Universidade do Minho

Universidade do Minho
Escola de Ciências
Centro de Matemática

Universidade do Minho
Escola de Ciências
Departamento de Matemática e Aplicações

Núcleo de Investigação em Políticas Económicas

Evaluation schemes in the ring of quaternionic polynomials

M. Irene Falcão[a,c] Fernando Miranda[a,c] Ricardo Severino[c] M. Joana Soares[b,c]

[a] CMAT - Centre of Mathematics, University of Minho, Portugal
[b] NIPE - Economic Policies Research Unit, University of Minho, Portugal
[c] Department of Mathematics and Applications, University of Minho, Portugal

Information

Keywords:
Quaternions,
Polynomial evaluation,
Error analysis.

Original publication:
BIT Numerical Mathematics, 58(1) (2018), 51–72
DOI:10.1007/s10543-017-0667-8
Springer Link

Abstract

In this paper we focus on computational aspects associated with polynomial problems in the ring of one-sided quaternionic polynomials. The complexity and error bounds of quaternion arithmetic are considered and several evaluation schemes are analyzed from their complexity point of view. The numerical stability of generalized Horner's and Goertzel's algorithms to evaluate polynomials with quaternion floating-point coefficients is addressed. Numerical tests illustrate the behavior of the algorithms from the point of view of performance and accuracy.

1 Introduction

The processes of factoring and evaluating real or complex polynomials are very important problems and have received a lot of attention over the years [10, 12, 14]. In the ring of quaternionic polynomials new problems arise, mainly because the structure of the zero-set of quaternionic polynomials is quite different from the complex case. In particular, the relation between the factors and the zeros of a polynomial is not so simple now.

In this paper we address the problem of *evaluating* a polynomial, i.e., given a polynomial, find its value at a given argument.

The paper is organized as follows: in Section 2 we introduce the algebra of real quaternions, the complexity of the elementary arithmetic operations and the fundamentals of quaternion floating-point arithmetic. Section 3 contains a review of the main results concerning the ring of one-sided left quaternionic polynomials (i.e. quaternionic polynomials whose coefficients are on the left-hand side of the variable). Section 4 is entirely dedicated to several polynomial evaluation schemes, each of one associated to a particular form in which the polynomial can be given: expanded form, nested form, factor form or one of two remainder forms. Of these, we would like to point out the method which we have designated by Niven's method — acknowledging in this way the work of [17] — calling the attention to the fact that this procedure can be seen as a quaternionic version of the well-known algorithm of Goertzel [5]; in addition, we show that this method also corresponds to a particular case of the algorithm proposed in [19] for the efficient evaluation of powers. The main objective of Section 5 is to obtain forward error bounds for the Horner's and Niven's algorithms to polynomial evaluation.

In this context, we introduce the notion, similar to the classical case, of the condition number of the evaluation of a polynomial at a given point, and express the relative error bound in the approximations produced by each of the aforementioned methods in terms of this number. Finally, Section 6 illustrates the results of the paper by examples.

2 Quaternion arithmetic

2.1 The quaternion algebra

Let $\mathbb{H}$ denote the algebra of real quaternions. $\mathbb{H}$ is a vector space of dimension 4 over $\mathbb{R}$ with basis $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$ and multiplication rules $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -1$, $\mathbf{ij} = -\mathbf{ji} = \mathbf{k}$. Given a quaternion $x = x_0 + x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k}$, with $x_0, x_1, x_2, x_3 \in \mathbb{R}$, its *conjugate* $\overline{x}$ is defined as $\overline{x} = x_0 - x_1\mathbf{i} - x_2\mathbf{j} - x_3\mathbf{k}$; the *real part* of x is the number x_0 and is denoted by $\operatorname{Re} x$ and the *vector part* of x, denoted by $\underline{x}$, is given by $\underline{x} = x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k}$; the *norm* of x, $|x|$, is given by $|x| = \sqrt{x\overline{x}} = \sqrt{x_0^2 + x_1^2 + x_2^2 + x_3^2}$; if $x \neq 0$, the *inverse* of x, denoted by x^{-1}, is the (unique) quaternion such that $xx^{-1} = x^{-1}x = 1$ and is given by $x^{-1} = \frac{\overline{x}}{|x|^2}$. Note that the norm is multiplicative, i.e. we have $|xy| = |x||y|$, for all $x, y \in \mathbb{H}$.

2.2 Complexity

In the next sections we are going to present several algorithms to perform operations on polynomials together with their computational costs. We will assume that for quaternion numbers $x = x_0 + x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k}$ and $y = y_0 + y_1\mathbf{i} + y_2\mathbf{j} + y_3\mathbf{k}$ we compute

$$x \pm y = x_0 \pm y_0 + (x_1 \pm y_1)\mathbf{i} + (x_2 \pm y_2)\mathbf{j} + (x_3 \pm y_3)\mathbf{k} \tag{2.1}$$

$$kx = kx_0 + kx_1\mathbf{i} + kx_2\mathbf{j} + kx_3\mathbf{k},\ k \in \mathbb{R} \tag{2.2}$$

$$\begin{aligned} xy := x \times y = (x_0y_0 - x_1y_1 - x_2y_2 - x_3y_3) + (x_1y_0 + x_0y_1 - x_3y_2 + x_2y_3)\mathbf{i} \\ + (x_2y_0 + x_3y_1 + x_0y_2 - x_1y_3)\mathbf{j} + (x_3y_0 - x_2y_1 + x_1y_2 + x_0y_3)\mathbf{k} \end{aligned} \tag{2.3}$$

$$\frac{1}{x} = \frac{x_0 - x_1\mathbf{i} - x_2\mathbf{j} - x_3\mathbf{k}}{x_0^2 + x_1^2 + x_2^2 + x_3^2} \tag{2.4}$$

Table 1 summarizes the number of flops[1] required by each of the arithmetic operations (2.1)-(2.4).

Table 1: Operations cost

Operations	$\times_{\mathbb{R}}$	$\pm_{\mathbb{R}}$	$\div_{\mathbb{R}}$	flops
$\pm_{\mathbb{H}}$: quaternion $\pm$ quaternion	–	4	–	4
$\times_{\text{scalar}}$: real $\times$ quaternion	4	–	–	4
$\times_{\mathbb{H}}$: quaternion $\times$ quaternion	16	12	–	28
$\text{inv}_{\mathbb{H}}$: 1/quaternion	4	3	4	11

Remark 2.1. *It is possible to perform a quaternionic multiplication using only 8 real multiplications, but using 27 additions. In our days, with machines where the multiplications are as fast as additions, it is not important anymore to distinguish these type of operations and so multiplication performed as indicated in* (2.3) *is preferable.*

[1] In the context of this paper, a flop is any of the four elementary arithmetic operations $+$, $-$, $\times$, $\div$.

2.3 Quaternion floating-point arithmetic

To carry out error analysis of algorithms in quaternion arithmetic we recall first the principles of real floating-point arithmetic. We mostly follow [10] (see also [22] and [7]) and adapt the results to the quaternion context, considering the operations (2.1)-(2.4).

Let $\mathbb{F} \subset \mathbb{R}$ denote the set of all normalized floating-point numbers and let $\mathrm{fl} : \mathbb{R} \to \mathbb{F}$ denote rounding to the nearest according to IEEE 754. Floating-point operations in IEEE 754 satisfy the following standard model (assuming that no underflow nor overflow occurs), in which $x, y \in \mathbb{F}$ and $\mathrm{op} \in \{+, -, \times, \div\}$:

$$\mathrm{fl}(x \ \mathrm{op} \ y) = (x \ \mathrm{op} \ y)(1+\delta_1) = \frac{x \ \mathrm{op} \ y}{1+\delta_2}, \quad |\delta_1|,\ |\delta_2| \leq \mathbf{u},$$

where $\mathbf{u}$ is the unit roundoff of the system. For binary IEEE 754 double precision, $\mathbf{u} = 2^{-53}$.

Throughout the paper, we assume that, in the operations performed, overflow or underflow never occurs.

The following results play an important role in the error analysis performed in this section.

Lemma 2.1 ([10, Lemma 3.1]). *If $|\delta_i| \leq \mathbf{u}$ and $\rho_i = \pm 1$, $i = 1, \ldots, n$ and $n\mathbf{u} < 1$, then*

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1+\theta_n, \qquad \text{where} \qquad |\theta_n| \leq \frac{n\mathbf{u}}{1-n\mathbf{u}} =: \gamma_n. \tag{2.5}$$

Lemma 2.2 ([10, Lemma 3.3]). *For any positive integer k let θ_k denote a quantity bounded according to $|\theta_k| \leq \gamma_k$ with γ_k defined by (2.5). The following relations hold:*

i. $(1+\theta_k)(1+\theta_j) = 1+\theta_{j+k}$;

ii. $\frac{1+\theta_k}{1+\theta_j} = \begin{cases} 1+\theta_{k+j}, & j \leq k, \\ 1+\theta_{k+2j}, & j > k \end{cases}$;

iii. $\gamma_k\gamma_j \leq \gamma_{\min(k,j)}$, *if* $\max(j,k)\,\mathbf{u} \leq \frac{1}{2}$;

iv. $i\gamma_k \leq \gamma_{ik}$;

v. $\gamma_k + \mathbf{u} \leq \gamma_{k+1}$;

vi. $\gamma_k + \gamma_j + \gamma_k\gamma_j \leq \gamma_{k+j}$.

Theorem 2.1 ([11, Theorem 4.2]). *For $n \in \mathbb{N}$ and given $x = (x_i), y = (y_i)$, with $x_i, y_i \in \mathbb{F}$ for $i = 1, \ldots, n$, any order of evaluation of the inner-product $x^{\mathrm{T}}y$ produces an approximation $\mathrm{fl}(x^{\mathrm{T}}y)$ such that*

$$|\,\mathrm{fl}(x^{\mathrm{T}}y) - x^{\mathrm{T}}y| \leq n\mathbf{u}\sum_{i=1}^{n}|x_i||y_i|.$$

We denote by $\mathbb{F}_{\mathbb{H}} := \mathbb{F} + \mathbb{F}\mathbf{i} + \mathbb{F}\mathbf{j} + \mathbb{F}\mathbf{k}$ the set of quaternion floating-point numbers. As in the real (or complex) case we denote by $\mathrm{fl}(\cdot)$ the result of a floating-point computation, where all operations inside parentheses are done in floating-point working precision in the obvious way ([10, 7]). We adapt [10, Lemma 3.5] to derive the following error bounds of the quaternion operations.

Theorem 2.2. *For $x, y \in \mathbb{F}_{\mathbb{H}}$ and $k \in \mathbb{F}$ we have, with δ denoting a quaternion:*

i. $\mathrm{fl}(x \pm y) = (x \pm y)(1+\delta),\ |\delta| \leq \mathbf{u}$;

ii. $\mathrm{fl}(kx) = (kx)(1+\delta),\ |\delta| \leq \mathbf{u}$;

iii. $\mathrm{fl}(xy) = (xy)(1+\delta),\ |\delta| \leq 8\mathbf{u}$;

iv. $\mathrm{fl}(x\,\underline{y}) = (x\,\underline{y})(1+\delta),\ |\delta| \leq 3\sqrt{3}\mathbf{u}$;

v. $\mathrm{fl}(\frac{1}{x}) = \frac{1}{x}(1+\delta),\ |\delta| \leq \gamma_5$;

vi. $\mathrm{fl}(|x|^2) = |x|^2(1+\delta), \quad |\delta| \leq 4\mathbf{u}$.

Proof. We only prove relation **iii.**, the other results being obtained in a similar manner. From (2.3), we have $e := \mathrm{fl}(xy) - xy = e_0 + e_1\mathbf{i} + e_2\mathbf{j} + e_3\mathbf{k}$, where

$$\begin{aligned}
e_0 &= \mathrm{fl}(x_0y_0 - x_1y_1 - x_2y_2 - x_3y_3) - (x_0y_0 - x_1y_1 - x_2y_2 - x_3y_3),\\
e_1 &= \mathrm{fl}(x_1y_0 + x_0y_1 - x_3y_2 + x_2y_3) - (x_1y_0 + x_0y_1 - x_3y_2 + x_2y_3),\\
e_2 &= \mathrm{fl}(x_2y_0 + x_3y_1 + x_0y_2 - x_1y_3) - (x_2y_0 + x_3y_1 + x_0y_2 - x_1y_3),\\
e_3 &= \mathrm{fl}(x_3y_0 - x_2y_1 + x_1y_2 + x_0y_3) - (x_3y_0 - x_2y_1 + x_1y_2 + x_0y_3).
\end{aligned}$$

By using the result of Theorem 2.1, it follows that

$$|e_i| \leq 4\mathbf{u}\, A_i, \quad i = 0, 1, 2, 3,$$

where

$$\begin{aligned}
A_0 &:= |x_0||y_0| + |x_1||y_1| + |x_2||y_2| + |x_3||y_3|,\\
A_1 &:= |x_1||y_0| + |x_0||y_1| + |x_3||y_2| + |x_2||y_3|,\\
A_2 &:= |x_2||y_0| + |x_3||y_1| + |x_0||y_2| + |x_1||y_3|,\\
A_3 &:= |x_3||y_0| + |x_2||y_1| + |x_1||y_2| + |x_0||y_3|.
\end{aligned}$$

Using the fact that $(|a| + |b| + |c| + |d|)^2 \leq 4(a^2 + b^2 + c^2 + d^2)$, we can write

$$\begin{aligned}
A_0^2 &\leq 4\left(x_0^2y_0^2 + x_1^2y_1^2 + x_2^2y_2^2 + x_3^2y_3^2\right),\\
A_1^2 &\leq 4\left(x_1^2y_0^2 + x_0^2y_1^2 + x_3^2y_2^2 + x_2^2y_3^2\right),\\
A_2^2 &\leq 4\left(x_2^2y_0^2 + x_3^2y_1^2 + x_0^2y_2^2 + x_1^2y_3^2\right),\\
A_3^2 &\leq 4\left(x_3^2y_0^2 + x_2^2y_1^2 + x_1^2y_2^2 + x_0^2y_3^2\right).
\end{aligned}$$

Therefore, we obtain

$$\begin{aligned}
|e| &= \sqrt{e_0^2 + e_1^2 + e_2^2 + e_3^2} \leq 4\mathbf{u}\sqrt{A_0^2 + A_1^2 + A_2^2 + A_3^2}\\
&\leq 8\mathbf{u}\sqrt{(x_0^2 + x_1^2 + x_2^2 + x_3^2)(y_0^2 + y_1^2 + x_2^2 + y_3^2)}\\
&= 8\mathbf{u}|x||y| = 8\mathbf{u}|xy|,
\end{aligned}$$

where in the last equality we used the fact that the norm is multiplicative. □

3 The ring of quaternionic polynomials

3.1 One-sided left polynomials

In this paper we focus on the so-called *one-sided* or *unilateral left* quaternionic polynomials, i.e. polynomials whose coefficients are located only on the left-hand side of the powers of x,

$$p(x) = a_nx^n + a_{n-1}x^{n-1} + \cdots + a_1x + a_0, \quad a_k \in \mathbb{H}. \tag{3.1}$$

As usual, if $a_n \neq 0$, we say that the *degree* of the polynomial p is n, write $\deg p = n$ and refer to a_n as the leading coefficient of the polynomial. When $a_n = 1$, we say that p is *monic*. The set of polynomials of the form (3.1) is a ring with respect to the operations of addition $(+)$ and multiplication $(*)$ defined in the same way as in the commutative case, where the variable x is assumed to commute with the quaternion coefficients, which we will denote by $\mathbb{H}[x]$. The *evaluation* of $p \in \mathbb{H}[x]$ at a given quaternion α is defined as

$$p(\alpha) = a_n\alpha^n + a_{n-1}\alpha^{n-1} + \cdots + a_1\alpha + a_0.$$

We say that α is a *zero* of p if $p(\alpha) = 0$.

The next result underlines the fact that the evaluation at α is in general not a ring homomorphism from $\mathbb{H}[x]$ to $\mathbb{H}$, i.e. factoring does not preserve evaluation.

Theorem 3.1 ([13]). *Let $f, g \in \mathbb{H}[x]$, $p = f * g$ and $\alpha \in \mathbb{H}$. Then*

$$p(\alpha) = \begin{cases} 0, & \text{if } g(\alpha) = 0, \\ f(\tilde{\alpha})\, g(\alpha), & \text{if } g(\alpha) \neq 0 \end{cases} \qquad \text{where} \quad \tilde{\alpha} = g(\alpha)\, \alpha\, (g(\alpha))^{-1}. \tag{3.2}$$

In particular, if α is a zero of p which is not a zero of g, then $\tilde{\alpha}$ is a zero of p.

3.2 Factorization

We summarize now the results needed in the sequel. For more details we refer the reader mainly to [13, 17, 20].

Theorem 3.2 (Euclidean division [13]). *If f and g are polynomials in $\mathbb{H}[x]$ (with $\deg g \leq \deg f$ and $g \neq 0$), then there exist unique polynomials q and r in $\mathbb{H}[x]$ such that*

$$f(x) = q(x) * g(x) + r(x) \tag{3.3}$$

with $r = 0$ or $\deg r < \deg g$.

Theorem 3.3 (Factor Theorem [6, 13]). *Let $p \in \mathbb{H}[x]$ and $\alpha \in \mathbb{H}$. Then, α is a zero of p if and only if there exists $q \in \mathbb{H}[x]$ such that $p(x) = q(x) * (x - \alpha)$.*

Theorem 3.4 (Fundamental Theorem of Algebra [17]). *Any non-constant polynomial in $\mathbb{H}[x]$ always has a zero in $\mathbb{H}$.*

The following result is an immediate consequence of the aforementioned theorems.

Theorem 3.5. - Factorization into linear terms *Any monic polynomial p of degree n $(n \geq 1)$ in $\mathbb{H}[x]$ admits a factorization into linear factors, i.e. there exist $x_1, \ldots, x_n \in \mathbb{H}$, such that*

$$p_n(x) = (x - x_n) * (x - x_{n-1}) * \cdots * (x - x_1). \tag{3.4}$$

4 Polynomial evaluation

An evaluation algorithm depends, of course, on the form in which the polynomial is (or can be) written. In what follows we are going to consider a polynomial p_n of degree n in the following forms:

- *Expanded form*
$$p_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \tag{4.1}$$
- *Nested form*
$$p_n(x) = ((\ldots (a_n x + a_{n-1}) * x + \ldots) * x + a_1) * x + a_0 \tag{4.2}$$
- *Factor form*
$$p_n(x) = a_n (x - x_n) * (x - x_{n-1}) * \cdots * (x - x_1) \tag{4.3}$$
- *Remainder form*
$$p_n(x) = q_{n-1}(x) * (x - \alpha) + c_0, \ \alpha \in \mathbb{H}, \tag{4.4}$$
where $c_0 = p_n(\alpha)$.
- *"Niven" remainder form*
$$p_n(x) = q_{n-2}(x) * (x - \bar{\alpha}) * (x - \alpha) + c_1 x + c_0, \ \alpha \in \mathbb{H}. \tag{4.5}$$
Naturally, if p_n is written as (4.5), then $p_n(\alpha) = c_1 \alpha + c_0$.

We would like to call attention to the equations (4.4) and (4.5) which correspond to the use of the Euclidean division algorithm (3.3) with $b(x) = x - \alpha$ and $b(x) = (x - \bar{\alpha}) * (x - \alpha)$, respectively. The polynomial

$$\Psi_\alpha := (x - \bar{\alpha}) * (x - \alpha) = x^2 - 2 \operatorname{Re} \alpha\, x + |\alpha|^2 \tag{4.6}$$

is known as the characteristic polynomial of the quaternion α. The designation of "Niven" remainder form was motivated by its use in the seminal paper [17].

4.1 Direct evaluation

The direct evaluation of a polynomial p_n with degree n at α in its expanded form (4.1) requires $\left[\frac{n(n+1)}{2}\right] \times_{\mathbb{H}}$ and $[n] \pm_{\mathbb{H}}$, corresponding to a total of $14n^2 + 18n$ flops.

Algorithm 1: Direct evaluation

$p_0 = a_0$;
for $k = 0 : n-1$
    $p_{k+1} = p_k + a_{k+1}\alpha^{k+1}$;
end

When the powers of α are recursively computed, as in Algorithm 1' below, the process requires $[2n-1] \times_{\mathbb{H}}$ and $[n] \pm_{\mathbb{H}}$ which gives a total of $60n - 28$ flops.

Algorithm 1': Direct evaluation with recursion

$t_1 = \alpha$;
$p_1 = a_0 + a_1 t_1$;
for $k = 1 : n-1$
    $t_{k+1} = \alpha t_k$;
    $p_{k+1} = p_k + a_{k+1} t_{k+1}$;
end

4.2 Horner's rule

The value $p = p_n(\alpha)$ can be obtained from (4.2) as

$$p = ((\dots(a_n\alpha + a_{n-1})\alpha + \dots)\alpha + a_1)\alpha + a_0$$

or from the following recursive scheme.

Algorithm 2: Horner's rule

$c_n = a_n$;
for $k = n-1 : -1 : 0$
    $c_k = c_{k+1}\alpha + a_k$;
end
$p = c_0$;

These nested and recursive forms are both quaternionic versions of the well-known Horner's method. Horner's rule provides a computationally efficient method of evaluating real polynomials p_n and can also be used for a compact presentation of the division of p_n by the linear polynomial $x - \alpha$, usually designated by *synthetic division*. In the quaternionic case, as in the commutative case, the coefficients of the quotient polynomial q_{n-1} in (4.4) are also generated by Horner's method for $p_n(\alpha)$, i.e.

$$q_{n-1}(x) = c_n x^{n-1} + c_{n-1}x^{n-2} + \dots + c_2 x + c_1.$$

The computational cost of Horner's scheme can be obtained as follows: $[n] \times_{\mathbb{H}}$ and $[n] \pm_{\mathbb{H}}$, which corresponds to $32n$ flops.

4.3 Factor form evaluation

If p_n is known in the form (4.3), the evaluation process is not so simple as in the classical case, where, as in Horner's scheme, n multiplications and n additions are needed. Here, due to the fact that the evaluation map

is not a homomorphism (see Section 2) one has to apply (3.2) recursively. In fact, if q_n has a factorization of the form (3.4) and writing

$$q_n(x) = r_n(x) = (x - x_n) * r_{n-1}(x),$$

where $r_{n-1}(x) = (x - x_{n-1}) * \cdots * (x - x_1)$, one has

$$q_n(\alpha) = r_n(\alpha) = \left(r_{n-1}(\alpha)\alpha r_{n-1}(\alpha)^{-1} - x_n\right) r_{n-1}(\alpha) = r_{n-1}(\alpha)\alpha - x_n r_{n-1}(\alpha).$$

Hence $p = p_n(\alpha) = a_n r_n(\alpha)$. The corresponding algorithm is as follows:

Algorithm 3: Factor form evaluation

$r_1 = \alpha - x_1$;
for $k = 1 : n - 1$
  $r_{k+1} = r_k\alpha - x_{k+1}r_k$;
end
$p = a_n\, r_n$;

This algorithm requires $[2n - 1]\times_{\mathbb{H}}$ and $[n]\pm_{\mathbb{H}}$ corresponding to a total of $60n - 28$ flops as the direct evaluation of Algorithm 1'.

4.4 Efficient evaluation of powers

In [19], the authors observed that since any quaternion α is a zero of its characteristic polynomial Ψ_α, any positive integer power of α can be written as

$$\alpha^n = A_n\alpha + B_n,$$

with real A_n and B_n. The following algorithm corresponds essentially to the algorithm proposed by the authors to recursively obtain the values of A_n and B_n and hence to compute $t = \alpha^n$.

Algorithm 4: Efficient evaluation of powers

$r = 2\,\mathrm{Re}(\alpha); s = |\alpha|^2$;
$A_1 = 1; B_1 = 0$;
for $k = 1 : n - 1$
  $A_{k+1} = rA_k + B_k$;
  $B_{k+1} = -sA_k$;
end
$t = A_n\alpha + B_n$;

As a consequence, the value $p = p_n(\alpha)$ can be obtained from

$$p = A\alpha + B, \tag{4.7}$$

where

$$A = a_1 + \sum_{k=2}^{n} A_k a_k \qquad \text{and} \qquad B = a_0 + \sum_{k=2}^{n} B_k a_k. \tag{4.8}$$

This algorithm involves the following costs: the computation of s and r requires 1 flop and 7 flops, respectively. A_n and B_n together cost $3n - 3$ flops, while the computation of A and B requires $8n - 8$ flops. Finally, considering the 32 flops needed for obtaining p using (4.7), we reach a total of $19n + 21$ flops.

Observe that the computation of α^n uses $3n + 10$ flops, while the direct approach needs $28n - 28$. This algorithm also requires less flops for the evaluation of $p_n(\alpha)$ than Horner's rule.

4.5 Niven's algorithm

The determination of "Niven" remainder form (4.5), corresponding to the division of p_n by the characteristic polynomial of α (cf. (4.6)), can also be presented in a compact form by the use of an expanded synthetic division corresponding to a slight modification of Algorithm 2. Algorithm 5 addresses this issue.

Algorithm 5: Niven's algorithm

$r = 2\,\mathrm{Re}(\alpha); s = |\alpha|^2;$
$c_{n+1} = 0;\ c_n = a_n;$
for $k = n-1 : -1 : 1$
    $c_k = a_k + r\,c_{k+1} - s\,c_{k+2};$
end
$c_0 = a_0 - s\,c_2;$
$p = c_1\alpha + c_0;$

In this case, the coefficients of the quotient polynomial q_{n-2} are also generated by the expanded synthetic division, i.e.

$$p_n(x) = (c_n x^{n-2} + c_{n-1}x^{n-3} + \cdots + c_3 x + c_2) * \Psi_\alpha(x) + c_1 x + c_0. \tag{4.9}$$

Concerning the complexity of the algorithm, we observe that the computation of the coefficients c_k requires $16n$ flops. Therefore, Niven's algorithm costs only $16n+32$ flops, which represents, for $n > 3$, an improvement over the other methods presented in this section. A similar procedure for evaluating complex functions is also referred in [12]. Note, however, that if $\alpha \in \mathbb{R}$, Horner's method requires only $8n$ flops, whilst Niven's procedure requires $16n + 2$ flops. For this reason, in what follows, when referring to Niven's method to compute $p_n(\alpha)$, we assume that $\alpha \in \mathbb{H} \setminus \mathbb{R}$.

Remark 4.1. *Observe that when $p_n(x) = x^n$, Niven's recurrence relations can easily be written as*

$$\begin{aligned} c_k &= rc_{k+1} + b_{k+1} \\ b_k &= -sc_{k+1};\ k = n-1:-1:1, \end{aligned}$$

since $a_k = 0,\ k = 0, \ldots, n-1$. Also $c_n = 1$ and $b_n = c_{n+1} = 0$. Therefore, Algorithm 4 turns out to be a particular case of Niven's algorithm for evaluating powers (with $A_j = c_{n-j-1}$ and $B_j = b_{n-j-1};\ j = 1, \ldots, n$). However the use of (4.7)-(4.8) to evaluate $p_n(\alpha)$ for more general polynomials is less efficient than Algorithm 5.

Table 2 contains a summary of the operations cost of the evaluation algorithms presented in this section.

Table 2: Complexity of the evaluation algorithms

Algorithm	flops
Direct evaluation - Algorithm 1	$14n^2 + 18n$
Direct evaluation - Algorithm 1'	$60n - 28$
Horner's rule - Algorithm 2	$32n$
Factor form evaluation - Algorithm 3	$60n - 28$
Powers evaluation - Algorithm 4	$19n + 21$
Niven's scheme - Algorithm 5	$16n + 32$

Example 1. *Evaluation of the polynomial*

$$p_4(x) = x^4 + (1 + \mathbf{j} - \mathbf{k})x^3 + (1 - 3\mathbf{i} + \mathbf{j} + \mathbf{k})x + 2 + 2\mathbf{j}$$

at $x = \mathbf{i}$.

[Algorithm 1] $p_4(\mathbf{i}) = \mathbf{i}^4 + (1+\mathbf{j}-\mathbf{k})\mathbf{i}^3 + (1-3\mathbf{i}+\mathbf{j}+\mathbf{k})\mathbf{i} + 2 + 2\mathbf{j} = 6 + 4\mathbf{j}$

[Algorithm 1']

$$\begin{array}{ll} t_1 = \mathbf{i} & p_1 = 5 + \mathbf{i} + 3\mathbf{j} - \mathbf{k} \\ t_2 = \mathbf{i}t_1 = -1 & p_2 = p_1 = 5 + \mathbf{i} + 3\mathbf{j} - \mathbf{k} \\ t_3 = \mathbf{i}t_2 = -\mathbf{i} & p_3 = p_2 + (1+\mathbf{j}-\mathbf{k})(-\mathbf{i}) = 5 + 4\mathbf{j} \\ t_4 = \mathbf{i}t_3 = 1 & p_4 = p_3 + 1 = 6 + 4\mathbf{j} \end{array}$$

[Algorithm 2]

	1	$1+\mathbf{j}-\mathbf{k}$	0	$1-3\mathbf{i}+\mathbf{j}+\mathbf{k}$	$2+2\mathbf{j}$
$\mathbf{i}$		$\mathbf{i}$	$-1+\mathbf{i}-\mathbf{j}-\mathbf{k}$	$-1-\mathbf{i}-\mathbf{j}+\mathbf{k}$	$4+2\mathbf{j}$
	1	$1+\mathbf{i}+\mathbf{j}-\mathbf{k}$	$-1+\mathbf{i}-\mathbf{j}-\mathbf{k}$	$-4\mathbf{i}+2\mathbf{k}$	$\underbrace{6+4\mathbf{j}}_{c_0}$

[Algorithm 3] *It is easy to see that*

$$p_4(x) = (x - x_4) * (x - x_3) * (x - x_2) * (x - x_1),$$

where $x_1 = -\mathbf{i}$, $x_2 = 1 + \mathbf{i}$, $x_3 = -1 - \mathbf{j}$, $x_4 = -1 + \mathbf{k}$. *Therefore*

$$\begin{array}{ll} p_1(\mathbf{i}) = \mathbf{i} + \mathbf{i} = 2\mathbf{i} & p_3(\mathbf{i}) = p_2(\mathbf{i})\mathbf{i} + (1+\mathbf{j})p_2(\mathbf{i}) = 2 - 2\mathbf{i} + 2\mathbf{k} \\ p_2(\mathbf{i}) = p_1(\mathbf{i})\mathbf{i} - (1+\mathbf{i})p_1(\mathbf{i}) = -2\mathbf{i} & p_4(\mathbf{i}) = p_3(\mathbf{i})\mathbf{i} - (-1+\mathbf{k})p_3(\mathbf{i}) = 6 + 4\mathbf{j} \end{array}$$

[Algorithm 4]

k	2	3	4
A_k	0	-1	0
B_k	-1	0	1

$$\begin{array}{l} A = a_1 - a_3 = -3\mathbf{i} + 2\mathbf{k} \\ B = a_0 - 2a_2 + a_4 = 3 + 2\mathbf{j} \\ p_4(\mathbf{i}) = A\mathbf{i} + B = 6 + 4\mathbf{j} \end{array}$$

[Algorithm 5]

	1	$1+\mathbf{j}-\mathbf{k}$	0	$1-3\mathbf{i}+\mathbf{j}+\mathbf{k}$	$2+2\mathbf{j}$
-1			-1	$-1-\mathbf{j}+\mathbf{k}$	1
0		0	0	0	
	1	$1+\mathbf{j}-\mathbf{k}$	-1	$\underbrace{-3\mathbf{i}+2\mathbf{k}}_{c_1}$	$\underbrace{3+2\mathbf{j}}_{c_0}$

$$p_4(\mathbf{i}) = c_1\mathbf{i} + c_0 = 6 + 4\mathbf{j}$$

5 Error analysis

In this section we analyze the rounding errors of the Horner's and Niven's algorithms, presented in Section 4, for evaluating a one-sided left polynomial of degree n, $p_n = \sum_{i=0}^{n} a_i x^i$, with $a_i \in \mathbb{F}_{\mathbb{H}}$, at a point $\alpha = \alpha_0 + \alpha_1\mathbf{i} + \alpha_2\mathbf{j} + \alpha_3\mathbf{k} \in \mathbb{F}_{\mathbb{H}}$.

5.1 Horner's algorithm

Denote by $\text{Horner}(p_n, \alpha)$ the result of the evaluation at $x = \alpha$ of the polynomial p_n through Horner's algorithm.

Theorem 5.1. *A forward error bound in polynomial evaluation by Horner's rule is*

$$|p_n(\alpha) - \text{Horner}(p_n, \alpha)| \leq \gamma_{9n}\hat{p}_n(|\alpha|),$$

where

$$\hat{p}_n(x) := \sum_{i=0}^{n} |a_i| x^i. \tag{5.1}$$

Proof. The proof is an adaptation of the corresponding proof for the real case, given in [10, p. 95]; see also [8, p.65], where the complex case was considered, and [22], where the numerical stability of Horner's rule was studied for the first time.

First, we observe that any quaternion x of the form $x = (1+\delta)\alpha$, with $\delta, \alpha \in \mathbb{H}$ can be written as $x = \alpha(1+\tilde{\delta})$, with $|\delta| = |\tilde{\delta}|$ and also that Lemma 2.1, which was established in [10] for real numbers δ_i, is valid for quaternions numbers δ_i (where, naturally, in that case, the symbol $|\cdot|$ refers to the norm of a quaternion). Then, using the error bounds of the quaternion floating-point arithmetic operations given in Theorem 2.2 — cases **i.** and **iii.** — and the result **vi.** of Lemma 2.2, one can easily show that

$$\mathsf{Horner}(p_n, \alpha) = \sum_{i=0}^{n} a_i(1+\delta_i)\alpha^i$$

where $|\delta_i| \leq \gamma_{9i+1}$ for $i = 0, \ldots, n-1$ and $|\delta_n| \leq \gamma_{9n}$. We thus have

$$|p_n(\alpha) - \mathsf{Horner}(p_n, \alpha)| \leq \sum_{i=0}^{n} |a_i||\delta_i||\alpha|^i \leq \gamma_{9n} \sum_{i=0}^{n} |a_i||\alpha|^i = \gamma_{9n}\hat{p}_n(|\alpha|),$$

as intended. □

If $p_n(\alpha) \neq 0$, then

$$\frac{|p_n(\alpha) - \mathsf{Horner}(p_n, \alpha)|}{|p_n(\alpha)|} \leq \gamma_{9n} \frac{\hat{p}_n(|\alpha|)}{|p_n(\alpha)|}.$$

Following the classical case, we introduce the notation

$$\mathrm{cond}(p_n, \alpha) := \frac{\hat{p}_n(|\alpha|)}{|p_n(\alpha)|} \tag{5.2}$$

and call this quantity the *condition number* for the evaluation of the polynomial p_n at α.

Thus, the relative error in polynomial evaluation by Horner's rule can be written as

$$\frac{|p_n(\alpha) - \mathsf{Horner}(p_n, \alpha)|}{|p_n(\alpha)|} \leq \gamma_{9n}\,\mathrm{cond}(p_n, \alpha). \tag{5.3}$$

Since the condition number can be arbitrary large, the above expression shows that, as in the classical case, one can not guarantee accurate polynomial evaluation by Horner's rule.

5.2 Niven's algorithm

At this stage we should point out that Niven's algorithm can be seen as the generalization to the quaternionic case of the Goertzel's algorithm for complex polynomial evaluation (see [5], [12, pp. 468]). Goertzel's method is just an extension of Clenshaw's method [1] for evaluating Chebyshev series by three-term recurrence and is a popular technique in digital signal processing. The stability of Goertzel's algorithm has been considered by several authors (see [4, 16, 18]), but for our purpose here we mostly follow [21].

Two aspects of the quaternion algebra play an important role in the generalization of the error analysis of Goertzel's algorithm to quaternionic context. The first one is the *de Moivre formula* for quaternions.

Theorem 5.2 ([9, Theorem 2.14])**.** *Any quaternion $x = x_0 + x_1\mathbf{i} + x_2\mathbf{j} + x_3\mathbf{k} \in \mathbb{H}$, $\underline{x} \neq 0$, can be written in the trigonometric form*

$$x = |x|(\cos\theta + \omega(x)\sin\theta), \ -\pi < \theta \leq \pi, \tag{5.4}$$

where

$$\cos\theta = \frac{x_0}{|x|}, \ \sin\theta = \frac{|\underline{x}|}{|x|} \text{ and } \omega(x) = \frac{\underline{x}}{|\underline{x}|}. \tag{5.5}$$

Note that $\omega(x) \in S^2$, where S^2 designates the unit sphere in $\mathbb{R}^3$. Moreover, a de Moivre formula can be established:

$$(\cos\theta + \omega(x)\sin\theta)^n = \cos n\theta + \omega(x)\sin n\theta. \tag{5.6}$$

The second key result concerns the fact that the explicit form of the coefficients of a three-term recurrence relation presented in the paper by Goertzel [5] can be extended in a straightforward manner to the more general situation considered here. In fact, if we replace the last two lines of Niven's algorithm by the equivalent relations

$$\begin{array}{l} u = a_0 + \alpha_0\, c_1 - s\, c_2;\ \ v = c_1\, \underline{\alpha}; \\ p = u + v; \end{array} \tag{5.7}$$

one can easily derive:

Theorem 5.3. *The quantities computed by Niven's algorithm are given explicitly by*

$$c_j = \sum_{k=j}^{n} a_k |\alpha|^{k-j} U_{k-j}(t),\ \ j = 1, 2, \ldots, n, \tag{5.8}$$

$$u = \sum_{k=0}^{n} a_k |\alpha|^k T_k(t), \quad v = \left(\sum_{k=1}^{n} a_k |\alpha|^{k-1} U_{k-1}(t) \right) \underline{\alpha}, \quad t = \frac{\alpha_0}{|\alpha|}, \tag{5.9}$$

where $T_k(t)$ and $U_k(t)$ are the Chebyshev polynomials of the first kind and of the second kind, respectively.

Remark 5.1. *It is well known (see e.g. [2]) that for $|t| < 1$*

$$T_k(t) = \cos k\theta \qquad \textit{and} \qquad U_k(t) = \frac{\sin(k+1)\theta}{\sin\theta}, \qquad \textit{where} \qquad t = \cos\theta.$$

Hence the trigonometric form (5.4) of a quaternion x and the de Moivre formula (5.6) allow us to write $x^k = |x|^k(\cos k\theta + \omega(x)\sin k\theta) = |x|^k(T_k(t) + \omega(x)\sin\theta U_{k-1}(t))$. This, together with (5.5), implies that

$$x^k = |x|^k T_k(t) + \underline{x}|x|^{k-1} U_{k-1}(t),\ \ k = 0, 1, \ldots, n. \tag{5.10}$$

We are now in the position to present the error analysis of Niven's algorithm. Here, as observed previously, we follow closely the analysis for the complex case given in [21], whose approach is particularly suitable for a quaternionic extension.

Denoting by $\mathrm{Niven}(p_n, \alpha)$ the result of evaluating at $x = \alpha$ a polynomial p_n of degree n, through (5.7), and retaining, for simplicity of the analysis, only error terms of first degree, the following error bound can be established.

Theorem 5.4. *A forward error bound in polynomial evaluation by Niven's algorithm is*

$$|p_n(\alpha) - \mathrm{Niven}(p_n, \alpha)| \leq \vartheta_n\, \mathbf{u}\, \hat{p}_n(|\alpha|) + \mathcal{O}(\mathbf{u}^2),$$

where

$$\vartheta_n := 12n(n+1) + (1 + 3\sqrt{3})n + 1 \tag{5.11}$$

and $\hat{p}_n$ is the polynomial defined by (5.1).

Proof. We shall analyze the effects of roundoff by backward error analysis. Assume that the arithmetic operations in Niven's recurrence relation are performed in the order indicated and denote by $\tilde{u}, \tilde{v}$, $\tilde{c}_k$, $\tilde{s}$ and $\tilde{p}$ the quantities actually computed by Niven's algorithm through (5.7) (we consider that no roundoff error is generated in the process of computing r). Recall from Theorem 2.2 that

$$\tilde{s} = fl(|\alpha|^2) = s(1+\delta), \quad |\delta| \leq 4\mathbf{u}.$$

Therefore $\tilde{c}_{n+1} = 0$, $\tilde{c}_n = a_n$ and, for $j = n-1, \ldots, 1$, we get

$$\tilde{c}_j = (a_j + \eta_j) + r\, \tilde{c}_{j+1} - s\, \tilde{c}_{j+2}, \tag{5.12}$$

where

$$|\eta_j| \leq \gamma_2 |a_j| + 2\gamma_3 |\alpha|\, |\tilde{c}_{j+1}| + \gamma_6 |\alpha|^2 |\tilde{c}_{j+2}|. \tag{5.13}$$

We also have

$$\tilde{u} = (a_0 + \eta_0) + \alpha_0\,\tilde{c}_1 - s\,\tilde{c}_2, \tag{5.14}$$

where

$$|\eta_0| \le \gamma_2|a_0| + \gamma_3|\alpha||\,\tilde{c}_1| + \gamma_6|\alpha|^2|\tilde{c}_2|. \tag{5.15}$$

We can combine the two inequalities (5.13) and (5.15) in the form

$$|\eta_j| \le \gamma_6\big(|a_j| + |\alpha||\tilde{c}_{j+1}| + |\alpha|^2|\tilde{c}_{j+2}|\big),\ j = n-1,\dots,0,$$

or, alternatively, as

$$|\eta_j| \le 6\mathbf{u}\big(|a_j| + |\alpha||\tilde{c}_{j+1}| + |\alpha|^2|\tilde{c}_{j+2}|\big) + \mathcal{O}(\mathbf{u}^2),\ j = n-1,\dots,0. \tag{5.16}$$

Taking into account (5.8)-(5.9), relations (5.12) and (5.14) give

$$\tilde{c}_j = \sum_{k=j}^{n}(a_k + \eta_k)|\alpha|^{k-j}U_{k-j}(t),\ j = n-1,\dots,1, \tag{5.17}$$

$$\tilde{u} = \sum_{k=0}^{n}(a_k + \eta_k)|\alpha|^{k}T_k(t). \tag{5.18}$$

Writing $\tilde{c}_j = c_j + e_j$, $j = 1,\dots,n-1$ and $\tilde{u} = u + e_0$ we have

$$|e_j| \le \sum_{k=j}^{n}|\eta_k||\alpha|^{k-j}|U_{k-j}(t)| \quad \text{and} \quad |e_0| \le \sum_{k=0}^{n}|\eta_k||\alpha|^{k}|T_k(t)|.$$

The well-known properties of Chebyshev polynomials, $|T_k(t)| \le 1$ and $|U_k(t)| \le k+1$ allow us to write the last relations as

$$|e_j| \le \sum_{k=j}^{n}(k-j+1)|\eta_k||\alpha|^{k-j} \le (n-j+1)\sum_{k=j}^{n}|\eta_k||\alpha|^{k-j} \quad \text{and} \quad |e_0| \le \sum_{k=0}^{n}|\eta_k||\alpha|^{k}.$$

The same properties produce (cf. (5.8)-(5.9))

$$|c_j| \le (n-j+1)\sum_{k=j}^{n}|a_k||\alpha|^{k-j} \qquad \text{and} \qquad |u| \le \sum_{k=0}^{n}|a_k||\alpha|^{k}.$$

Introducing the notation

$$g_j := \sum_{k=j}^{n}|a_k||\alpha|^{k-j},\ j = 0,1,\dots,n,$$

the last inequalities can be written simply as

$$|c_j| \le (n-j+1)g_j,\ j = 1,2,\dots,n, \qquad \text{and} \qquad |u| \le g_0 = \hat{p}_n(|\alpha|),$$

where $\hat{p}_n$ is the polynomial defined by (5.1). We also have, from (5.16),

$$\begin{aligned}
|\eta_j| &\leq 6\mathbf{u}\Big(|a_j| + \sum_{k=j+1}^{n}(|a_k|+|\eta_k|)|\alpha|^{k-j}(k-j-1) \\
&\qquad + \sum_{k=j+2}^{n}(|a_k|+|\eta_k|)|\alpha|^{k-j}(k-j-2)\Big) + \mathcal{O}(\mathbf{u}^2) \\
&\leq 6\mathbf{u}\Big(|a_j| + \sum_{k=j+1}^{n}(|a_k|+\mathcal{O}(\mathbf{u}))|\alpha|^{k-j}(k-j-1) \\
&\qquad + \sum_{k=j+2}^{n}(|a_k|+\mathcal{O}(\mathbf{u}))|\alpha|^{k-j}(k-j-2)\Big) + \mathcal{O}(\mathbf{u}^2) \\
&= 6\mathbf{u}\Big(|a_j| + \sum_{k=j+1}^{n}|a_k||\alpha|^{k-j}(k-j-1) \\
&\qquad + \sum_{k=j+2}^{n}|a_k||\alpha|^{k-j}(k-j-2)\Big) + \mathcal{O}(\mathbf{u}^2) \\
&\leq 12\mathbf{u}g_j(n-j) + \mathcal{O}(\mathbf{u}^2). \qquad (5.19)
\end{aligned}$$

Observing that $\sum_{j=0}^{n} g_j|\alpha|^j = \sum_{j=0}^{n}\sum_{k=j}^{n}|a_k||\alpha|^k \leq (n+1)g_0$, it follows that

$$\begin{aligned}
\sum_{j=0}^{n}|\eta_j||\alpha|^j &\leq 12\mathbf{u}\sum_{j=0}^{n}(n-j)g_j|\alpha|^j + \mathcal{O}(\mathbf{u}^2) \\
&\leq 12n(n+1)\,\mathbf{u}\,g_0 + \mathcal{O}(\mathbf{u}^2). \qquad (5.20)
\end{aligned}$$

Next, observe that since $p_n(\alpha) = u + v = u + c_1\,\underline{\alpha}$, we have (see Theorem 2.2 – iv)

$$\tilde{p} = \tilde{u} + \tilde{v} + \xi, \quad \text{with} \quad |\xi| \leq \mathbf{u}\big(|\hat{u}| + (1+3\sqrt{3})|\tilde{c}_1||\alpha|\big) + \mathcal{O}(\mathbf{u}^2). \qquad (5.21)$$

Since

$$|\tilde{u}| \leq |u| + |e_0| \leq g_0 + \mathcal{O}(\mathbf{u})$$

and

$$|\tilde{c}_1||\alpha| \leq (|c_1| + |e_1|)|\alpha| \leq ng_1|\alpha| + \mathcal{O}(\mathbf{u}) \leq ng_0 + \mathcal{O}(\mathbf{u})$$

we obtain

$$|\xi| \leq ((1+3\sqrt{3})n + 1)\,\mathbf{u}\,g_0 + \mathcal{O}(\mathbf{u}^2). \qquad (5.22)$$

But, (5.17)-(5.18) produce

$$\begin{aligned}
\tilde{u} + \tilde{c}_1\underline{\alpha} &= \sum_{k=0}^{n}(a_k+\eta_k)|\alpha|^k T_k(t) + \left(\sum_{k=1}^{n}(a_k+\eta_k)|\alpha|^{k-1}U_{k-1}(t)\right)\underline{\alpha} \\
&= a_0 + \eta_0 + \sum_{k=1}^{n}(a_k+\eta_k)\big(|\alpha|^k T_k(t) + \underline{\alpha}|\alpha|^{k-1}U_{k-1}(t)\big) \\
&= \sum_{k=0}^{n}(a_k+\eta_k)\alpha^k, \qquad (5.23)
\end{aligned}$$

where we have used (5.10) and the identity $T_0(t) = 1$. Therefore from (5.21)-(5.22) we have

$$|\tilde{p} - p_n(\alpha)| \leq |\tilde{u} + \tilde{c}_1\underline{\alpha} - u - c_1\underline{\alpha}| + |\xi|.$$

Using (5.8)-(5.9), (5.17)-(5.18) and (5.23) we obtain

$$\begin{aligned}|\tilde{p} - p_n(\alpha)| &\leq \left|\sum_{k=0}^{n}(a_k+\eta_k)\alpha^k - \sum_{k=0}^{n} a_k\alpha^k\right| + |\xi| \\ &\leq \sum_{k=0}^{n}|\eta_k||\alpha|^k + ((1+3\sqrt{3})n+1)\,\mathbf{u}\,g_0 + \mathcal{O}(\mathbf{u}^2).\end{aligned}$$

Finally, from (5.20) we obtain

$$|p_n(\alpha) - \mathsf{Niven}(p_n,\alpha)| \leq \vartheta_n \mathbf{u}\, g_0 + \mathcal{O}(\mathbf{u}^2) = \vartheta_n \mathbf{u}\, \hat{p}_n(|\alpha|) + \mathcal{O}(\mathbf{u}^2),$$

where ϑ_n is given by (5.11), which completes the proof. □

If $p_n(\alpha) \neq 0$, then the relative error in polynomial evaluation by Niven's algorithm can be bounded as

$$\frac{|p_n(\alpha) - \mathsf{Niven}(p_n,\alpha)|}{|p_n(\alpha)|} \leq \vartheta_n \mathbf{u}\,\mathrm{cond}(p_n,\alpha) + \mathcal{O}(\mathbf{u}^2). \tag{5.24}$$

6 Numerical tests

In this section we report some of the experiments we have carried out to illustrate the accuracy and computing time of the algorithms presented along the paper. Our numerical tests were performed using IEEE 754 double precision with Matlab R2016a ($\mathbf{u} = 2^{-53}$). In order to exactly evaluate a polynomial, we have used the Mathematica package `QuaternionAnalysis` [3, 15] which was endowed with the ability to perform operations on quaternion polynomials.

More detailed informations about the package can be obtained from the website

http://w3.math.uminho.pt/QuaternionAnalysis.

6.1 Performance comparisons

We have compared, in terms of computing time, four algorithms: the recursive direct approach of Algorithm 1', the Horner's rule (Algorithm 2), the Pogorui & Shapiro efficient evaluation of powers (Algorithm 4) and finally, Niven's method (Algorithm 5). We do not present results for Algorithm 1 and Algorithm 4 for the obvious reasons: the first one is very time consuming, while the second requires the knowledge of the factor-terms of the polynomial under consideration and therefore cannot be directly compared with the other methods.

We have implemented straightforwardly in Matlab the four aforementioned methods, taking into account the quaternion arithmetic operations (2.1)-(2.3) and identifying a quaternion with a 1×4 array of real numbers.

All the simulations have been performed on a computer with Intel Core i7-4700Hp processor at 2.40 GHz with 8 GB of RAM and 6 MB of cache memory. We have chosen polynomials with random quaternionic coefficients $a_k = a_{k,0} + a_{k,1}\mathbf{i} + a_{k,2}\mathbf{j} + a_{k,3}\mathbf{k}$, where $a_{k,i} \in [-5,5]$, with degrees varying from 5 to 250. These polynomials were evaluated at 500 random quaternion points in the same range of a_k. The tests confirm the complexity analysis carried out in Section 4 (cf. Table 2), as illustrated in Figure 1.

6.2 Accuracy comparisons

In order to test the accuracy of Horner's and Niven's methods we have adapted the example presented in [8, p. 69] to the quaternionic case. Instead of considering the evaluation of complex polynomials of the form $(x-(1+\mathbf{i}))^n$ we focus on quaternionic polynomials

$$p_n(x) = (x - (1+\mathbf{i}-\mathbf{j}-\mathbf{k}))^n;\ n = 3,\ldots,20, \tag{6.1}$$

written in expanded form and consider the evaluation of p_n at two different points.

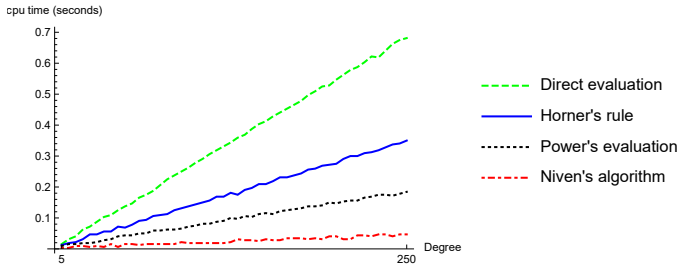

Figure 1: Average measured running time for Algorithms 1' (Direct evaluation), 2 (Horner's algorithm), 4 (Efficient computation of powers) and 5 (Niven's algorithm)

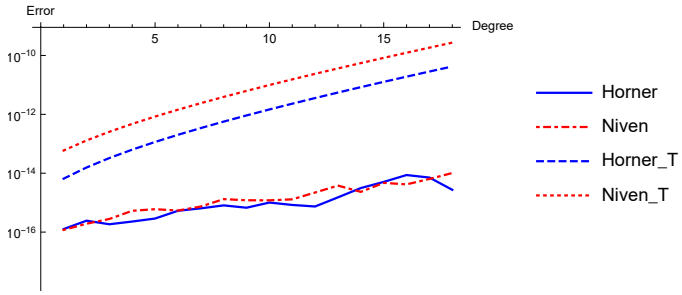

Figure 2: Relative accuracy of Horner's and Niven's algorithms for evaluating the polynomial p_n given by (6.1) at $\alpha_1 = \mathrm{fl}(1 + \frac{1}{2}\mathbf{i} + \frac{1}{3}\mathbf{j} + \frac{1}{4}\mathbf{k})$. A priori error bounds (Horner_T and Niven_T) are also represented. Here, the condition number varies from 3 (for $n = 3$) to 5×10^2 (for $n = 20$).

We first consider a point $\alpha_1 = \mathrm{fl}(1 + \frac{1}{2}\mathbf{i} + \frac{1}{3}\mathbf{j} + \frac{1}{4}\mathbf{k})$ for which the condition number $\mathrm{cond}(p_n, \alpha_1)$ varies from 3 (for $n = 3$) to 5×10^2 (for $n = 20$). Following [8] we present in Figure 2 the relative accuracy

$$\frac{|p_n(\alpha_1) - \mathrm{Horner}(p_n, \alpha_1)|}{|p_n(\alpha_1)|} \quad \text{and} \quad \frac{|p_n(\alpha_1) - \mathrm{Niven}(p_n, \alpha_1)|}{|p_n(\alpha_1)|}.$$

The a priori errors (5.3) and (5.24) are also plotted in the same figure.

Due to the magnitude of the condition number, one can expect accurate enough approximations to $p_n(\alpha_1)$ (cf. (5.3) and (5.24)). This is clearly illustrated in Figure 2.

We have also chosen a point $\alpha_2 = \mathrm{fl}(1.333 + 1.333\mathbf{i} - 1.333\mathbf{j} - 1.333\mathbf{k})$ such that the condition number $\mathrm{cond}(p_n, \alpha_2)$ rapidly grows with the degree n of the polynomial, varying from 3×10^2 (for $n = 3$) to 5×10^{16} (for $n = 20$). Our purpose was to call the attention to the fact that the classical ill-conditioning issues associated with polynomial evaluation (see e.g. [8]) also appear, as expected, in the quaternionic case.

To illustrate the above considerations, in Figure 3 we present a plot of the errors against the condition number corresponding to the evaluation of p_n at α_2 for $n = 3$ to $n = 20$.

In Figures 2 and 3 we can observe that the error bounds (5.3) and (5.24) are always pessimistic compared to the actual measured errors. The same behavior was observed in the other test examples we have performed.

7 Final remarks

The increasing interest in quaternionic numerical methods, particularly root-finding methods, justifies the importance and motivation for the study of polynomial evaluation schemes from the point of view of their complexity and stability, conducted in this paper. Such study was mainly focused on two algorithms: the Horner's rule for quaternions and what we have called Niven's algorithm – a generalization to $\mathbb{H}$ of the classical Goertzel's algorithm.

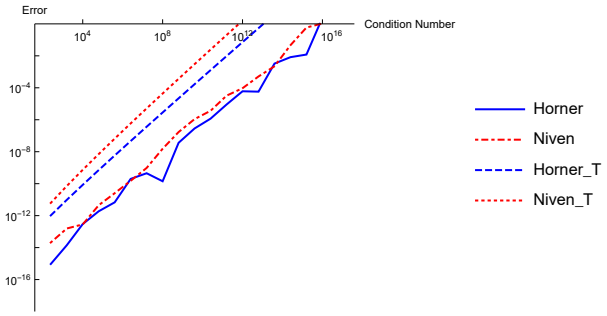

Figure 3: Relative accuracy of Horner's and Niven's algorithms for evaluating the polynomials p_n given by (6.1) at $\alpha_2 = \mathrm{fl}(1.333 + 1.333\mathbf{i} - 1.333\mathbf{j} - 1.333\mathbf{k})$. A priori error bounds (Horner_T and Niven_T) are also represented. Here, as in Figure 2, the degree of the polynomials varies from $n = 3$ to $n = 20$.

The theoretical analysis carried out in the paper shows that Niven's method requires less operations than Horner's rule and that both methods produce accurate approximations to the value of $p_n(\alpha)$, as far as the condition number $\mathrm{cond}(p_n, \alpha)$ is not large.

Not surprisingly, for large values of the condition number both methods suffer from instability producing a computed value with few (or even none) exact digits. Such behavior is similar to the one exhibited by classical polynomial evaluation schemes. In the classical case, several techniques are known to overcome such difficulties (see e.g. [21, 8, 7]). In the near future, quaternionic versions of these procedures will also be considered.

Acknowledgements

Research at CMAT was financed by Portuguese Funds through FCT - Fundação para a Ciência e a Tecnologia, within the Project UID/MAT/00013/2013. Research at NIPE was carried out within the funding with COMPETE reference number POCI-01-0145-FEDER-006683 (UID/ECO/03182/2013), with the FCT/MEC's (Fundação para a Ciência e a Tecnologia, I.P.) financial support through national funding and by the ERDF through the Operational Programme on "Competitiveness and Internationalization - COMPETE 2020" under the PT2020 Partnership Agreement.

References

[1] C.W. Clenshaw. A note on the summation of Chebyshev series. *Math. Comp.*, 9(51):118–120 (1955)

[2] P.J. Davis. *Interpolation and approximation*. Blaisdell Publishing Co. Ginn and Co. New York-Toronto-London (1963)

[3] M.I. Falcão and F. Miranda. Quaternions: A Mathematica package for quaternionic analysis. *Lecture Notes in Comput. Sci.*, 6784:200–214 (2011)

[4] W.M. Gentleman. An error analysis of Goertzel's (Watt's) method for computing Fourier coefficients. *Comput. J.*, 12:160–165 (1969)

[5] G. Goertzel. An algorithm for the evaluation of finite trigonometric series. *Amer. Math. Monthly*, 65:34–35 (1958)

[6] B. Gordon and T.S. Motzkin. On the zeros of polynomials over division rings I. *Trans. Amer. Math. Soc.*, 116:218–226 (1965)

[7] S. Graillat, P. Langlois, and N. Louvet. Algorithms for accurate, validated and fast polynomial evaluation. *Japan J. Indust. Appl. Math.*, 26(2-3):191–214 (2009)

[8] S. Graillat and V. Ménissier-Morain. Accurate summation, dot product and polynomial evaluation in complex floating point arithmetic. *Information and Computation*, 216:57–71 (2012)

[9] K. Gürlebeck, K. Habetha, and W. Sprößig. *Holomorphic Functions in the Plane and n-dimensional space*. Birkhäuser Verlag, Basel (2008) Translated from the 2006 German original.

[10] N.J. Higham. *Accuracy and Stability of Numerical Algorithms*. 2nd edn. SIAM, Philadelphia (2002)

[11] C.-P. Jeannerod and S. M. Rump. Improved error bounds for inner products in floating-point arithmetic. *SIAM J. Matrix Anal. Appl.*, 34(2):338–344 (2013)

[12] D.E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. 2nd edn. Addison-Wesley, Reading (1981)

[13] T.-Y. Lam. *A First Course in Noncommutative Rings*. Graduate Texts in Mathematics. Springer-Verlag, New York (1991)

[14] J.M. McNamee. *Numerical Methods for Roots of Polynomials*, volume 16 of *Part I*. Elsevier B. V., Amsterdam (2007)

[15] F. Miranda and M.I. Falcão. QuaternionAnalysis Package: User's Guide, Technical Report (2014) http://w3.math.uminho.pt/QuaternionAnalysis

[16] A.C.R. Newbery. Error analysis for Fourier series evaluation. *Math. Comp.*, 27(123):639–644 (1973)

[17] I. Niven. Equations in quaternions. *Amer. Math. Monthly*, 48:654–661 (1941)

[18] J. Oliver. Rounding error propagation in polynomial evaluation schemes. *J. Comput. Appl. Math.*, 5(2):85–97 (1979)

[19] A. Pogorui and M. Shapiro. On the structure of the set of zeros of quaternionic polynomials. *Complex Var. Theory Appl.*, 49(6):379–389 (2004)

[20] R. Serôdio and L.-S. Siu. Zeros of quaternion polynomials. *Appl. Math. Lett.*, 14(2):237–239 (2001)

[21] A. Smoktunowicz and I. Wróbel. On improving the accuracy of Horner's and Goertzel's algorithms. *Numer. Algorithms*, 38(4):243–258 (2005)

[22] J.H. Wilkinson. *Rounding Errors in Algebraic Processes*. Notes on Applied Science No. 32, Her Majesty's Stationery Office, London (1963). Also published by Prentice-Hall, Englewood Cliffs, NJ, USA. Reprinted by Dover, New York (1994)

[23] F. Zhang. Quaternions and matrices of quaternions. *Linear Algebra Appl.*, 251:21–57 (1997)