quaternion 2.0.0

Quaternion Package for GNU Octave

Lukas F. Reichlin
Juan Pablo Carbajal

Preface

The GNU Octave quaternion package from version 2 onwards was developed by Lukas F. Reichlin with important contributions by Juan Pablo Carbajal. This new package is intended as a replacement for quaternion-1.0.0 by A. Scottedward Hodel. It is loosely based on ideas from the Quaternion Toolbox for Matlab by Steve Sangwine and Nicolas Le Bihan with a special focus on code simplicity and vectorization. Its main features are:

- Matrices and n-dimensional arrays of quaternions.
- Overloaded operators due to the use of classes introduced with Octave 3.2.
- Operator semantics similar to Octave's built-in complex numbers.
- Fully vectorized code for crunching large quaternion arrays in a speedy manner.

Using the help function

Some functions of the quaternion package are listed with a leading `@quaternion/`. This is only needed to view the help text of the function, e.g. `help norm` shows the built-in function while `help @quaternion/norm` shows the overloaded function for quaternions. Note that there are quaternion functions like `unit` that have no built-in equivalent.

When just using the function, the leading `@quaternion/` must **not** be typed. Octave selects the right function automatically. So one can type `norm (q)` and `norm (matrix)` regardless of the class of the argument.

Table of Contents

1 Quaternions

1.1 quaternion

q = quaternion (*w*) [Function File]
q = quaternion (*x*, *y*, *z*) [Function File]
q = quaternion (*w*, *x*, *y*, *z*) [Function File]

Constructor for quaternions - create or convert to quaternion.

```
q = w + x*i + y*j + z*k
```

Arguments *w*, *x*, *y* and *z* can be scalars or matrices, but they must be real and of equal size. If scalar part *w* or components *x*, *y* and *z* of the vector part are not specified, zero matrices of appropriate size are assumed.

Example

```
octave:1> q = quaternion (2)                                    =
q = 2 + 0i + 0j + 0k

octave:2> q = quaternion (3, 4, 5)
q = 0 + 3i + 4j + 5k

octave:3> q = quaternion (2, 3, 4, 5)
q = 2 + 3i + 4j + 5k

octave:4> w = [2, 6, 10; 14, 18, 22];                           =
octave:5> x = [3, 7, 11; 15, 19, 23];
octave:6> y = [4, 8, 12; 16, 20, 24];
octave:7> z = [5, 9, 13; 17, 21, 25];
octave:8> q = quaternion (w, x, y, z)
q.w =
    2    6   10
   14   18   22

q.x =
    3    7   11
   15   19   23

q.y =
    4    8   12
   16   20   24

q.z =
    5    9   13
   17   21   25

octave:9>
```

1.2 qi

qi [Function File]

Create x-component of a quaternion's vector part.

```
q = w + x*qi + y*qj + z*qk
```

Example

```
octave:1> q1 = quaternion (1, 2, 3, 4)                                   =
q1 = 1 + 2i + 3j + 4k
octave:2> q2 = 1 + 2*qi + 3*qj + 4*qk
q2 = 1 + 2i + 3j + 4k
octave:3>
```

1.3 qj

qj [Function File]

Create y-component of a quaternion's vector part.

```
q = w + x*qi + y*qj + z*qk
```

Example

```
octave:1> q1 = quaternion (1, 2, 3, 4)                                   =
q1 = 1 + 2i + 3j + 4k
octave:2> q2 = 1 + 2*qi + 3*qj + 4*qk
q2 = 1 + 2i + 3j + 4k
octave:3>
```

1.4 qk

qk [Function File]

Create z-component of a quaternion's vector part.

```
q = w + x*qi + y*qj + z*qk
```

Example

```
octave:1> q1 = quaternion (1, 2, 3, 4)                                   =
q1 = 1 + 2i + 3j + 4k
octave:2> q2 = 1 + 2*qi + 3*qj + 4*qk
q2 = 1 + 2i + 3j + 4k
octave:3>
```

1.5 q2rot

[*axis*, *angle*] = q2rot (*q*) [Function File]

Extract vector/angle form of a unit quaternion *q*.

Inputs

q Unit quaternion describing the rotation.

Outputs

axis Eigenaxis as a 3-d unit vector [x, y, z].

angle Rotation angle in radians. The positive direction is determined by the right-hand rule applied to *axis*.

Example

```
octave:1> axis = [0, 0, 1]                                              =
axis =
   0   0   1
octave:2> angle = pi/4
angle =  0.78540
octave:3> q = rot2q (axis, angle)
q = 0.9239 + 0i + 0j + 0.3827k
octave:4> [vv, th] = q2rot (q)
vv =
   0   0   1
th =  0.78540
octave:5> theta = th*180/pi
theta =  45.000
octave:6>
```

1.6 rot2q

q = `rot2q` (*axis*, *angle*) [Function File]

Create unit quaternion *q* which describes a rotation of *angle* radians about the vector *axis*. This function uses the active convention where the vector *axis* is rotated by *angle* radians. If the coordinate frame should be rotated by *angle* radians, also called the passive convention, this is equivalent to rotating the *axis* by -*angle* radians.

Inputs

axis Vector `[x, y, z]` describing the axis of rotation.

angle Rotation angle in radians. The positive direction is determined by the right-hand rule applied to *axis*.

Outputs

q Unit quaternion describing the rotation.

Example

```
octave:1> axis = [0, 0, 1];                                             =
octave:2> angle = pi/4;
octave:3> q = rot2q (axis, angle)
q = 0.9239 + 0i + 0j + 0.3827k
octave:4> v = quaternion (1, 1, 0)
v = 0 + 1i + 1j + 0k
octave:5> vr = q * v * conj (q)
vr = 0 + 0i + 1.414j + 0k
octave:6>
```

2 Methods

2.1 @quaternion/abs

qabs = abs (*q*) [Function File]

Modulus of a quaternion.

```
q = w + x*i + y*j + z*k
abs (q) = sqrt (w.^2 + x.^2 + y.^2 + z.^2)
```

2.2 @quaternion/blkdiag

q = blkdiag (*q1*, *q2*, . . .) [Function File]

Block-diagonal concatenation of quaternions.

2.3 @quaternion/cat

q = cat (*dim*, *q1*, *q2*, . . .) [Function File]

Concatenation of quaternions along dimension *dim*.

2.4 @quaternion/columns

nc = columns (*q*) [Function File]

Return number of columns *nc* of quaternion array *q*.

2.5 @quaternion/conj

q = conj (*q*) [Function File]

Return conjugate of a quaternion.

```
q = w + x*i + y*j + z*k
conj (q) = w - x*i - y*j - z*k
```

2.6 @quaternion/diag

q = diag (*v*) [Function File]

q = diag (*v*, *k*) [Function File]

Return a diagonal quaternion matrix with quaternion vector V on diagonal K. The second argument is optional. If it is positive, the vector is placed on the K-th super-diagonal. If it is negative, it is placed on the -K-th sub-diagonal. The default value of K is 0, and the vector is placed on the main diagonal.

2.7 @quaternion/diff

qdot = diff (*q*, *omega*) [Function File]

Derivative of a quaternion.

Let Q be a quaternion to transform a vector from a fixed frame to a rotating frame. If the rotating frame is rotating about the [x, y, z] axes at angular rates [wx, wy, wz], then the derivative of Q is given by

```
Q' = diff(Q, omega)
```

If the passive convention is used (rotate the frame, not the vector), then

```
Q' = diff(Q,-omega)
```

2.8 @quaternion/exp

qexp = `exp` (*q*) [Function File]

Exponential of a quaternion.

2.9 @quaternion/inv

qinv = `inv` (*q*) [Function File]

Return inverse of a quaternion.

2.10 @quaternion/ispure

flg = `ispure` (*q*) [Function File]

Return 1 if scalar part of quaternion is zero, otherwise return 0

2.11 @quaternion/log

qlog = `log` (*q*) [Function File]

Logarithmus naturalis of a quaternion.

2.12 @quaternion/norm

n = `norm` (*q*) [Function File]

Norm of a quaternion.

2.13 @quaternion/rows

nr = `rows` (*q*) [Function File]

Return number of rows *nr* of quaternion array *q*.

2.14 @quaternion/size

nvec = `size` (*q*) [Function File]
n = `size` (*q*, *dim*) [Function File]
[*nx*, *ny*, ...] = `size` (*q*) [Function File]

Return size of quaternion arrays.

Inputs

q Quaternion object.

dim If given a second argument, `size` will return the size of the corresponding dimension.

Outputs

nvec Row vector. The first element is the number of rows and the second element the number of columns. If *q* is an n-dimensional array of quaternions, the n-th element of *nvec* corresponds to the size of the n-th dimension of *q*.

n Scalar value. The size of the dimension *dim*.

nx Number of rows.

ny Number of columns.

... Sizes of the 3rd to n-th dimensions.

2.15 @quaternion/unit

qn = unit (*q*) [Function File]

Normalize quaternion to length 1 (unit quaternion).

```
q = w + x*i + y*j + z*k
unit (q) = q ./ sqrt (w.^2 + x.^2 + y.^2 + z.^2)
```

3 Overloaded Operators

3.1 @quaternion/ctranspose

Conjugate transpose of a quaternion. Used by Octave for "q'".

3.2 @quaternion/eq

Equal to operator for two quaternions. Used by Octave for "q1 == q2".

3.3 @quaternion/horzcat

Horizontal concatenation of quaternions. Used by Octave for "[q1, q2]".

3.4 @quaternion/ldivide

Element-wise left division for quaternions. Used by Octave for "q1 .\ q2".

3.5 @quaternion/minus

Subtraction of two quaternions. Used by Octave for "q1 - q2".

3.6 @quaternion/mldivide

Matrix left division for quaternions. Used by Octave for "q1 \ q2".

3.7 @quaternion/mpower

Matrix power operator of quaternions. Used by Octave for "q^x".

3.8 @quaternion/mrdivide

Matrix right division for quaternions. Used by Octave for "q1 / q2".

3.9 @quaternion/mtimes

Matrix multiplication of two quaternions. Used by Octave for "q1 * q2".

3.10 @quaternion/plus

Addition of two quaternions. Used by Octave for "q1 + q2".

3.11 @quaternion/power

Power operator of quaternions. Used by Octave for "q.^x". Exponent x can be scalar or of appropriate size.

3.12 @quaternion/rdivide

Element-wise right division for quaternions. Used by Octave for "q1 ./ q2".

3.13 @quaternion/subsasgn

Subscripted assignment for quaternions. Used by Octave for "q.key = value".

3.14 @quaternion/subsref

Subscripted reference for quaternions. Used by Octave for "q.w".

3.15 @quaternion/times

Element-wise multiplication of two quaternions. Used by Octave for "q1 .* q2".

3.16 @quaternion/transpose

Transpose of a quaternion. Used by Octave for "q.'".

3.17 @quaternion/uminus

Unary minus of a quaternion. Used by Octave for "-q".

3.18 @quaternion/uplus

Unary plus of a quaternion. Used by Octave for "+q".

3.19 @quaternion/vertcat

Vertical concatenation of quaternions. Used by Octave for "[q1; q2]".