Symbolic Computations Over the Algebra of Coquaternions

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 - Centre for Research in Economics and Management, University of Minho, Portugal
[c] DMAT - Department of Mathematics, University of Minho, Portugal

Information

Keywords:
Coquaternions, polynomial computation, symbolic computation.

Original publication:
4th International Conference on Numerical and Symbolic Computation: Developments and Applications – SYMCOMP 2019 Proceeding, 2019, 141-155, ISBN: 978-989-99410-5-2

Abstract

Coquaternions, introduced by Sir James Cockle in 1849, form a four dimensional real algebra generalizing complex numbers. In recent years one can observe an emerging interest among mathematicians and physicists on the study of these numbers. In this work we present a Mathematica package for implementing the algebra of coquaternions. This package provides the basic mathematical tools necessary for manipulating coquaternions and coquaternionic polynomials, reflecting, in its present form, the recent interests of the authors in the area.

1 Introduction

Quaternions, introduced in 1843 by the Irish mathematician William Rowan Hamilton (1805-1865) as a generalization of complex numbers [14], have become a powerful tool for solving problems in almost all applied sciences [17], unlocking also new approaches in many branches of applied mathematics. This increasing interest in using quaternions has motivated the emergence of several software packages to perform computations in the algebra of the real quaternions (see, for example, [5, 9, 10, 21]), or more generally, in Clifford Algebras (see [1, 2] and the references therein for details).

In 1849, the English mathematician James Cockle, introduced another *system of quadruple algebra* [4], using the term *coquaternion* to refer to elements of such system. Although coquaternions, also known in the literature as split quaternions, are not as popular as the Hamilton's quaternions, one can say that recently they have also been an active research area [3, 13, 16, 19, 20]. Despite this, and as far as we know, there are no computational tools specially design to work with coquaternions and polynomial computations.

This paper describes a Mathematica add-on application `Coquaternions` whose main purpose is to define arithmetic for coquaternions. Drawing on our past experience in developing the package `QuaternionAnalysis` [5], we present a collection of functions providing the basic mathematical tools for computations over the coquaternion algebra.

This package is a fundamental tool supporting the recent interests of the authors in the area [6, 7, 12] and it should be considered work in progress in its present form. The actual version of the package is available at the webpage http://w3.math.uminho.pt/Coquaternions.

2 The Algebra of Coquaternions

Let $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$ be an orthonormal basis of the Euclidean vector space $\mathbb{R}^4$. The algebra of real coquaternions, which we denote by $\mathbb{H}_{\mathsf{coq}}$, is generated by the product given according to the following rules

$$\mathbf{i}^2 = -1, \ \mathbf{j}^2 = \mathbf{k}^2 = 1, \ \ \mathbf{ij} = -\mathbf{ji} = \mathbf{k}.$$

In the `Coquaternions` package, a coquaternion $\mathsf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ is an object of the form `Coquaternion[q0,q1,q2,q3]`, whose entries are numeric quantities or symbols (in such case the package assumes that all symbols represent real numbers).

The package adds rules to `Plus`, `Minus`, `Times`, `Power` and `NonCommutativeMultiply` to make it easy to perform basic arithmetic operations. After loading the package

```
In[1]:= <<Coquaternions`
```

and defining two general coquaternions $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ and $p = p_0 + p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k}$, one can carry out operations on coquaternions in a very simple way:

```
In[2]:= q=Coquaternion[q0,q1,q2,q3]; p=Coquaternion[p0,p1,p2,p3];

In[3]:= q+2p                                                    (*Plus and Times*)

Out[3]= Coquaternion[q0 + 2p0, q1 + 2p1, q2 + 2p2, q3 + 2p3]

In[4]:= q**p                                           (*NonCommutativeMultiply*)

Out[4]= Coquaternion[q0p0 − q1p1 + q2p2 + q3p3, q0p1 + q1p0 − q2p3 + q3p2, q0p2 − q1p3 + q2p0 + q3p1,
            q0p3 + q1p2 − q2p1 + q3p0]

In[5]:= Power[Coquaternion[q0,q1,q2,q3],2]                        (*Integer Power*)

Out[5]= Coquaternion[q0^2 − q1^2 + q2^2 + q3^2, 2q0q1, 2q0q2, 2q0q3]
```

We introduce now some basic definitions in $\mathbb{H}_{\mathsf{coq}}$. Given $\mathsf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \in \mathbb{H}_{\mathsf{coq}}$, its *conjugate* $\overline{\mathsf{q}}$ is defined as $\overline{\mathsf{q}} = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k}$; the number q_0 is called the *real part* of q and denoted by $\mathrm{Re}\,\mathsf{q}$ and the *vector part* of q, denoted by $\underline{\mathsf{q}}$, is given by $\underline{\mathsf{q}} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$. We denote by $\mathrm{tr}\,\mathsf{q}$ and call *trace* of q the quantity given by $\mathrm{tr}\,\mathsf{q} = \mathsf{q} + \overline{\mathsf{q}} = 2\,\mathrm{Re}\,\mathsf{q}$ and call *determinant* of q and denote by $\det \mathsf{q}$ the quantity given by $\det \mathsf{q} = \mathsf{q}\,\overline{\mathsf{q}} = q_0^2 + q_1^2 - q_2^2 - q_3^2$.

Unlike $\mathbb{C}$ and $\mathbb{H}$, $\mathbb{H}_{\mathsf{coq}}$ is not a division algebra. In fact, a coquaternion q is invertible if and only if $\det \mathsf{q} \neq 0$. In that case, we have $\mathsf{q}^{-1} = \frac{\overline{\mathsf{q}}}{\det \mathsf{q}}$. A coquaternion q is called *space-like*, *light-like* or *time-like* if $\det \mathsf{q} < 0$, $\det \mathsf{q} = 0$ or $\det \mathsf{q} > 0$, respectively; the sets of such coquaternions are denoted by $\mathbb{S}$, $\mathbb{L}$ and $\mathbb{T}$, respectively and we say that two coquaternions have the same nature if both belong to the same set $\mathbb{S}$, $\mathbb{L}$ or $\mathbb{T}$. Finally, we endow $\mathbb{H}_{\mathsf{coq}}$ with the *semi-norm* $||\mathsf{q}|| = \sqrt{|\det \mathsf{q}|}$ and call q a *unit coquaternion* if $||\mathsf{q}|| = 1$.

The following standard functions are extended to coquaternion objects `Abs`, `Conjugate`, `Norm`, `Det`, `Re`, `Tr`. They can be used to verified some properties of coquaternions.

Given $\mathsf{p}, \mathsf{q} \in \mathbb{H}_{\mathsf{coq}}$ and $\alpha \in \mathbb{R}$, we have:

1. $\overline{\overline{\mathsf{q}}} = \mathsf{q}$; $\overline{\mathsf{p} + \mathsf{q}} = \overline{\mathsf{p}} + \overline{\mathsf{q}}$; $\overline{\mathsf{p}\,\mathsf{q}} = \overline{\mathsf{q}}\,\overline{\mathsf{p}}$;

```
In[6]:= (p*)* == p && (p + q)* == p* + q* && (p** q)* == q* ** p*

Out[6]= True
```

2. $\mathrm{Re}\,\mathsf{q} = \mathrm{Re}\,\overline{\mathsf{q}}$; $\mathrm{Re}(\mathsf{p}\,\mathsf{q}) = \mathrm{Re}(\mathsf{q}\,\mathsf{p})$;

```
In[7]:= Re[q] == Re[q*] && Re[p ** q] == Re[q ** p]

Out[7]= True
```

3. $\det \mathsf{q} = \det(\overline{\mathsf{q}})$; $\det(\mathsf{pq}) = \det(\mathsf{qp})$; $\det(\alpha\mathsf{q}) = \alpha^2 \det \mathsf{q}$;

```
In[8]:= Det[q] == Det[q*] && Det[p**q] == Det[q ** p] && Det[α q] == α^2 Det[q]

Out[8]= True
```

4. $\det \mathsf{q} = (\operatorname{Re} \mathsf{q})^2 + \det \underline{\mathsf{q}}$;

```
In[9]:= Det[q] == Re[q]^2 + Det[Vec[q]]

Out[9]= True
```

5. $\mathsf{q}^2 = (\operatorname{tr} \mathsf{q})\mathsf{q} - \det \mathsf{q}$;

```
In[10]:= q ** q == Tr[q] q - Det[q]

Out[10]= True
```

6. $\mathbb{H}_{\mathsf{coq}}$ is not a division algebra.

```
In[11]:= Coquaternion[1, 0, 1, 0] ** q

Out[11]= Coquaternion[q0 + q2, q1 − q3, q0 + q2, q3 − q1]

In[12]:= Solve[List @@ % == {1, 0, 0, 0}, {q0, q1, q2, q3}]

Out[12]= { }

In[13]:= {1/Coquaternion[1, 1, 0, 1], 1/Coquaternion[1, 0, 1, 0]}

          Coquaternion::NotInvertible : Non invertible Coquaternion

Out[13]= {Coquaternion[1, −1, 0, −1], Null}
```

7. $\mathbb{H}_{\mathsf{coq}}$ has zero divisors and nilpotent elements.

```
In[14]:= Coquaternion[1,0,1,0] ** Coquaternion[1,0,-1,0]

Out[14]= Coquaternion[0, 0, 0, 0]

In[15]:= Power[Coquaternion[0,1,1,0],2]

Out[15]= Coquaternion[0, 0, 0, 0]
```

Since $\det \mathsf{q} \geq \det \underline{\mathsf{q}}$, if q is space-like $\underline{\mathsf{q}}$ is of the same nature and a light-like coquaternion can not have a time-like vector part. The following examples illustrate the use of the function `Nature` to classify coquaternions.

```
In[16]:= q=Coquaternion[1,0,2,0]; Nature/@{q,Vec[q]}
Out[16]= {Spacelike, Spacelike}

In[17]:= q=Coquaternion[3,0,2,0]; Nature/@{q,Vec[q]}
Out[17]= {Timelike, Spacelike}

In[18]:= q=Coquaternion[1,1,1,0]; Nature/@{q,Vec[q]}
Out[18]= {Timelike, Lightlike}

In[19]:= q=Coquaternion[1,2,1,0]; Nature/@{q,Vec[q]}
Out[19]= {Timelike, Timelike}

In[20]:= q=Coquaternion[0,1,1,0]; Nature/@{q,Vec[q]}
Out[20]= {Lightlike, Lightlike}

In[21]:= q=Coquaternion[Sqrt@3,1,2,0]; Nature/@{q,Vec[q]}
Out[21]= {Lightlike, Spacelike}
```

Function	Description
`Abs[q]`	extends the absolute value function to coquaternion objects
`Conjugate[q]`	extends the conjugate function to coquaternion objects
`Coquaternion[q0,q1,q2,q3]`	the coquaternion $q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$
`CoquaternionQ[q]`	gives `True` if q is a coquaternion and `False` otherwise
`Det[q]`	extends the determinant function to coquaternion objects
`LightlikeQ[q]`	gives `True` if q is Light-like and `False` otherwise
`Nature[q]`	classifies q as Space-like, Light-like or Time-like
`Norm[q]`	extends the norm function to coquaternion objects
`Power[q]`	extends the power function to coquaternion objects
`Re[q]`	extends the real part function to coquaternion objects
`SpacelikeQ[q]`	gives `True` if q is Space-like and `False` otherwise
`TimelikeQ[q]`	gives `True` if q is Time-like and `False` otherwise
`Tr[q]`	extends the trace function to coquaternion objects
`Vec[q]`	the vector part of q

Table 1: Basic operations on coquaternions

3 Representation of Coquaternions

Complex representation

Any coquaternion q $= q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ can be written as

$$\mathsf{q} = (q_0 + \mathbf{i}q_1) + (q_2 + \mathbf{i}q_3)\mathbf{j}$$

and represented by the complex matrix

$$\begin{pmatrix} q_0 + \mathbf{i}q_1 & q_2 + \mathbf{i}q_3 \\ q_2 - \mathbf{i}q_3 & q_0 - \mathbf{i}q_1 \end{pmatrix}$$

Thus, the product of two coquaternions can be expressed as the product of the two corresponding complex matrices.

```
 In[22]:= p=Coquaternion[1,0,2,1];q=Coquaternion[1,1,-2,1];

 In[23]:= CoquaternionToComplex[q]

Out[23]= {1 + i, -2 + i}

 In[24]:= ComplexToCoquaternion[%]

Out[24]= Coquaternion[1, 1, -2, 1]

 In[25]:= Dot@@CoquaternionToComplexMatrix/@{q,p}//ComplexMatrixToCoquaternion

Out[25]= Coquaternion[-2, 5, -1, 4]

 In[26]:= q**p

Out[26]= Coquaternion[-2, 5, -1, 4]
```

Real 2×2 matrix representation

It is well-known that the algebra of coquaternions is isomorphic to $\mathcal{M}_2(\mathbb{R})$, the algebra of real 2×2 matrices, with the map $\Phi : \mathcal{M}_2(\mathbb{R}) \to \mathbb{H}_{\mathsf{coq}}$ defined by

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto \mathsf{a} = \frac{1}{2}\big((a+d) + (b-c)\mathbf{i} + (b+c)\mathbf{j} + (a-d)\mathbf{k}\big),$$

establishing the isomorphism. The inverse of Φ is the map $\Psi : \mathbb{H}_{\mathsf{coq}} \to \mathcal{M}_2(\mathbb{R})$ defined by

$$\mathsf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \mapsto Q = \begin{pmatrix} q_0 + q_3 & q_1 + q_2 \\ q_2 - q_1 & q_0 - q_3 \end{pmatrix}.$$

This means, in particular, that all the results contained in the previous section can be derived in terms of 2×2 real matrices.

```
In[27]:= CoquaternionToMatrix[Coquaternion[q0,q1,q2,q3]] // MatrixForm
```

Out[27]/MatrixForm=

$$\begin{pmatrix} q0+q3 & q1+q2 \\ -q1+q2 & q0-q3 \end{pmatrix}$$

```
In[28]:= CoquaternionToMatrix[Coquaternion[a,b,c,d]]//MatrixForm
```

Out[28]/MatrixForm=

$$\begin{pmatrix} a+d & b+c \\ -b+c & a-d \end{pmatrix}$$

```
In[29]:= CoquaternionToMatrix[q]//MatrixForm
```

Out[29]/MatrixForm=

$$\begin{pmatrix} 2 & -1 \\ -3 & 0 \end{pmatrix}$$

```
In[30]:= MatrixToCoquaternion[%]
```

Out[30]= Coquaternion$[1, 1, -2, 1]$

```
In[31]:= CoquaternionToMatrix[q].CoquaternionToMatrix[p]//MatrixForm
```

Out[31]/MatrixForm=

$$\begin{pmatrix} 2 & 4 \\ -6 & -6 \end{pmatrix}$$

```
In[32]:= CoquaternionToMatrix[q**p]//MatrixForm
```

Out[32]/MatrixForm=

$$\begin{pmatrix} 2 & 4 \\ -6 & -6 \end{pmatrix}$$

Real 4×4 matrix representation

The left and right representations of the coquaternion q $= q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$, associated with the left and right product are, respectively,

$$L_{\mathsf{q}} = \begin{pmatrix} q_0 & -q_1 & q_2 & q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{pmatrix} \quad \text{and} \quad R_{\mathsf{q}} = \begin{pmatrix} q_0 & -q_1 & q_2 & q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & -q_3 & q_0 & -q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{pmatrix}$$

```
In[33]:= CoquaternionTo4DMatrixL[Coquaternion[q0,q1,q2,q3]]//MatrixForm
```

Out[33]/MatrixForm=

$$\begin{pmatrix} q0 & -q1 & q2 & q3 \\ q1 & q0 & q3 & -q2 \\ q2 & q3 & q0 & -q1 \\ q3 & -q2 & q1 & q0 \end{pmatrix}$$

```
In[34]:= CoquaternionTo4DMatrixR[Coquaternion[q0,q1,q2,q3]]//MatrixForm
```

Out[34]/MatrixForm=

$$\begin{pmatrix} q0 & -q1 & q2 & q3 \\ q1 & q0 & -q3 & q2 \\ q2 & -q3 & q0 & q1 \\ q3 & q2 & -q1 & q0 \end{pmatrix}$$

```
In[35]:= (Lq=CoquaternionTo4DMatrixL[q])//MatrixForm
```

Out[35]/MatrixForm=

$$\begin{pmatrix} 1 & -1 & -2 & 1 \\ 1 & 1 & 1 & 2 \\ -2 & 1 & 1 & -1 \\ 1 & 2 & 1 & 1 \end{pmatrix}$$

```
In[36]:= (Rq=CoquaternionTo4DMatrixR[q])//MatrixForm
```

Out[36]/MatrixForm=

$$\begin{pmatrix} 1 & -1 & -2 & 1 \\ 1 & 1 & -1 & -2 \\ -2 & -1 & 1 & 1 \\ 1 & -2 & -1 & 1 \end{pmatrix}$$

```
In[37]:= Coquaternion@@(Lq.List@@p)==q**q
Out[37]= True

In[38]:= Coquaternion@@(Rq.List@@p)==p**q
Out[38]= True
```

Polar Form

Any coquaternion $\mathsf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} \in \mathbb{S} \cup \mathbb{T}$, such that $\underline{\mathsf{q}} \notin \mathbb{L}$, has a polar representation [18, 11] in one of the forms

$$\mathsf{q} = \begin{cases} \|\mathsf{q}\|\big(\sinh\phi_{\mathsf{q}} + \boldsymbol{\omega}_{\underline{\mathsf{q}}}\cosh\phi_{\mathsf{q}}\big), & \text{if } \mathsf{q} \in \mathbb{S}, \\ \|\mathsf{q}\|\big(\operatorname{sgn} \mathrm{q}_0\cosh\psi_{\mathsf{q}} + \boldsymbol{\omega}_{\underline{\mathsf{q}}}\sinh\psi_{\mathsf{q}}\big), & \text{if } \mathsf{q} \in \mathbb{T} \text{ and } \underline{\mathsf{q}} \in \mathbb{S}, \\ \|\mathsf{q}\|\big(\cos\theta_{\mathsf{q}} + \boldsymbol{\omega}_{\underline{\mathsf{q}}}\sin\theta_{\mathsf{q}}\big), & \text{if } \mathsf{q} \in \mathbb{T} \text{ and } \underline{\mathsf{q}} \in \mathbb{T}, \end{cases} \tag{3.1a}$$

where sgn is the usual sign function, ϕ_{q}, ψ_{q} and θ_{q} are such that

$$\sinh\phi_{\mathsf{q}} = \frac{q_0}{\|\mathsf{q}\|},\quad \sinh\psi_{\mathsf{q}} = \frac{\|\underline{\mathsf{q}}\|}{\|\mathsf{q}\|},\quad \cos\theta_{\mathsf{q}} = \frac{q_0}{\|\mathsf{q}\|},\quad \theta_{\mathsf{q}} \in (0,\pi) \tag{3.1b}$$

and

$$\boldsymbol{\omega}_{\underline{\mathsf{q}}} = \frac{\underline{\mathsf{q}}}{\|\underline{\mathsf{q}}\|} \tag{3.1c}$$

is a unit coquaternion satisfying $\boldsymbol{\omega}_{\underline{\mathsf{q}}}^2 = 1$, if $\underline{\mathsf{q}} \in \mathbb{S}$ and $\boldsymbol{\omega}_{\underline{\mathsf{q}}}^2 = -1$, if $\underline{\mathsf{q}} \in \mathbb{T}$.

The polar form (3.1) is implemented in `Coquaternions` through the use of `PolarForm[q]` which gives a list of one of the forms

$$\{\mathsf{nat} \to \mathsf{Spacelike}, r \to \|\mathsf{q}\|, \varphi \to \phi_{\mathsf{q}}, \omega \to \boldsymbol{\omega}_{\underline{\mathsf{q}}}\},$$
$$\{\mathsf{nat} \to \mathsf{TimeSpacelike}, r \to \|\mathsf{q}\|, \varphi \to \psi_{\mathsf{q}}, \omega \to \boldsymbol{\omega}_{\underline{\mathsf{q}}}, \mathsf{sgn} \to \operatorname{sgn} q_0\},$$
$$\{\mathsf{nat} \to \mathsf{TimeTimelike}, r \to \|\mathsf{q}\|, \varphi \to \theta_{\mathsf{q}}, \omega \to \boldsymbol{\omega}_{\underline{\mathsf{q}}}\},$$

according to (3.1a)-(3.1c). The function `FromPolarForm` reads a list of one of the above forms and gives the corresponding coquaternion.

```
In[39]:= PolarForm[Coquaternion[1,1,2,2]]
```

Out[39]= $\left\{\texttt{nat}\to\texttt{Spacelike}, r\to\sqrt{6}, \varphi\to\texttt{ArcSinh}\left[\frac{1}{\sqrt{6}}\right], \omega\to\texttt{Coquaternion}\left[0,\frac{1}{\sqrt{7}},\frac{2}{\sqrt{7}},\frac{2}{\sqrt{7}}\right]\right\}$

```
In[40]:= FromPolarForm[{nat,r,φ,ω}/.%]
Out[40]= Coquaternion[1,1,2,2]

In[41]:= PolarForm[Coquaternion[-3,1,1,2]]
```

Out[41]= $\left\{\texttt{nat}\to\texttt{TimeSpacelike}, r\to\sqrt{5}, \varphi\to\texttt{ArcSinh}\left[\frac{2}{\sqrt{5}}\right], \omega\to\texttt{Coquaternion}\left[0,\frac{1}{2},\frac{1}{2},1\right], \texttt{sgn}\to-1\right\}$

```
In[42]:= FromPolarForm[{nat,r,φ,ω,sgn}/.%]
Out[42]= Coquaternion[-3,1,1,2]

In[43]:= PolarForm[Coquaternion[4,3,2,1]]
```

Out[43]= $\left\{\texttt{nat}\to\texttt{TimeTimelike}, r\to2\sqrt{5}, \varphi\to\texttt{ArcCos}\left[\frac{2}{\sqrt{5}}\right], \omega\to\texttt{Coquaternion}\left[0,\frac{3}{2},1,\frac{1}{2}\right]\right\}$

```
In[44]:= FromPolarForm[{nat,r,φ,ω}/.%]
```

```
Out[44]= Coquaternion[4, 3, 2, 1]

 In[45]:= PolarForm[Coquaternion[1,1,1,1]]

          PolarForm::LightLikeCoq : No polar form.
```

Function	Description
`ComplexMatrixToCoquaternion[m]`	gives the coquaternion corresponding to $m \in \mathcal{M}_2(\mathbb{C})$
`ComplexToCoquaternion[{a,b}]`	gives the coquaternion $a + b\mathbf{j}$ to $a, b \in \mathbb{C}$
`CoquaternionToComplex[q]`	gives a list $\{q_0 + \mathbf{i}q_1, q_2 + \mathbf{i}q_3\}$ for the complex form of q
`CoquaternionToComplexMatrix[q]`	gives the complex 2×2 representation matrix of q
`CoquaternionToMatrix[q]`	gives the real 2×2 representation matrix of q
`CoquaternionTo4DMatrixL[q]`	gives the real 4×4 left representation matrix of q
`CoquaternionTo4DMatrixR[q]`	gives the real 4×4 right representation matrix of q
`FromPolarForm[l]`	gives the coquaternion whose "polar coordinates" are l
`MatrixToCoquaternion[m]`	gives the coquaternion corresponding to $m \in \mathcal{M}_2(\mathbb{R})$
`PolarForm[q]`	gives the polar form of a coquaternion (in case of existence)

Table 2: Representation of coquaternions

4 Coquaternionic Polynomials

The study of quaternionic polynomials and, in particular, methods for determining and classifying their zero-sets have become an active research area. Recently we have developed a Mathematica tool - `QPolynomial` - a collection of functions for manipulating, evaluating and factoring quaternionic polynomials [9, 10]. In contrast, polynomials defined over the algebra of coquaternions have received much less attention from researchers (see, for example, [8], [12] and [15] and references therein).

In this section we describe a work in progress Mathematica tool - `CoqPolynomial`, relying on the package `Coquaternions`, for dealing with some polynomial problems in $\mathbb{H}_{\text{coq}}$. Starting with the basic definitions, we denote by $\mathbb{H}_{\text{coq}}[x]$ the set of polynomials of the form

$$P(x) = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0, \;\; c_i \in \mathbb{H}_{\text{coq}}, \tag{4.1}$$

i.e., the set of polynomials whose coefficients are only located on the left of the variable, with the addition and multiplication of such polynomials defined as in the commutative case, where the variable is assumed to commute with the coefficients. This is a ring, referred to as the ring of (left) *one-sided* or *unilateral* polynomials in $\mathbb{H}_{\text{coq}}$. From now on we use the simplified designation *coquaternionic polynomials* for polynomials in $\mathbb{H}_{\text{coq}}[x]$ and we usually omit the reference to the variable and write simply P when referring to an element $P(x) \in \mathbb{H}_{\text{coq}}[x]$.

As usual, if $c_n \neq 0$, we say that the *degree* of the polynomial P is n and refer to c_n as the leading coefficient of the polynomial. When $c_n = 1$, we say that P is *monic*. If the coefficients c_i in (4.1) are real, then we say that P is a *real polynomial*. Given a polynomial P of the form (4.1), its *conjugate polynomial* is the polynomial defined by

$$\overline{P}(x) = \bar{c}_n x^n + \bar{c}_{n-1} x^{n-1} + \cdots + \bar{c}_1 x + \bar{c}_0.$$

Following the structure of `QPolynomial`, a coquaternionic polynomial in `CoqPolynomial` is an object of the form `Polynomial[`c_n`,`c_{n-1}`,...,`c_1`,`c_0`]` accordingly to (4.1). For such objects, rules as `Plus`, `NonCommutativeMultiply`, `Power` and functions as `Conjugate`, `Eval`, etc. are defined; its use can be illustrated as follows:

```
 In[46]:= P1 = Polynomial[1,Coquaternion[2,-1,1,0]];
           P2 = Polynomial[Coquaternion[1,0,0,1],Coquaternion[0,0,1,1]];
           α = Coquaternion[0,1,1,0];
 In[47]:= P1 + P2
Out[47]= Polynomial[Coquaternion[2, 0, 0, 1], Coquaternion[2, −1, 2, 1]]

 In[48]:= α ** P1
Out[48]= Polynomial[Coquaternion[0, 1, 1, 0], Coquaternion[2, 2, 2, 2]]

 In[49]:= P1 ** P2
Out[49]= Polynomial[Coquaternion[1, 0, 0, 1], Coquaternion[2, −2, 3, 3], Coquaternion[1, −1, 3, 1]]

 In[50]:= P3 = Polynomial[Coquaternion[1,0,0,-1],Coquaternion[1,2,1,1]];
           P2 ** P3
Out[50]= Polynomial[Coquaternion[1, 4, 4, 3], Coquaternion[2, 0, 3, −1],

 In[51]:= Conjugate[P1 ** P2]
Out[51]= Polynomial[Coquaternion[1, 0, 0, −1], Coquaternion[2, 2, −3, −3], Coquaternion[1, 1, −3, −1]

 In[52]:= Conjugate[P2] ** Conjugate[P2]] == %
Out[52]= True
```

For a coquaternionic polynomial P, the *evaluation map* at a given element $q \in \mathbb{H}_{\mathsf{coq}}$ is defined by

$$P(q) = c_n q^n + c_{n-1} q^{n-1} \; + \; \cdots + c_1 q + c_0.$$

If $P(q) = 0$, then we say that q is a *zero* (or a *root*) P.

```
 In[53]:= Eval[P1,α]
Out[53]= Coquaternion[2, 0, 2, 0]

 In[54]:= Eval[P2][α]
Out[54]= Coquaternion[0, 2, 3, 1]

 In[55]:= Eval[P1,Coquaternion[-2,1,-1,0]]
Out[55]= 0
```

The evaluation map is not a homomorphism from the ring $\mathbb{H}_{\mathsf{coq}}[x]$ into $\mathbb{H}_{\mathsf{coq}}$; given two polynomials $L, R \in \mathbb{H}[x]$, in general, we do not have $(LR)(q) = L(q)R(q)$.

```
 In[56]:= Eval[P1 ** P2][α]
Out[56]= Coquaternion[6, 4, 8, −4]

 In[57]:= Eval[P1][α] ** Eval[P2][α]
Out[57]= Coquaternion[6, 2, 6, −2]
```

We now list and illustrate some results concerning the evaluation, at a given element $q \in \mathbb{H}_{\mathsf{coq}}$, of the product of two polynomials. Consider the polynomials

$$L(x) = \sum_{i=0}^{n} a_i x^i, \quad R(x) = \sum_{j=0}^{m} b_j x^j, \quad P(x) = L(x)R(x).$$

1. $P(q) = \sum_{i=0}^{n} a_i \, R(q) \, q^i$.

```
In[58]:= L = Polynomial[1,Coquaternion[1,2,3,4]];

            R = Polynomial[1,Coquaternion[1,-1,2,0]];

            q = Coquaternion[-1,-1,2,0];

            P = L ** R
```

```
Out[58]= Polynomial[1, Coquaternion[2, 1, 5, 4], Coquaternion[9, 9, 1, 11]]

 In[59]:= Eval[P,q]

Out[59]= Coquaternion[22, 16, −8, 14]

 In[60]:= l = List @@ L;
           l[[2]] ** Eval[R,q] + l[[1]] ** Eval[R,q] ** q

Out[60]= Coquaternion[22, 16, −8, 14]
```

2. If $R(q) = 0$, then $P(q) = 0$.

```
 In[61]:= PolynomialZeroQ[R,Conjugate@q] && PolynomialZeroQ[P,Conjugate@q]

Out[61]= True
```

3. If $R(q)$ is non-singular, then $P(q) = L(\tilde{q})R(q)$, where $\tilde{q} = R(q)q(R(q))^{-1}$.

```
 In[62]:= InvertibleQ[Eval[R,q]]

Out[62]= True

 In[63]:= qtilde = Eval[R,q] ** q ** (1/Eval[R,q])

Out[63]= Coquaternion[−1, −1, 2, 0]

 In[64]:= Eval[L,qtilde] ** Eval[R,q]

Out[64]= Coquaternion[22, 16, −8, 14]
```

4. If $L(x)$ is a real polynomial, then $(LR)(q) = (RL)(q) = R(q)L(q)$.

```
 In[65]:= LL = Polynomial[1,2,3];
           {Eval[LL ** R,q], Eval[R ** LL,q], Eval[LL,q] ** Eval[R,q]} // Union

Out[65]= {Coquaternion[0, −10, 20, 0]}
```

5. If $q \in \mathbb{R}$, then $(LR)(q) = L(q)R(q)$.

```
 In[66]:= Eval[P,1]

Out[66]= Coquaternion[12, 10, 6, 15]

 In[67]:= Eval[L,1] ** Eval[R,1]

Out[67]= Coquaternion[12, 10, 6, 15]
```

6. If $R(q)$ is non-singular, then q is a zero of P iff $R(q)q(R(q))^{-1}$ is a zero of L.

```
 In[68]:= L = Polynomial[1,Coquaternion[-1,-5,1,5]];
           P = L ** R;zeroP = Coquaternion[1,-1,1,1];
           PolynomialZeroQ[P,zeroP]

Out[68]= True

 In[69]:= InvertibleQ[Eval[R,zeroP]]

Out[69]= True

 In[70]:= zeroL = Eval[R,zeroP] ** zeroP ** (1/Eval[R,zeroP])
```

```
Out[70]= Coquaternion[1, 5, −1, −5]

 In[71]:= PolynomialZeroQ[L,zeroL]

Out[71]= True

 In[72]:= L=Polynomial[1,Coquaternion[-1,0,1,1]];

           P=L**R;zeroL=Coquaternion[1,0,-1,-1];

           PolynomialZeroQ[L,zeroL]

Out[72]= True

 In[73]:= zeroP=Eval[Conjugate@R,zeroL]**zeroL**(1/Eval[Conjugate@R,zeroL])
```

Out[73]= `Coquaternion`$[1, \frac{16}{5}, -3, \frac{9}{5}]$

```
 In[74]:= PolynomialZeroQ[P,zeroP]

Out[74]= True
```

Function	Description
`Conjugate[P]`	gives the conjugate of P
`Eval[P][q]`	gives P(q)
`Polynomial[`c_n`,`c_{n-1}`,...,`c_1`,`c_0`]`	the object $c_n x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$
`PolynomialZeroQ[P, q]`	gives `True` if $P(q) = 0$

Table 3: Some functions of `CoqPolynomial`

As we have already referred, the collection of Mathematica functions here described provides the basic mathematical tools necessary for manipulating coquaternions and coquaternionic polynomials, reflecting, in its present form, the recent interests of the authors in the area. In the near future we intend to endow `CoqPolynomial` with the ability to compute roots of coquaternionic polynomials, by implementing the recent algorithm proposed in [12]. Details on this implementation will appear in a forthcoming paper.

Acknowledgments

This paper was financed by national funds of the FCT – Portuguese Foundation for Science and Technology – within the projects UID/ECO/03182/2019 and UID/MAT/00013/2013.

References

[1] Abłamowicz, R., "Computations with Clifford and Graßmann algebras", *Adv. Appl. Clifford Algebras*, Vol. **19(3-4)**, pp. 499–545, 2009

[2] Abłamowicz, R., Fauser, B., "Mathematics of Clifford – a Maple package for Clifford and Graßmann algebras", *Adv. Appl. Clifford Algebras*, Vol. **15(2)**, pp. 157–181, 2005

[3] Brody, D.C., Graefe, E.M., "On complexified mechanics and coquaternions", *J. Phys. A: Math. Theory*, Vol. **44**, pp. 1–9, 2011

[4] Cockle, J., "On systems of algebra involving more than one imaginary; and on equations of the fifth degree", *Philos. Mag.*, Vol. **35(238)**, pp. 434–437, 1849

[5] Falcão, M.I., Miranda, F., "Quaternions: A Mathematica Package for Quaternionic Analysis", *Lecture Notes in Comput. Sci.*, Vol. **6784**, pp. 200–214, 2011

[6] Falcão, M.I., Miranda, F., Severino, R., Soares, M.J., "Basins of attraction for a quadratic coquaternionic map", *Chaos, Solitons & Fractals*, Vol. **104**, pp. 716–724, 2017

[7] Falcão, M.I., Miranda, F., Severino, R., Soares, M.J., "Iteration of quadratic maps on coquaternions", *Int. J. Bifurcation and Chaos*, Vol. **27(12)**:1730039, 2017

[8] Falcão, M.I., Miranda, F., Severino, R., Soares, M.J., "Polynomials over quaternions and coquaternions: a unified approach", *Lecture Notes in Comput. Sci.*, Vol. **10405**, pp. 379–393, 2017

[9] Falcão, M.I., Miranda, F., Severino, R., Soares, M.J., "Computational aspects of quaternionic polynomials - Part I :: Manipulating, evaluating and factoring", *The Mathematica Journal*, Vol. **20(4)**, 2018

[10] Falcão, M.I., Miranda, F., Severino, R., Soares, M.J., "Computational aspects of quaternionic polynomials - Part II :: Root-finding methods", *The Mathematica Journal*, Vol. **20(5)**, 2018

[11] Falcão, M.I., Miranda, F., Severino, R., Soares, M.J., "On the roots of coquaternions", *Adv. Appl. Clifford Algebras*, Vol. **28:97**, 2018

[12] Falcão, M.I., Miranda, F., Severino, R., Soares, M.J., "The number of zeros of unilateral polynomials over coquaternions revisited", *Linear and Multilinear Algebra*, 2018, DOI: 10.1080/03081087.2018.1450828

[13] Gao, C., Chen, X., Shen, Y.-G., "Quintessence and phantom emerging from the split-complex field and the split-quaternion field", *Gen. Relativ. Gravit.*, Vol. **48:11**, 2016

[14] Hamilton, W.R., "On a new species of imaginary quantities connected with a theory of quaternions", *Proc. R. Ir. Acad*, Vol. **2**, pp. 424–434, Nov. 13, 1843

[15] Janovská, D., Opfer, G., "Zeros and singular points for one-sided coquaternionic polynomials with an extension to other $\mathbb{R}^4$ algebras", *Electron. Trans. Numer. Anal.*, Vol. **41**, pp. 133–158, 2014

[16] Kula, L., Yayli, Y., "Split quaternions and rotations in semi Euclidean space E_2^4", *J. Korean Math. Soc.*, Vol. **44**, pp. 1313–1327, 2007

[17] Malonek, H.R., "Quaternions in applied sciences. A historical perspective of a mathematical concept", *17th Inter. Conf. on the Appl. of Computer Science and Mathematics on Architecture and Civil Engineering*, Weimar, 2003

[18] Özdemir, M., "The roots of a split quaternion", *Appl. Math. Letters*, Vol. **22**, pp. 258–263, 2009

[19] Özdemir, M., Ergin, A., "Some geometric applications of timelike quaternions", *Proc. 16th Int. Conf. Jangjeon Math. Soc.*, Vol. **16**, pp. 108–115, 2005

[20] Özdemir, M., Ergin, A., "Rotations with unit timelike quaternions in Minkowski 3-space", *J. Geometry Phys.*, Vol. **56(2)**, pp. 322–336, 2006

[21] Sangwine, J., Le Bihan, N., "Quaternion Toolbox for Matlab", 2005, `http://qtfm.sourceforge.net`