THUNDER

Generated by Doxygen 1.8.5

Thu Sep 13 2018 13:32:23

Contents

1	File	Index			1
	1.1	File Lis	st		1
2	File	Docum	entation		3
	2.1	include	e/Complex.	h File Reference	3
		2.1.1	Detailed	Description	4
		2.1.2	Function	Documentation	4
			2.1.2.1	ABS	4
			2.1.2.2	COMPLEX	4
			2.1.2.3	COMPLEX_POLAR	4
			2.1.2.4	CONJUGATE	5
			2.1.2.5	gsl_real_imag_sum	5
			2.1.2.6	IMAG	5
			2.1.2.7	operator*	5
			2.1.2.8	operator*	6
			2.1.2.9	operator*=	6
			2.1.2.10	operator+	6
			2.1.2.11	operator+=	6
			2.1.2.12	operator	7
			2.1.2.13	operator-=	7
			2.1.2.14	operator/	7
			2.1.2.15	operator/=	7
			2.1.2.16	REAL	8
	2.2	include	e/Geometry	y/Euler.h File Reference	8
		2.2.1	Detailed	Description	9
		2.2.2	Function	Documentation	9
			2.2.2.1	alignZ	9
			2.2.2.2	angle	10
			2.2.2.3	angle	10
			2.2.2.4	angle	10
			2225	direction	10

iv CONTENTS

	2.2.2.6	quaternion	10
	2.2.2.7	quaternion	11
	2.2.2.8	quaternion	11
	2.2.2.9	quaternion_conj	11
	2.2.2.10	quaternion_mul	11
	2.2.2.11	randRotate2D	12
	2.2.2.12	randRotate3D	12
	2.2.2.13	reflect3D	12
	2.2.2.14	rotate2D	12
	2.2.2.15	rotate2D	12
	2.2.2.16	rotate3D	12
	2.2.2.17	rotate3D	13
	2.2.2.18	rotate3D	13
	2.2.2.19	rotate3DX	13
	2.2.2.20	rotate3DY	13
	2.2.2.21	rotate3DZ	13
	2.2.2.22	swingTwist	14
Index			15

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

 $include/ {\color{red}Complex.h}$

Complex.h defines complex number related operations like +,-,*,/,|a|, $|a|^2$ and so on 3 include/Geometry/Euler.h

Euler.h contains several functions, for operations of quaternions, converting between Euler angles, rotation matrices and unit quaternions and sampling rotation matrices from even distribution

2 File Index

Chapter 2

File Documentation

2.1 include/Complex.h File Reference

Complex.h defines complex number related operations like $+,-,*,/,|a|, |a|^2$ and so on.

```
#include <gsl/gsl_complex.h>
#include <gsl/gsl_complex_math.h>
#include <math.h>
#include "Config.h"
#include "Precision.h"
#include "Typedef.h"
```

Functions

- Complex COMPLEX POLAR (const RFLOAT phi)
- Complex CONJUGATE (const Complex &a)

Get the conjugate result based on value a.

RFLOAT ABS (const Complex &a)

Calculate the |a| of complex number a.

- RFLOAT **ABS2** (const Complex &a)
- Complex COMPLEX (RFLOAT a, RFLOAT b)

Construct a complex number with a as real part and b as image part.

RFLOAT REAL (const Complex &a)

Get the real part of the complex number a.

RFLOAT IMAG (const Complex &a)

Get the image part of the complex number a.

• RFLOAT gsl_real_imag_sum (const Complex &a)

Calcuate the sum of the complex number a's real part and image part.

• Complex operator+ (const Complex &a, const Complex &b)

Implement the add operation between two complex numbers, e.g. c = a + b, where a and b are complex numbers.

Complex operator- (const Complex &a, const Complex &b)

Implement the sub operation between two complex numbers, e.g. c = a - b, where a and b are complex numbers.

• Complex operator* (const Complex &a, const Complex &b)

Implement the mul operation between two complex numbers, e.g. c = a * b, where a and b are complex numbers.

Complex operator/ (const Complex &a, const Complex &b)

Implement the div operation between two complex numbers, e.g. c = a / b, where a and b are complex numbers.

void operator+= (Complex &a, const Complex &b)

Implement the += operation between two complex numbers, e.g. a += b, where a and b are complex numbers.

void operator= (Complex &a, const Complex &b)

Implement the -= operation between two complex numbers, e.g. a -= b, where a and b are complex numbers.

void operator*= (Complex &a, const Complex &b)

Implement the *= operation between two complex numbers, e.g. a *= b, where a and b are complex numbers.

void operator/= (Complex &a, const Complex &b)

Implement the /= operation between two complex numbers, e.g. a /= b, where a and b are complex numbers.

Complex operator* (const Complex a, const RFLOAT x)

Implement the mul operation between a complex number and a RFLOAT number, e.g. c = a + b, where a is a complex number and b is a RFLOAT number.

- Complex **operator*** (const RFLOAT x, const Complex a)
- void **operator***= (Complex &a, const RFLOAT x)
- Complex operator/ (const Complex a, const RFLOAT x)
- void **operator**/= (Complex &a, const RFLOAT x)

2.1.1 Detailed Description

Complex.h defines complex number related operations like $+,-,*,/,|a|, |a|^2$ and so on.

2.1.2 Function Documentation

2.1.2.1 RFLOAT ABS (const Complex & a) [inline]

Calculate the |a| of complex number a.

Returns

The |a| of complex number a.

Parameters

in	а	The number whose a needs to be calculated

2.1.2.2 Complex COMPLEX (RFLOAT a, RFLOAT b) [inline]

Construct a complex number with a as real part and b as image part.

Returns

A initialized complex number.

Parameters

in	а	Real part
in	b	Image part

2.1.2.3 Complex COMPLEX_POLAR (const RFLOAT phi) [inline]

Get the complex number representation c, given the angle ϕ in polar coordinate.

Returns

Complex polar representation.

in	phi	ϕ - Angle value ϕ to be converted

2.1.2.4 Complex CONJUGATE (const Complex & a) [inline]

Get the conjugate result based on value a.

Returns

Conjugate of a.

Parameters

-			
	in	а	Complex number whose conjuate value needs to be returned

2.1.2.5 RFLOAT gsl_real_imag_sum (const Complex & a) [inline]

Calcuate the sum of the complex number a's real part and image part.

Returns

The sum of a's real part and image part

Parameters

ſ	in	3	Complex number to be operated
	T11	а	Complex number to be operated

2.1.2.6 RFLOAT IMAG (const Complex & a) [inline]

Get the image part of the complex number a.

Returns

Complex number a's image part

Parameters

in a Complex number to be operated

2.1.2.7 Complex operator* (const Complex & a, const Complex & b) [inline]

Implement the mul operation between two complex numbers, e.g. c = a * b, where a and b are complex numbers.

Returns

the result of c = a * b.

in a First operand used to perform sub operation between two complex numbers
--

in	b	Second operand used to perform sub operation between two complex nu	mbers
----	---	---	-------

2.1.2.8 Complex operator* (const Complex a, const RFLOAT x) [inline]

Implement the mul operation between a complex number and a RFLOAT number, e.g. c = a + b, where a is a complex number and b is a RFLOAT number.

Returns

the result of c = a * b.

Parameters

in	а	First operand with type of complex used to perform mul operation.
in	X	Second operand with type of RFLOAT used to perform mul operation.

2.1.2.9 void operator*= (Complex & a, const Complex & b) [inline]

Implement the *= operation between two complex numbers, e.g. a *= b, where a and b are complex numbers.

Returns

the result of a *= b.

Parameters

in	а	First operand used to perform *= operation between two complex numbers
in	b	Second operand used to perform *= operation between two complex numbers

2.1.2.10 Complex operator+ (const Complex & a, const Complex & b) [inline]

Implement the add operation between two complex numbers, e.g. c = a + b, where a and b are complex numbers.

Returns

the result of c = a + b.

Parameters

in	а	First operand used to perform add operation between two complex numbers
in	b	Second operand used to perform add operation between two complex num-
		bers

2.1.2.11 void operator+= (Complex & a, const Complex & b) [inline]

Implement the += operation between two complex numbers, e.g. a += b, where a and b are complex numbers.

Returns

the result of a += b.

i	n	а	First operand used to perform += operation between two complex numbers
i	n	b	Second operand used to perform += operation between two complex numbers

2.1.2.12 Complex operator-(const Complex & a, const Complex & b) [inline]

Implement the sub operation between two complex numbers, e.g. c = a - b, where a and b are complex numbers.

Returns

the result of c = a - b.

Parameters

in	а	First operand used to perform sub operation between two complex numbers
in	b	Second operand used to perform sub operation between two complex numbers

2.1.2.13 void operator-= (Complex & a, const Complex & b) [inline]

Implement the -= operation between two complex numbers, e.g. a -= b, where a and b are complex numbers.

Returns

the result of a -= b.

Parameters

in	а	First operand used to perform -= operation between two complex numbers
in	b	Second operand used to perform -= operation between two complex numbers

2.1.2.14 Complex operator/ (const Complex & a, const Complex & b) [inline]

Implement the div operation between two complex numbers, e.g. c = a / b, where a and b are complex numbers.

Returns

the result of c = a / b.

Parameters

in	а	First operand used to perform div operation between two complex numbers
in	b	Second operand used to perform div operation between two complex numbers

2.1.2.15 void operator/= (Complex & a, const Complex & b) [inline]

Implement the /= operation between two complex numbers, e.g. a /= b, where a and b are complex numbers.

Returns

the result of a = b.

Parameters

in	а	First operand used to perform /= operation between two complex numbers
in	b	Second operand used to perform /= operation between two complex numbers

```
2.1.2.16 RFLOAT REAL (const Complex & a) [inline]
```

Get the real part of the complex number a.

Returns

Complex number a's real part

Parameters

in	а	Complex number to be operated
----	---	-------------------------------

2.2 include/Geometry/Euler.h File Reference

Euler.h contains several functions, for operations of quaternions, converting between Euler angles, rotation matrices and unit quaternions and sampling rotation matrices from even distribution.

```
#include <cmath>
#include <gsl/gsl_math.h>
#include "Macro.h"
#include "Typedef.h"
#include "Precision.h"
#include "Random.h"
#include "Functions.h"
```

Functions

void quaternion_mul (dvec4 &dst, const dvec4 &a, const dvec4 &b)

Calculate the product of two quaternions q_1 and q_2 .

· dvec4 quaternion_conj (const dvec4 &quat)

Calculate the conjugate quaternion of a quaternion.

• void angle (double &phi, double &theta, const dvec3 &src)

Calculate ϕ and θ given a certain direction \mathbf{v} .

• void angle (double &phi, double &theta, double &psi, const dmat33 &src)

Calculate ϕ , θ and ψ of the rotation represented by the rotation matrix \mathbf{R} .

void angle (double &phi, double &theta, double &psi, const dvec4 &src)

Calculate ϕ , θ and ψ of the rotation represented by the unit quaternion \mathbf{q} .

• void quaternion (dvec4 &dst, const double phi, const double theta, const double psi)

Calculate the unit quaternion \mathbf{q} for representing the rotation, given 3 Euler angles ϕ , θ and ψ .

• void quaternion (dvec4 &dst, const double phi, const dvec3 &axis)

Calculate the unit quaternion \mathbf{q} for representing the rotation, given the rotation axis \mathbf{r} and the rotation angle around this axis ϕ .

void quaternion (dvec4 &dst, const dmat33 &src)

Calculate the unit quaternion \mathbf{q} for representing the rotation, given the rotation matrix \mathbf{R} .

void rotate2D (dmat22 &dst, const dvec2 &vec)

Calculate the rotation matrix (2D) \mathbf{R} , which rotates the unit vector $\mathbf{v_0} = \{1,0\}$ to the given unit vector \mathbf{v} .

void rotate2D (dmat22 &dst, const double phi)

Calculate the rotation matrix (2D) \mathbf{R} , given the rotation angle ϕ .

• void direction (dvec3 &dst, const double phi, const double theta)

Caclulate the unit direction vector \mathbf{v} , given the rotation angle ϕ and θ .

void rotate3D (dmat33 &dst, const double phi, const double theta, const double psi)

Caclulate the rotation matrix **R**, given the rotation angle ϕ , θ and ψ .

void rotate3D (dmat33 &dst, const dvec4 &src)

Calculate the rotation matrix \mathbf{R} , given the unit quaternion \mathbf{q} which represents this rotation.

void rotate3DX (dmat33 &dst, const double phi)

Calculate the rotation matrix \mathbf{R} which represents the rotation along X-axis with rotation angle ϕ .

• void rotate3DY (dmat33 &dst, const double phi)

Calculate the rotation matrix \mathbf{R} which represents the rotation along Y-axis with rotation angle ϕ .

void rotate3DZ (dmat33 &dst, const double phi)

Calculate the rotation matrix \mathbf{R} which represents the rotation along Z-axis with rotation angle ϕ .

void alignZ (dmat33 &dst, const dvec3 &vec)

Calculate the rotation matrix \mathbf{R} which aligns a direction vector \mathbf{v} to Z-axis.

void rotate3D (dmat33 &dst, const double phi, const dvec3 &axis)

Calculate the rotation matrix \mathbf{R} which represents the rotation along the axis \mathbf{v} with rotation angle ϕ .

• void reflect3D (dmat33 &dst, const dvec3 &plane)

Calculate the transformation matrix \mathbf{M} of reflection against a certian plane, which is represented by its normal vector \mathbf{n} .

void swingTwist (dvec4 &swing, dvec4 &twist, const dvec4 &src, const dvec3 &vec)

Calculate the two quaternions \mathbf{q}_s and \mathbf{q}_t , which represent swing and twist along axis \mathbf{v} respectively, representing the rotation represented by quaternion \mathbf{q} .

void randRotate2D (dmat22 &rot)

Sample a 2D rotation matrix **R** from even distribution.

void randRotate3D (dmat33 &rot)

Sample a 3D rotation matrix R from even distribution.

2.2.1 Detailed Description

Euler.h contains several functions, for operations of quaternions, converting between Euler angles, rotation matrices and unit quaternions and sampling rotation matrices from even distribution. Quaternions are a number system that extends the complex numbers. Unit quaternions provide a convenient mathematical notation for representing rotations of objects in 3D. Compared to Euler angles, they are simpler to compose and aovid the problem of glimbal lock. Compared to rotation matrices, they are more compact and more efficient. Moroever, unlike Euler angles, unit quaternions do not rely on the choosing and order of the rotation axes.

To be noticed, Euler angles in this file follow the standard of ZXZ Euler system. In other words, Euler angle set $\{\phi, \theta, \psi\}$ stands for rotating along Z axis with ϕ , followed by rotating along X axis with θ , and followed by rotating along Z axis with ψ .

2.2.2 Function Documentation

2.2.2.1 void alignZ (dmat33 & dst, const dvec3 & vec)

Calculate the rotation matrix ${\bf R}$ which aligns a direction vector ${\bf v}$ to Z-axis.

out	dst	R
in	vec	v

2.2.2.2 void angle (double & phi, double & theta, const dvec3 & src)

Calculate ϕ and θ given a certain direction \mathbf{v} .

v must be a unit vector. Output value ϕ ranges $[0, 2\pi)$, and θ ranges $[0, \pi]$.

Parameters

out	phi	φ
out	theta	θ
in	src	v

2.2.2.3 void angle (double & phi, double & theta, double & psi, const dmat33 & src)

Calculate ϕ , θ and ψ of the rotation represented by the rotation matrix \mathbf{R} .

R must be an orthogonal matirx and determinant of which equals to 1. In other words, $RR^T = I$ and $\det A = 1$. Output value ϕ ranges $[0, 2\pi)$, θ ranges $[0, \pi]$, and ψ ranges $[0, 2\pi)$.

Parameters

out	phi	φ
out	theta	$\mid heta \mid$
out	psi	Ψ
in	src	R

2.2.2.4 void angle (double & phi, double & theta, double & psi, const dvec4 & src)

Calculate ϕ , θ and ψ of the rotation represented by the unit quaternion \mathbf{q} .

Parameters

out	phi	φ
out	theta	θ
out	psi	Ψ
in	src	q

2.2.2.5 void direction (dvec3 & dst, const double phi, const double theta)

Caclulate the unit direction vector \mathbf{v} , given the rotation angle ϕ and θ .

Parameters

out	dst	v
in	phi	ϕ
in	theta	θ

2.2.2.6 void quaternion (dvec4 & dst, const double phi, const double theta, const double psi)

Calculate the unit quaternion q for representing the rotation, given 3 Euler angles ϕ , θ and ψ .

out	dst	q
in	phi	ϕ
in	theta	θ
in	psi	Ψ

2.2.2.7 void quaternion (dvec4 & dst, const double phi, const dvec3 & axis)

Calculate the unit quaternion \mathbf{q} for representing the rotation, given the rotation axis \mathbf{r} and the rotation angle around this axis ϕ .

This rotation axis \mathbf{r} must be a unit vector, while the rotation angle ϕ ranges $(-\infty, +\infty)$.

Parameters

out	dst	q
in	phi	ϕ
in	axis	r

2.2.2.8 void quaternion (dvec4 & dst, const dmat33 & src)

Calculate the unit quaternion q for representing the rotation, given the rotation matrix R.

Parameters

out	dst	q	
in	src	R	

2.2.2.9 dvec4 quaternion_conj (const dvec4 & quat)

Calculate the conjugate quaternion of a quaternion.

Returns

the conjugate quaternion

Parameters

1		La mustamia n
ın	quat	a quaternion

2.2.2.10 void quaternion_mul (dvec4 & dst, const dvec4 & a, const dvec4 & b)

Calculate the product of two quaternions q_1 and q_2 .

Assuming that $\mathbf{q_1} = (w_1, x_1, y_1, z_1)$ and $\mathbf{q_2} = (w_2, x_2, y_2, z_2)$, the product can be calculated as

$$\begin{pmatrix} w_1 \\ x_1 \\ y_1 \\ z_1 \end{pmatrix} \times \begin{pmatrix} w_2 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \\ w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2 \\ w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2 \\ w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2 \end{pmatrix}$$

Parameters

out	dst	product, a quaternion
in	а	left multiplier, q ₁
in	b	right multiplier, \mathbf{q}_2

2.2.2.11 void randRotate2D (dmat22 & rot)

Sample a 2D rotation matrix ${\bf R}$ from even distribution.

Parameters

out	rot	R

2.2.2.12 void randRotate3D (dmat33 & rot)

Sample a 3D rotation matrix ${\bf R}$ from even distribution.

Parameters

out	rot	R

2.2.2.13 void reflect3D (dmat33 & dst, const dvec3 & plane)

Calculate the transformation matrix M of reflection against a certian plane, which is represented by its normal vector \mathbf{n} .

Parameters

out	dst	M
in	plane	n

2.2.2.14 void rotate2D (dmat22 & dst, const dvec2 & vec)

Calculate the rotation matrix (2D) \mathbf{R} , which rotates the unit vector $\mathbf{v_0} = \{1,0\}$ to the given unit vector \mathbf{v} .

Parameters

out	dst	R
in	vec	v

2.2.2.15 void rotate2D (dmat22 & dst, const double phi)

Calculate the rotation matrix (2D) ${f R}$, given the rotation angle ϕ .

Parameters

out	dst	R
in	phi	φ

2.2.2.16 void rotate3D (dmat33 & dst, const double phi, const double theta, const double psi)

Caclulate the rotation matrix **R**, given the rotation angle ϕ , θ and ψ .

out	dst	R
in	phi	ϕ
in	theta	θ
in	psi	Ψ

2.2.2.17 void rotate3D (dmat33 & dst, const dvec4 & src)

Calculate the rotation matrix \mathbf{R} , given the unit quaternion \mathbf{q} which represents this rotation.

Parameters 4 6 1

out	dst	R
in	src	q

2.2.2.18 void rotate3D (dmat33 & dst, const double phi, const dvec3 & axis)

Calculate the rotation matrix R which represents the rotation along the axis v with rotation angle ϕ .

Parameters

out	dst	R
in	phi	ϕ
in	axis	v

2.2.2.19 void rotate3DX (dmat33 & dst, const double phi)

Calculate the rotation matrix ${f R}$ which represents the rotation along X-axis with rotation angle ϕ .

Parameters

out	dst	
in	phi	

2.2.2.20 void rotate3DY (dmat33 & dst, const double phi)

Calculate the rotation matrix ${f R}$ which represents the rotation along Y-axis with rotation angle ϕ .

Parameters

out	dst	R
in	phi	φ

2.2.2.21 void rotate3DZ (dmat33 & dst, const double phi)

Calculate the rotation matrix ${f R}$ which represents the rotation along Z-axis with rotation angle ϕ .

out	dst	R

	, .	1
าก	ו וחמ	σ
T-11	Pili	Ψ

2.2.2.22 void swingTwist (dvec4 & swing, dvec4 & twist, const dvec4 & src, const dvec3 & vec)

Calculate the two quaternions q_s and q_t , which represent swing and twist along axis v respectively, representing the rotation represented by quaternion q.

out	swing	$\mathbf{q_s}$
out	twist	q _t
in	src	q
in	vec	v

Index

ABS	Complex.h, 5
Complex.h, 4	IMAG
alignZ Euler.h, 9	Complex.h, 5
	include/Complex.h, 3
angle Euler.h, 10	include/Geometry/Euler.h, 8
Luiei.ii, 10	•
COMPLEX	operator*
Complex.h, 4	Complex.h, 5, 6
COMPLEX_POLAR	operator*=
Complex.h, 4	Complex.h, 6
CONJUGATE	operator+
Complex.h, 5	Complex.h, 6
Complex.h	operator+=
ABS, 4	Complex.h, 6
COMPLEX, 4	operator-
COMPLEX_POLAR, 4	Complex.h, 7
CONJUGATE, 5	operator-=
gsl_real_imag_sum, 5	Complex.h, 7
IMAG, 5	operator/ Complex.h, 7
operator*, 5, 6	operator/=
operator*=, 6	Complex.h, 7
operator+, 6	Complex.ii, 7
operator+=, 6	quaternion
operator-, 7	Euler.h, 10, 11
operator-=, 7 operator/, 7	quaternion_conj
operator/=, 7	Euler.h, 11
REAL, 8	quaternion_mul
HEAL, 0	Euler.h, 11
direction	DEAL
Euler.h, 10	REAL
	Complex.h, 8
Euler.h	randRotate2D
alignZ, 9	Euler.h, 12
angle, 10	randRotate3D
direction, 10	Euler.h, 12 reflect3D
quaternion, 10, 11 quaternion_conj, 11	Euler.h, 12
quaternion_conj, 11 quaternion_mul, 11	rotate2D
randRotate2D, 12	Euler.h, 12
randRotate3D, 12	rotate3D
reflect3D, 12	Euler.h, 12, 13
rotate2D, 12	rotate3DX
rotate3D, 12, 13	Euler.h, 13
rotate3DX, 13	rotate3DY
rotate3DY, 13	Euler.h, 13
rotate3DZ, 13	rotate3DZ
swingTwist, 14	Euler.h, 13
gsl_real_imag_sum	swingTwist

16 INDEX

Euler.h, 14