Rotations of Moment of Inertia Tensor using Quaternions

Mikica B Kocic, 2012-04-22, v0.3

There is a wonderful connection between complex numbers (and quaternions, as their extension) and geometry in which, translations correspond to additions, rotations and scaling to multiplications, and reflections to conjugations.

However, the beauty of describing spatial rotations with quaternions is shadowed by an expression like:

$$\begin{pmatrix} 1-2\left(y^2+z^2\right) & 2xy-2wz & 2wy+2xz \\ 2xy+2wz & 1-2\left(x^2+z^2\right) & 2yz-2wx \\ 2xz-2wy & 2wx+2yz & 1-2\left(x^2+y^2\right) \end{pmatrix} \cdot$$

$$\cdot \mathcal{J} \cdot \begin{pmatrix} 1-2\left(y^2+z^2\right) & 2xy-2wz & 2wy+2xz \\ 2xy+2wz & 1-2\left(x^2+z^2\right) & 2yz-2wx \\ 2xz-2wy & 2wx+2yz & 1-2\left(x^2+y^2\right) \end{pmatrix}^{\mathrm{T}}$$

which actually represents such simple transformation as the rotation of moment of inertia tensor using quaternions [2].

The purpose of this document is to express spatial rotations of a moment of inertia tensor in the pure quaternionic form and to highlight the algebraical meaning behind such rotations by proving some interesting theorems.

■ Conventions and Definitions

Load quaternions package (see `Quat.nb` for the annotated source code)

```
Get[ "Quat.m", Path -> { NotebookDirectory [] } ];
```

▼ Conventions

In this document, matrices, vectors and matrix representation of quaternions are shown in uppercase font (roman-bold font in text) and quaternions and other numbers are shown in lowercase font (italic font in text). Since the symbol `I` in *Mathematica* is used for imaginary unit, the selected symbol for moment of inertia is the uppercase script J i.e. $\mathcal{J}$. The identity matrix is suffixed with a number of dimensions, i.e. `I4` represents the identity matrix in $\mathbb{R}^4$.

Matrices 4×4 originated from quaternions may be geometrically partitioned into scalar (real), vector (pure imaginary) and cross-product parts [3]. Such partitions are visualized using row and column divider lines throughout this document:

$$\left(\begin{array}{c|ccc} \textit{scalar} & \square & \textit{vector} & \square \\ \hline \square & \square & \square & \square \\ \textit{vector}^{\mathrm{T}} & \square & \textit{cross product} & \square \\ \square & \square & \square & \square \end{array}\right)$$

▼ General symbols and variables used in theorems (definitions)

Hamilton's quaternions q, q_1 and q_2 with real number components (but not with complex number components as biquaternions; see definition of default `$Assumptions` bellow):

```
q = Q[ w, x, y, z ];
```

```
q1 = Q[ w1, x1, y1, z1 ];
```

```
q2 = Q[ w2, x2, y2, z2 ];
```

Identity matrix `I4` in $\mathbb{R}^4$:

```
I4 = IdentityMatrix[4];
```

Common $\mathbb{R}^4$ matrices **A** and **B**:

```
A = ( Aww | Awx  Awy  Awz
      Axw | Axx  Axy  Axz
      Ayw | Ayx  Ayy  Ayz
      Azw | Azx  Azy  Azz );
```

```
B = ( Bww | Bwx  Bwy  Bwz
      Bxw | Bxx  Bxy  Bxz
      Byw | Byx  Byy  Byz
      Bzw | Bzx  Bzy  Bzz );
```

Symmetrical $\mathbb{R}^4$ matrix representing moment of inertia tensor $\mathcal{J}$:

```
𝒥 = ( 1 |  0    0    0
      0 | Ixx  Ixy  Ixz
      0 | Ixy  Iyy  Iyz
      0 | Ixz  Iyz  Izz );
```

Default assumption used for all asserted statements and expressions in this notebook is that previously defined entities q, q_1, q_2, **A**, **B** and $\mathcal{J}$ are on field of $\mathbb{R}$, i.e. that their components are elements in $\mathbb{R}$:

```
$Assumptions = Flatten@{ ToList /@ { q, q1, q2 }, A, B, 𝒥} ∈ Reals;
```

■ Different Types of Multiplications

The ordirary notation for multiplication in *Mathematica* is

A . **B**	multiplication between matrices ("dot" multiplication)
a × **b**	cross-product between 3D vectors
p ** q	quaternion multiplication (both in `Quat.m` and *Mathematica* Quaternions package)

Beside the ordinary notation for multiplication, this document also introduces the following notation based on the center dot (·) operator.

A · **B**	:=	multiplication between matrices
$p \cdot q$	:=	quaternion multiplication
$q \cdot$ **A**	:=	column-wise quaternion-matrix multiplication

Note that the center dot operator in *Mathematica* is without built-in meaning.

▼ Matrix multiplication

Define center dot operator **A** · **B** as matrix multiplication (otherwise, *Mathematica* uses plain dot "." for matrix multiplications).

```
CenterDot[ m__ ?MatrixQ ] := Dot[ m ]
```

▼ Quaternion multiplication

Mathematica uses non-commutative multiply (**) operator as symbol for quaternion multiplication. The `Quat.m` package and this document follow this convention. Here we define also the center dot operator as symbol for quaternion multiplication, just for convinience and esthetics

```
CenterDot[ q__ ?QQ ] := NonCommutativeMultiply[ q ]
```

▼ Column-wise (or horizontal) quaternion-matrix multiplication

Let us now define multiplication between quaternion q and matrix $\mathbf{M}$, where it is assumed that matrix is a row vector of quaternions in columns, i.e. $\mathbf{M} = (\, q_1 \quad q_2 \quad \cdots \quad q_n \,)$, so that quaternion multiplication is done between q and q_j in every column j.

$$q \cdot \left(\begin{array}{c|ccc} q_{11} & q_{12} & & q_{1n} \\ \hline q_{21} & q_{22} & & q_{2n} \\ q_{31} & q_{32} & \cdots & q_{3n} \\ q_{41} & q_{42} & & q_{4n} \end{array}\right) := \left(\begin{array}{c|ccc} \overline{q}_{11} & \overline{q}_{12} & & \overline{q}_{1n} \\ \hline \overline{q}_{21} & \overline{q}_{22} & & \overline{q}_{2n} \\ \overline{q}_{31} & \overline{q}_{32} & \cdots & \overline{q}_{3n} \\ \overline{q}_{41} & \overline{q}_{42} & & \overline{q}_{4n} \end{array}\right)$$

Since the quaternion multiplication is not commutative, there are several cases to distinguish:

a) Left multiplication $q \cdot \mathbf{M}$, where a quaternion in each column in $\mathbf{M}$ is multiplied by the quaternion q from the left:

```
CenterDot[ q_?QQ, mat_?MatrixQ ] := Transpose@
  Table[
   ToList[ q · ToQ@mat⟦All,j⟧ ], {j, 1, Dimensions[mat]⟦2⟧}
  ]
```

b) Right multiplication $\mathbf{M} \cdot q$, where a quaternion in each column in $\mathbf{M}$ is multiplied by the quaternion q from the right:

```
CenterDot[ mat_?MatrixQ, q_?QQ ] := Transpose@
  Table[
   ToList[ ToQ@mat⟦All,j⟧ · q ], {j, 1, Dimensions[mat]⟦2⟧}
  ]
```

c) Compound multiplication from both sides $p \cdot \mathbf{M} \cdot q$, where multiplication of each column in M by the quaternion p from the left and q from the right:

```
CenterDot[ q1_?QQ, mat_?MatrixQ, q2_?QQ ] := Transpose@
  Table[
   ToList[ q1 · ToQ@mat⟦All,j⟧ · q2 ], {j, 1, Dimensions[mat]⟦2⟧}
  ]
```

The multiplication between a quaternion and a matrix is associative, i.e. it holds:

$$q_1 \cdot \mathbf{M} \cdot q_2 == q_1 \cdot (\mathbf{M} \cdot q_2) == (q_1 \cdot \mathbf{M}) \cdot q_2$$

which is proved as a theorem in the following sections.

Note that for a single column matrix $\mathbf{M}$, the multiplication between a quaternion and a matrix falls back to an ordinary quaternion multiplication.

▼ Row-wise (or vertical) quaternion-matrix multiplication

Row-wise quaternion-matrix multiplication can be defined by transposing matrices and using column-wise multiplication:

$$\left(q_1 \cdot \mathbf{M}^{\mathsf{T}} \cdot q_2 \right)^{\mathsf{T}}$$

▼ Conj[] matrix operator

Conj[] operator conjugates only the vector part of the matrix, but not the scalar and the cross product parts:

```
Conj[ q_?MatrixQ ] :=
```

$$\left(\begin{array}{c|ccc} q_{[\![1,1]\!]} & -q_{[\![1,2]\!]} & -q_{[\![1,3]\!]} & -q_{[\![1,4]\!]} \\ \hline -q_{[\![2,1]\!]} & q_{[\![2,2]\!]} & q_{[\![2,3]\!]} & q_{[\![2,4]\!]} \\ -q_{[\![3,1]\!]} & q_{[\![3,2]\!]} & q_{[\![3,3]\!]} & q_{[\![3,4]\!]} \\ -q_{[\![4,1]\!]} & q_{[\![4,2]\!]} & q_{[\![4,3]\!]} & q_{[\![4,4]\!]} \end{array}\right)$$

Note that Conj[] is **not** a conjugate of a quaternion! The conjugate of a quaternion corresponds to the plain transpose of the matrix.

Idea behind quaternion-matrix multiplication

Basic idea behind the quaternion-matrix multiplication comes from an expression:

```
Assert[
 Q[1, 0, 0, 0] == q · Q[1, 0, 0, 0] · q*,
 Assumptions → |q|^2 == 1
]
```

```
⊢ True (after Simplify) ∎
```

for which it is assumed to be also valid when transforming identity matrix **I4** that is equvalent representation of the quaternion *Q*[1, 0, 0, 0] in matrix form, i.e. that there should be kind of multiplication where:

```
Assert[
 (1  0  0  0)^T == q · (1  0  0  0)^T · q*,
 Assumptions → |q|^2 == 1
]
```

```
⊢ True (after Simplify) ∎
```

■ Theorems

▼ Quaternion-matrix multiplication

Quaternion multiplication associativity (sanity-check):

```
q1 · q · q2 == q1 · (q · q2) == (q1 · q) · q2 // Assert
```

```
⊢ True ∎
```

Quaternion-matrix multiplication associativity theorems:

```
q1 · A · q2 == q1 · (A · q2) == (q1 · A) · q2 // Assert
```

```
⊢ True ∎
```

```
q · A · q* == (q · A) · q* // Assert
```

```
⊢ True ∎
```

```
q · A · q* == q · (A · q*) // Assert
```

```
⊢ True ∎
```

```
(q · I4)^T == q* · I4 // Assert
```

```
⊢ True ∎
```

```
(I4 · q)^T == I4 · q* // Assert
```

```
⊢ True ∎
```

```
(q · I4) · A == q · A // Assert
```

```
⊢ True ∎
```

```
(I4 · q) · A == A · q // Assert
```

```
⊢ True ∎
```

```
(q1 · I4) · (I4 · q2) == q1 · I4 · q2 // Assert
```

```
⊢ True ∎
```

```
(I4 · q2) · ( (q1 · I4) · A ) == (q1 · A) · q2 == q1 · A · q2 // Assert
```

⊢ True (after Simplify) ∎

```
(I4 · q2) · (q1 · I4) · A == q1 · A · q2 ⋀
  (I4 · q2) · (q1 · I4) · A == q1 · A · q2 ⋀
  (q1 · I4) · (I4 · q2) · A == q1 · A · q2 ⋀
  (q1 · I4 · q2) · A == q1 · A · q2  // Assert
```

⊢ True (after Simplify) ∎

Conjugations of quaternion-matrix multiplications:

```
Conj[ q · A ] == Conj[A] · q*  // Assert
```

⊢ True ∎

```
Conj[ A · q ] == q* · Conj[A] // Assert
```

⊢ True ∎

```
Conj[ q1 · A · q2] == Conj[ q1 · (A · q2)] == Conj[A · q2] · q1* == q2* · Conj[A] · q1* // Assert
```

⊢ True (after Simplify) ∎

```
Conj[ q · A · q*] == q · Conj[A] · q* // Assert
```

⊢ True (after Simplify) ∎

▼ Left and right rotation matrices

Because the quaternion multiplication is bilinear, it can be expressed in a matrix form, and in two different ways ([1]). Multiplication on the left qp gives $\mathbf{L}_q \cdot \mathbf{p}$ where p is now treated as a 4-dimensional column vector $\mathbf{p}$ and multiplication on the right pq gives $\mathbf{p} \cdot \mathbf{R}_q$.

Let us define $\mathbf{L}_q$ and $\mathbf{R}_q$ as quaternion-multiplication with identity matrix:

```
ClearAll[ L, R, Q ]
```

```
L_q_ := q · I4
R_q_ := I4 · q
Q_q_ := L_q · R_q*
```

Proof that the quaternion multiplication can be decomposed into L&R matrix parts and that $\mathbf{L}_q \, \mathbf{R}_{q^*} \, \mathbf{p}$ corresponds to $q\,p\,q^*$:

```
Assert[
 L_q · R_q* · ToVector[q1] == ToVector[ q · q1 · q* ],
 Assumptions → |q|^2 == 1
]
```

⊢ True (after Simplify) ∎

▼ Properties of L, R and Q matrices

```
L_q* == (L_q)^T  // Assert
```

⊢ True ∎

```
R_q* == (R_q)^T  // Assert
```

⊢ True ∎

```
( q · R_q* ) · A == q · ( R_q* · A ) == ( L_q · A ) · q*  // Assert
```

⊢ True (after Simplify) ∎

```
Qq == Lq · Rq* == (Rq*ᵀ · Lqᵀ)ᵀ == (Rq · Lq*)ᵀ == Rq* · Lq // Assert
```

⊢ True ∎

```
Qq* == Rq · Lq* == Lq* · Rq // Assert
```

⊢ True ∎

```
Qq · Qq* == |q|^4 I4 // Assert
```

⊢ True (after Simplify) ∎

```
Conj[Lq · A] · q* == Conj[ q · (Lq · A) ] // Assert
```

⊢ True (after Simplify) ∎

```
Conj[Lq · A] · q* == Conj[Lq · A] · q* // Assert
```

⊢ True ∎

▼ L & R matrices components

```
Grid@{
  { "Lq", "Lq*", "Rq", "Rq*" },
  { Lq // QForm, Lq* // QForm, Rq // QForm, Rq* // QForm }
 }
```

L_q

$$\begin{pmatrix} w & -x & -y & -z \\ x & w & -z & y \\ y & z & w & -x \\ z & -y & x & w \end{pmatrix}$$

L_{q^*}

$$\begin{pmatrix} w & x & y & z \\ -x & w & z & -y \\ -y & -z & w & x \\ -z & y & -x & w \end{pmatrix}$$

R_q

$$\begin{pmatrix} w & -x & -y & -z \\ x & w & z & -y \\ y & -z & w & x \\ z & y & -x & w \end{pmatrix}$$

R_{q^*}

$$\begin{pmatrix} w & x & y & z \\ -x & w & -z & y \\ -y & z & w & -x \\ -z & -y & x & w \end{pmatrix}$$

▼ Rotation matrix Q components

```
Grid@{
  { "Qq == Lq·Rq* == Rq*·Lq", "Qq* == Lq*·Rq == Rq·Lq*" },
  { Simplify[ Qq, Assumptions → |q|^2 == 1 ] // QForm,
    Simplify[ Qq*, Assumptions → |q|^2 == 1 ] // QForm }
 }
```

$Q_q == L_q \cdot R_{q^*} == R_{q^*} \cdot L_q$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-2y^2-2z^2 & 2xy-2wz & 2(wy+xz) \\ 0 & 2(xy+wz) & 1-2x^2-2z^2 & -2wx+2yz \\ 0 & -2wy+2xz & 2(wx+yz) & 1-2x^2-2y^2 \end{pmatrix}$$

$Q_{q^*} == L_{q^*} \cdot R_q == R_q \cdot L_{q^*}$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-2y^2-2z^2 & 2(xy+wz) & -2wy+2xz \\ 0 & 2xy-2wz & 1-2x^2-2z^2 & 2(wx+yz) \\ 0 & 2(wy+xz) & -2wx+2yz & 1-2x^2-2y^2 \end{pmatrix}$$

▼ Shoemake's rotation matrix from quaternion

Classical Shoemake's form of the rotation matrix from a quaternion is given as (a transformation matrix based on the Cayley-Klein parameters; see [2]):

```
RotationMatrix4[ q ] // QForm
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-2(y^2+z^2) & 2xy-2wz & 2wy+2xz \\ 0 & 2xy+2wz & 1-2(x^2+z^2) & -2wx+2yz \\ 0 & -2wy+2xz & 2wx+2yz & 1-2(x^2+y^2) \end{pmatrix}$$

▼ Scalar, vector and cross-product parts of L&R matrices

Scalar part:

$$\frac{1}{2}\left(L_q + L_{q^*}\right) == \frac{1}{2}\left(R_q + R_{q^*}\right) == \left(\begin{array}{c|ccc} w & 0 & 0 & 0 \\ \hline 0 & w & 0 & 0 \\ 0 & 0 & w & 0 \\ 0 & 0 & 0 & w \end{array}\right) \text{// Assert}$$

⊢ True ∎

Vector part:

$$\frac{1}{2}\left(L_{q^*} - R_q\right) == \frac{1}{2}\left(R_{q^*} - L_q\right) == \left(\begin{array}{c|ccc} 0 & x & y & z \\ \hline -x & 0 & 0 & 0 \\ -y & 0 & 0 & 0 \\ -z & 0 & 0 & 0 \end{array}\right) \text{// Assert}$$

⊢ True ∎

$$\frac{1}{2}\left(L_q - R_{q^*}\right) == \frac{1}{2}\left(R_q - L_{q^*}\right) == \left(\begin{array}{c|ccc} 0 & x & y & z \\ \hline -x & 0 & 0 & 0 \\ -y & 0 & 0 & 0 \\ -z & 0 & 0 & 0 \end{array}\right)^{\mathsf{T}} \text{// Assert}$$

⊢ True ∎

Cross product part:

$$\frac{1}{2}\left(L_q - R_q\right) == \left(\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -z & y \\ 0 & z & 0 & -x \\ 0 & -y & x & 0 \end{array}\right) \text{// Assert}$$

⊢ True ∎

$$\frac{1}{2}\left(L_{q^*} - R_{q^*}\right) == \left(\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \hline 0 & 0 & -z & y \\ 0 & z & 0 & -x \\ 0 & -y & x & 0 \end{array}\right)^{\mathsf{T}} \text{// Assert}$$

⊢ True ∎

$\mathbf{Q}\cdot\mathbf{Q} + 2\,\mathbf{Q}\cdot$(vector part) form.
Note that the vector part is a pure imaginary part of a quaternion.

$$\text{ToQ@0} == q^2 + 2\,q \cdot \text{ToQ@Im}[q^*] - |q|^2 \text{ // Assert}$$

⊢ True ∎

$$\text{Assert}\Big[\, Q_q == L_q \cdot L_q + 2\,L_q \cdot \left(\begin{array}{c|ccc} 0 & x & y & z \\ \hline -x & 0 & 0 & 0 \\ -y & 0 & 0 & 0 \\ -z & 0 & 0 & 0 \end{array}\right)$$

$$== L_q \cdot L_q + 2\,L_q \cdot \left(\frac{1}{2}\left(R_{q^*} - L_q\right)\right)$$

$$== L_q \cdot R_{q^*}$$

$$\Big]$$

⊢ True (after Simplify) ∎

$$\frac{1}{2}\,L_{q^{-1}} \cdot \left(L_q \cdot L_q - Q_q\right) == \frac{1}{2}\left(L_q - R_{q^*}\right) \text{ // Assert}$$

⊢ True (after Simplify) ∎

Equality of rotation matrix from quaternion and Euler-Rodrigues' formula

Euler-Rodrigues' formula for the rotation matrix corresponding to a rotation by an angle θ about a fixed axis specified by the unit vector **u**.

Reference: http://mathworld.wolfram.com/RodriguesRotationFormula.html

```
EulerRodrigues[θ_, {x_, y_, z_}] :=
 Cos[θ] {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}} + (1 - Cos[θ]) {{x x, y x, z x}, {x y, y y, z y}, {x z, y z, z z}} + Sin[θ] {{0, -z, y}, {z, 0, -x}, {-y, x, 0}}
```

Proof that the rotation matrix from a quaternion (as defined in Quat.m package) is equivalent to Euler-Rodrigues' formula.

```
Block[{x, y, z, θ, q},
 q = ToQ$AngleAxis[θ, {x, y, z}];
 Assert[
  RotationMatrix[q] == EulerRodrigues[θ, {x, y, z}],
  Assumptions →
   -π ≤ θ ≤ π ⋀ x^2 + y^2 + z^2 == 1 ⋀ {x, y, z, θ} ∈ Reals
 ]
]
```

⊢ True (after Simplify) ∎

▼ Rotation matrix theorems

```
RM = RotationMatrix4[q];
RM // QForm
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-2\left(y^2+z^2\right) & 2xy-2wz & 2wy+2xz \\ 0 & 2xy+2wz & 1-2\left(x^2+z^2\right) & -2wx+2yz \\ 0 & -2wy+2xz & 2wx+2yz & 1-2\left(x^2+y^2\right) \end{pmatrix}$$

```
Assert[
 RM == Qq == Lq · Rq* ⋀
  RM^T == Qq* == Rq*^T · Lq^T,
 Assumptions → |q|^2 == 1
]
```

⊢ True (after Simplify) ∎

```
Assert[
 RM · ToVector[q1]
       == Lq · Rq* · ToVector[q1]
       == ToVector[q · q1 · q*],
 Assumptions → |q|^2 == 1
]
```

⊢ True (after Simplify) ∎

```
Assert[
 RM · A · RM^T
       == Q_q · A · Q_q*
       == L_q · R_q* · A · L_q* · R_q
       == ( q · ( q · A · q* )^T · q* )^T,
 Assumptions → |q|^2 == 1
]
```

⊢ True (after Simplify) ∎

```
Assert[
 I4
       == (q · I4 · q*) · I4 · (q · I4 · q*)^T
       == (L_q · R_q*) · I4 · (L_q · R_q*)^T
       == L_q · R_q* · I4 · R_q*^T · L_q^T
       == L_q · (R_q* · I4 · R_q*^T) · L_q^T,
 Assumptions → |q|^2 == 1
]
```

⊢ True (after Simplify) ∎

```
Assert[
 RM · A · RM^T
       == (q · I4 · q*) · A · (q · I4 · q*)^T
       == (L_q · R_q*) · A · (L_q · R_q*)^T
       == L_q · R_q* · A · R_q*^T · L_q^T
       == L_q · (R_q* · A · R_q*^T) · L_q^T,
 Assumptions → |q|^2 == 1
]
```

⊢ True (after Simplify) ∎

Proof that both $\mathbf{Q}_q$ and $\mathbf{R}_M$ are orthogonal matrices:

```
Assert[
 Q_q · Q_q^T
       == (L_q · R_q*) · (L_q · R_q*)^T
       == L_q · R_q* · (R_q*)^T · (L_q)^T
       == L_q · R_q* · R_q · L_q*
       == I4,
 Assumptions → |q|^2 == 1
]
```

⊢ True (after Simplify) ∎

```
Assert[
 Qq · Qq^T
       == Qq · Qq*
       == RM · RM^T
       == I4,
 Assumptions → |q|^2 == 1
]
```

⊢ True (after Simplify) ∎

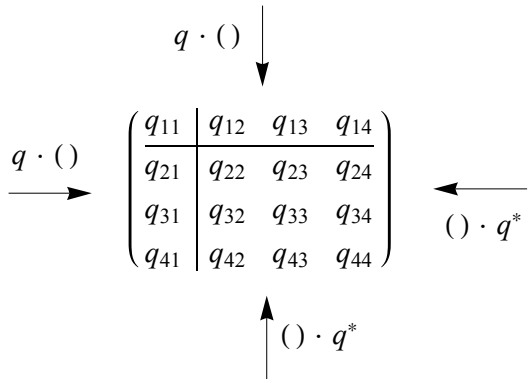
▼ Quaternionic meaning of R(*q*)·A·R(*q*)^T

The hint how quaternion multplication affects a (moment of inertia or any other) tensor **A** can be seen from the earlier proven theorem:

```
Assert[
 RM · A · RM^T == ( q · ( q · A · q* )^T · q* )^T,
 Assumptions → |q|^2 == 1
]
```

⊢ True (after Simplify) ∎

which shows that the tensor **A** is rotated, at first, by rotating quaternions across each column $\mathbf{A}_c = q \cdot \mathbf{A} \cdot q^*$, and at second, by rotating quaternions across each row $\mathbf{A}_{rc} = \left(q \cdot (\mathbf{A}_c)^T \cdot q^*\right)^T$.



The meaning of the compound multiplications across columns and rows is extracting and affecting individual components of quaternions embedded from the previous rotations of the tensor. The moment of inertia tensor can always be reduced to its principal axes in diagonal form. In a matrix representation of a quaternion, the diagonal of the matrix contains the repeated scalar (real) part of the quaternion. (For any quaternion q there exist quaternion p such that qp is scalar.) The principal moment of inertia, on the other hand, has on its diagonal three different scalars in general case, which gives origin to three different quaternions that are *embedded* and independently shuffled across-columns/rows during rotations.

■ References

[1] Eberly, David H. with a contrib. by Shoemake, Ken - *Game Physics*, Morgan Kaufmann, 2004, Chapter 10 "Quaternions", pp. 507-544, http://books.google.se/books?id=a9SzFHPJ0mwC

[2] Shoemake, Ken - *Uniform random rotations*, in: D. Kirk (Ed.), *Graphics Gems III*, Academic Press, London, 1992, pp. 124-132, http://books.google.se/books?id=xmW_u3mQLmQC

[3] Shoemake, Ken - *Quaternions*, Department of Computer and Information Science, University of Pennsylvania, 2005-10-07, downloaded from C+MS course CS171 at Caltech: http://courses.cms.caltech.edu/cs171/quatut.pdf