Quaternion

5.3

Generated by Doxygen 1.8.14

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Module Documentation

5.1 Models

Modules

- Utils

5.1.1 Detailed Description

5.2 Utils

Modules

- Quaternion

5.2.1 Detailed Description

5.3 Quaternion

Files

- file quat.hh

  Define the quaternion class.
- file quat_inline.hh

  Define inline methods for the quaternion class.
- file quat_messages.hh

  Define the class QuatMessages, the class that specifies the message IDs used in the quaternion model.
- file quat.cc

  Define basic methods for the quaternion class.
- file quat_from_mat.cc

  Define left_quat_from_transformation (), which computes the parent-to-child left quaternion from the input transformation matrix.
- file quat_messages.cc

  Implement the class QuatMessages.
- file quat_norm.cc

  Define quaternion normalization methods.
- file quat_to_eigenrot.cc

  Define Quaternion::left_quat_to_eigen_rotation, which computes the eigen rotation corresponding to a quaternion.
- file quat_to_mat.cc

  Define Quaternion::left_quat_to_transformation, which computes the parent- to-child transformation matrix from the parent-to-child left quaternion.

Namespaces

- jeod

  Namespace jeod.

5.3.1 Detailed Description

Chapter 6

Namespace Documentation

6.1 jeod Namespace Reference

Namespace jeod.

Data Structures

- class Quaternion

  Implement quaternions to the extent needed to represent orientations.
- class QuatMessages

  Specifies the message IDs used in the orbital elements model.

6.1.1 Detailed Description

Namespace jeod.

Chapter 7

Data Structure Documentation

7.1 jeod::Quaternion Class Reference

Implement quaternions to the extent needed to represent orientations.

`#include <quat.hh>`

Public Member Functions

- Quaternion ()

  Construct a quaternion; default constructor.
- Quaternion (const double s)

  Construct a pure real quaternion.
- Quaternion (const double s, const double v[3])

  Construct from a scalar and a vector.
- Quaternion (const double arr[4])

  Construct from a double array.
- Quaternion (const double T[3][3])

  Construct a left transformation unit quaternion.
- void set_to_zero ()

  Set all components of the quaternion to zero.
- void make_identity ()

  Make the quaternion represent an identity transform.
- operator double * ()

  Make a quaternion look like a double array.
- void copy_to (double arr[4]) const

  Copy a quaternion to a four vector, with the scalar part copied to arr[0] and the vector part to arr[1] to arr[3].
- void copy_from (const double arr[4])

  Copy a quaternion from a four vector, with the scalar part of the quaternion in arr[0] and the vector part in arr[1] to arr[3].
- void scale (const double scale)

  Scale the quaternion by a real.
- void scale (const double scale, Quaternion &quat) const

  Scale the quaternion by a real, leaving original intact.
- double norm_sq () const

Compute the square of the norm of the quaternion.

- void normalize ()

  Normalize the quaternion, making the scalar part of the quaternion non-negative.
- void normalize (Quaternion &quat) const

  Form the normalized quaternion, leaving original intact.
- void normalize_integ ()

  Normalize the quaternion, but do not make the scalar part non-negative.
- void normalize_integ (Quaternion &quat) const

  Form the normalized quaternion, leaving original intact.
- void conjugate ()

  Replace the quaternion with its conjugate.
- void conjugate (Quaternion &quat) const

  Form the conjugate of a quaternion, leaving original intact.
- void multiply (const Quaternion &quat, Quaternion &prod) const

  *Post-multiply this quaternion by another quaternion: prod = this * quat.*
- void multiply (const Quaternion &quat)

  *Post-multiply this quaternion by another quaternion: this = this * quat.*
- void conjugate_multiply (const Quaternion &quat, Quaternion &prod) const

  *Post-multiply this quaternion's conjugate by another quaternion: prod = conj(this) * quat.*
- void conjugate_multiply (const Quaternion &quat)

  *Post-multiply this quaternion's conjugate by another quaternion: this = conj(this) * quat.*
- void multiply_conjugate (const Quaternion &quat, Quaternion &prod) const

  *Post-multiply this quaternion by another's conjugate: prod = this * conj(quat).*
- void multiply_conjugate (const Quaternion &quat)

  *Post-multiply this quaternion by another's conjugate: this = this * conj(quat).*
- void multiply_left (const Quaternion &quat, Quaternion &prod) const

  *Pre-multiply this quaternion by another quaternion: prod = quat * this.*
- void multiply_left (const Quaternion &quat)

  *Pre-multiply this quaternion by another quaternion: this = quat * this.*
- void multiply_left_conjugate (const Quaternion &quat, Quaternion &prod) const

  *Pre-multiply this quaternion by another's conjugate: prod = conj(quat) * this.*
- void multiply_left_conjugate (const Quaternion &quat)

  *Pre-multiply this quaternion by another's conjugate: this = conj(quat) * this.*
- void multiply_vector_left (const double vec[3], Quaternion &prod) const

  *Pre-multiply this quaternion by a pure imaginary quaternion, the latter represented by a vector: prod = [0, vec] * quat.*
- void multiply_vector_right (const double vec[3], Quaternion &prod) const

  *Post-multiply this quaternion by a pure imaginary quaternion, the latter represented by a vector: prod = quat * [0, vec].*
- void left_quat_from_transformation (const double T[3][3])

  Compute the parent-to-child left quaternion from the input transformation matrix.
- void left_quat_to_transformation (double T[3][3]) const

  Compute the parent-to-child transformation matrix from the parent-to-child left quaternion.
- void left_quat_from_eigen_rotation (double eigen_angle, const double eigen_axis[3])

  Construct the quaternion corresponding to an eigen rotation.
- void left_quat_to_eigen_rotation (double *eigen_angle, double eigen_axis[3]) const

  Compute the eigen rotation corresponding to a quaternion.
- void eigen_compare (const Quaternion &compare_to, double *eigen_angle, double eigen_axis[3]) const

  *Compute eigen decomposition of this*conj(quat).*
- void left_quat_transform (const double vec_in[3], double vec_out[3]) const

  Transform a vector.
- void compute_left_quat_deriv (const double ang_vel[3], Quaternion &qdot) const

Compute the time derivative of a left quaternion.

- void compute_left_quat_second_deriv (const double ang_vel[3], const double ang_acc[3], Quaternion &qdot) const

  Compute the time derivative of a left quaternion.

Static Public Member Functions

- static void normalize_integ (double arr[4])

  Normalize the quaternion, but do not make the scalar part non-negative.
- static void compute_left_quat_deriv (const double quat[4], const double ang_vel[3], double qdot[4])

  Compute the time derivative of a left quaternion.
- static void compute_left_quat_second_deriv (const double quat[4], const double ang_vel[3], const double ang_acc[3], double qddot[4])

  Compute the second time derivative of a left quaternion.
- static Quaternion compute_slerp (Quaternion &q1, Quaternion &q2, const double T)

  Compute the minimum interpolation quaternion between a start quarternion and end quaternion.

Data Fields

- double scalar {1.0}

  The scalar, or real, part of the quaternion.
- double vector [3] {}

  The vectorial, or imaginary, part of the quaternion.

Friends

- class InputProcessor
- void init_attrjeod__Quaternion ()

7.1.1 Detailed Description

Implement quaternions to the extent needed to represent orientations.

Definition at line 86 of file quat.hh.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Quaternion() [1/5]

```
jeod::Quaternion::Quaternion ( )
```

Construct a quaternion; default constructor.

Definition at line 53 of file quat.cc.

References scalar, and vector.

7.1.2.2 Quaternion() [2/5]

```
jeod::Quaternion::Quaternion (
            const double real_part )  [explicit]
```

Construct a pure real quaternion.

Parameters

in	*real_part*	Scalar

Definition at line 65 of file quat.cc.

7.1.2.3 Quaternion() [3/5]

```
jeod::Quaternion::Quaternion (
            const double s,
            const double v[3] )  [inline]
```

Construct from a scalar and a vector.

Parameters

in	*s*	Scalar part
in	*v*	Vector part

Definition at line 83 of file quat_inline.hh.

References vector.

7.1.2.4 Quaternion() [4/5]

```
jeod::Quaternion::Quaternion (
            const double arr[4] )  [inline], [explicit]
```

Construct from a double array.

Parameters

in	*arr*	Quaternion source

Definition at line 95 of file quat_inline.hh.

References copy_from().

7.1.2.5 Quaternion() [5/5]

```
jeod::Quaternion::Quaternion (
            const double T[3][3] )  [explicit]
```

Construct a left transformation unit quaternion.

Parameters

in	*T*	Transformation matrix

Definition at line 74 of file quat.cc.

References left_quat_from_transformation().

7.1.3 Member Function Documentation

7.1.3.1 compute_left_quat_deriv() [1/2]

```
void jeod::Quaternion::compute_left_quat_deriv (
            const double ang_vel[3],
            Quaternion & qdot ) const  [inline]
```

Compute the time derivative of a left quaternion.

Parameters

in	*ang_vel*	Angular velocity Units: r/s
out	*qdot*	Quaternion derivative

Definition at line 466 of file quat_inline.hh.

References multiply_vector_left().

7.1.3.2 compute_left_quat_deriv() [2/2]

```
void jeod::Quaternion::compute_left_quat_deriv (
            const double quat[4],
            const double ang_vel[3],
            double qdot[4] )  [inline], [static]
```

Compute the time derivative of a left quaternion.

Parameters

in	*quat*	Quaternion as 4-vector
in	*ang_vel*	Angular velocity Units: r/s
out	*qdot*	Derivative as 4-vector

Definition at line 495 of file quat_inline.hh.

7.1.3.3 compute_left_quat_second_deriv() [1/2]

```
void jeod::Quaternion::compute_left_quat_second_deriv (
            const double ang_vel[3],
            const double ang_acc[3],
            Quaternion & qddot ) const  [inline]
```

Compute the time derivative of a left quaternion.

Parameters

in	*ang_vel*	Angular velocity Units: r/s
in	*ang_acc*	Angular acceleration Units: r/s2
out	*qddot*	Quaternion 2nd deriv

Definition at line 479 of file quat_inline.hh.

References multiply_left().

7.1.3.4 compute_left_quat_second_deriv() [2/2]

```
void jeod::Quaternion::compute_left_quat_second_deriv (
            const double quat[4],
            const double ang_vel[3],
            const double ang_acc[3],
            double qddot[4] )  [inline], [static]
```

Compute the second time derivative of a left quaternion.

Parameters

in	*quat*	Quaternion as 4-vector
in	*ang_vel*	Angular velocity Units: r/s
in	*ang_acc*	Angular acceleration Units: r/s2
out	*qddot*	2nd derivative as 4-vector

Definition at line 511 of file quat_inline.hh.

7.1.3.5 compute_slerp()

```
Quaternion jeod::Quaternion::compute_slerp (
            Quaternion & q1,
            Quaternion & q2,
            const double T )  [static]
```

Compute the minimum interpolation quaternion between a start quarternion and end quaternion.

Parameters

in	*q1*	Starting quaternion
in	*q2*	Ending quaternion
in	*T*	Interpolation coefficient between 0.0 and 1.0 representing a rotational scale factor between the intial and final quaternion. When the compute_slerp method is used in a loop to rotate an object from a start and end orientation, a smaller step or change in T results in a smoother object rotation

Definition at line 91 of file quat.cc.

References normalize(), scalar, jeod::QuatMessages::undefined, and vector.

7.1.3.6 conjugate() [1/2]

```
void jeod::Quaternion::conjugate ( )  [inline]
```

Replace the quaternion with its conjugate.

Definition at line 215 of file quat_inline.hh.

References vector.

7.1.3.7 conjugate() [2/2]

```
void jeod::Quaternion::conjugate (
            Quaternion & quat ) const  [inline]
```

Form the conjugate of a quaternion, leaving original intact.

Parameters

out	*quat*	Conjugated quaternion

Definition at line 224 of file quat_inline.hh.

References scalar, and vector.

7.1.3.8 conjugate_multiply() [1/2]

```
void jeod::Quaternion::conjugate_multiply (
            const Quaternion & quat,
            Quaternion & prod ) const  [inline]
```

Post-multiply this quaternion's conjugate by another quaternion: prod = conj(this) * quat.

Parameters

in	*quat*	Right multiplicand
out	*prod*	Quaternion product

Definition at line 270 of file quat_inline.hh.

References scalar, and vector.

7.1.3.9 conjugate_multiply() [2/2]

```
void jeod::Quaternion::conjugate_multiply (
            const Quaternion & quat )  [inline]
```

Post-multiply this quaternion's conjugate by another quaternion: this = conj(this) * quat.

Parameters

in	*quat*	Right multiplicand

Definition at line 283 of file quat_inline.hh.

References scalar, and vector.

7.1.3.10 copy_from()

```
void jeod::Quaternion::copy_from (
            const double arr[4] )  [inline]
```

Copy a quaternion from a four vector, with the scalar part of the quaternion in arr[0] and the vector part in arr[1] to arr[3].

Parameters

in	*arr*	Quaternion source

Definition at line 136 of file quat_inline.hh.

References scalar, and vector.

Referenced by Quaternion().

7.1.3.11 copy_to()

```
void jeod::Quaternion::copy_to (
             double arr[4] ) const  [inline]
```

Copy a quaternion to a four vector, with the scalar part copied to arr[0] and the vector part to arr[1] to arr[3].

Parameters

out	*arr*	Copy of quaternion

Definition at line 123 of file quat_inline.hh.

References scalar, and vector.

7.1.3.12 eigen_compare()

```
void jeod::Quaternion::eigen_compare (
             const Quaternion & quat,
             double * eigen_angle,
             double eigen_axis[3] ) const  [inline]
```

Compute eigen decomposition of this∗conj(quat).

Parameters

in	*quat*	Quaternion to compare to
out	*eigen_angle*	Eigen angle Units: r
out	*eigen_axis*	Eigen axis

Definition at line 454 of file quat_inline.hh.

References left_quat_to_eigen_rotation(), and multiply_conjugate().

7.1.3.13 left_quat_from_eigen_rotation()

```
void jeod::Quaternion::left_quat_from_eigen_rotation (
             double eigen_angle,
             const double eigen_axis[3] )  [inline]
```

Construct the quaternion corresponding to an eigen rotation.

Parameters

in	*eigen_angle*	Eigen angle Units: r
in	*eigen_axis*	Eigen axis

Definition at line 149 of file quat_inline.hh.

References scalar, and vector.

7.1.3.14 left_quat_from_transformation()

```
void jeod::Quaternion::left_quat_from_transformation (
             const double T[3][3] )
```

Compute the parent-to-child left quaternion from the input transformation matrix.

Assumptions and Limitations

- Matrix is orthonormal.

Parameters

in	*T*	Transformation matrix

Definition at line 115 of file quat_from_mat.cc.

References scalar, and vector.

Referenced by Quaternion().

7.1.3.15 left_quat_to_eigen_rotation()

```
void jeod::Quaternion::left_quat_to_eigen_rotation (
             double * eigen_angle,
             double eigen_axis[3] ) const
```

Compute the eigen rotation corresponding to a quaternion.

Assumptions and Limitations

- Quaternion is normalized.

Parameters

out	*eigen_angle*	Eigen angle Units: r
out	*eigen_axis*	Eigen axis

Definition at line 47 of file quat_to_eigenrot.cc.

References scalar, and vector.

Referenced by eigen_compare().

7.1.3.16 left_quat_to_transformation()

```
void jeod::Quaternion::left_quat_to_transformation (
             double T[3][3] ) const
```

Compute the parent-to-child transformation matrix from the parent-to-child left quaternion.

Assumptions and Limitations

- Quaternion is normalized.

Parameters

out	*T*	Transformation matrix

Definition at line 82 of file quat_to_mat.cc.

References scalar, and vector.

7.1.3.17 left_quat_transform()

```
void jeod::Quaternion::left_quat_transform (
             const double vec_in[3],
             double vec_out[3] ) const  [inline]
```

Transform a vector.

Parameters

in	*vec_in*	Vector to be transformed
out	*vec_out*	Transformed vector

Definition at line 433 of file quat_inline.hh.

References scalar, and vector.

7.1.3.18 make_identity()

```
void jeod::Quaternion::make_identity ( )  [inline]
```

Make the quaternion represent an identity transform.

Definition at line 112 of file quat_inline.hh.

References scalar, and vector.

7.1.3.19 multiply() [1/2]

```
void jeod::Quaternion::multiply (
            const Quaternion & quat,
            Quaternion & prod ) const  [inline]
```

Post-multiply this quaternion by another quaternion: prod = this * quat.

Parameters

in	*quat*	Right multiplicand
out	*prod*	Quaternion product

Definition at line 236 of file quat_inline.hh.

References scalar, and vector.

7.1.3.20 multiply() [2/2]

```
void jeod::Quaternion::multiply (
            const Quaternion & quat )  [inline]
```

Post-multiply this quaternion by another quaternion: this = this * quat.

Parameters

in	*quat*	Right multiplicand

Definition at line 249 of file quat_inline.hh.

References scalar, and vector.

7.1.3.21 multiply_conjugate() [1/2]

```
void jeod::Quaternion::multiply_conjugate (
            const Quaternion & quat,
            Quaternion & prod ) const  [inline]
```

Post-multiply this quaternion by another's conjugate: prod = this * conj(quat).

Parameters

in	*quat*	Right multiplicand
out	*prod*	Quaternion product

Definition at line 304 of file quat_inline.hh.

References scalar, and vector.

Referenced by eigen_compare().

7.1.3.22 multiply_conjugate() [2/2]

```
void jeod::Quaternion::multiply_conjugate (
            const Quaternion & quat )  [inline]
```

Post-multiply this quaternion by another's conjugate: this = this * conj(quat).

Parameters

in	*quat*	Right multiplicand

Definition at line 317 of file quat_inline.hh.

References scalar, and vector.

7.1.3.23 multiply_left() [1/2]

```
void jeod::Quaternion::multiply_left (
            const Quaternion & quat,
            Quaternion & prod ) const  [inline]
```

Pre-multiply this quaternion by another quaternion: prod = quat * this.

Parameters

in	*quat*	Left multiplicand
out	*prod*	Quaternion product

Definition at line 338 of file quat_inline.hh.

References scalar, and vector.

Referenced by compute_left_quat_second_deriv().

7.1.3.24 multiply_left() [2/2]

```
void jeod::Quaternion::multiply_left (
            const Quaternion & quat )  [inline]
```

Pre-multiply this quaternion by another quaternion: this = quat * this.

Parameters

in	*quat*	Left multiplicand

Definition at line 351 of file quat_inline.hh.

References scalar, and vector.

7.1.3.25 multiply_left_conjugate() [1/2]

```
void jeod::Quaternion::multiply_left_conjugate (
            const Quaternion & quat,
            Quaternion & prod ) const  [inline]
```

Pre-multiply this quaternion by another's conjugate: prod = conj(quat) * this.

Parameters

in	*quat*	Left multiplicand
out	*prod*	Quaternion product

Definition at line 372 of file quat_inline.hh.

References scalar, and vector.

7.1.3.26 multiply_left_conjugate() [2/2]

```
void jeod::Quaternion::multiply_left_conjugate (
            const Quaternion & quat )  [inline]
```

Pre-multiply this quaternion by another's conjugate: this = conj(quat) * this.

Parameters

in	*quat*	Left multiplicand

Definition at line 385 of file quat_inline.hh.

References scalar, and vector.

7.1.3.27 multiply_vector_left()

```
void jeod::Quaternion::multiply_vector_left (
            const double vec[3],
            Quaternion & prod ) const  [inline]
```

Pre-multiply this quaternion by a pure imaginary quaternion, the latter represented by a vector: prod = [0, vec] $*$ quat.

Parameters

in	*vec*	Right multiplicand
out	*prod*	Quaternion product

Definition at line 407 of file quat_inline.hh.

References scalar, and vector.

Referenced by compute_left_quat_deriv().

7.1.3.28 multiply_vector_right()

```
void jeod::Quaternion::multiply_vector_right (
            const double vec[3],
            Quaternion & prod ) const  [inline]
```

Post-multiply this quaternion by a pure imaginary quaternion, the latter represented by a vector: prod = quat $*$ [0, vec].

Parameters

in	*vec*	Right multiplicand
out	*prod*	Quaternion product

Definition at line 421 of file quat_inline.hh.

References scalar, and vector.

7.1.3.29 norm_sq()

```
double jeod::Quaternion::norm_sq ( ) const  [inline]
```

Compute the square of the norm of the quaternion.

Returns

Square of the norm of the quaternion

Definition at line 187 of file quat_inline.hh.

References scalar, and vector.

Referenced by normalize(), and normalize_integ().

7.1.3.30 normalize() [1/2]

```
void jeod::Quaternion::normalize ( )
```

Normalize the quaternion, making the scalar part of the quaternion non-negative.

Definition at line 47 of file quat_norm.cc.

References norm_sq(), scalar, and scale().

Referenced by compute_slerp(), and normalize().

7.1.3.31 normalize() [2/2]

```
void jeod::Quaternion::normalize (
            Quaternion & quat ) const  [inline]
```

Form the normalized quaternion, leaving original intact.

Parameters

out	*quat*	Normalized quaternion

Definition at line 196 of file quat_inline.hh.

References normalize().

7.1.3.32 normalize_integ() [1/3]

```
void jeod::Quaternion::normalize_integ ( )
```

Normalize the quaternion, but do not make the scalar part non-negative.

Definition at line 83 of file quat_norm.cc.

References norm_sq(), and scale().

Referenced by normalize_integ().

7.1.3.33 normalize_integ() [2/3]

```
void jeod::Quaternion::normalize_integ (
            Quaternion & quat ) const  [inline]
```

Form the normalized quaternion, leaving original intact.

Parameters

out	*quat*	Normalized quaternion

Definition at line 206 of file quat_inline.hh.

References normalize_integ().

7.1.3.34 normalize_integ() [3/3]

```
void jeod::Quaternion::normalize_integ (
            double quat[4] )  [static]
```

Normalize the quaternion, but do not make the scalar part non-negative.

Parameters

quat	Quaternion to be normalized.

Definition at line 108 of file quat_norm.cc.

7.1.3.35 operator double *()

```
jeod::Quaternion::operator double * ( )  [inline]
```

Make a quaternion look like a double array.

Definition at line 128 of file quat.hh.

7.1.3.36 scale() [1/2]

```
void jeod::Quaternion::scale (
            const double fact )  [inline]
```

Scale the quaternion by a real.

Parameters

in	*fact*	Scale factor

Definition at line 166 of file quat_inline.hh.

References scalar, and vector.

Referenced by normalize(), and normalize_integ().

7.1.3.37 scale() [2/2]

```
void jeod::Quaternion::scale (
            const double fact,
            Quaternion & quat ) const  [inline]
```

Scale the quaternion by a real, leaving original intact.

Parameters

in	*fact*	Scale factor
out	*quat*	Scaled quaternion

Definition at line 177 of file quat_inline.hh.

References scalar, and vector.

7.1.3.38 set_to_zero()

```
void jeod::Quaternion::set_to_zero ( )  [inline]
```

Set all components of the quaternion to zero.

Definition at line 103 of file quat_inline.hh.

References scalar, and vector.

7.1.4 Friends And Related Function Documentation

7.1.4.1 init_attrjeod__Quaternion

```
void init_attrjeod__Quaternion ( )  [friend]
```

7.1.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 88 of file quat.hh.

7.1.5 Field Documentation

7.1.5.1 scalar

```
double jeod::Quaternion::scalar {1.0}
```

The scalar, or real, part of the quaternion.

trick_units(–)

Definition at line 93 of file quat.hh.

Referenced by compute_slerp(), conjugate(), conjugate_multiply(), copy_from(), copy_to(), left_quat_from_↩ eigen_rotation(), left_quat_from_transformation(), left_quat_to_eigen_rotation(), left_quat_to_transformation(), left_quat_transform(), make_identity(), multiply(), multiply_conjugate(), multiply_left(), multiply_left_conjugate(), multiply_vector_left(), multiply_vector_right(), norm_sq(), normalize(), Quaternion(), scale(), and set_to_zero().

7.1.5.2 vector

```
double jeod::Quaternion::vector[3] {}
```

The vectorial, or imaginary, part of the quaternion.

trick_units(–)

Definition at line 98 of file quat.hh.

Referenced by compute_slerp(), conjugate(), conjugate_multiply(), copy_from(), copy_to(), left_quat_from_↩ eigen_rotation(), left_quat_from_transformation(), left_quat_to_eigen_rotation(), left_quat_to_transformation(), left_quat_transform(), make_identity(), multiply(), multiply_conjugate(), multiply_left(), multiply_left_conjugate(), multiply_vector_left(), multiply_vector_right(), norm_sq(), Quaternion(), scale(), and set_to_zero().

The documentation for this class was generated from the following files:

- quat.hh
- quat_inline.hh
- quat.cc
- quat_from_mat.cc
- quat_norm.cc
- quat_to_eigenrot.cc
- quat_to_mat.cc

7.2 jeod::QuatMessages Class Reference

Specifies the message IDs used in the orbital elements model.

`#include <quat_messages.hh>`

Public Member Functions

- QuatMessages ()=delete
- QuatMessages (const QuatMessages &)=delete
- QuatMessages & operator= (const QuatMessages &)=delete

Static Public Attributes

- static const char * undefined = "utils/quaternion/" "undefined"

  Issued an undefined behaviour is encountered.
- static const char * invalid_entry = "utils/quaternion/" "invalid_entry"

  Issued when function input is invalid.

Friends

- class InputProcessor
- void init_attrjeod__QuatMessages ()

7.2.1 Detailed Description

Specifies the message IDs used in the orbital elements model.

Definition at line 81 of file quat_messages.hh.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 QuatMessages() [1/2]

```
jeod::QuatMessages::QuatMessages ( )  [delete]
```

7.2.2.2 QuatMessages() [2/2]

```
jeod::QuatMessages::QuatMessages (
            const QuatMessages &  )  [delete]
```

7.2.3 Member Function Documentation

7.2.3.1 operator=()

```
QuatMessages& jeod::QuatMessages::operator= (
            const QuatMessages &  )  [delete]
```

7.2.4 Friends And Related Function Documentation

7.2.4.1 init_attrjeod__QuatMessages

```
void init_attrjeod__QuatMessages ( )  [friend]
```

7.2.4.2 InputProcessor

```
friend class InputProcessor  [friend]
```

Definition at line 83 of file quat_messages.hh.

7.2.5 Field Documentation

7.2.5.1 invalid_entry

```
char const * jeod::QuatMessages::invalid_entry = "utils/quaternion/" "invalid_entry"  [static]
```

Issued when function input is invalid.

trick_units(–)

Definition at line 93 of file quat_messages.hh.

7.2.5.2 undefined

```
char const * jeod::QuatMessages::undefined = "utils/quaternion/" "undefined"  [static]
```

Issued an undefined behaviour is encountered.

trick_units(–)

Definition at line 88 of file quat_messages.hh.

Referenced by jeod::Quaternion::compute_slerp().

The documentation for this class was generated from the following files:

- quat_messages.hh
- quat_messages.cc

Chapter 8

File Documentation

8.1 quat.cc File Reference

Define basic methods for the quaternion class.

```
#include "utils/math/include/numerical.hh"
#include "utils/math/include/vector3.hh"
#include "../include/quat.hh"
#include "../include/quat_messages.hh"
#include "utils/message/include/message_handler.hh"
#include <cmath>
#include <fstream>
#include <iomanip>
```

Namespaces

- jeod

  Namespace jeod.

8.1.1 Detailed Description

Define basic methods for the quaternion class.

8.2 quat.hh File Reference

Define the quaternion class.

```
#include <cstdlib>
#include "utils/sim_interface/include/jeod_class.hh"
#include "quat_inline.hh"
```

Data Structures

- class jeod::Quaternion

  Implement quaternions to the extent needed to represent orientations.

Namespaces

- jeod

  Namespace jeod.

8.2.1 Detailed Description

Define the quaternion class.

8.3 quat_from_mat.cc File Reference

Define left_quat_from_transformation (), which computes the parent-to-child left quaternion from the input transformation matrix.

```
#include <cmath>
#include "../include/quat.hh"
```

Namespaces

- jeod

  Namespace jeod.

8.3.1 Detailed Description

Define left_quat_from_transformation (), which computes the parent-to-child left quaternion from the input transformation matrix.

8.4 quat_inline.hh File Reference

Define inline methods for the quaternion class.

```
#include <cmath>
#include "quat.hh"
#include "utils/math/include/vector3.hh"
```

Namespaces

- jeod

  Namespace jeod.

8.4.1 Detailed Description

Define inline methods for the quaternion class.

8.5 quat_messages.cc File Reference

Implement the class QuatMessages.

```
#include "utils/message/include/make_message_code.hh"
#include "../include/quat_messages.hh"
```

Namespaces

- jeod

  Namespace jeod.

Macros

- #define MAKE_QUAT_MESSAGE_CODE(id) JEOD_MAKE_MESSAGE_CODE(QuatMessages, "utils/quaternion/", id)

8.5.1 Detailed Description

Implement the class QuatMessages.

8.5.2 Macro Definition Documentation

8.5.2.1 MAKE_QUAT_MESSAGE_CODE

```
#define MAKE_QUAT_MESSAGE_CODE(
              id ) JEOD_MAKE_MESSAGE_CODE(QuatMessages, "utils/quaternion/", id)
```

Definition at line 43 of file quat_messages.cc.

8.6 quat_messages.hh File Reference

Define the class QuatMessages, the class that specifies the message IDs used in the quaternion model.

```
#include "utils/sim_interface/include/jeod_class.hh"
```

Data Structures

- class jeod::QuatMessages

  Specifies the message IDs used in the orbital elements model.

Namespaces

- jeod

  Namespace jeod.

8.6.1 Detailed Description

Define the class QuatMessages, the class that specifies the message IDs used in the quaternion model.

8.7 quat_norm.cc File Reference

Define quaternion normalization methods.

```
#include <cmath>
#include "../include/quat.hh"
```

Namespaces

- jeod

  Namespace jeod.

8.7.1 Detailed Description

Define quaternion normalization methods.

8.8 quat_to_eigenrot.cc File Reference

Define Quaternion::left_quat_to_eigen_rotation, which computes the eigen rotation corresponding to a quaternion.

```
#include <cmath>
#include "utils/math/include/vector3.hh"
#include "../include/quat.hh"
```

Namespaces

- jeod

  Namespace jeod.

8.8.1 Detailed Description

Define Quaternion::left_quat_to_eigen_rotation, which computes the eigen rotation corresponding to a quaternion.

8.9 quat_to_mat.cc File Reference

Define Quaternion::left_quat_to_transformation, which computes the parent- to-child transformation matrix from the parent-to-child left quaternion.

```
#include "utils/math/include/vector3.hh"
#include "../include/quat.hh"
```

Namespaces

- jeod

  Namespace jeod.

8.9.1 Detailed Description

Define Quaternion::left_quat_to_transformation, which computes the parent- to-child transformation matrix from the parent-to-child left quaternion.

Index