnes.fr/index.php/, 2019.

[8] P. R. Girard. The quaternion group and modern physics. *European Journal of Physics*, 5(1):25–32, jan 1984.

[9] W. R. Hamilton. *Lectures on quaternions*. Hodges and Smith, Dublin, 1853.

[10] C.-P. Jeannerod, N. Louvet, and J.-M. Muller. Further analysis of Kahan's algorithm for the accurate computation of 2×2 determinants. *Mathematics of Computation*, 82(284):2245–2264, October 2013.

[11] W. Kahan. Branch cuts for complex elementary functions. In *The State of the Art in Numerical Analysis*, pages 165–211. Clarendon Press, Oxford, 1987.

[12] W. Kahan. Lecture notes on the status of IEEE-754. Available at http://www.cs.berkeley.edu/~wkahan/ieee754status/IEEE754.PDF, 1997.

[13] D. E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, Reading, MA, 3rd edition, 1998.

[14] J. B. Kuipers. Quaternions and rotation sequences. In *Proceedings of the International Conference on Geometry, Integrability and Quantization*, pages 127–143, Sofia, Bulgaria, 2000. Coral Press Scientific Publishing.

[15] O. Møller. Quasi double-precision in floating-point addition. *BIT*, 5:37–50, 1965.

[16] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser Boston, 2018. ACM G.1.0; G.1.2; G.4; B.2.0; B.2.4; F.2.1., ISBN 978-3-319-76525-9.

[17] Y. Nievergelt. Scalar fused multiply-add instructions produce floating-point matrix arithmetic provably accurate to the penultimate digit. *ACM Transactions on Mathematical Software*, 29(1):27–48, 2003.

[18] T. Ogita, S. M. Rump, and S. Oishi. Accurate sum and dot product. *SIAM Journal on Scientific Computing*, 26(6):1955–1988, 2005.

[19] D. M. Priest. Efficient scaling for complex division. *ACM Transactions on Mathematical Software*, 30(4), December 2004.

[20] O. Rodrigues. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ses déplacements considérés indépendamment des causes qui peuvent les produire (in french). *J. de Mathématiques Pures et Appliquées*, 5:380–440, 1840.

[21] C. Rucker. Integrating rotations using nonunit quaternions. *IEEE Robotics and Automation Letters*, 3(4):2979–2986, Oct 2018.

[22] S. Sarabandi and F. Thomas. Accurate computation of quaternions from rotation matrices. In *International Symposium on Advances in Robot Kinematics*, pages 39–46. Springer, 2018.

[23] S. Sarabandi and F. Thomas. A Survey on the Computation of Quaternions From Rotation Matrices. *Journal of Mechanisms and Robotics*, 11(2), 03 2019. 021006.

[24] K. Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, page 245–254, New York, NY, USA, 1985. Association for Computing Machinery.

[25] C. Song, G. Islas, and K. Schilling. Inverse dynamics based model predictive control for spacecraft rapid attitude maneuver. *IFAC-PapersOnLine*, 52(12):111 – 116, 2019. 21st IFAC Symposium on Automatic Control in Aerospace ACA 2019.