poly3D

Richard J. Mathar, arXiv:0809.2369 [math-ph]

Generated by Doxygen 1.8.2

Mon Oct 15 2012 12:12:56

Contents

1 Todo List **2**

2 Data Structure Index **2**

2.1 Data Structures 2

3 File Index **2**

3.1 File List 2

4 Data Structure Documentation **2**

4.1 point3D Class Reference 2

4.1.1 Detailed Description 3

4.1.2 Constructor & Destructor Documentation 3

4.1.3 Member Function Documentation 3

4.1.4 Field Documentation 4

4.2 poly3D Class Reference 4

4.2.1 Detailed Description 5

4.2.2 Constructor & Destructor Documentation 5

4.2.3 Member Function Documentation 6

4.2.4 Field Documentation 7

4.3 trino3D Class Reference 7

4.3.1 Detailed Description 8

4.3.2 Constructor & Destructor Documentation 8

4.3.3 Member Function Documentation 8

4.3.4 Field Documentation 9

4.4 Zern3DRadi Class Reference 9

4.4.1 Detailed Description 9

4.4.2 Constructor & Destructor Documentation 10

4.4.3 Field Documentation 10

5 File Documentation **10**

5.1 poly3D.cpp File Reference 10

5.1.1 Function Documentation 10

5.2 poly3D.h File Reference 12

5.2.1 Macro Definition Documentation 12

5.2.2 Function Documentation 12

Index **14**

1 Todo List

Global poly3D::operator+= (const trino3D &oth)

allow for annihilation of coefficients and elimination of associated products..

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

point3D 2

poly3D 4

trino3D 7

Zern3DRadi 9

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

poly3D.cpp 10

poly3D.h 12

4 Data Structure Documentation

4.1 point3D Class Reference

Public Member Functions

- point3D ()
- point3D (const double X, const double Y, const double Z)
- point3D (const double cart[TURB3D_DIM])
- point3D (const point3D &orig)
- double dist () const
- double dist (const point3D &oth) const
- void scale (const double radius)
- void normalize (const double newleng)
- void phithet (double rtp[3]) const
- point3D & operator= (const point3D &right)
- point3D & operator*= (const double c)
- point3D & operator+= (const point3D &right)
- point3D & operator-= (const point3D &right)

Data Fields

- double xyz [3]

4.1.1 Detailed Description

A point in a 3D domain.

Since

2008-09-22

4.1.2 Constructor & Destructor Documentation

4.1.2.1 point3D::point3D ( )

Default Ctor at the origin of coordinates

4.1.2.2 point3D::point3D (const double *X*, const double *Y*, const double *Z*)

Ctor from individual cartesian coordinates

4.1.2.3 point3D::point3D (const double *cart[TURB3D_DIM]*)

Ctor with given 3D coordinates

4.1.2.4 point3D::point3D (const point3D & *orig*)

Copy constructor.

4.1.3 Member Function Documentation

4.1.3.1 double point3D::dist ( ) const

Distance to the origin

4.1.3.2 double point3D::dist (const point3D & *oth*) const

Distance to another point

4.1.3.3 void point3D::normalize (const double *newleng*)

Scale components so the new length equals a given number

Parameters

newleng	the distance to the origin (or vector length) on return

4.1.3.4 point3D & point3D::operator*= (const double *c*)

Multiply by a constant in the sense that this is a vector shortened/lengthed.

Parameters

c	the constant to multply with

Returns

the new vector that results

4.1.3.5 point3D & point3D::operator+= (const point3D & *right*)

Add a vector to define the new location.

Parameters

right	the vector to add

Returns

the new vector that results

4.1.3.6 point3D & point3D::operator-= (const point3D & *right*)

Subtract a point to define the difference vector.

Parameters

right	

Returns

the difference vector that results

4.1.3.7 point3D & point3D::operator= (const point3D & *right*)

Assigment Operator.

Parameters

right	the right hand side of the assignment

Returns

the new vector that results

4.1.3.8 void point3D::phithet (double *rtp[3]*) const

Convert point to spherical angular coordinates.

Parameters

rtp	radius, theta and phi on return. r = sqrt(x^2+y^2+z^2) ; x=r sin(theta) cos(phi) y=r sin(theta) sin(phi) z=r cos(theta)

4.1.3.9 void point3D::scale (const double *radius*)

Scale the points such that those with distance 'radius' appear to be at distance 1

4.1.4 Field Documentation

4.1.4.1 double point3D::xyz[3]

the 3 Cartesian coordinates.

4.2 poly3D Class Reference

Public Member Functions

- poly3D ()

- poly3D (int nl)
- poly3D (const int l, const int m, const bool cnots)
- poly3D & operator∗= (const double cof)
- poly3D (const int n, const int l)
- poly3D (const int n, const int l, const int m, const bool cnots)
- double at (const point3D &pt) const
- void gradat (const point3D &pt, double gr[3]) const
- int hastype (const int e[TURB3D_DIM]) const
- int hastype (const trino3D &t) const
- poly3D & operator+= (const trino3D &oth)
- poly3D & operator∗= (const trino3D &t)
- poly3D & operator+= (const poly3D &oth)
- poly3D & operator∗= (const poly3D &oth)

Data Fields

- vector< trino3D > compo

4.2.1 Detailed Description

Polynomial in 3D

Since

2008-09-24

4.2.2 Constructor & Destructor Documentation

4.2.2.1 poly3D::poly3D ( )

Default ctor.

4.2.2.2 poly3D::poly3D (int *nl*)

Constructor of a radial polynomial r^nl with r^2=x^2+y^2+z^2

Parameters

nl	the power of r. Must be an even integer. nl must be an even integer which is not checked in this revision.

4.2.2.3 poly3D::poly3D (const int *l*, const int *m*, const bool *cnots*)

Constructor for a vector Harmonics r^l Y_l^m.

Parameters

l	the angular momentum quantum number
m	the magnetic quantum number in the range 0<=m<=l
cnots	true if this is a cosine type, false if a sine type there is no check that m=0 is only used with cnots equal to true

4.2.2.4 poly3D::poly3D (const int *n*, const int *l*)

Constructor for the radial Zernike polynomials R_n^l(r)/r^l.

Parameters

n	the main power
l	the angular momentum quantum number

4.2.2.5 poly3D::poly3D (const int *n*, const int *l*, const int *m*, const bool *cnots*)

Constructor for a general Zernike term R_n^l Y_lm

Parameters

n	the main power
l	the angular momentum quantum number
m	the magnetic quantum number in the range 0<=m<=l
cnots	true if this is a cosine type, false if a sine type there is no check that m=0 is only used with cnots equal to true

4.2.3 Member Function Documentation

4.2.3.1 double poly3D::at (const point3D & *pt*) const

Evaluate it at a point in 3D param pt the location of the 3D point return the value, which is the sum over all terms.

4.2.3.2 void poly3D::gradat (const point3D & *pt*, double *gr[3]*) const

Gradient evaluation at a specified point in 3D space.

Parameters

pt	the point at which the gradient is computed
gr	on return the three components of the gradient

4.2.3.3 int poly3D::hastype (const int *e[TURB3D_DIM]*) const

Test whether a trinomial of an exponential signature is already one of the terms.

Parameters

e	the three nonzero exponents of the reference

Returns

the index of the first term of that signature, or -1 if not found.

4.2.3.4 int poly3D::hastype (const trino3D & *t*) const

Test whether a trinomial of an exponential signature is already one of the terms.

Parameters

t	the trinomial of the reference

Returns

the index of the first term of that signature, or -1 if not found.

4.2.3.5 poly3D & poly3D::operator*= (const double *cof*)

Multiply by a constant

Parameters

c	the constant to multply with

Returns

the product that results

4.2.3.6 poly3D & poly3D::operator*= (const trino3D & *t*)

Multiply all components with the other trinomial term.

Parameters

oth	the trinomial on the right hand side of the equation.

4.2.3.7 poly3D & poly3D::operator*= (const poly3D & *oth*)

Multiply by another polynomial.

Parameters

oth	the polynomial on the right hand side of the equation.

4.2.3.8 poly3D & poly3D::operator+= (const trino3D & *oth*)

Add another trinomial term to this one

Parameters

oth	the trinomial on the right hand side of the equation.

Todo allow for annihilation of coefficients and elimination of associated products..

4.2.3.9 poly3D & poly3D::operator+= (const poly3D & *oth*)

Add another polynomial to this one

Parameters

oth	the polynomial on the right hand side of the equation.

4.2.4 Field Documentation

4.2.4.1 vector<trino3D> poly3D::compo

The individual terms.

4.3 trino3D Class Reference

Public Member Functions

- trino3D (const double c, const int exx, const int exy, const int exz)
- double at (const point3D &pt) const
- void gradat (const point3D &pt, double gr[3]) const
- bool istype (const int e[3]) const

- bool istype (const trino3D &oth) const
- trino3D & operator∗= (const double c)

Data Fields

- double coef
- int expo [TURB3D_DIM]

4.3.1 Detailed Description

A term in a 3D polynomial

Since

2008-09-24

4.3.2 Constructor & Destructor Documentation

4.3.2.1 trino3D::trino3D (const double *c*, const int *exx*, const int *exy*, const int *exz*)

Ctor

Parameters

c	the coeffient in front
exx	the power of x
exy	the power of y
exz	the power of z

4.3.3 Member Function Documentation

4.3.3.1 double trino3D::at (const point3D & *pt*) const

Evaluation a a specific point in 3D space

4.3.3.2 void trino3D::gradat (const point3D & *pt*, double *gr[3]*) const

Gradient evaluation at a specified point in 3D space.

Parameters

pt	the point at which the gradient is computed
gr	on return the three components of the gradient

4.3.3.3 bool trino3D::istype (const int *e[3]*) const

Test whether the polynomial is of some type of given exponents. Check whether the current polynomial is compatible with respect to addition.

Parameters

p	the three non-negative exponents of the reference type

Returns

true of the current type matches all three exponents of the reference type.

4.3.3.4 bool trino3D::istype (const trino3D & *oth*) const

Test whether the polynomial is of some type of given exponents. Check whether the current polynomial is compatible with respect to addition.

Parameters

oth	the trinomial to match against

Returns

true of the current type matches all three exponents of the reference type.

4.3.3.5 trino3D & trino3D::operator∗= (const double *c*)

Multiply by a constant

Parameters

c	the constant to multply with

Returns

the product that results

4.3.4 Field Documentation

4.3.4.1 double trino3D::coef

The expansion coefficient

4.3.4.2 int trino3D::expo[TURB3D_DIM]

The non-negative powers of x, y and z

4.4 Zern3DRadi Class Reference

Public Member Functions

- Zern3DRadi (const int lowIdx, const int upIdx)

Data Fields

- int n
- int l
- vector< double > coef
- int alpha

4.4.1 Detailed Description

Zernike 3D radial polynomial

Since

2008-09-25

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Zern3DRadi::Zern3DRadi (const int *lowIdx,* const int *upIdx*)

4.4.3 Field Documentation

4.4.3.1 int Zern3DRadi::alpha

auxiliary mixed excess index

4.4.3.2 vector<double> Zern3DRadi::coef

4.4.3.3 int Zern3DRadi::l

upper index, representing the family

4.4.3.4 int Zern3DRadi::n

main quantum number, lower index

5 File Documentation

5.1 poly3D.cpp File Reference

Functions

- point3D operator+ (const point3D &left, const point3D &right)
- point3D operator- (const point3D &left, const point3D &right)
- point3D operator∗ (const double c, const point3D &right)
- ostream & operator<< (ostream &os, const point3D &some)
- trino3D operator∗ (const trino3D &left, const trino3D &right)
- ostream & operator<< (ostream &os, const trino3D &some)
- poly3D operator∗ (const poly3D &left, const poly3D &right)
- ostream & operator<< (ostream &os, const poly3D &some)

5.1.1 Function Documentation

5.1.1.1 point3D operator∗ (const double *c,* const point3D & *right*)

Multiply length by a factor.

Parameters

c	multiplier.

Returns

stretched (c >1) or shrinked (c<1) or reverted (c<0) vector.

5.1.1.2 trino3D operator∗ (const trino3D & *left,* const trino3D & *right*)

Multiply two 3D trinomials.

Parameters

left	the trinomial left to the multiplication sign
right	the trinomial right of the multiplication sign

Returns

the product. The coefficient is the product of the coeffienst and the exponents are the sums of the individual exponents of x, y and z.

5.1.1.3 poly3D operator* (const poly3D & *left*, const poly3D & *right*)

Multiply two 3D polynomials.

Parameters

left	the polynomial left to the multiplication sign
right	the polynomial right of the multiplication sign

Returns

the polynomial which is the sum over all products of the components.

5.1.1.4 point3D operator+ (const point3D & *left*, const point3D & *right*)

Add two 3D points in the sense that the first is a point, the 2nd a vector for translation.

Parameters

left	the point/vector left to the summation sign
right	the point/vector right of the summation sign

Returns

the sum. This contains the sum of the two inputs in each component.

5.1.1.5 point3D operator- (const point3D & *left*, const point3D & *right*)

Subtract two 3D points in the sense that the first is a point, the 2nd a point and the result is the vector from the second to the first.

Parameters

left	the point/vector left to the subtraction sign
right	the point/vector right of the subtraction sign

Returns

The difference.

5.1.1.6 ostream& operator<< (ostream & *os*, const point3D & *some*)

Print a Position.

Parameters

os	the output stream to print to
some	the term to be printed

5.1.1.7 ostream& operator<< (ostream & *os*, const trino3D & *some*)

Print a trinomial term

Parameters

os	the output stream to print to
some	the term to be printed

5.1.1.8 ostream& operator<< (ostream & *os*, const poly3D & *some*)

5.2 poly3D.h File Reference

Data Structures

- class point3D
- class trino3D
- class Zern3DRadi
- class poly3D

Macros

- #define TURB3D_DIM 3

Functions

- point3D operator+ (const point3D &left, const point3D &right)
- point3D operator- (const point3D &left, const point3D &right)
- point3D operator∗ (const double c, const point3D &right)
- ostream & operator<< (ostream &os, const point3D &some)
- trino3D operator∗ (const trino3D &left, const trino3D &right)
- ostream & operator<< (ostream &os, const trino3D &some)
- poly3D operator∗ (const poly3D &left, const poly3D &right)
- ostream & operator<< (ostream &os, const poly3D &some)

5.2.1 Macro Definition Documentation

5.2.1.1 #define TURB3D_DIM 3

5.2.2 Function Documentation

5.2.2.1 point3D operator∗ (const double *c*, const point3D & *right*)

Multiply length by a factor.

Parameters

c	multiplier.

Returns

stretched (c >1) or shrinked (c<1) or reverted (c<0) vector.

5.2.2.2 trino3D operator∗ (const trino3D & *left*, const trino3D & *right*)

Multiply two 3D trinomials.

Parameters

left	the trinomial left to the multiplication sign
right	the trinomial right of the multiplication sign

Returns

the product. The coefficient is the product of the coeffienst and the exponents are the sums of the individual exponents of x, y and z.

5.2.2.3 poly3D operator∗ (const poly3D & *left*, const poly3D & *right*)

Multiply two 3D polynomials.

Parameters

left	the polynomial left to the multiplication sign
right	the polynomial right of the multiplication sign

Returns

the polynomial which is the sum over all products of the components.

5.2.2.4 point3D operator+ (const point3D & *left*, const point3D & *right*)

Add two 3D points in the sense that the first is a point, the 2nd a vector for translation.

Parameters

left	the point/vector left to the summation sign
right	the point/vector right of the summation sign

Returns

the sum. This contains the sum of the two inputs in each component.

5.2.2.5 point3D operator- (const point3D & *left*, const point3D & *right*)

Subtract two 3D points in the sense that the first is a point, the 2nd a point and the result is the vector from the second to the first.

Parameters

left	the point/vector left to the subtraction sign
right	the point/vector right of the subtraction sign

Returns

The difference.

5.2.2.6 ostream& operator<< (ostream & *os*, const point3D & *some*)

Print a Position.

Parameters

os	the output stream to print to
some	the term to be printed

5.2.2.7 ostream& operator<< (ostream & *os*, const trino3D & *some*)

Print a trinomial term

Parameters

os	the output stream to print to
some	the term to be printed

5.2.2.8 ostream& operator<< (ostream & *os*, const poly3D & *some*)

Index

alpha
  Zern3DRadi, 9
at
  poly3D, 5
  trino3D, 7

coef
  trino3D, 8
  Zern3DRadi, 9
compo
  poly3D, 6

dist
  point3D, 2

expo
  trino3D, 8

gradat
  poly3D, 5
  trino3D, 7

hastype
  poly3D, 5

istype
  trino3D, 7, 8

l
  Zern3DRadi, 9

n
  Zern3DRadi, 9
normalize
  point3D, 2

operator<<
  poly3D.cpp, 10, 11
  poly3D.h, 12, 13
operator∗
  poly3D.cpp, 9, 10
  poly3D.h, 11, 12
operator∗=
  point3D, 2
  poly3D, 6
  trino3D, 8
operator+
  poly3D.cpp, 10
  poly3D.h, 12
operator+=
  point3D, 3
  poly3D, 6
operator-
  poly3D.cpp, 10
  poly3D.h, 12
operator-=
  point3D, 3
operator=
  point3D, 3

phithet
  point3D, 3
point3D, 1
  dist, 2
  normalize, 2
  operator∗=, 2
  operator+=, 3
  operator-=, 3
  operator=, 3
  phithet, 3
  point3D, 2
  point3D, 2
  scale, 3
  xyz, 3
poly3D, 4
  at, 5
  compo, 6
  gradat, 5
  hastype, 5
  operator∗=, 6
  operator+=, 6
  poly3D, 4, 5
  poly3D, 4, 5
poly3D.cpp, 9
  operator<<, 10, 11
  operator∗, 9, 10
  operator+, 10
  operator-, 10
poly3D.h, 11
  operator<<, 12, 13
  operator∗, 11, 12
  operator+, 12
  operator-, 12
  TURB3D_DIM, 11

scale
  point3D, 3

TURB3D_DIM
  poly3D.h, 11
trino3D, 7
  at, 7
  coef, 8
  expo, 8
  gradat, 7
  istype, 7, 8
  operator∗=, 8
  trino3D, 7
  trino3D, 7

xyz
  point3D, 3

Zern3DRadi, 8

alpha, 9
coef, 9
l, 9
n, 9
Zern3DRadi, 9
Zern3DRadi, 9