Computational Aspects of Quaternionic Polynomials

Part II: Root-Finding Methods

M. Irene Falcão
Fernando Miranda
Ricardo Severino
M. Joana Soares

This article explores the numerical mathematics and visualization capabilities of Mathematica in the framework of quaternion algebra. In this context, we discuss computational aspects of the recently introduced Newton and Weierstrass methods for finding the roots of a quaternionic polynomial.

Introduction

Since Niven proved in his pioneering work [1] that every nonconstant polynomial of the form

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0, \text{ with } a_n \neq 0 \text{ and } a_k \in \mathbb{H} \text{ (the quaternions)}, \tag{1}$$

has at least one zero in $\mathbb{H}$, thereby extending the fundamental theorem of algebra to quaternionic polynomials, the use of such polynomials has been considered by different authors and in different contexts. Quaternionic polynomials ([2]) have found a wealth of applications in a number of different areas and have motivated the design of efficient methods for numerically approximating their zeros (see e.g. [3–8]).

This article discusses two numerical methods to approximate the zeros (or roots) of polynomials of the form (1). They can be seen as the quaternionic versions of the well-known Newton and Weierstrass iterative root-finding methods and they both rely on quaternion arithmetic. Here we explain in detail how we have used Mathematica to produce the numerical results recently presented in [9–11].

All the computations in this article require the package `QuaternionAnalysis`, available for download at w3.math.uminho.pt/QuaternionAnalysis (see [12] and [13]).

Newton-Like Methods

Theoretical Framework

We introduce the basic definitions and results needed; we refer to Part 1 of this article [2] for recalling the main aspects of the quaternion algebra $\mathbb{H}$ and to [14] for details on quaternionic calculus.

The real vector space $\mathbb{R}^4$ can be identified with $\mathbb{H}$ by means of

$$\mathbf{x} := (x_0, x_1, x_2, x_3) \in \mathbb{R}^4 \longleftrightarrow x = x_0 + i\,x_1 + j\,x_2 + k\,x_3 \in \mathbb{H},$$

where i, j and k are Hamilton's imaginary units. Thus, throughout the article, we do not distinguish an element in $\mathbb{R}^4$ from the corresponding quaternion in $\mathbb{H}$, unless we need to stress the context.

Using the simplified notation $\underline{x} := i\,x_1 + j\,x_2 + k\,x_3$ for the vector part of x, any arbitrary nonreal quaternion x can be written as

$$x = x_0 + \underline{x} = x_0 + \omega(\underline{x})\,|\,\underline{x}\,|, \tag{2}$$

where $|\,\underline{x}\,|$ is the norm of $\underline{x}$ and $\omega(\underline{x})$ is the quaternion

$$\omega(\underline{x}) = \frac{\underline{x}}{|\,\underline{x}\,|}, \tag{3}$$

also referred to as the sign of $\underline{x}$. In addition, since $\omega(\underline{x})^2 = -1$ and $|\,\omega(\underline{x})\,| = 1$, one can say that $\omega(\underline{x})$ behaves like the complex imaginary unit, and for this reason we call (2) the complex-like form of the quaternion x.

In what follows, we consider domains $\Omega \subset \mathbb{R}^4 \cong \mathbb{H}$ and functions $f : \Omega \to \mathbb{H}$ that can be written in the form

$$f(x) = f(x_0 + \boldsymbol{\omega}\, r) = u(x_0, r) + \boldsymbol{\omega}\, v(x_0, r), \tag{4}$$

where $r := |\,\underline{x}\,|$, $\boldsymbol{\omega} := \omega(\underline{x})$ and u and v are real-valued functions. Continuity and differentiability are defined coordinate-wise.

We define on the set $C^1(\Omega, \mathbb{H})$ the so-called radial operators

$$\partial_{\text{rad}} := \frac{1}{2}(\partial_0 - \boldsymbol{\omega}\,\partial_r) \text{ and } \overline{\partial}_{\text{rad}} := \frac{1}{2}(\partial_0 + \boldsymbol{\omega}\,\partial_r),$$

where $\partial_0 := \frac{\partial}{\partial x_0}$ and $\partial_r := \frac{\partial}{\partial x_r}$.

We introduce the following concept.

Definition 1

Let f be a function of the form (4), $x \in \Omega$ *and* $h = h_0 + \omega(\underline{x})\,h_r$, *with* $h_0, h_r \in \mathbb{R}$. *Such a function f is called radially holomorphic (or radially regular) in Ω if*

$$\lim_{h \to 0}\,(f(x+h) - f(x))\,h^{-1}$$

exists. In that case, this limit is called the radial derivative of f at x and is denoted by f'.

Theorem 1

A function f of the form (4) *is radially holomorphic iff $\overline{\partial}_{\text{rad}} f = 0$. In that case, we have $f' = \partial_{\text{rad}} f = \partial_0 f = \partial_0 u(x_0, r) + \omega\, \partial_0 v(x_0, r)$.*

It follows at once that any quaternionic polynomial of the form (1) but with $a_k \in \mathbb{R}$ is radially holomorphic and its radial derivative is

$$P'(x) = n\, a_n\, x^{n-1} + (n-1)\, a_{n-1}\, x^{n-2} + \ldots + a_1. \tag{5}$$

▫ Quaternionic Newton Method

For holomorphic complex functions f of one complex variable, the well-known Newton method for finding a zero x^* consists of approximating $x^* \in \mathbb{C}$ by means of the iterative process

$$\text{2D–NM}: z_{k+1} = z_k - \frac{f(z_k)}{f'(z_k)}, k = 0, 1, 2, \ldots, \tag{6}$$

with $z_k \in \mathbb{C}$ sufficiently close to $x^* \in \mathbb{R}^2$ and $f'(z_k) \neq 0$. Identifying a real quaternion with a vector in $\mathbb{R}^4$, the problem of solving any quaternionic equation can always be transformed into the problem of solving a system of four nonlinear equations, whose solutions, in turn, can be obtained by using the multivariate version of (6):

$$\text{4D–NM}: z_{k+1} = z_k - (J f(z_k))^{-1} f(z_k),\ k = 0, 1, 2, \ldots, \tag{7}$$

with $z_k \in \mathbb{R}^4$ sufficiently close to $x^* \in \mathbb{R}^4$ and a nonsingular Jacobian matrix $J f(z_k)$. Not surprisingly, recent experiments performed by some of the authors of this article ([9], [10]) have shown the substantial gain in computational effort that can be achieved when using a direct quaternionic approach to this problem.

Newton methods in the quaternion context were formally adapted for the first time by Janovská and Opfer in [7], where the authors solved equations of the form $x^n = a, a \in \mathbb{H}$. Later, Kalantari in [15], using algebraic-combinatorial arguments, proposed a Newton method for finding roots of special quaternionic polynomials. In [9], the equivalence between the classical multivariate Newton method (7) and quaternionic versions of Newton methods for a class of functions was established.

Due to the noncommutativity of multiplication for quaternions, the quotient of two quaternions p and q may be interpreted in two different ways: either as $p\, q^{-1}$ (the right quotient) or $q^{-1}\, p$ (the left quotient). This leads naturally to considering two versions of Newton iteration in the quaternionic setting:

$$\mathbb{H} - \text{NM}_{\text{right}}: z_{k+1} = z_k - f(z_k)\,(f'(z_k))^{-1},\ k = 0, 1, 2, \ldots, \tag{8}$$

$$\mathbb{H} - \text{NM}_{\text{left}}: z_{k+1} = z_k - (f'(z_k))^{-1} f(z_k)\,,\ k = 0, 1, 2, \ldots. \tag{9}$$

The derivative in equations (8) and (9) has been considered in [9] and [10] as the radial derivative of a radially holomorphic function. In fact, in Corollary 2 of [9] it was proved that for such functions, equations (7), (8) and (9) produce, for each z_0, the same sequence, provided that $J f(z_k),\ k = 0, 1, \ldots$ is nonsingular. Here is a more general result.

Theorem 2 ([9], Theorem 4)

Let $f(x) = \sum_{i=0}^{s} \alpha_i f_i(x)$ be a function defined on the set $C^1(\Omega, \mathbb{H})$ such that the f_i, $i = 0, \ldots, s$, are radially holomorphic functions in Ω and the α_i are quaternions not all zero. If z^* is a root of f such that $J f(z^*)$ is nonsingular and $J f$ is Lipschitz continuous on a neighborhood of z^*, then for all $z_0 \in \mathbb{H}$ sufficiently close to z^* such that $\omega(\underline{z_0})$ commutes with all $\omega(\underline{\alpha_i})$, the Newton processes

$$\mathbb{H}-\mathrm{NM}_{\mathrm{right}} : z_{k+1} = z_k - f(z_k)\left(\sum_{i=0}^{s} \alpha_i f_i{}'(z_k)\right)^{-1}, k = 0, 1, 2, \ldots \tag{10}$$

$$\mathbb{H}-\mathrm{NM}_{\mathrm{left}} : z_{k+1} = z_k - \left(\sum_{i=0}^{s} \alpha_i f_i{}'(z_k)\right)^{-1} f(z_k), k = 0, 1, 2, \ldots \tag{11}$$

both produce the same sequence as (7), which converges quadratically to z^*.

Each step k of the iterative schemes (10) and (11) is implemented in the function `NewtonIterativeFunction`, which has as arguments the quaternion z_k and the indication of the version: `"right"` for (10) or `"left"` for (11). At each step, a test of the value of $f'(z_k)$ is also performed. We recall again that all the functions presented here require the package `QuaternionAnalysis`.

```
Needs["QuaternionAnalysis`"]
```

```
⚠ SetCoordinates: The coordinates system is set to {X0, X1, X2, X3}.
```

```
NewtonIterativeFunction[f_, Df_, q_, "right", t_: 10^-16] :=
 If[Abs@(Df @@ q) > t, q - f @@ q ** (1 / Df @@ q), Null]
```

```
NewtonIterativeFunction[f_, Df_, q_, "left", t_: 10^-16] :=
 If[Abs@(Df @@ q) > t, q - (1 / Df @@ q) ** f @@ q, Null]
```

The $\mathbb{H}$-Newton methods consist of the successive application of the iterative schemes (9) or (10) through the function `NewtonIterations`, using a stopping criteria based on the incremental size $|z_{k+1} - z_k|$ and on the maximum number of iterations `ItMax`.

```
TestConvergence[z1_, z2_, eps_] := Norm[z1 - z2] > eps
```

```
NewtonIterations[f_, Df_, z0_, method_, eps_: 10^-12,
  ItMax_: 20] :=
 NestWhileList[NewtonIterativeFunction[f, Df, #, method] &,
  z0, TestConvergence[##, eps] &, 2, ItMax]
```

Example 1

Consider the radially holomorphic polynomial $p(x) = x^3 - x$, whose only roots in $\mathbb{H}$ are the real isolated roots -1, 1 and 0. For the concepts of isolated and spherical roots, we refer the reader to [2], Definition 4.

```
f1[X0_, X1_, X2_, X3_] =
  QPower[Quaternion[X0, X1, X2, X3], 3] -
   Quaternion[X0, X1, X2, X3];
Df1 = Function[{X0, X1, X2, X3},
   Evaluate[Map[D[#, X0] &, f1[X0, X1, X2, X3]]]];
```

The use of the initial guess $z_0 = 1 - i + j - k$ requires nine iterations to get an approximation to the root 0 with precision 10^{-12}. The fact that both methods produce the same sequence is also confirmed.

```
NewtonIterations[f1, Df1, Quaternion[1., -1., 1., -1.],
  "right"] // Column

Quaternion[1., -1., 1., -1.]
Quaternion[0.713376, -0.611465, 0.611465, -0.611465]
Quaternion[0.540286, -0.323683, 0.323683, -0.323683]
Quaternion[0.426575, -0.0788592, 0.0788592, -0.0788592]
Quaternion[-0.0117115, 0.167662, -0.167662, 0.167662]
Quaternion[-0.00409338, 0.0225044, -0.0225044, 0.0225044]
Quaternion[-0.0000369029,
 0.0000658452, -0.0000658452, 0.0000658452]
Quaternion[-2.77941 × 10^-12,
 1.17485 × 10^-12, -1.17485 × 10^-12, 1.17485 × 10^-12]
Quaternion[0., 0., 0., 0.]
Quaternion[0., 0., 0., 0.]
```

```
NewtonIterations[f1, Df1, Quaternion[1., -1., 1., -1.],
  "left"] // Column

Quaternion[1., -1., 1., -1.]
Quaternion[0.713376, -0.611465, 0.611465, -0.611465]
Quaternion[0.540286, -0.323683, 0.323683, -0.323683]
Quaternion[0.426575, -0.0788592, 0.0788592, -0.0788592]
Quaternion[-0.0117115, 0.167662, -0.167662, 0.167662]
Quaternion[-0.00409338, 0.0225044, -0.0225044, 0.0225044]
Quaternion[-0.0000369029,
 0.0000658452, -0.0000658452, 0.0000658452]
Quaternion[-2.77941 × 10^-12,
 1.17485 × 10^-12, -1.17485 × 10^-12, 1.17485 × 10^-12]
Quaternion[0., 0., 0., 0.]
Quaternion[0., 0., 0., 0.]
```

The use of the initial guesses $z_0 = 1 + j$ and $z_0 = 1 - j$ requires 14 iterations to get an approximation to the roots 1 and -1, respectively.

```
NewtonIterations[f1, Df1, Quaternion[1., 0., 1., 0.],
  "right"] // Column
```

```
Quaternion[1., 0., 1., 0.]
Quaternion[0.756757, 0., 0.540541, 0.]
Quaternion[0.6375, 0., 0.146216, 0.]
Quaternion[0.782518, 0., -0.5633, 0.]
Quaternion[0.654848, 0., -0.172497, 0.]
Quaternion[0.765795, 0., 0.433337, 0.]
Quaternion[0.679227, 0., 0.049088, 0.]
Quaternion[1.42625, 0., -0.397221, 0.]
Quaternion[1.11157, 0., -0.203991, 0.]
Quaternion[0.98741, 0., -0.0588488, 0.]
Quaternion[0.994677, 0., 0.00153626, 0.]
Quaternion[1.00004, 0., -0.0000249862, 0.]
Quaternion[1., 0., -2.95062×10^-9, 0.]
Quaternion[1., 0., -1.22895×10^-17, 0.]
Quaternion[1., 0., 0., 0.]
```

```
NewtonIterations[f1, Df1, Quaternion[1., 0., 1., 0.],
  "left"] // Column
```

```
Quaternion[1., 0., 1., 0.]
Quaternion[0.756757, 0., 0.540541, 0.]
Quaternion[0.6375, 0., 0.146216, 0.]
Quaternion[0.782518, 0., -0.5633, 0.]
Quaternion[0.654848, 0., -0.172497, 0.]
Quaternion[0.765795, 0., 0.433337, 0.]
Quaternion[0.679227, 0., 0.049088, 0.]
Quaternion[1.42625, 0., -0.397221, 0.]
Quaternion[1.11157, 0., -0.203991, 0.]
Quaternion[0.98741, 0., -0.0588488, 0.]
Quaternion[0.994677, 0., 0.00153626, 0.]
Quaternion[1.00004, 0., -0.0000249862, 0.]
Quaternion[1., 0., -2.95062×10^-9, 0.]
Quaternion[1., 0., -1.22895×10^-17, 0.]
Quaternion[1., 0., 0., 0.]
```

```
NewtonIterations[f1, Df1, Quaternion[-1., 0., 1., 0.],
  "right"] // Column
```

```
Quaternion[-1., 0., 1., 0.]
Quaternion[-0.756757, 0., 0.540541, 0.]
Quaternion[-0.6375, 0., 0.146216, 0.]
Quaternion[-0.782518, 0., -0.5633, 0.]
Quaternion[-0.654848, 0., -0.172497, 0.]
Quaternion[-0.765795, 0., 0.433337, 0.]
Quaternion[-0.679227, 0., 0.049088, 0.]
Quaternion[-1.42625, 0., -0.397221, 0.]
Quaternion[-1.11157, 0., -0.203991, 0.]
Quaternion[-0.98741, 0., -0.0588488, 0.]
Quaternion[-0.994677, 0., 0.00153626, 0.]
Quaternion[-1.00004, 0., -0.0000249862, 0.]
Quaternion[-1., 0., -2.95062×10^-9, 0.]
Quaternion[-1., 0., -1.22895×10^-17, 0.]
Quaternion[-1., 0., 0., 0.]
```

```
NewtonIterations[f1, Df1, Quaternion[-1., 0., 1., 0.],
  "left"] // Column
```

```
Quaternion[-1., 0., 1., 0.]
Quaternion[-0.756757, 0., 0.540541, 0.]
Quaternion[-0.6375, 0., 0.146216, 0.]
Quaternion[-0.782518, 0., -0.5633, 0.]
Quaternion[-0.654848, 0., -0.172497, 0.]
Quaternion[-0.765795, 0., 0.433337, 0.]
Quaternion[-0.679227, 0., 0.049088, 0.]
Quaternion[-1.42625, 0., -0.397221, 0.]
Quaternion[-1.11157, 0., -0.203991, 0.]
Quaternion[-0.98741, 0., -0.0588488, 0.]
Quaternion[-0.994677, 0., 0.00153626, 0.]
Quaternion[-1.00004, 0., -0.0000249862, 0.]
Quaternion[-1., 0., -2.95062×10^-9, 0.]
Quaternion[-1., 0., -1.22895×10^-17, 0.]
Quaternion[-1., 0., 0., 0.]
```

Example 2

The polynomial $p(x) = x^3 + x$ has a real root 0 and the sphere of zeros $[i]$. Since the polynomial is radially holomorphic, both methods produce the same sequence. Here we would like to call attention to the convergence to the spherical root.

```
f2[X0_, X1_, X2_, X3_] =
  QPower[Quaternion[X0, X1, X2, X3], 3] +
   Quaternion[X0, X1, X2, X3];
Df2 = Function[{X0, X1, X2, X3},
   Evaluate[Map[D[#, X0] &, f2[X0, X1, X2, X3]]]];
```

```
(s1 = NewtonIterations[f2, Df2, Quaternion[1., -1., 1., -1.],
     "right"]) // Last
```

```
Quaternion[-1.46937 × 10^-39, -0.57735, 0.57735, -0.57735]
```

```
NewtonIterations[f2, Df2, Quaternion[-1., 1., 0., 1.],
  "right"] // Last
```

```
Quaternion[2.40741 × 10^-35, 0.707107, 0., 0.707107]
```

```
NewtonIterations[f2, Df2, Quaternion[-1., 2., 3., 4.],
  "right"] // Last
```

```
Quaternion[-8.83524 × 10^-29, 0.371391, 0.557086, 0.742781]
```

As pointed out in Example 3 of [10], the behavior of the Newton methods in case of convergence to values generating a spherical root $[\alpha]$ is clear: if z_0 is the initial guess, then the Newton sequence converges to the root $r \in [\alpha]$ such that $\omega(\underline{r}) = \omega(\underline{z_0})$. This phenomenon can be easily seen from the preceding results or by computing the sign (3) of the vector part of the iterations.

```
W /@ s1 // Column
```

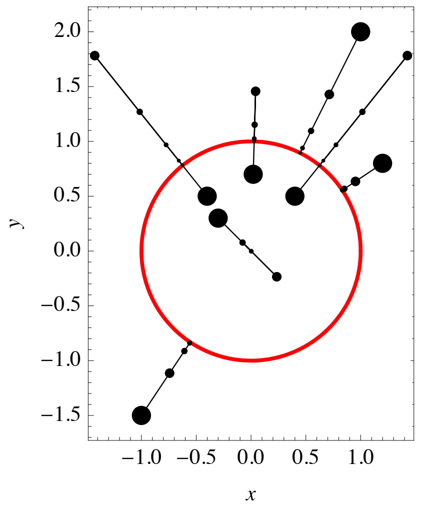
```
Quaternion[0., -0.57735, 0.57735, -0.57735]
Quaternion[0., -0.57735, 0.57735, -0.57735]
Quaternion[0., -0.57735, 0.57735, -0.57735]
Quaternion[0., -0.57735, 0.57735, -0.57735]
Quaternion[0., -0.57735, 0.57735, -0.57735]
Quaternion[0., -0.57735, 0.57735, -0.57735]
Quaternion[0., -0.57735, 0.57735, -0.57735]
Quaternion[0., -0.57735, 0.57735, -0.57735]
Quaternion[0., -0.57735, 0.57735, -0.57735]
Quaternion[0., -0.57735, 0.57735, -0.57735]
```

When the initial guess z_0 is chosen in $\Omega = \{(0, x, y, 0) : x, y \in \mathbb{R}\}$, then all the subsequent iterations belong to Ω.

```
Quaternion[0, x, y, 0] - f2[0, x, y, 0] ** (1 / Df2[0, x, y, 0])
```

$$\text{Quaternion}\left[0, \frac{2x(x^2+y^2)}{-1+3x^2+3y^2}, \frac{2y(x^2+y^2)}{-1+3x^2+3y^2}, 0\right]$$

```
InitialGuess =
  {
   Quaternion[0., 1., 2., 0.],
   Quaternion[0., -1., -1.5, 0.],
   Quaternion[0., -.3, .3, 0.],
   Quaternion[0., .02, .7, 0.],
   Quaternion[0., 1.2, .8, 0.],
   Quaternion[0., .4, .5, 0.],
   Quaternion[0., -.4, .5, 0.]
  };
seqs4D =
  Apply[List, NewtonIterations[f2, Df2, #, "right"] & /@
    InitialGuess, {2}];
seqs2D = Map[Take[#, {2, 3}] &, seqs4D, {2}];
seqs2D = Map[(Transpose@{#1, Range[Length@#1]}) &, seqs2D];

Show[
 Plot[{Sqrt[1 - x^2], -Sqrt[1 - x^2]}, {x, -2, 2},
  PlotRange → All, PlotStyle → {Red}],
 Graphics[
  {Circle[#, .03] & /@
     (List @@@ (Rest@Most@# & /@ InitialGuess)),
   Apply[{AbsolutePointSize[8 / #2], Point[#1]} &,
    seqs2D, {2}],
   Map[Line, Map[First, seqs2D, {2}]]}],
 Frame → True, FrameLabel → {x, y}, AspectRatio → Automatic,
 Ticks → None, Axes → False
]
```



Example 3

Now consider the polynomial $p(x) = x^3 - j x^2 - x + j$ with the three isolated roots $r_1 = 1$, $r_2 = -1$ and $r_3 = j$ (cf. [9], Example 3). This polynomial is not radially holomorphic, which means that we cannot anticipate the behavior of Newton methods unless we choose initial guesses z_0 such that Theorem 2 applies, that is, such that $\omega(\underline{z_0})$ commutes with $\omega(j) = j$. In other words, z_0 must be of the form $a + j\,b$.

```
f3[X0_, X1_, X2_, X3_] =
  QPower[Quaternion[X0, X1, X2, X3], 3] -
   Quaternion[0, 0, 1, 0] **
    QPower[Quaternion[X0, X1, X2, X3], 2] -
   Quaternion[X0, X1, X2, X3] + Quaternion[0, 0, 1, 0];
Df3 = Function[{X0, X1, X2, X3},
   Evaluate[Map[D[#, X0] &, f3[X0, X1, X2, X3]]]];
```

```
NewtonIterations[f3, Df3, Quaternion[1., 0., 2., 0.],
  "right"] // Last
```

```
Quaternion[-7.98722 × 10^-30, 0., 1., 0.]
```

What happens if the assumptions of Theorem 2 are not valid? In fact, as we next illustrate, although the left and right Newton methods do not give the same sequence, we can observe convergence in both cases.

The choice $z_0 = 1 + i + 2\,j$ leads, in both versions, to convergence to the root $r_3 = j$.

```
NewtonIterations[f3, Df3, Quaternion[1., 1., 2., 0.],
  "right"] // Column
```

```
Quaternion[1., 1., 2., 0.]
Quaternion[0.660633, 0.479638, 1.54299, -0.126697]
Quaternion[0.401696, 0.13592, 1.20369, -0.150778]
Quaternion[0.166646, -0.0369085, 0.992312, -0.0718172]
Quaternion[0.00536146, -0.0240926, 0.979014, 0.0111292]
Quaternion[-0.000219403, 0.0000688575, 1.00114, 0.000301303]
Quaternion[-4.97446 × 10^-7, -1.31574 × 10^-7, 1., 3.09445 × 10^-8]
Quaternion[-1.32829 × 10^-12,
 -3.07868 × 10^-14, 1., -1.30901 × 10^-13]
Quaternion[-4.12682 × 10^-24, 3.47749 × 10^-25, 1., -8.18143 × 10^-26]
Quaternion[0., 0., 1., 0.]
```

```
NewtonIterations[f3, Df3, Quaternion[1., 1., 2., 0.],
  "left"] // Column
```

```
Quaternion[1., 1., 2., 0.]
Quaternion[0.660633, 0.678733, 1.38009, 0.0904977]
Quaternion[0.405278, 0.409805, 0.971158, 0.179709]
Quaternion[0.168383, 0.0923196, 0.760201, 0.227761]
Quaternion[-0.0277294, -0.144549, 0.875723, -0.0347233]
Quaternion[0.00497575, 0.0313041, 0.978504, 0.0187528]
Quaternion[-0.000198594,
 -0.00150765, 0.998969, -0.000453679]
Quaternion[4.08499 × 10^-7,
 2.92572 × 10^-6, 0.999999, 1.53546 × 10^-6]
Quaternion[-1.19923 × 10^-12,
 -9.84354 × 10^-12, 1., -2.11736 × 10^-12]
Quaternion[2.14176 × 10^-23, 1.70721 × 10^-22, 1., 6.14242 × 10^-23]
Quaternion[0., 0., 1., 0.]
```

With the choice $z_0 = 1.31 + 2\,i$, the right version of the Newton method converges to the root $r_3 = j$, while the left version converges to $r_1 = 1$.

```
NewtonIterations[f3, Df3, Quaternion[1.31, 2., 0., 0.],
  "right"] // Column
```

```
Quaternion[1.31, 2., 0., 0.]
Quaternion[0.908706, 1.38734, 0.43934, -0.212237]
Quaternion[0.638046, 0.879595, 0.76845, -0.380062]
Quaternion[0.420486, 0.387238, 0.99051, -0.449127]
Quaternion[0.215586, -0.014328, 1.06647, -0.288126]
Quaternion[0.0625773, -0.0814078, 1.03496, -0.0333192]
Quaternion[0.00554934, -0.00499996, 1.005, 0.00839568]
Quaternion[0.0000568226, 0.0000905714, 1.00009, 0.0000566505]
Quaternion[1.01771 × 10^-8, 6.43985 × 10^-9, 1., -1.0288 × 10^-8]
Quaternion[3.29762 × 10^-16, -2.09405 × 10^-16, 1., -1.31078 × 10^-16]
Quaternion[1.47911 × 10^-31, -9.86076 × 10^-32, 1., 1.2326 × 10^-31]
```

```
NewtonIterations[f3, Df3, Quaternion[1.31, 2., 0., 0.],
  "left"] // Column
```

```
Quaternion[1.31, 2., 0., 0.]
Quaternion[0.908706, 1.22532, -0.0502774, 0.103775]
Quaternion[0.671407, 0.631535, -0.0591902, 0.169317]
Quaternion[0.602177, 0.107295, 0.00150777, 0.174004]
Quaternion[0.92336, -0.247162, 0.184631, -0.104646]
Quaternion[0.876118, 0.00103716, -0.0397039, -0.0468976]
Quaternion[1.00287, 0.0069243, 0.0191523, 0.0130866]
Quaternion[0.999358, 0.0000841244, -0.000162339, 0.0000748251]
Quaternion[1., -1.56162 × 10^-7, 3.95316 × 10^-7, -4.20693 × 10^-8]
Quaternion[1., -9.51746 × 10^-14, 1.57162 × 10^-13, 1.93423 × 10^-14]
Quaternion[1., 3.70575 × 10^-26, 5.10885 × 10^-18, -2.89733 × 10^-26]
```

It is interesting that the 4D Newton method (7) gives convergence to the other root $r_2 = -1$, as observed in [9].

Following [9] and [10], consider a function `Iterations` that gives the number of iterations required for each process to converge, within a certain precision, to one of the solutions of the problem under consideration, using z_0 as the initial guess.

```
Iterations[f_, Df_, z0_, method_, eps_, ItMax_] := Module[
  {seq = NewtonIterations[f, Df, z0, method, eps, ItMax],
   nseq},
  nseq = Length@seq;
  If[Last@seq === Null || nseq > ItMax, Null, nseq]
 ]
```

We now consider different initial guesses z_0 by choosing points in special regions $\Omega = \Omega(x, y) \subset \mathbb{R}^4$ and we show density plots of `Iterations`. The white regions that may appear correspond to a choice of $z_0 \in \Omega$ for which the method under consideration does not reach the level of precision `eps` with `ItMax` iterations. The default choices of `eps = 10^-2` and `ItMax = 20` usually lead to realistic plots that require some minutes to be produced. A smoother density can be obtained by increasing the option `PlotPoints`.

```
ViewSolution[f_, Df_, domain_, method_, eps_: 10^-2,
  ItMax_: 20] :=
DensityPlot[Iterations[f, Df, N@First@domain, method,
   eps, ItMax], Evaluate[Sequence @@ (Rest@domain)],
  PlotRange -> {0, ItMax}, PlotPoints → 50,
  AspectRatio -> Automatic, MaxRecursion -> 1,
  FrameTicksStyle -> Directive[7],
  ColorFunctionScaling -> False,
  ColorFunction -> (ColorData["Rainbow"][# / ItMax] &),
  PlotLegends → Automatic]
```

Example 4

We consider again the polynomial $p(x) = x^3 - j\,x^2 - x + j$ of Example 3, whose roots are the isolated roots -1, 1 and j. The following code produces the plots corresponding to the choice of z_0 in one of the following regions:

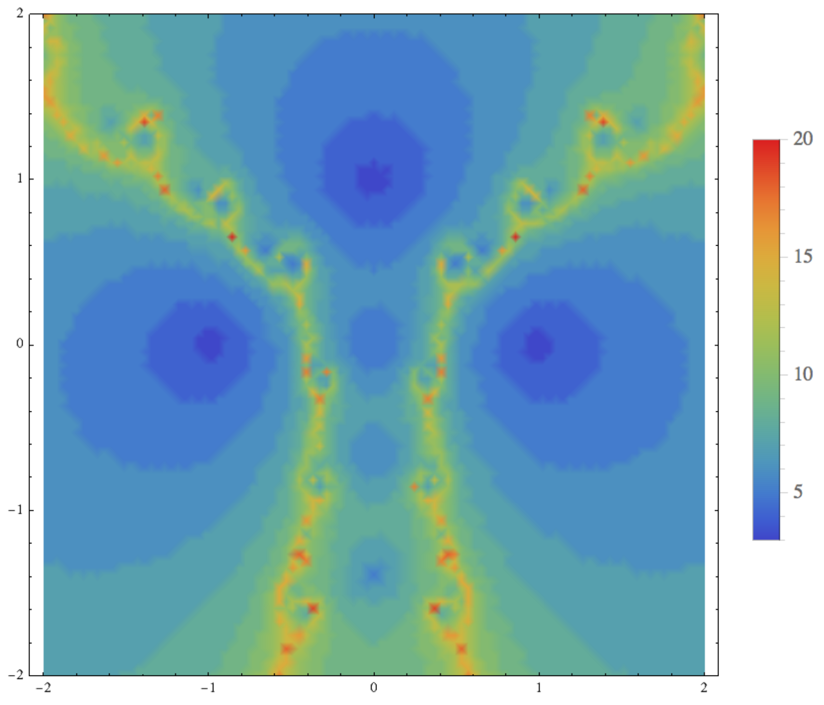
$$\Omega_1 = \{(x, 0, y, 0) : x, y \in [-2, 2]\}\},$$

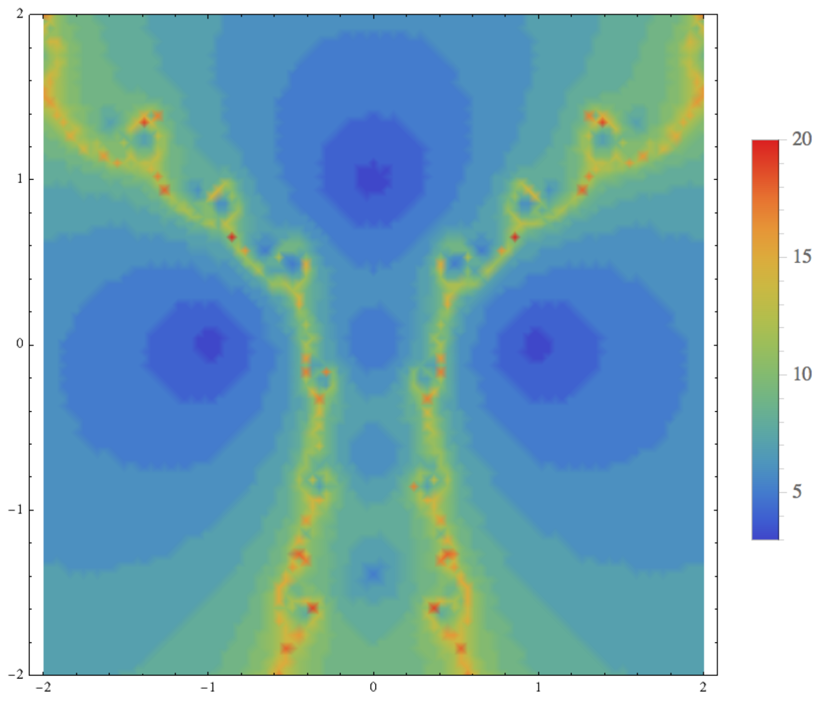
$$\Omega_2 = \{(x, y, 0, 0) : x, y \in [-2, 2]\}\},$$

$$\Omega_3 = \{(0, x, y, 0) : x, y \in [-2, 2]\}\}.$$

As was already pointed out, Theorem 2 can be applied only in Ω_1; this is why both methods produce the same plots in this case.

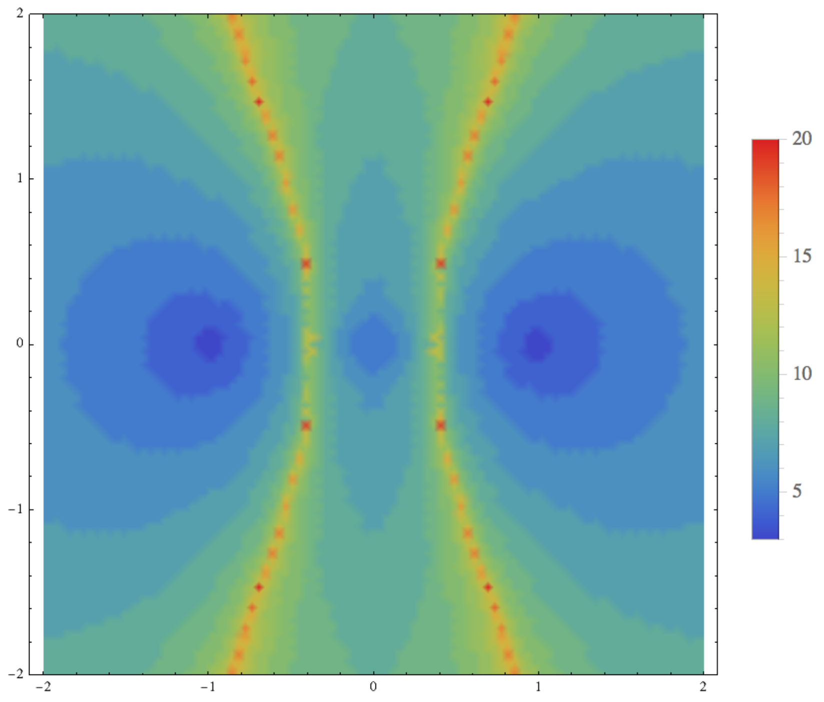
```
Show[ViewSolution[f3, Df3,
  {Quaternion[x, 0, y, 0], {x, -2, 2}, {y, -2, 2}}, "left"]]
Show[ViewSolution[f3, Df3,
  {Quaternion[x, 0, y, 0], {x, -2, 2}, {y, -2, 2}}, "right"]]
```

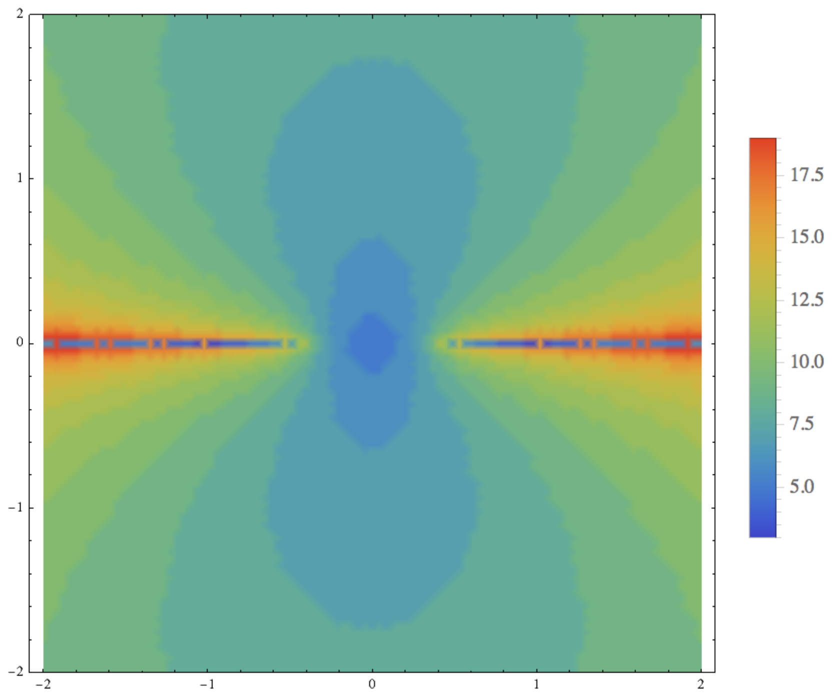




Here is the behavior of the $\mathbb{H}$-Newton methods in Ω_2.

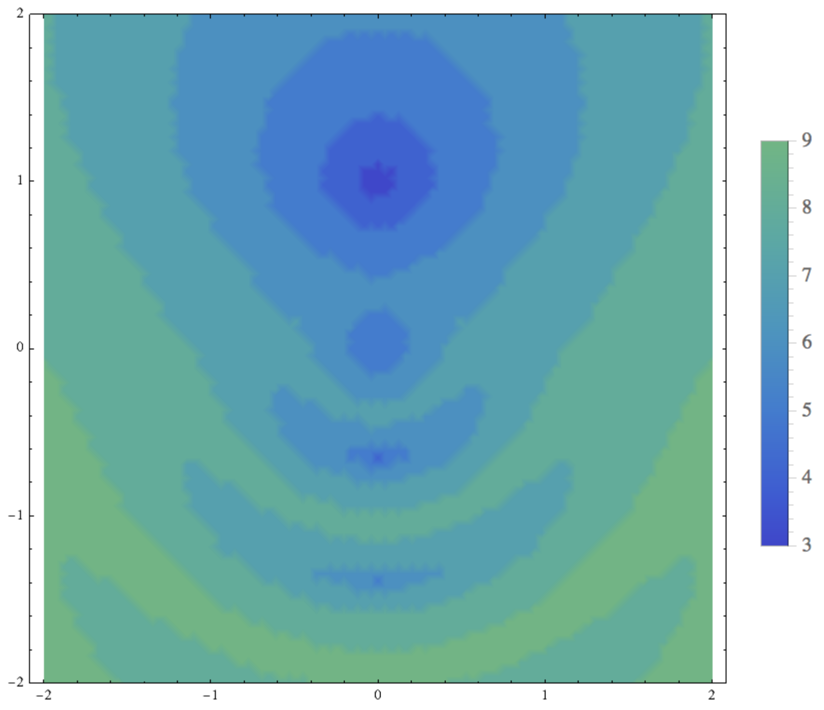
```
Show[ViewSolution[f3, Df3,
  {Quaternion[x, y, 0, 0], {x, -2, 2}, {y, -2, 2}}, "left"]]
Show[ViewSolution[f3, Df3,
  {Quaternion[x, y, 0, 0], {x, -2, 2}, {y, -2, 2}}, "right"]]
```

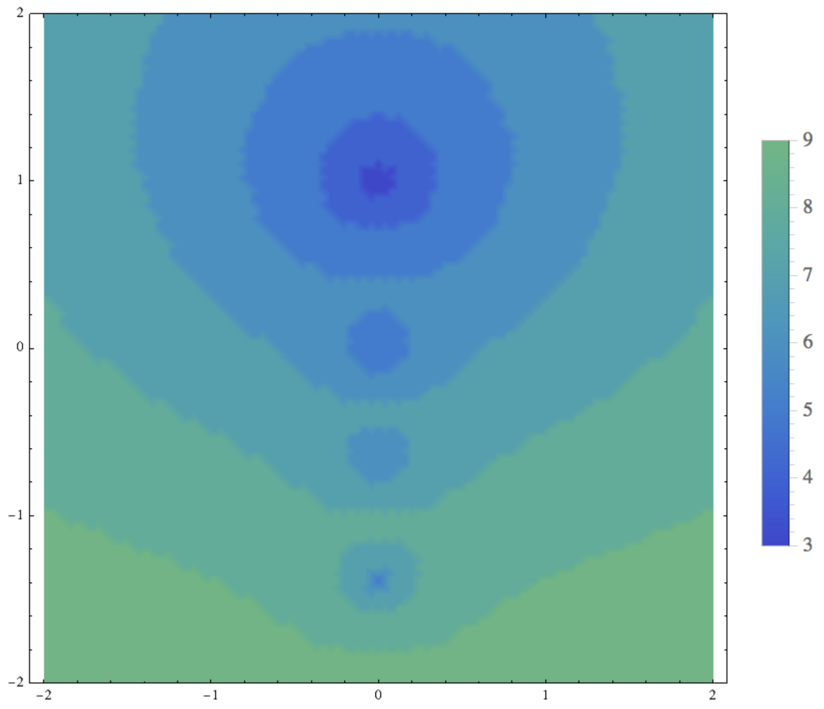




Here is the behavior of the $\mathbb{H}$-Newton methods in Ω_3.

```
Show[ViewSolution[f3, Df3,
  {Quaternion[0, x, y, 0], {x, -2, 2}, {y, -2, 2}}, "left"]]
Show[ViewSolution[f3, Df3,
  {Quaternion[0, x, y, 0], {x, -2, 2}, {y, -2, 2}}, "right"]]
```





▫ Basin of Attraction of a Root

The plots produced by `ViewSolution` give information on the number of iterations required by each of the quaternionic Newton methods to converge within a certain precision to any of the roots of the polynomial under consideration. However, those plots do not give any information about the root and how the convergence occurs. This issue can be easily overcome by plotting the basins of attraction of the roots with respect to the iterative function. More precisely, we introduce a new input parameter in the function `NewtonIterations` with the information of the root for which we want to compute the basin of attraction. A new function `NewtonIterationsSphere` takes into account the existence of spheres of zeros. The functions `Iterations` and `IterationsSphere` give the number of iterations needed to observe convergence to an isolated root or a spherical one, respectively. These functions return `Null` when the corresponding convergence test fails.

```
TestConvergenceSphere[z1_, z2_, eps_] :=
 Abs[Re[z1] - Re[z2]] > eps || Abs[AbsVec[z1] - AbsVec[z2]] > eps

NewtonIterations[f_, Df_, z0_, root_, method_, eps_,
  ItMax_] :=
 NestWhileList[NewtonIterativeFunction[f, Df, #, method] &,
  z0, TestConvergence[#, root, eps] &, 1, ItMax]

NewtonIterationsSphere[f_, Df_, z0_, root_, method_,
  eps_, ItMax_] :=
 NestWhileList[NewtonIterativeFunction[f, Df, #, method] &,
  N[z0], TestConvergenceSphere[#, root, eps] &, 1, ItMax]

Iterations[f_, Df_, z0_, root_, method_, eps_, ItMax_] :=
 Module[
  {seq = NewtonIterations[f, Df, z0, root, method, eps, ItMax],
   nseq}, nseq = Length@seq;
If[Last@seq === Null || nseq > ItMax, Null, nseq]]

IterationsSphere[f_, Df_, z0_, root_, method_, eps_,
  ItMax_] :=
 Module[
  {seq = NewtonIterationsSphere[f, Df, z0, root, method,
     eps, ItMax], nseq}, nseq = Length@seq;
If[Last@seq === Null || nseq > ItMax, Null, nseq]]
```

The functions that plot the basin of attraction of an isolated root `Basin` or a spherical root `BasinSphere` have an input parameter `color` associated to that root. The color coding used is the following: if the initial guess z_0, chosen in a domain $\Omega = \Omega(x, y) \subset \mathbb{R}^4$, causes the process to converge to a certain isolated root r to which the color c_r was associated, then the point z_0 is plotted with the color c_r. For a sphere of zeros $[s]$, all the points z_0 that converge to a point in $[s]$ have the color assigned to $[s]$. Dark shades of a color mean fast convergence, while lighter-colored points lead to slower convergence. As before, white regions mean that the method does not converge.

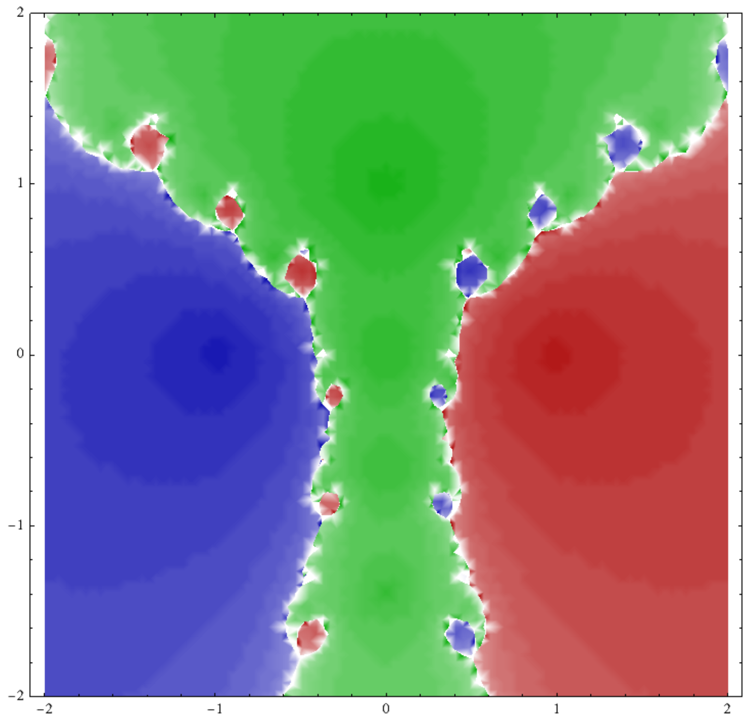
```
Basin[f_, Df_, domain_, root_, method_, color_,
  eps_: 10^-2, ItMax_: 20] :=
DensityPlot[Iterations[f, Df, N@First@domain, root,
   method, eps, ItMax], Evaluate[Sequence @@ (Rest@domain)],
  PlotRange -> {0, ItMax}, PlotPoints → 50,
  AspectRatio -> Automatic, MaxRecursion -> 1,
  FrameTicksStyle -> Directive[7],
  ColorFunctionScaling -> False,
  ColorFunction -> (Lighter[color, # / ItMax] &)]

BasinSphere[f_, Df_, domain_, root_, method_, color_,
  eps_: 10^-2, ItMax_: 20] :=
DensityPlot[IterationsSphere[f, Df, N@First@domain,
   root, method, eps, ItMax],
  Evaluate[Sequence @@ (Rest@domain)],
  PlotRange -> {0, ItMax}, PlotPoints → 50,
  AspectRatio -> Automatic, MaxRecursion -> 1,
  FrameTicksStyle -> Directive[7],
  ColorFunctionScaling -> False,
  ColorFunction -> (Lighter[color, # / ItMax] &)]
```

Example 5

We consider once more the polynomial $p(x) = x^3 - j\,x^2 - x + j$ of Example 4, now from the perspective of the basins of attraction of each of the roots -1, 1 and j. We associate with these roots the colors red, blue and green, respectively, and consider the domains Ω_1, Ω_2 and Ω_3, described in Example 4. The corresponding plots can be obtained as follows (it can take some time to produce the figures).

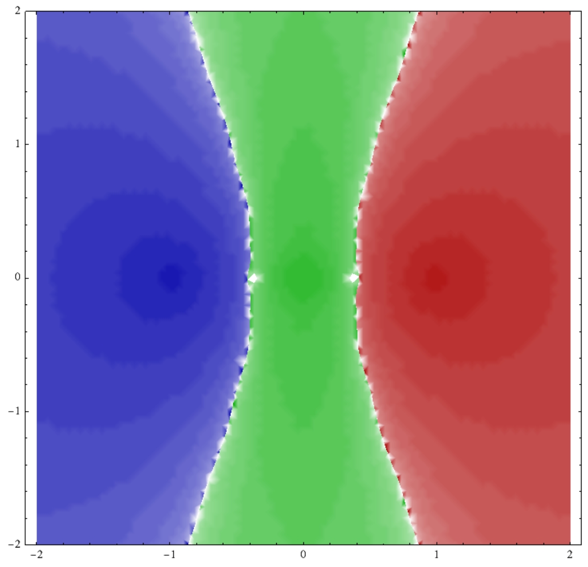
Here are the basins of attraction in Ω_1 (left).

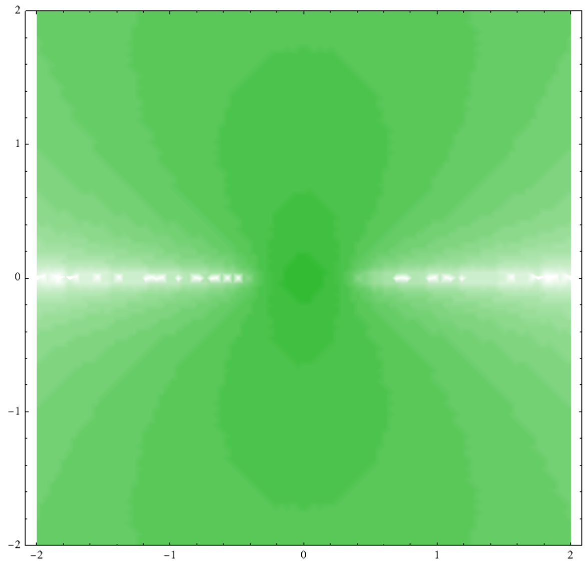
```
Show[Basin[f3, Df3, {Quaternion[x, 0, y, 0], {x, -2, 2},
    {y, -2, 2}}, 1, "left", Darker@Red],
  Basin[f3, Df3, {Quaternion[x, 0, y, 0], {x, -2, 2},
    {y, -2, 2}}, -1, "left", Darker@Blue],
  Basin[f3, Df3, {Quaternion[x, 0, y, 0], {x, -2, 2},
    {y, -2, 2}}, Quaternion[0, 0, 1, 0], "left", Darker@Green],
  PlotRange → All]
```



Here are the basins of attraction in Ω_2 (left and right).

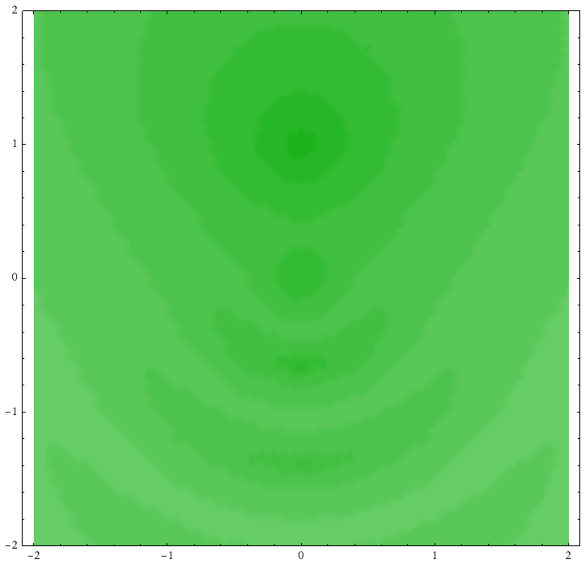
```
Quiet@
 Show[Basin[f3, Df3, {Quaternion[x, y, 0, 0], {x, -2, 2},
     {y, -2, 2}}, 1, "left", Darker@Red],
  Basin[f3, Df3, {Quaternion[x, y, 0, 0], {x, -2, 2},
     {y, -2, 2}}, -1, "left", Darker@Blue],
  Basin[f3, Df3, {Quaternion[x, y, 0, 0], {x, -2, 2},
     {y, -2, 2}}, Quaternion[0, 0, 1, 0], "left",
   Darker@Green], PlotRange → All]
```

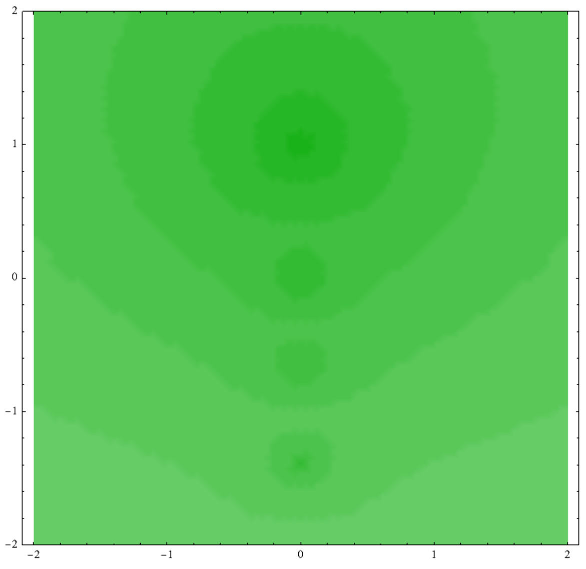




Here are the basins of attraction in Ω_3 (left and right).

```
Show[Basin[f3, Df3, {Quaternion[0, x, y, 0], {x, -2, 2},
   {y, -2, 2}}, 1, "left", Darker@Red],
 Basin[f3, Df3, {Quaternion[0, x, y, 0], {x, -2, 2},
   {y, -2, 2}}, -1, "left", Darker@Blue],
 Basin[f3, Df3, {Quaternion[0, x, y, 0], {x, -2, 2},
   {y, -2, 2}}, Quaternion[0, 0, 1, 0], "left", Darker@Green],
 PlotRange → All]
```



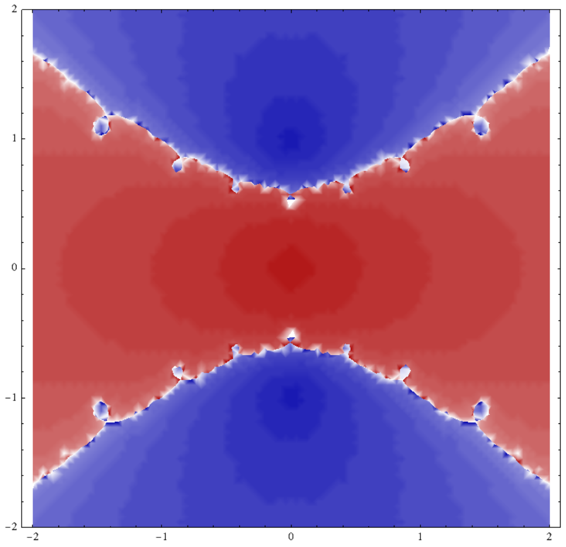


Example 6

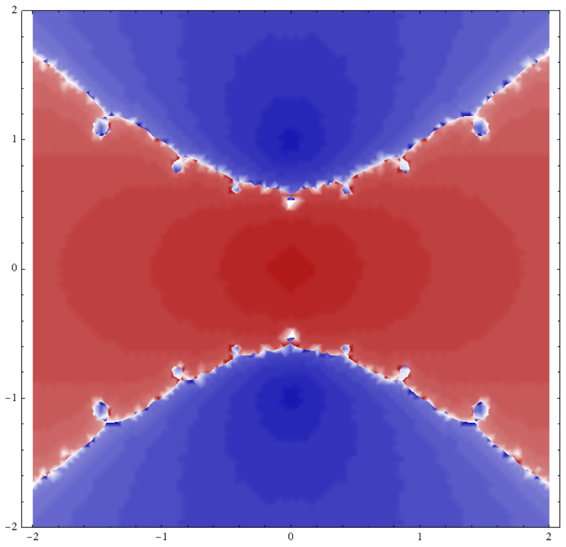
This example concerns the polynomial $p(x) = x^3 + x$ studied in Example 2, which has an isolated root 0 (red) and a sphere of zeros $[i]$ (blue). The corresponding plots can be obtained as follows.

Here are the basins of attraction in Ω_1 (left).

```
Show[Basin[f2, Df2, {Quaternion[x, 0, y, 0], {x, -2, 2},
    {y, -2, 2}}, 0,
   "left", Darker@Red],
 BasinSphere[f2, Df2,
  {Quaternion[x, 0, y, 0], {x, -2, 2}, {y, -2, 2}},
  Quaternion[0, 1, 0, 0], "left", Darker@Blue],
 PlotRange → All]
```

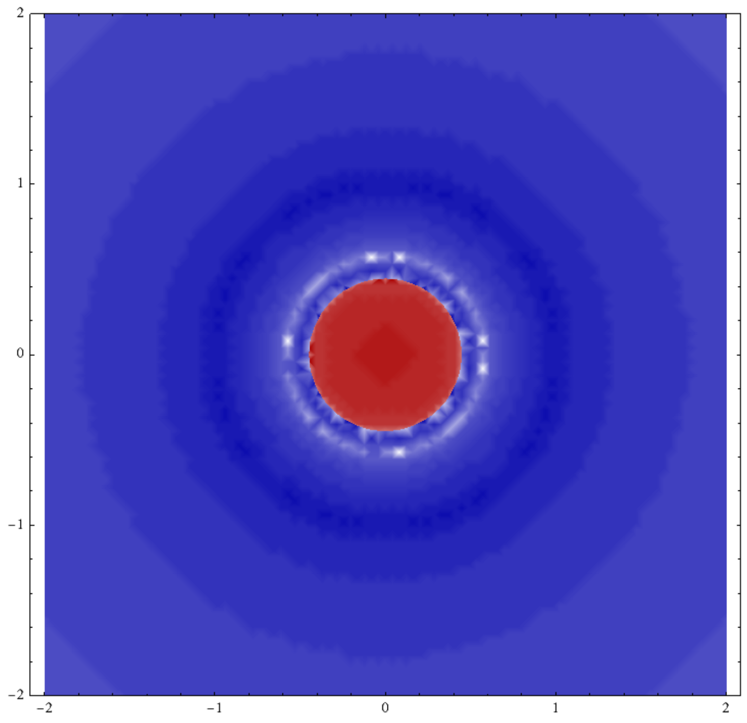


Here are the basins of attraction in Ω_2 (left); as expected, the behavior is similar to that in Ω_1, since $[j] = [i]$.



Here are the basins of attraction in Ω_3 (left).

```
Show[Basin[f2, Df2, {Quaternion[0, x, y, 0], {x, -2, 2},
    {y, -2, 2}}, 0, "left", Darker@Red],
 BasinSphere[f2, Df2,
  {Quaternion[0, x, y, 0], {x, -2, 2}, {y, -2, 2}},
  Quaternion[0, 1, 0, 0], "left", Darker@Blue],
 PlotRange → All]
```



Weierstrass Method

Theoretical Framework

The Weierstrass method is one of the most popular iterative methods for obtaining simultaneously approximations to all the roots of a polynomial with complex coefficients. The method was first proposed by Weierstrass [16] in 1891 and later rediscovered and derived in different ways by Durand [17] in 1960, Dočev [18] in 1962 and Kerner [19] and Prešić [20] in 1966.

Let P be a complex monic polynomial of degree n with roots $\zeta_1, \ldots, \zeta_n$ and let $z_1{}^{(0)}, \ldots, z_n{}^{(0)}$ be n distinct numbers. The classical Weierstrass method for approximating the roots ζ_i is defined by the iterative scheme:

$$z_i{}^{(k+1)} = z_i{}^{(k)} - \frac{P(z_i{}^{(k)})}{\prod_{j=1, j \neq i}^{n} z_i{}^{(k)} - z_j{}^{(k)}}; \; i = 1, \ldots, n; \; k = 0, 1, \ldots. \tag{12}$$

If the roots $\zeta_1, \ldots, \zeta_n$ are distinct and $z_1{}^{(0)}, \ldots, z_n{}^{(0)}$ are sufficiently good initial approximations to these roots, then the method converges at a quadratic rate, as was first proved by Dočev [18]. The iteration procedure (12) computes one approximation at a time based on the already computed approximations. For this reason, it is usually referred to as the *total-step* or *parallel* mode. The convergence of the method can be accelerated by using a variant—the so-called *single-step, serial* or *sequential mode*—that makes use of the most recent updated approximations to the roots as soon as they are available:

$$z_i{}^{(k+1)} = z_i{}^{(k)} - \frac{P(z_i{}^{(k)})}{\prod_{j=1}^{i-1} (z_i{}^{(k)} - z_j{}^{(k+1)}) \prod_{j=i+1}^{n} (z_i{}^{(k)} - z_j{}^{(k)})}; \qquad (13)$$
$$i = 1, \ldots, n; \;\; k = 0, 1, \ldots.$$

In a recent article [11], we adapted the Weierstrass method to the quaternion algebra setting. We refer to [2] and references therein to recall the main concepts and properties of the ring $\mathbb{H}[x]$ of unilateral quaternionic polynomials. In particular, we recall the factorization of polynomials P in $\mathbb{H}[x]$ into linear terms and the relation between zeros and factors of P.

Theorem 3—Factorization into linear terms

Any monic polynomial P of degree $n \geq 1$ in $\mathbb{H}[x]$ *admits a factorization into linear factors; that is, there exist $x_1, \ldots, x_n$ such that*

$$P(x) = (x - x_n) \ldots (x - x_1). \qquad (14)$$

Theorem 4—Zeros from factors

Consider a polynomial P whose factor terms are $x_1, \ldots, x_n$; that is, P admits a factorization of the form (14)*. If the similarity classes $[x_i]$, $i = 1, \ldots, n$, are distinct, then P has exactly n zeros $\zeta_1, \ldots, \zeta_n$, which are given by:*

$$\zeta_i = \overline{\mathcal{R}}(x_i)\, x_i\, \mathcal{R}(x_i)\Big)^{-1}, \; i = 1, \ldots, n \qquad (15)$$

where

$$\mathcal{R}_i(x) = \begin{cases} 1, & \textit{if } i = 1 \\ (x - x_{i-1}) \ldots (x - x_1), & \textit{otherwise} \end{cases} \qquad (16)$$

Weierstrass Algorithm

Following the idea of the Weierstrass method in its sequential version (13), the next results show how to obtain sequences converging, at a quadratic rate, to the factor terms in (14) of a given polynomial P. Moreover, by making use of Theorem 4, it is possible to construct sequences converging quadratically to the roots of P.

Theorem 5 ([11])

Let P be a polynomial in $\mathbb{H}[x]$ *of degree n with simple roots and, for $i = 1, \ldots,$ n; $k = 0, 1, 2, \ldots$, let*

$$z_i{}^{(k+1)} = z_i{}^{(k)} - \left(\overline{\mathcal{L}_i{}^{(k)}}\, P\, \overline{\mathcal{R}_i{}^{(k)}}\right)\left(z_i{}^{(k)}\right)\left(C_i{}^{(k)}\, z_i{}^{(k)}\right)^{-1}, \qquad (17)$$

where

$$\mathcal{L}_i^{(k)}(x) := \left(x - z_n^{(k)}\right)\left(x - z_{n-1}^{(k)}\right)\ldots\left(x - z_{i+1}^{(k)}\right), \quad i = 1, \ldots, n-1 \text{ and } \mathcal{L}_n^{(k)}(x) := 1, \tag{18}$$

$$\mathcal{R}_i^{(k)}(x) := \left(x - z_{i-1}^{(k+1)}\right)\ldots\left(x - z_2^{(k+1)}\right)\ldots\left(x - z_1^{(k+1)}\right), \quad i = 2, \ldots, n \text{ and } \mathcal{R}_1^{(k)}(x) := 1 \tag{19}$$

and

$$C_i^{(k)}(x) := \prod_{j=1}^{i-1} \Psi_{z_j^{(k+1)}}(x) \prod_{j=i+1}^{n} \Psi_{z_j^{(k)}}(x), \tag{20}$$

with Ψ_q *denoting the characteristic polynomial of* q*, that is,* $\Psi_q = (x-q)(x-\overline{q})$*. If the initial approximations* $z_i^{(0)}$ *are sufficiently close to the factor terms* x_i *in a factorization of* P *in the form (*14*), then the sequences* $\{z_i^{(k)}\}$ *converge quadratically to* x_i*. Moreover, the sequences* $\left\{\zeta_i^{(k)}\right\}$ *defined by*

$$\zeta_i^{(k+1)} = \overline{\mathcal{R}_i^{(k)}}(z_i^{(k+1)})\, z_i^{(k+1)}\left(\mathcal{R}_i^{(k)}(z_i^{(k+1)})\right)^{-1}, \; k = 1, \ldots, n \tag{21}$$

converge quadratically to the roots ζ_i *of* P*.*

The functions $\overline{\mathcal{L}_i^{(k)}}$, $\overline{\mathcal{R}_i^{(k)}}$ and $C_i^{(k)}$ are implemented as the functions `Polynomial`$\mathcal{L}$, `Polynomial`$\mathcal{R}$ and `Polynomial`C, respectively. The support file `QPolynomial` associated with [2] needs to be loaded.

```
<< QPolynomial`

Polynomialℒ[fact_, i_] := Module[
  {n = Length@fact},
  Which[
   i < n - 1,
   (NonCommutativeMultiply @@
     (Polynomial[1, -Conjugate@#] & /@ (Drop[fact, i]))),
   i == n - 1, Polynomial[1, -Conjugate@#] & @@
    (Drop[fact, i]),
   i == n, Polynomial[1]
   ]
  ]

Polynomialℛ[fact_, i_] := Module[
  {n = Length@fact},
  Which[
   i == 1, Polynomial[1],
   i == 2, Polynomial[1, -Conjugate@#] & @@
    (Drop[fact, i - n - 1]),
   i ≤ n,
   (NonCommutativeMultiply @@
     (Polynomial[1, -Conjugate@#] & /@
       (Drop[fact, i - n - 1])))
   ]
  ]
```

```
PolynomialC[factors : {_, _}, i_] :=
 Eval[CharacteristicPolynomial @@ Drop[factors, {i}]] @
  factors⟦i⟧

PolynomialC[factors_, i_] := Module[
  {factori = factors⟦i⟧},
  NonCommutativeMultiply @@
   Map[Eval[#] @ factori &,
    CharacteristicPolynomial /@ Drop[factors, {i}]]
 ]
```

The iterative functions associated with (17) and (21) are built into the `WeierstrassIteration` function.

```
WeierstrassIteration[pol_, fact_, ___] := Module[
  {n = Length@fact, factors = fact, roots = {}, RPoli,
   RPoliz},
  For[i = 1, i ≤ n, i++,
   RPoli = PolynomialR[factors, i];
   factors⟦i⟧ =
    factors⟦i⟧ -
     Eval[PolynomialL[factors, i] ** pol ** RPoli][
       factors⟦i⟧] ** (1 / PolynomialC[factors, i]);
   RPoliz = Eval[RPoli][factors⟦i⟧];
   AppendTo[roots, RPoliz ** factors⟦i⟧ ** (1 / RPoliz)];
  ];
  {pol, factors, roots}
 ]
```

The quaternionic Weierstrass iterative method is implemented in the function `WeierstrassMethod`.

```
TestConvergenceW[pol_, x1_, x2_, eps_] :=
  Max@ (Abs[Norm /@ (Last@x1) - Norm /@ (Last@x2)]) > eps ||
   Max@ (Abs[Re /@ (Last@x1) - Re /@ (Last@x2)]) > eps ||
   Max@ (Norm /@ Eval[pol, Last@x2]) > eps;

WeierstrassMethod[pol_Polynomial, fact_, eps_: 10.^-12,
   max_Integer: 50] :=
  Module[{out, last, errorroots, errorfact},
   If[Length@pol == Length@fact + 1,
    out = NestWhileList[WeierstrassIteration @@ # &,
      {pol, fact}, TestConvergenceW[pol, #1, #2, eps] &,
      2, max];
    last = Take[Reverse@out, {1, 2}];
    errorroots =
     Max[Max@Abs[Re@ (last⟦1, 3⟧) - Re /@ (last⟦2, 3⟧)],
      Max@ (Abs[Norm /@ (last⟦1, 3⟧) - Norm /@ (last⟦2, 3⟧)])];
    {If[errorroots < eps, "Convergence", "Divergence"],
     Length@out - 1, N@errorroots, N@last⟦1, 3⟧,
     N@last⟦1, 2⟧},
    Message[WeierstrassMethod::Dimensions]
   ]
  ];
```

The usual convergence test $\max_i |\zeta_i^{(k+1)} - \zeta_i^{(k)}| < \varepsilon_1$ has been replaced in `TestCon⋱ vergenceW` by

$$\max_i \left|\operatorname{Re}\left(\zeta_i^{(k+1)}\right) - \operatorname{Re}\left(\zeta_i^{(k)}\right)\right| < \varepsilon_1 \text{ and} \max_i \left| \left|\zeta_i^{(k+1)}\right| - \left|\zeta_i^{(k)}\right| \right| < \varepsilon_1$$

in order to let the function `WeierstrassMethod` recognize a sphere of zeros. Since we also include a test on the value of $\max_i \left|P\left(\zeta_i^{(k+1)}\right)\right|$, there is no risk of misidentifying an isolated root.

Example 7

We consider now the application of the Weierstrass method to the computation of the roots of the polynomial $p(x) = x^3 - j x^2 - x + j$ of Example 3, which we recall are $r_1 = 1$, $r_2 = -1$ and $r_3 = j$. All of the initial approximations $z_1^{(0)}$, $z_2^{(0)}$ and $z_3^{(0)}$ have to lie in distinct congruence classes.

```
WeierstrassMethod[Polynomial[1, Quaternion[0, 0, -1, 0],
   -1, Quaternion[0, 0, 1, 0]],
  {1., 2., Quaternion[1., 0., 1., 0.]}]
```

```
{Convergence, 6, 1.66533×10^-15, {Quaternion[1., 0., 0., 0.],
  Quaternion[-1., 0., -1.97215×10^-31, 0.],
  Quaternion[-1.83671×10^-40, 0., 1., 0.]},
 {Quaternion[1., 0., 0., 0.],
  Quaternion[-1., 0., -1.97215×10^-31, 0.],
  Quaternion[-1.83671×10^-40, 0., 1., 0.]}}
```

Some explanation of the output is needed. The first entry indicates the convergence or divergence of the method. The second entry is the error in the approximations to the zeros. The last two entries contain approximations to the roots and factors terms. Since there are two real roots and just one nonreal root, the roots and factor terms coincide.

Example 8

Our next test example is a polynomial that also fulfills the assumptions of Theorem 5 and has simple zeros (see [11], Example 1). First, we check that the polynomial

$$p(x) = (x + 2\,i)\,(x + 1 + k)\,(x - 2)\,(x - 1)\,(x - 2 + j)\,(x - 1 + i) \tag{22}$$

has the roots

$$\begin{aligned} &\zeta_1 = 1 - i,\ \zeta_2 = 2 - \frac{2}{3} i - \frac{1}{3} j + \frac{2}{3} k,\ \zeta_3 = 1, \zeta_4 = 2, \\ &\zeta_5 = -1 - \frac{29}{39} i + \frac{14}{39} j - \frac{22}{39} k,\ \zeta_6 = -\frac{224}{113} i - \frac{30}{113} k . \end{aligned} \tag{23}$$

Recall that the function `ZerosFromChain` included in `QPolynomial` can be used.

```
factors = {Quaternion[1, -1, 0, 0], Quaternion[2, 0, -1, 0],
   1, 2, Quaternion[-1, 0, 0, -1], Quaternion[0, -2, 0, 0]};
zeros = ZerosFromChain[factors]
```

$$\left\{\text{Quaternion}[1, -1, 0, 0],\ \text{Quaternion}\left[2, -\frac{2}{3}, -\frac{1}{3}, \frac{2}{3}\right],\ 1, 2,\ \text{Quaternion}\left[-1, -\frac{29}{39}, \frac{14}{39}, -\frac{22}{39}\right],\ \text{Quaternion}\left[0, -\frac{224}{113}, 0, -\frac{30}{113}\right]\right\}$$

The Weierstrass method applied to the extended form of *p* produces the following results.

```
WeierstrassMethod[PolynomialFromChain[factors],
 {Quaternion[0.5, 0., 0., 0.], Quaternion[1.5, 0., -1., 0.],
  Quaternion[1.5, 1., -1., 1.], Quaternion[1.5, 1., -1., 0.],
  Quaternion[-0.5, 0., 0., 0.],
  Quaternion[-1., -2., 0., 0.]}]
```

$$\{\text{Convergence}, 22, 9.10383\times10^{-14},$$
$$\{\text{Quaternion}[1., -2.93099\times10^{-15}, 4.64073\times10^{-16}, -7.10543\times10^{-17}], \text{Quaternion}[2., 2.07834\times10^{-14}, -3.9968\times10^{-15}, -6.4837\times10^{-15}],$$
$$\text{Quaternion}[-1., -0.74359, 0.358974, -0.564103],$$
$$\text{Quaternion}[1., -1., 3.31679\times10^{-15}, 3.44169\times10^{-15}],$$
$$\text{Quaternion}[2., -0.666667, -0.333333, 0.666667], \text{Quaternion}[-9.71011\times10^{-16}, -1.9823, -6.245\times10^{-17}, -0.265487]\},$$
$$\{\text{Quaternion}[1., -2.93099\times10^{-15}, 4.64073\times10^{-16}, -7.10543\times10^{-17}], \text{Quaternion}[2., 2.07834\times10^{-14}, -3.9968\times10^{-15}, -6.4837\times10^{-15}],$$
$$\text{Quaternion}[-1., -0.74359, 0.358974, -0.564103],$$
$$\text{Quaternion}[1., -0.801865, -0.540793, -0.254079],$$
$$\text{Quaternion}[2., 0.545455, -0.818182, -0.181818], \text{Quaternion}[-1.04668\times10^{-15}, -2., 2.49756\times10^{-14}, -2.61124\times10^{-15}]\}\}$$

```
N[zeros]
```

```
{Quaternion[1., -1., 0., 0.],
 Quaternion[2., -0.666667, -0.333333, 0.666667], 1.,
 2., Quaternion[-1., -0.74359, 0.358974, -0.564103],
 Quaternion[0., -1.9823, 0., -0.265487]}
```

The convergence to the roots is in a order different from the one given in (22) because the convergence to the factor terms also occurs in a sequence different from the one given in (23).

Example 9

The polynomial $p(x) = x^3 + (1 + j)\,x^2 + x + 1 + j$ has an isolated root $1 + j$ and a sphere of zeros $[i]$. The assumptions of Theorem 5 do not apply to this polynomial, but we can observe convergence to the roots as we increase the precision of the computations. When a polynomial has a spherical root, two of its factor terms are in the same congruence class. Therefore, as the iteration proceeds, the values $Q_i^{(k)}(z_i^{(k)})$ in (17) become close to zero and some care is required.

```
WeierstrassMethod[Polynomial[1, Quaternion[1, 0, 1, 0],
   1, Quaternion[1, 0, 1, 0]],
  {Quaternion[-1., 1., 0, 0], Quaternion[2, 0, 0, 0],
   Quaternion[1, 0, 0, 0]}]

{Divergence, 50, 1.82203×10^-8, {Quaternion[
    -9.42063×10^-9, 0.927843, -0.369866, -0.0480275],
   Quaternion[-1., -1.11022×10^-16, -1., -5.55112×10^-17],
   Quaternion[-1.13651×10^-8, 0.572246, -0.779212, 0.255662]},
  {Quaternion[-9.42063×10^-9, 0.927843, -0.369866,
    -0.0480275], Quaternion[-1., -0.559839,
    -0.236198, -0.794224], Quaternion[
    -1.13651×10^-8, -0.368004, -0.393937, 0.842251]}}
```

Using the usual precision, it was not possible to reach the required 10^{-12} tolerance. However, performing the calculations with more decimal places causes a fast convergence, under the same assumptions.

```
out = WeierstrassMethod[
   Polynomial[1, Quaternion[1, 0, 1, 0], 1,
    Quaternion[1, 0, 1, 0]],
   N[{Quaternion[-1, 1, 0, 0], Quaternion[2, 0, 0, 0],
     Quaternion[1, 0, 0, 0]}, 200]]

{Convergence, 10, 6.9419×10^-20, {Quaternion[
    1.89705×10^-46, 0.927843, -0.369866, -0.0480276],
   Quaternion[-1., -2.88136×10^-55, -1., -9.13966×10^-56],
   Quaternion[0., 0.788846, 0.29268, -0.540426]},
  {Quaternion[1.89705×10^-46, 0.927843,
    -0.369866, -0.0480276],
   Quaternion[-1., -0.559839, -0.236198, -0.794224],
   Quaternion[0., -0.368004, -0.393937, 0.842251]}}
```

The spherical root can be identified at once by observing that, up to the required precision, we have $[\zeta_1] = [\zeta_3]$.

```
ξ = out[[5]];

Abs[Re[ξ⟦1⟧] - Re[ξ⟦3⟧]]

1.89705 × 10^-46

Abs[Norm[ξ⟦1⟧] - Norm[ξ⟦3⟧]]

0.
```

Conclusion

This is the second article on several computational aspects of polynomials in the ring $\mathbb{H}[x]$. One can find in the literature methods for numerically approximating the zeros of quaternionic polynomials based on the use of complex techniques, but numerical methods relying on quaternion arithmetic remain scarce, with the exceptions of the Newton and Weierstrass methods discussed in this article. We developed several functions to implement those methods and we also added some visualization tools.

Acknowledgments

Research at the Centre of Mathematics (CMAT) was financed by Portuguese Funds through FCT - Fundação para a Ciência e a Tecnologia, within the Project UID/-MAT/00013/2013. Research at the Economics Politics Research Unit (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 European Regional Development Fund (ERDF) through the Operational Programme on "Competitiveness and Internationalization - COMPETE 2020" under the PT2020 Partnership Agreement.

References

[1] I. Niven, "Equations in Quaternions," *The American Mathematical Monthly*, **48**(10), 1941 pp. 654–661. www.jstor.org/stable/2303304.

[2] M. I. Falcão, F. Miranda, R. Severino, and M. J. Soares, "Computational Aspects of Quaternionic Polynomials: Part 1," *The Mathematica Journal*, **20**(4), 2018. doi.org/10.3888/tmj.20-4.

[3] R. Farouki, G. Gentili, C. Giannelli, A. Sestini and C. Stoppato, "A Comprehensive Characterization of the Set of Polynomial Curves with Rational Rotation-Minimizing Frames," *Advances in Computational Mathematics*, **43**(1), 2017 pp. 1–24. doi.org/10.1007/s10444-016-9473-0.

[4] R. Pereira, P. Rocha and P. Vettori, "Algebraic Tools for the Study of Quaternionic Behavioral Systems," *Linear Algebra and Its Applications*, **400**, 2005 pp. 121–140. doi.org/10.1016/j.laa.2005.01.008.

[5] R. Serôdio, E. Pereira and J. Vitória, "Computing the Zeros of Quaternion Polynomials," *Computers and Mathematics with Applications*, **42**(8-9) 2001 pp. 1229–1237. doi.org/10.1016/S0898-1221(01)00235-8.

[6] S. De Leo, G. Ducati, and V. Leonardi, "Zeros of Unilateral Quaternionic Polynomials," *The Electronic Journal of Linear Algebra*, **15**(1), 2006 pp. 297–313. doi.org/10.13001/1081-3810.1240.

[7] D. Janovská and G. Opfer, "Computing Quaternionic Roots by Newton's Method," *Electronic Transactions on Numerical Analysis*, **26**, 2007 pp. 82–102.

[8] D. Janovská and G. Opfer, "A Note on the Computation of All Zeros of Simple Quaternionic Polynomials," *SIAM Journal on Numerical Analysis*, **48**(1), 2010 pp. 244–256. doi.org/10.1137/090748871.

[9] M. I. Falcão, "Newton Method in the Context of Quaternion Analysis," *Applied Mathematics and Computation*, **236**, 2014 pp. 458–470. doi.org/10.1016/j.amc.2014.03.050.

[10] F. Miranda and M. I. Falcão, "Modified Quaternion Newton Methods," in *Computational Science and Its Applications (ICCSA 2014)*, Guimarães, Portugal, *Lecture Notes in Computer Science*, **8579** (B. Murgante et al., eds.), Berlin, Heidelberg: Springer, 2014 pp. 146–161. doi.org/10.1007/978-3-319-09144-0_ 11.

[11] M. I. Falcão, F. Miranda, R. Severino and M. J. Soares, "Weierstrass Method for Quaternionic Polynomial Root-Finding," *Mathematical Methods in the Applied Sciences*, 2017 pp. 1–15. doi:10.1002/mma.4623.

[12] M. I. Falcão and F. Miranda, "Quaternions: A Mathematica Package for Quaternionic Analysis," in *Computational Science and Its Applications (ICCSA 2011)*, *Lecture Notes in Computer Science*, **6784** (B. Murgante, O. Gervasi, A. Iglesias, D. Taniar and B. O. Apduhan, eds.), Berlin, Heidelberg: Springer, 2011 pp. 200–214. doi:10.1007/978-3-642-21931-3_17.

[13] F. Miranda and M. I. Falcão. "QuaternionAnalysis Mathematica Package." w3.math.uminho.pt/QuaternionAnalysis.

[14] K. Gürlebeck, K. Habetha and W. Sprössig, *Holomorphic Functions in the Plane and n-Dimensional Space,* Basel: Birkhäuser, 2008. doi.org/10.1007/978-3-7643-8272-8.

[15] B. Kalantari, "Algorithms for Quaternion Polynomial Root-Finding," *Journal of Complexity*, **29**(3–4) 2013 pp. 302–322. doi.org/10.1016/j.jco.2013.03.001.

[16] K. Weierstrass, "Neuer Beweis des Satzes, dass jede ganze rationale Function einer Veränderlichen dargestellt werden kann als ein Product aus linearen Functionen derselben Veränderlichen," in *Sitzungsberichte der Königlich Preussischen Akademie der Wissenschaften zu Berlin*, 1891.

[17] E. Durand, *Solutions numériques des équations algébriques. Tome I: Equations du type F(x); racines d'un polynôme,* Paris: Masson,1960.

[18] K. Dočev, "A Variant of Newton's Method for the Simultaneous Approximation of All Roots of an Algebraic Equation," *Fiziko-Matematichesko Spisanie. Bulgarska Akademiya na Naukite*, **5**(38), 1962 pp. 136–139.

[19] I. O. Kerner, "Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen," *Numerische Mathematik*, **8**(3), 1966 pp. 290–294. doi.org/10.1007/BF02162564.

[20] S. B. Prešić, "Un procédé itératif pour la factorisation des polynômes," *Comptes Rendus de l'Académie des Sciences Paris Série A*, **262**, 1966 pp. 862–863.

M. I. Falcão, F. Miranda, R. Severino and M. J. Soares, "Computational Aspects of Quaternionic Polynomials: Part 2," *The Mathematica Journal*, 2018. dx.doi.org/doi:10.3888/tmj.20-5.

Additional Material

1. The package `QuaternionAnalysis`.

Available at: w3.math.uminho.pt/QuaternionAnalysis

2. The file `QPolynomial.m`.

Available at: www.mathematica-journal.com/data/uploads/2018/05/QPolynomial.m

About the Authors

M. Irene Falcão is an associate professor in the Department of Mathematics and Applications of the University of Minho. Her research interests are numerical analysis, hypercomplex analysis and scientific software.

Fernando Miranda is an assistant professor in the Department of Mathematics and Applications of the University of Minho. His research interests are differential equations, quaternions and related algebras and scientific software.

Ricardo Severino is an assistant professor in the Department of Mathematics and Applications of the University of Minho. His research interests are dynamical systems, quaternions and related algebras and scientific software.

M. Joana Soares is an associate professor in the Department of Mathematics and Applications of the University of Minho. Her research interests are numerical analysis, wavelets mainly in applications to economics, and quaternions and related algebras.

M. Irene Falcão
CMAT - Centre of Mathematics
DMA - Department of Mathematics and Applications
University of Minho
Campus de Gualtar, 4710-057 Braga
Portugal
mif@math.uminho.pt

Fernando Miranda
CMAT - Centre of Mathematics
DMA - Department of Mathematics and Applications
University of Minho
Campus de Gualtar, 4710-057 Braga
Portugal
fmiranda@math.uminho.pt

Ricardo Severino
DMA - Department of Mathematics and Applications
University of Minho
Campus de Gualtar, 4710-057 Braga
Portugal
ricardo@math.uminho.pt

M. Joana Soares
NIPE - Economics Politics Research Unit
DMA - Department of Mathematics and Applications
University of Minho
Campus de Gualtar, 4710-057 Braga
Portugal
jsoares@math.uminho.pt