My Project for Splines interpolation

1.0

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

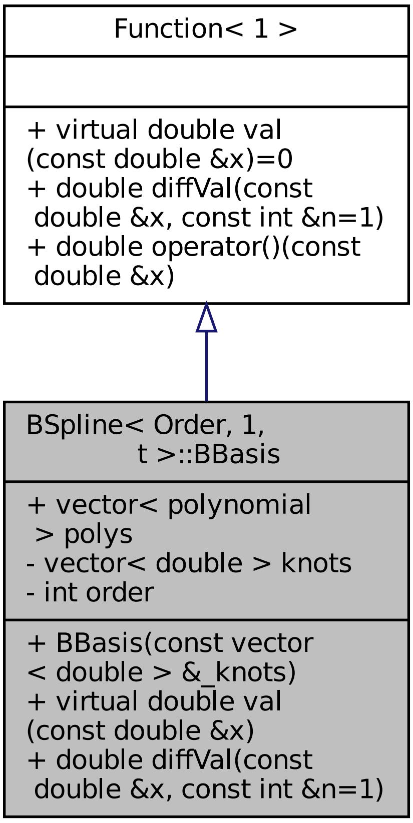
Class Documentation

4.1 BSpline< Order, 1, t >::BBasis Class Reference

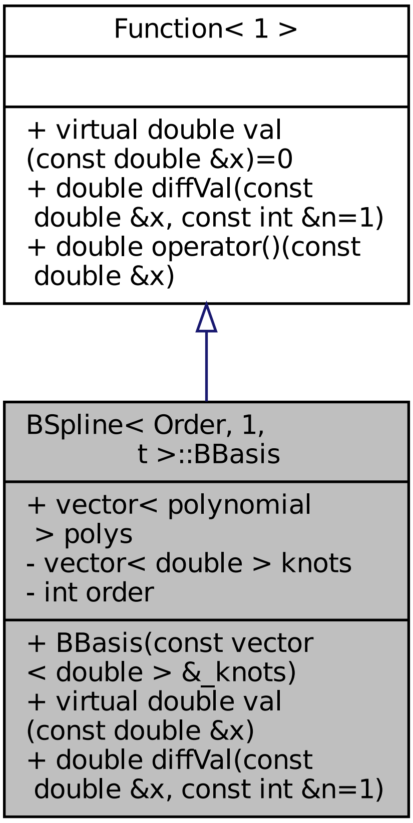
the Basis function of B-form splines

```
#include <splines.h>
```

Inheritance diagram for BSpline< Order, 1, t >::BBasis:


Function< 1 >
+ virtual double val (const double &x)=0
+ double diffVal(const double &x, const int &n=1)
+ double operator()(const double &x)
BSpline< Order, 1, t >::BBasis
+ vector< polynomial > polys
- vector< double > knots
- int order
+ BBasis(const vector < double > &_knots)
+ virtual double val (const double &x)
+ double diffVal(const double &x, const int &n=1)

Collaboration diagram for BSpline< Order, 1, t >::BBasis:


Function< 1 >
+ virtual double val (const double &x)=0
+ double diffVal(const double &x, const int &n=1)
+ double operator()(const double &x)
BSpline< Order, 1, t >::BBasis
+ vector< polynomial > polys
- vector< double > knots
- int order
+ BBasis(const vector < double > &_knots)
+ virtual double val (const double &x)
+ double diffVal(const double &x, const int &n=1)

Public Member Functions

- BBasis (const vector< double > &_knots)

  Construct a B splines Basis dependent on knots, initially it is zero order.
- virtual double val (const double &x)

  pure virtual function to return the value of function at x
- double diffVal (const double &x, const int &n=1)

Public Attributes

- vector< polynomial > polys

  express of function as polynomial in each interval

Private Attributes

- vector< double > knots

  knots of Basis function
- int order

  order of Basis function

Friends

- class Bspline

4.1.1 Detailed Description

template<int Order, BSplineType t>
class BSpline< Order, 1, t >::BBasis

the Basis function of B-form splines

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BBasis()

```
template<int Order, BSplineType t>
BSpline< Order, 1, t >::BBasis::BBasis (
             const vector< double > & _knots )  [inline]
```

Construct a B splines Basis dependent on knots, initially it is zero order.

Parameters

_knots	

4.1.3 Member Function Documentation

4.1.3.1 diffVal()

```
template<int Order, BSplineType t>
double BSpline< Order, 1, t >::BBasis::diffVal (
             const double & x,
             const int & n = 1 )  [inline]
```

4.1.3.2 val()

```
template<int Order, BSplineType t>
virtual double BSpline< Order, 1, t >::BBasis::val (
             const double & x )  [inline], [virtual]
```

pure virtual function to return the value of function at x

Parameters

x	independent variable of function

Returns

double a real number

Implements Function< 1 >.

4.1.4 Friends And Related Function Documentation

4.1.4.1 Bspline

```
template<int Order, BSplineType t>
friend class Bspline  [friend]
```

4.1.5 Member Data Documentation

4.1.5.1 knots

```
template<int Order, BSplineType t>
vector<double> BSpline< Order, 1, t >::BBasis::knots  [private]
```

knots of Basis function

4.1.5.2 order

```
template<int Order, BSplineType t>
int BSpline< Order, 1, t >::BBasis::order  [private]
```

order of Basis function

4.1.5.3 polys

```
template<int Order, BSplineType t>
vector<polynomial> BSpline< Order, 1, t >::BBasis::polys
```

express of function as polynomial in each interval

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

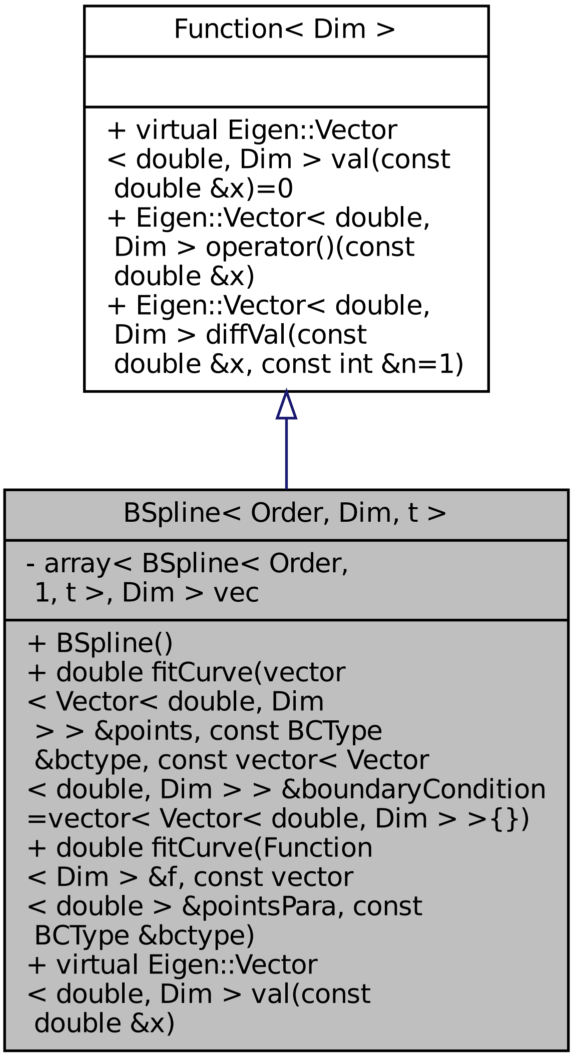
- splines.h

4.2 BSpline< Order, Dim, t > Class Template Reference

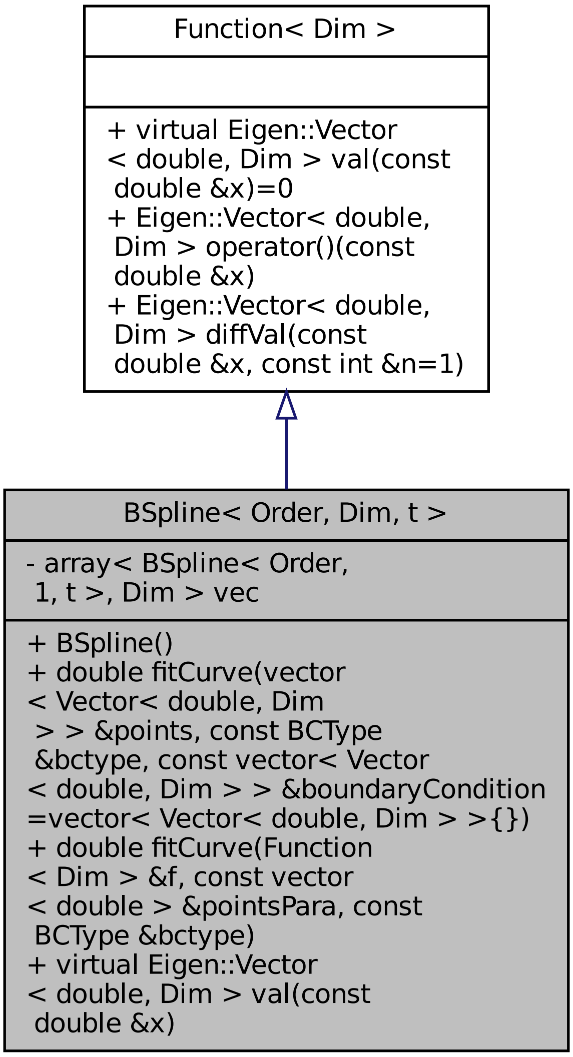
arbitrary order BSplines for curve in arbitrary dimension

```
#include <splines.h>
```

Inheritance diagram for BSpline< Order, Dim, t >:


Function< Dim >
+ virtual Eigen::Vector < double, Dim > val(const double &x)=0
+ Eigen::Vector< double, Dim > operator()(const double &x)
+ Eigen::Vector< double, Dim > diffVal(const double &x, const int &n=1)
BSpline< Order, Dim, t >
- array< BSpline< Order, 1, t >, Dim > vec
+ BSpline()
+ double fitCurve(vector < Vector< double, Dim > > &points, const BCType &bctype, const vector< Vector < double, Dim > > &boundaryCondition =vector< Vector< double, Dim > >{})
+ double fitCurve(Function < Dim > &f, const vector < double > &pointsPara, const BCType &bctype)
+ virtual Eigen::Vector < double, Dim > val(const double &x)

Collaboration diagram for BSpline< Order, Dim, t >:


Function< Dim >
+ virtual Eigen::Vector< double, Dim > val(const double &x)=0
+ Eigen::Vector< double, Dim > operator()(const double &x)
+ Eigen::Vector< double, Dim > diffVal(const double &x, const int &n=1)
BSpline< Order, Dim, t >
- array< BSpline< Order, 1, t >, Dim > vec
+ BSpline()
+ double fitCurve(vector< Vector< double, Dim > > &points, const BCType &bctype, const vector< Vector < double, Dim > > &boundaryCondition =vector< Vector< double, Dim > >{})
+ double fitCurve(Function < Dim > &f, const vector < double > &pointsPara, const BCType &bctype)
+ virtual Eigen::Vector < double, Dim > val(const double &x)

Public Member Functions

- BSpline ()

  default construct a new BSpline object
- double fitCurve (vector< Vector< double, Dim > > &points, const BCType &bctype, const vector< Vector< double, Dim > > &boundaryCondition=vector< Vector< double, Dim > >{})

  fitting a curve by points
- double fitCurve (Function< Dim > &f, const vector< double > &pointsPara, const BCType &bctype)

  fitting a curve by function
- virtual Eigen::Vector< double, Dim > val (const double &x)

  pure virtual function to return the value of function at x

Private Attributes

- array< BSpline< Order, 1, t >, Dim > vec

    splines for component of curve

4.2.1 Detailed Description

template<int Order, int Dim, BSplineType t = myDefault1>
class BSpline< Order, Dim, t >

arbitrary order BSplines for curve in arbitrary dimension

Template Parameters

Order	order of splines
Dim	dismension
t	type of B-form splines

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BSpline()

```
template<int Order, int Dim, BSplineType t = myDefault1>
BSpline< Order, Dim, t >::BSpline ( )  [inline]
```

default construct a new BSpline object

4.2.3 Member Function Documentation

4.2.3.1 fitCurve() [1/2]

```
template<int Order, int Dim, BSplineType t = myDefault1>
double BSpline< Order, Dim, t >::fitCurve (
             Function< Dim > & f,
             const vector< double > & pointsPara,
             const BCType & bctype )  [inline]
```

fitting a curve by function

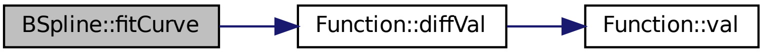
Parameters

f	function you want to fit
pointsPara	knots of parameter of function
bctype	boundary condition type

Returns

double cumulative chordal lengths

Here is the call graph for this function:



4.2.3.2 fitCurve() `[2/2]`

```
template<int Order, int Dim, BSplineType t = myDefault1>
double BSpline< Order, Dim, t >::fitCurve (
            vector< Vector< double, Dim > > & points,
            const BCType & bctype,
            const vector< Vector< double, Dim > > & boundaryCondition = vector<Vector<double,Dim> >{}
) [inline]
```

fitting a curve by points

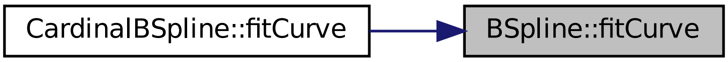
Parameters

points	a series of points on the curve you want to fit
bctype	boundary condition type
boundaryCondition	boundary condition

Returns

double: the endpoints of cumulative chordal lengths

Here is the caller graph for this function:



4.2.3.3 val()

```
template<int Order, int Dim, BSplineType t = myDefault1>
virtual Eigen::Vector<double,Dim> BSpline< Order, Dim, t >::val (
            const double & x )  [inline], [virtual]
```

pure virtual function to return the value of function at x

Parameters

x	independent variable of function

Returns

Eigen::Vector<double,Dim> a point in the Dim dimension space

Implements Function< Dim >.

4.2.4 Member Data Documentation

4.2.4.1 vec

```
template<int Order, int Dim, BSplineType t = myDefault1>
array< BSpline<Order,1,t>, Dim > BSpline< Order, Dim, t >::vec  [private]
```

splines for component of curve

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

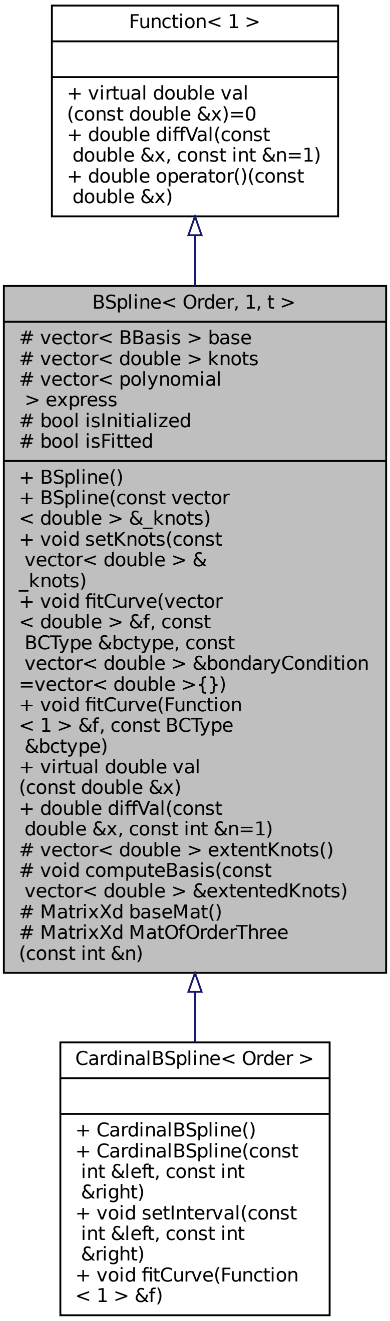
- splines.h

4.3 BSpline< Order, 1, t > Class Template Reference

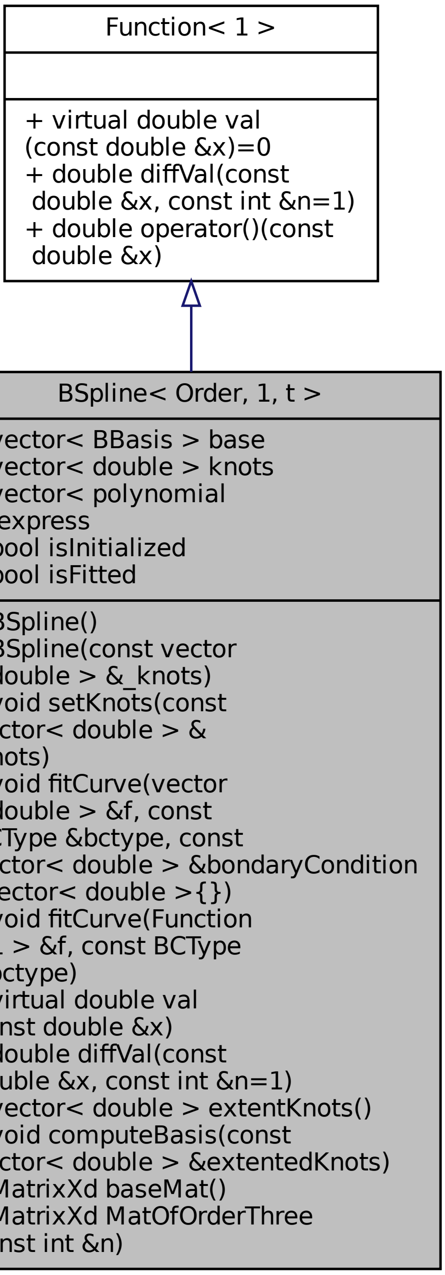
specialization for one dimension B-form splines

```
#include <splines.h>
```

Inheritance diagram for BSpline< Order, 1, t >:


Function< 1 >
+ virtual double val (const double &x)=0
+ double diffVal(const double &x, const int &n=1)
+ double operator()(const double &x)
BSpline< Order, 1, t >
# vector< BBasis > base
# vector< double > knots
# vector< polynomial > express
# bool isInitialized
# bool isFitted
+ BSpline()
+ BSpline(const vector < double > &_knots)
+ void setKnots(const vector< double > & _knots)
+ void fitCurve(vector < double > &f, const BCType &bctype, const vector< double > &bondaryCondition =vector< double >{})
+ void fitCurve(Function < 1 > &f, const BCType &bctype)
+ virtual double val (const double &x)
+ double diffVal(const double &x, const int &n=1)
# vector< double > extentKnots()
# void computeBasis(const vector< double > &extentedKnots)
# MatrixXd baseMat()
# MatrixXd MatOfOrderThree (const int &n)
CardinalBSpline< Order >
+ CardinalBSpline()
+ CardinalBSpline(const int &left, const int &right)
+ void setInterval(const int &left, const int &right)
+ void fitCurve(Function < 1 > &f)

Collaboration diagram for BSpline< Order, 1, t >:



Classes

- class BBasis

  the Basis function of B-form splines

Public Member Functions

- BSpline ()

    default Construct a new BSpline object
- BSpline (const vector< double > &_knots)

    Construct a BSpline which have setted interpolation knots and computed basis.
- void setKnots (const vector< double > &_knots)

    Set interpolation knots for splines and compute basis.
- void fitCurve (vector< double > &f, const BCType &bctype, const vector< double > &bondary↩ Condition=vector< double >{})

    compute the coefficient of basis and compute the express of spline to interpolate a series of points
- void fitCurve (Function< 1 > &f, const BCType &bctype)

    compute the coefficient of basis and compute the express of spline to interpolate a function
- virtual double val (const double &x)

    pure virtual function to return the value of function at x
- double diffVal (const double &x, const int &n=1)

Protected Member Functions

- vector< double > extentKnots ()

    extent the knots inputted with extra knots for basis computing
- void computeBasis (const vector< double > &extentedKnots)

    commpute basis on the extentedKnots, stored in the attribute,base , and set isInitialized as 1
- MatrixXd baseMat ()

    creat a matrix with condition that the value of splines is equal to the value of fitted funciton at the interpolation knots
- MatrixXd MatOfOrderThree (const int &n)

    creat a matrix for compute coefficient of basis for cubic spline with boundary conditon of n order derivative

Protected Attributes

- vector< BBasis > base

    a series of basis of spline on the interpolation knots
- vector< double > knots

    interpolation knots
- vector< polynomial > express

    expression of splines as the result of addition with basis multiply the coefficient which computed in the fitCurve
- bool isInitialized

    is the spline set the knots and compute the basis, if it's not, users can't fit the curve
- bool isFitted

    is the spline fitting some curve, if it's not, users can't get the value at any points of splines

4.3.1 Detailed Description

template<int Order, BSplineType t>
class BSpline< Order, 1, t >

specialization for one dimension B-form splines

Template Parameters

Order	order of splines
t	type of B-form splines

4.3.2 Constructor & Destructor Documentation

4.3.2.1 BSpline() [1/2]

```
template<int Order, BSplineType t>
BSpline< Order, 1, t >::BSpline ( )  [inline]
```

default Construct a new BSpline object

4.3.2.2 BSpline() [2/2]

```
template<int Order, BSplineType t>
BSpline< Order, 1, t >::BSpline (
              const vector< double > & _knots )  [inline]
```

Construct a BSpline which have setted interpolation knots and computed basis.

Parameters

_knots	interpolation knots

4.3.3 Member Function Documentation

4.3.3.1 baseMat()

```
template<int Order, BSplineType t>
MatrixXd BSpline< Order, 1, t >::baseMat ( )  [inline], [protected]
```

creat a matrix with condition that the value of splines is equal to the value of fitted funciton at the interpolation knots

Returns

MatrixXd

4.3.3.2 computeBasis()

```
template<int Order, BSplineType t>
void BSpline< Order, 1, t >::computeBasis (
             const vector< double > & extentedKnots )  [inline], [protected]
```

commpute basis on the extentedKnots, stored in the attribute,base , and set isInitialized as 1

Parameters

extentedKnots	

4.3.3.3 diffVal()

```
template<int Order, BSplineType t>
double BSpline< Order, 1, t >::diffVal (
             const double & x,
             const int & n = 1 )  [inline]
```

4.3.3.4 extentKnots()

```
template<int Order, BSplineType t>
vector<double> BSpline< Order, 1, t >::extentKnots ( )  [inline], [protected]
```

extent the knots inputted with extra knots for basis computing

Returns

vector<double>

4.3.3.5 fitCurve() [1/2]

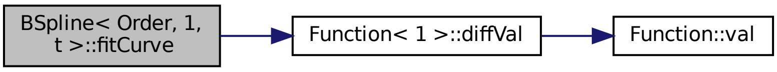
```
template<int Order, BSplineType t>
void BSpline< Order, 1, t >::fitCurve (
             Function< 1 > & f,
             const BCType & bctype )  [inline]
```

compute the coefficient of basis and compute the express of spline to interpolate a function

Parameters

f	function you want to interpolate
bctype	boundary condition

Here is the call graph for this function:



4.3.3.6 fitCurve() [2/2]

```
template<int Order, BSplineType t>
void BSpline< Order, 1, t >::fitCurve (
             vector< double > & f,
             const BCType & bctype,
             const vector< double > & bondaryCondition = vector<double>{} )  [inline]
```

compute the coefficient of basis and compute the express of spline to interpolate a series of points

Parameters

f	the value of curve at the interpolation knots
bctype	boundary condtion type
bondaryCondition	inputted extra condition

4.3.3.7 MatOfOrderThree()

```
template<int Order, BSplineType t>
MatrixXd BSpline< Order, 1, t >::MatOfOrderThree (
             const int & n )  [inline], [protected]
```

creat a matrix for compute coefficient of basis for cubic spline with boundary conditon of n order derivative

Parameters

n	order of derivative

Returns

MatrixXd

4.3.3.8 setKnots()

```
template<int Order, BSplineType t>
void BSpline< Order, 1, t >::setKnots (
            const vector< double > & _knots )  [inline]
```

Set interpolation knots for splines and compute basis.

Parameters

_knots	interpolation knots

4.3.3.9 val()

```
template<int Order, BSplineType t>
virtual double BSpline< Order, 1, t >::val (
            const double & x )  [inline], [virtual]
```

pure virtual function to return the value of function at x

Parameters

x	independent variable of function

Returns

double a real number

Implements Function< 1 >.

4.3.4 Member Data Documentation

4.3.4.1 base

```
template<int Order, BSplineType t>
vector<BBasis> BSpline< Order, 1, t >::base  [protected]
```

a series of basis of spline on the interpolation knots

4.3.4.2 express

```
template<int Order, BSplineType t>
vector<polynomial> BSpline< Order, 1, t >::express  [protected]
```

expression of splines as the result of addition with basis multiply the coefficient which computed in the fitCurve

4.3.4.3 isFitted

```
template<int Order, BSplineType t>
bool BSpline< Order, 1, t >::isFitted  [protected]
```

is the spline fitting some curve, if it's not, users can't get the value at any points of splines

4.3.4.4 isInitialized

```
template<int Order, BSplineType t>
bool BSpline< Order, 1, t >::isInitialized  [protected]
```

is the spline set the knots and compute the basis, if it's not, users can't fit the curve

4.3.4.5 knots

```
template<int Order, BSplineType t>
vector<double> BSpline< Order, 1, t >::knots  [protected]
```

interpolation knots

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

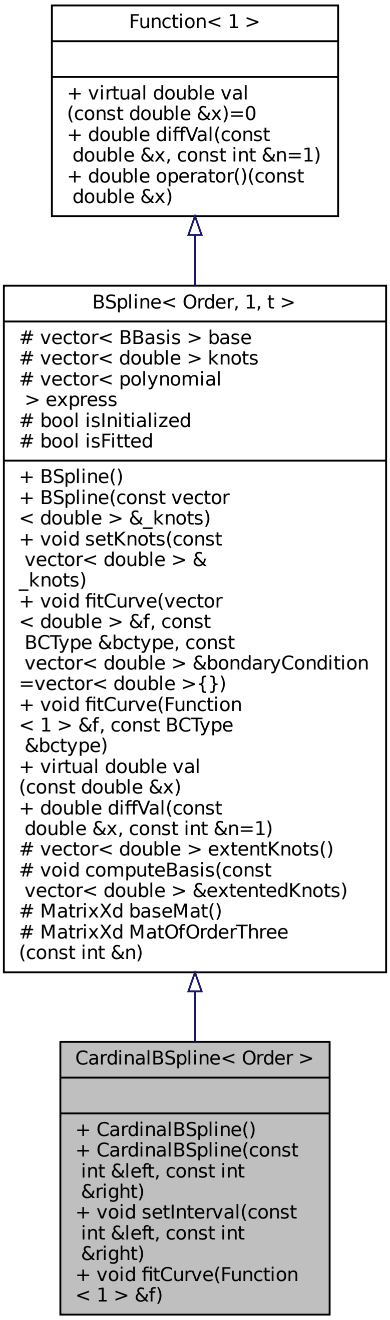
- splines.h

4.4 CardinalBSpline< Order > Class Template Reference

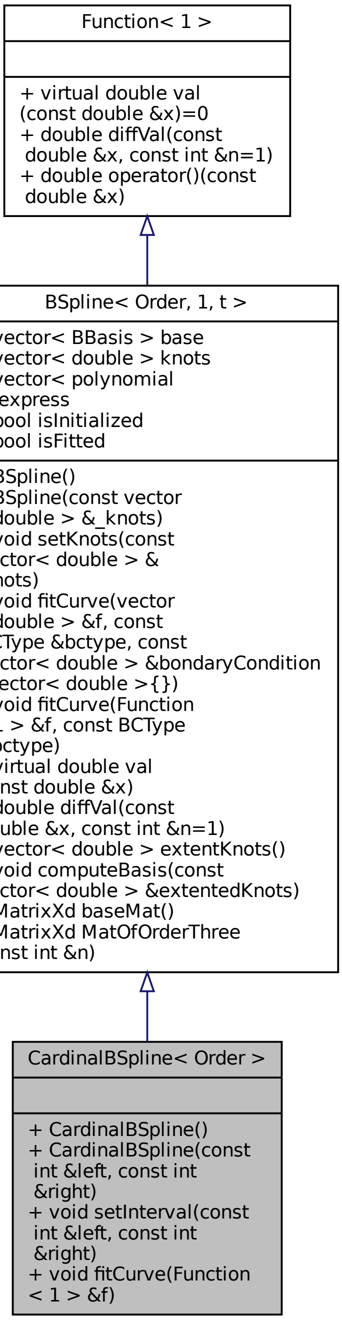
one dimension cardinal B-form splines

```
#include <splines.h>
```

Inheritance diagram for CardinalBSpline< Order >:



Collaboration diagram for CardinalBSpline< Order >:



Public Member Functions

- CardinalBSpline ()

    default construct a new Cardinal B Spline object

- CardinalBSpline (const int &left, const int &right)

    Construct a new Cardinal B Spline object with interpolation interval [left,right].

- void setInterval (const int &left, const int &right)

Set the Interval [left,right].
- void fitCurve (Function< 1 > &f)

  use cardinal B Splines to interpolate a function

Additional Inherited Members

4.4.1 Detailed Description

template<int Order>
class CardinalBSpline< Order >

one dimension cardinal B-form splines

Template Parameters

Order	order of splines

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CardinalBSpline() [1/2]

```
template<int Order>
CardinalBSpline< Order >::CardinalBSpline ( )  [inline]
```

default construct a new Cardinal B Spline object

4.4.2.2 CardinalBSpline() [2/2]

```
template<int Order>
CardinalBSpline< Order >::CardinalBSpline (
            const int & left,
            const int & right )  [inline]
```

Construct a new Cardinal B Spline object with interpolation interval [left,right].

Parameters

left	start point
right	end point

4.4.3 Member Function Documentation

4.4.3.1 fitCurve()

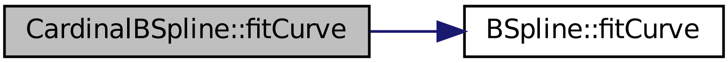
```
template<int Order>
void CardinalBSpline< Order >::fitCurve (
             Function< 1 > & f )  [inline]
```

use cardinal B Splines to interpolate a function

Parameters

f	function which you want to interpolate

Here is the call graph for this function:



4.4.3.2 setInterval()

```
template<int Order>
void CardinalBSpline< Order >::setInterval (
             const int & left,
             const int & right )  [inline]
```

Set the Interval [left,right].

Parameters

left	start point
right	end point

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

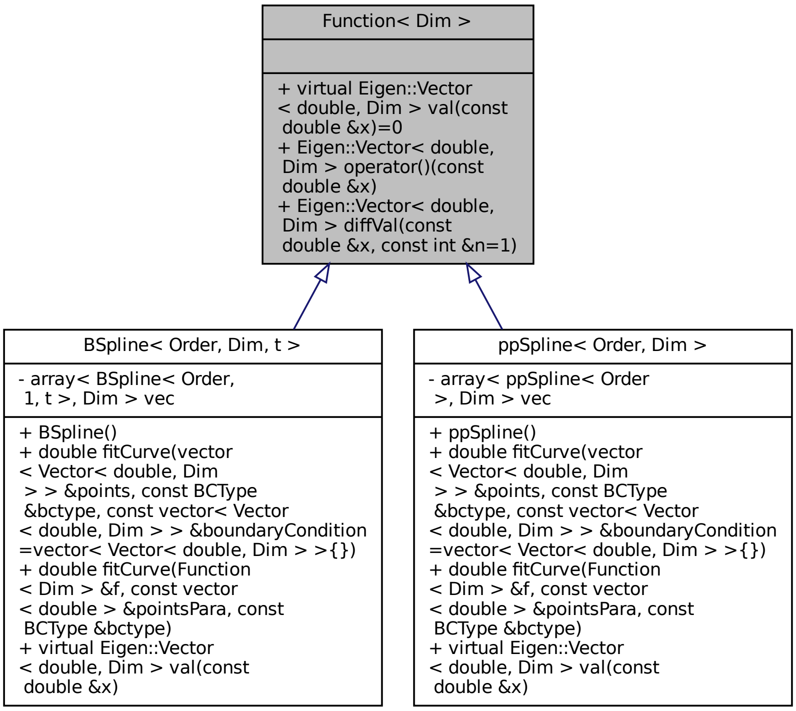
- splines.h

4.5 Function< Dim > Class Template Reference

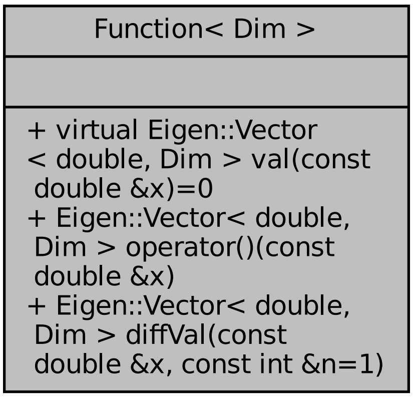
A function(math) abstract class.

```
#include <function.h>
```

Inheritance diagram for Function< Dim >:


Function< Dim >
+ virtual Eigen::Vector< double, Dim > val(const double &x)=0
+ Eigen::Vector< double, Dim > operator()(const double &x)
+ Eigen::Vector< double, Dim > diffVal(const double &x, const int &n=1)
BSpline< Order, Dim, t >
- array< BSpline< Order, 1, t >, Dim > vec
+ BSpline()
+ double fitCurve(vector< Vector< double, Dim > > &points, const BCType &bctype, const vector< Vector< double, Dim > > &boundaryCondition =vector< Vector< double, Dim > >{})
+ double fitCurve(Function< Dim > &f, const vector< double > &pointsPara, const BCType &bctype)
+ virtual Eigen::Vector< double, Dim > val(const double &x)
ppSpline< Order, Dim >
- array< ppSpline< Order >, Dim > vec
+ ppSpline()
+ double fitCurve(vector< Vector< double, Dim > > &points, const BCType &bctype, const vector< Vector< double, Dim > > &boundaryCondition =vector< Vector< double, Dim > >{})
+ double fitCurve(Function< Dim > &f, const vector< double > &pointsPara, const BCType &bctype)
+ virtual Eigen::Vector< double, Dim > val(const double &x)

Collaboration diagram for Function< Dim >:


Function< Dim >
+ virtual Eigen::Vector< double, Dim > val(const double &x)=0
+ Eigen::Vector< double, Dim > operator()(const double &x)
+ Eigen::Vector< double, Dim > diffVal(const double &x, const int &n=1)

Public Member Functions

- virtual Eigen::Vector< double, Dim > val (const double &x)=0

  pure virtual function to return the value of function at x
- Eigen::Vector< double, Dim > operator() (const double &x)

  override the operator () to use this class in a expression just like function in the math
- Eigen::Vector< double, Dim > diffVal (const double &x, const int &n=1)

  impletement a numerical derivative for any funcition, user can choose override to make it more precise

4.5.1 Detailed Description

template<int Dim>
class Function< Dim >

A function(math) abstract class.

Template Parameters

Dim	is dimension of function($R->R^{Dim}$)

4.5.2 Member Function Documentation

4.5.2.1 diffVal()

```
template<int Dim>
Eigen::Vector<double,Dim> Function< Dim >::diffVal (
             const double & x,
             const int & n = 1 )  [inline]
```

impletement a numerical derivative for any funcition, user can choose override to make it more precise

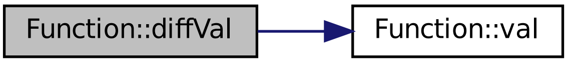
Parameters

x	independent variable of function
n	order of derivative

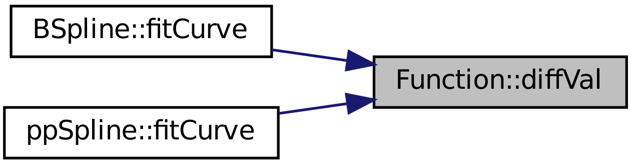
Returns

Eigen::Vector<double,Dim> a point in the Dim dimension space

Here is the call graph for this function:



Here is the caller graph for this function:



4.5.2.2 operator()()

```
template<int Dim>
Eigen::Vector<double,Dim> Function< Dim >::operator() (
              const double & x )  [inline]
```

override the operator () to use this class in a expression just like function in the math

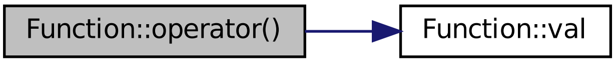
Parameters

x	independent variable of function

Returns

Eigen::Vector<double,Dim> a point in the Dim dimension space

Here is the call graph for this function:



4.5.2.3 val()

```
template<int Dim>
virtual Eigen::Vector<double,Dim> Function< Dim >::val (
                const double & x )  [pure virtual]
```

pure virtual function to return the value of function at x

Parameters

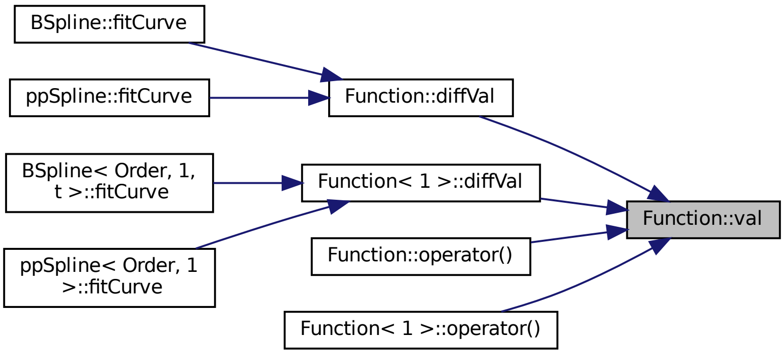
x	independent variable of function

Returns

Eigen::Vector<double,Dim> a point in the Dim dimension space

Implemented in ppSpline< Order, Dim >, and BSpline< Order, Dim, t >.

Here is the caller graph for this function:



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

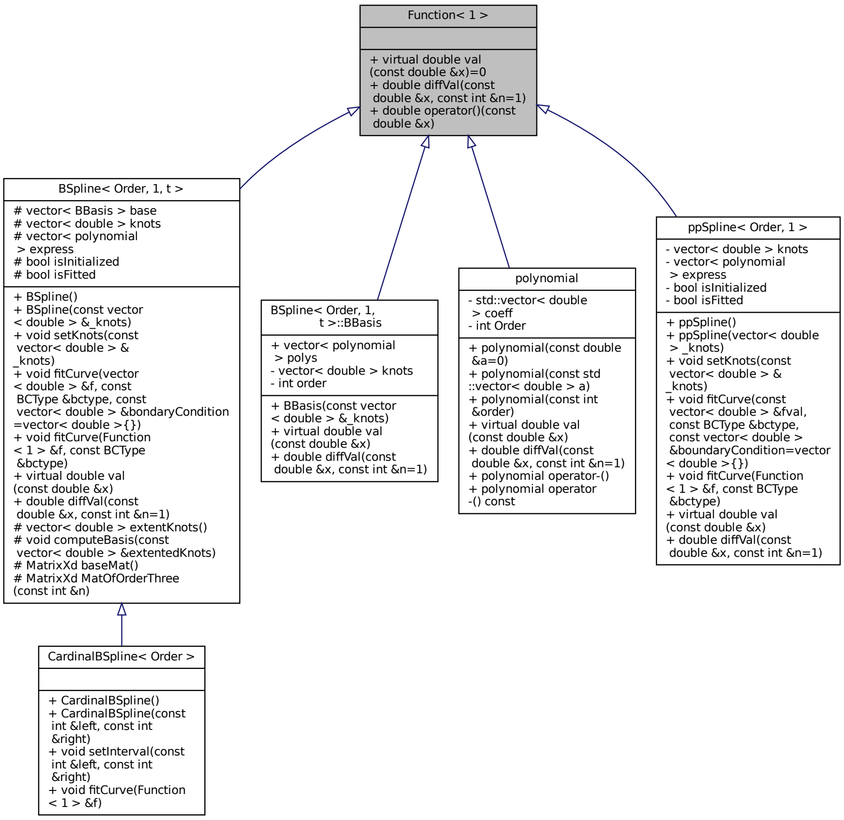
- function.h

4.6 Function< 1 > Class Reference

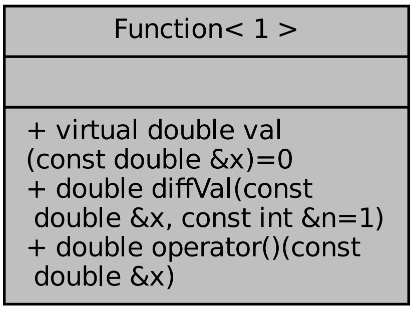
specialization for one dimension function, definition is as same as high dimension

```
#include <function.h>
```

Inheritance diagram for Function< 1 >:



Collaboration diagram for Function< 1 >:



Public Member Functions

- virtual double val (const double &x)=0

  pure virtual function to return the value of function at x
- double diffVal (const double &x, const int &n=1)

  impletement a numerical derivative for any funcition, user can choose override to make it more precise
- double operator() (const double &x)

  override the operator () to use this class in a expression just like function in the math

4.6.1 Detailed Description

specialization for one dimension function, definition is as same as high dimension

Template Parameters

4.6.2 Member Function Documentation

4.6.2.1 diffVal()

```
double Function< 1 >::diffVal (
             const double & x,
             const int & n = 1 )  [inline]
```

impletement a numerical derivative for any funcition, user can choose override to make it more precise

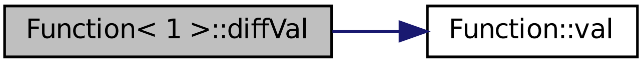
Parameters

x	independent variable of function
n	order of derivative

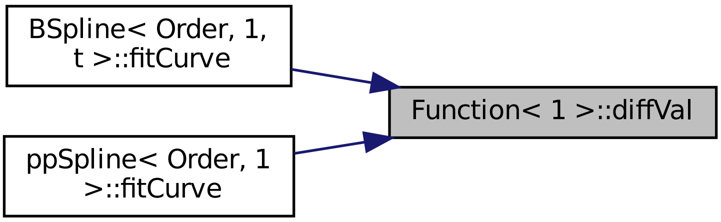
Returns

double a real number

Here is the call graph for this function:



Here is the caller graph for this function:


BSpline< Order, 1, t >::fitCurve
ppSpline< Order, 1 >::fitCurve
Function< 1 >::diffVal

4.6.2.2 operator()()

```
double Function< 1 >::operator() (
            const double & x )  [inline]
```

override the operator () to use this class in a expression just like function in the math

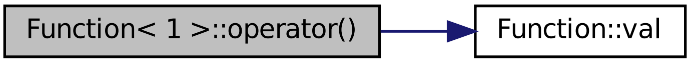
Parameters

x	independent variable of function

Returns

double a real number

Here is the call graph for this function:



4.6.2.3 val()

```
virtual double Function< 1 >::val (
              const double & x )  [pure virtual]
```

pure virtual function to return the value of function at x

Parameters

x	independent variable of function

Returns

double a real number

Implemented in ppSpline< Order, 1 >, BSpline< Order, 1, t >, BSpline< Order, 1, t >::BBasis, and polynomial.

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

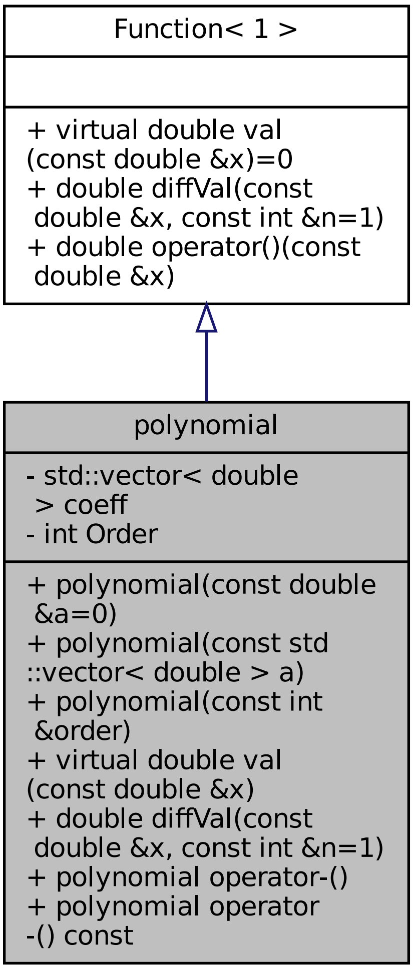
- function.h

4.7 polynomial Class Reference

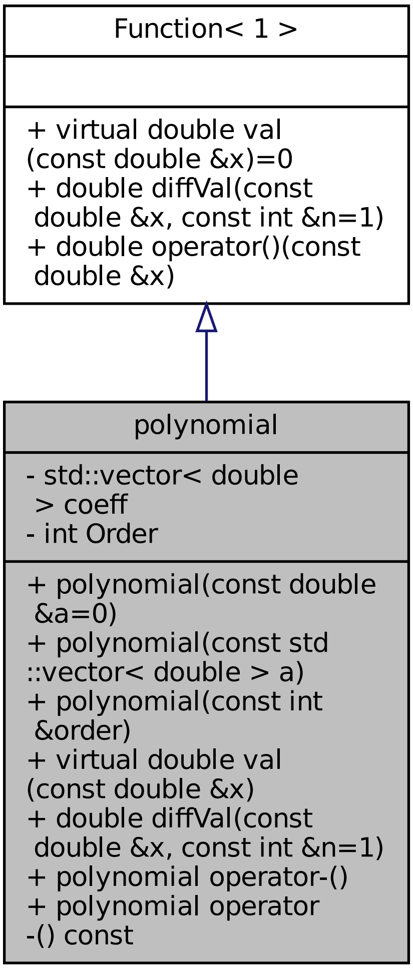
polynomial inherited from Function<1>

```
#include <function.h>
```

Inheritance diagram for polynomial:



Collaboration diagram for polynomial:



Public Member Functions

- polynomial (const double &a=0)

  Construct a polynomial as a constant function.
- polynomial (const std::vector< double > a)

  Construct a polynomial with coefficient.
- polynomial (const int &order)

  Construct a n order polynomial.
- virtual double val (const double &x)

  compute the value of polynomial at x
- double diffVal (const double &x, const int &n=1)

  compute the derivative of polynomial at x
- polynomial operator- ()

  overrider the operator - to get a polynomial whose coefficients are opposite of this
- polynomial operator- () const

  overrider the operator - to get a polynomial whose coefficients are opposite of this

Private Attributes

- std::vector< double > coeff

  coefficient of polynomial, and coeff[i] is coefficient of x^ i
- int Order

  order of polynomial

Friends

- polynomial operator+ (const polynomial &a, const polynomial &b)

  override the operator + to compute addition of two polynomials
- polynomial operator- (const polynomial &a, const polynomial &b)

  override the operator - to compute subtraction of two polynomials
- polynomial operator* (const polynomial &a, const polynomial &b)

  *override the operator * to compute multiplication of two polynomials*
- polynomial operator/ (const polynomial &a, const double &b)

  override the operator / to compute that a polynomial divide a real number

4.7.1 Detailed Description

polynomial inherited from Function<1>

4.7.2 Constructor & Destructor Documentation

4.7.2.1 polynomial() [1/3]

```
polynomial::polynomial (
            const double & a = 0 )  [inline]
```

Construct a polynomial as a constant function.

Parameters

a	polynomial = a

4.7.2.2 polynomial() [2/3]

```
polynomial::polynomial (
            const std::vector< double > a )  [inline]
```

Construct a polynomial with coefficient.

Parameters

a	a vector stored coefficient of polynomial

4.7.2.3 polynomial() [3/3]

```
polynomial::polynomial (
            const int & order )  [inline], [explicit]
```

Construct a n order polynomial.

Parameters

order	number of order

4.7.3 Member Function Documentation

4.7.3.1 diffVal()

```
double polynomial::diffVal (
            const double & x,
            const int & n = 1 )  [inline]
```

compute the derivative of polynomial at x

Parameters

x	independent variable of polynomial
n	order of derivative

Returns

double derivative of polynomial at x

4.7.3.2 operator-() [1/2]

```
polynomial polynomial::operator- ( )  [inline]
```

overrider the operator - to get a polynomial whose coefficients are opposite of this

Returns

polynomial whose coefficients are opposite of this

4.7.3.3 operator-() [2/2]

```
polynomial polynomial::operator- ( ) const  [inline]
```

overrider the operator - to get a polynomial whose coefficients are opposite of this

Returns

polynomial whose coefficients are opposite of this

4.7.3.4 val()

```
virtual double polynomial::val (
             const double & x )  [inline], [virtual]
```

compute the value of polynomial at x

Parameters

x	independent variable of polynomial

Returns

double

Implements Function< 1 >.

4.7.4 Friends And Related Function Documentation

4.7.4.1 operator*

```
polynomial operator* (
             const polynomial & a,
             const polynomial & b )  [friend]
```

override the operator * to compute multiplication of two polynomials

Parameters

a	one of polynomial to operator
b	the other polynomial to operator

Returns

polynomial result of multiplication of two polynomials

4.7.4.2 operator+

```
polynomial operator+ (
            const polynomial & a,
            const polynomial & b )  [friend]
```

override the operator + to compute addition of two polynomials

Parameters

a	one of polynomial to operator
b	the other polynomial to operator

Returns

polynomial result of addition of two polynomials

4.7.4.3 operator-

```
polynomial operator- (
            const polynomial & a,
            const polynomial & b )  [friend]
```

override the operator - to compute subtraction of two polynomials

Parameters

a	minuend
b	subtrahend

Returns

polynomial result of subtraction of two polynomials

4.7.4.4 operator/

```
polynomial operator/ (
            const polynomial & a,
            const double & b )  [friend]
```

override the operator / to compute that a polynomial divide a real number

Parameters

a	dividend, a polynomial
b	divisor, a real number

Returns

polynomial result of that a polynomial divide a real number

4.7.5 Member Data Documentation

4.7.5.1 coeff

```
std::vector<double> polynomial::coeff [private]
```

coefficient of polynomial, and coeff[i] is coefficient of x^i

4.7.5.2 Order

```
int polynomial::Order [private]
```

order of polynomial

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

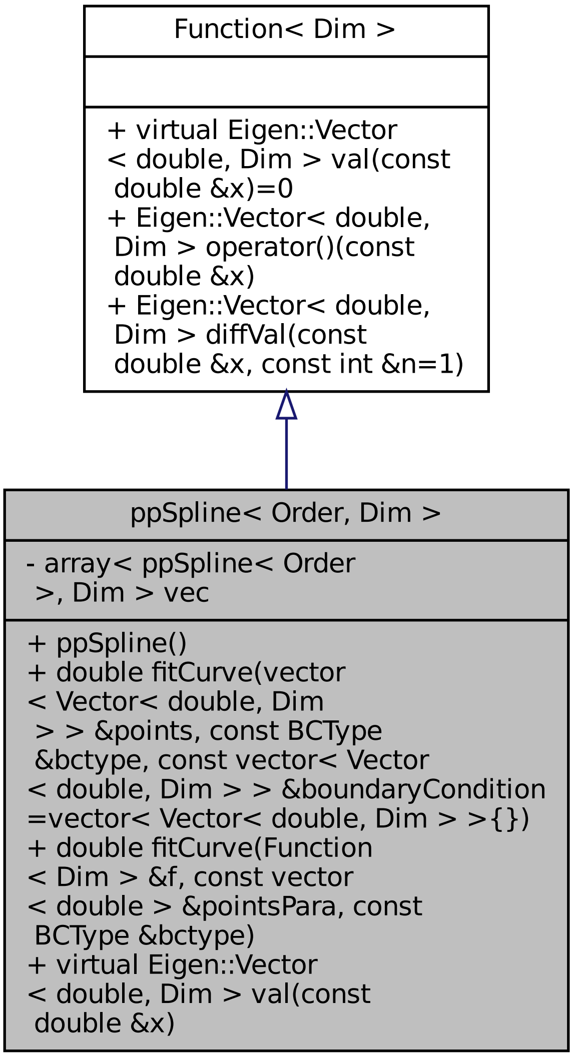
- function.h

4.8 ppSpline< Order, Dim > Class Template Reference

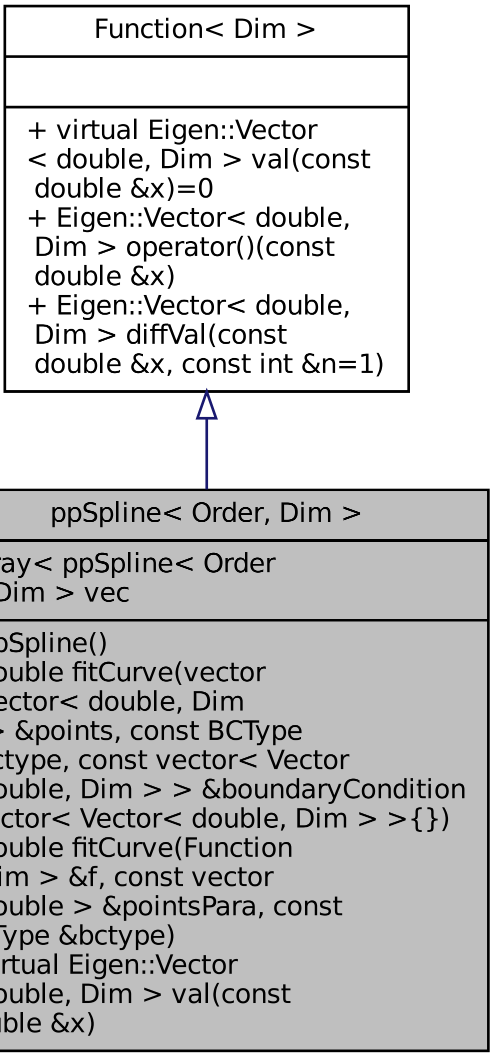
arbitrary order BSplines for curve in arbitrary dimension

```
#include <splines.h>
```

Inheritance diagram for ppSpline< Order, Dim >:


Function< Dim >
+ virtual Eigen::Vector< double, Dim > val(const double &x)=0
+ Eigen::Vector< double, Dim > operator()(const double &x)
+ Eigen::Vector< double, Dim > diffVal(const double &x, const int &n=1)
ppSpline< Order, Dim >
- array< ppSpline< Order >, Dim > vec
+ ppSpline()
+ double fitCurve(vector< Vector< double, Dim > > &points, const BCType &bctype, const vector< Vector< double, Dim > > &boundaryCondition =vector< Vector< double, Dim > >{})
+ double fitCurve(Function< Dim > &f, const vector< double > &pointsPara, const BCType &bctype)
+ virtual Eigen::Vector< double, Dim > val(const double &x)

Collaboration diagram for ppSpline< Order, Dim >:


Function< Dim >
+ virtual Eigen::Vector< double, Dim > val(const double &x)=0
+ Eigen::Vector< double, Dim > operator()(const double &x)
+ Eigen::Vector< double, Dim > diffVal(const double &x, const int &n=1)
ppSpline< Order, Dim >
- array< ppSpline< Order >, Dim > vec
+ ppSpline()
+ double fitCurve(vector< Vector< double, Dim > > &points, const BCType &bctype, const vector< Vector< double, Dim > > &boundaryCondition =vector< Vector< double, Dim > >{})
+ double fitCurve(Function< Dim > &f, const vector< double > &pointsPara, const BCType &bctype)
+ virtual Eigen::Vector< double, Dim > val(const double &x)

Public Member Functions

- ppSpline ()
- double fitCurve (vector< Vector< double, Dim > > &points, const BCType &bctype, const vector< Vector< double, Dim > > &boundaryCondition=vector< Vector< double, Dim > >{})

  fitting a curve by points
- double fitCurve (Function< Dim > &f, const vector< double > &pointsPara, const BCType &bctype)

  fitting a curve by function
- virtual Eigen::Vector< double, Dim > val (const double &x)

  pure virtual function to return the value of function at x

Private Attributes

- array< ppSpline< Order >, Dim > vec

  piecewise polynomial Splines for component of curve

4.8.1 Detailed Description

template<int Order, int Dim>
class ppSpline< Order, Dim >

arbitrary order BSplines for curve in arbitrary dimension

Template Parameters

Order	order of splines
Dim	dimension

4.8.2 Constructor & Destructor Documentation

4.8.2.1 ppSpline()

```
template<int Order, int Dim>
ppSpline< Order, Dim >::ppSpline ( )  [inline]
```

4.8.3 Member Function Documentation

4.8.3.1 fitCurve() [1/2]

```
template<int Order, int Dim>
double ppSpline< Order, Dim >::fitCurve (
            Function< Dim > & f,
            const vector< double > & pointsPara,
            const BCType & bctype )  [inline]
```

fitting a curve by function

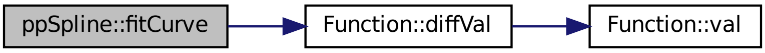
Parameters

f	function you want to fit
pointsPara	knots of parameter of function
bctype	boundary condition type

Returns

double: cumulative chordal lengths

Here is the call graph for this function:



4.8.3.2 fitCurve() [2/2]

```
template<int Order, int Dim>
double ppSpline< Order, Dim >::fitCurve (
             vector< Vector< double, Dim > > & points,
             const BCType & bctype,
             const vector< Vector< double, Dim > > & boundaryCondition = vector<Vector<double,Dim> >{}
) [inline]
```

fitting a curve by points

Parameters

points	a series of points on the curve you want to fit
bctype	boundary condition type
boundaryCondition	boundary condition

Returns

double: the endpoints of cumulative chordal lengths

4.8.3.3 val()

```
template<int Order, int Dim>
virtual Eigen::Vector<double,Dim> ppSpline< Order, Dim >::val (
             const double & x ) [inline], [virtual]
```

pure virtual function to return the value of function at x

Parameters

x	independent variable of function

Returns

Eigen::Vector<double,Dim> a point in the Dim dimension space

Implements Function< Dim >.

4.8.4 Member Data Documentation

4.8.4.1 vec

```
template<int Order, int Dim>
array< ppSpline<Order>, Dim > ppSpline< Order, Dim >::vec  [private]
```

piecewise polynomial Splines for component of curve

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

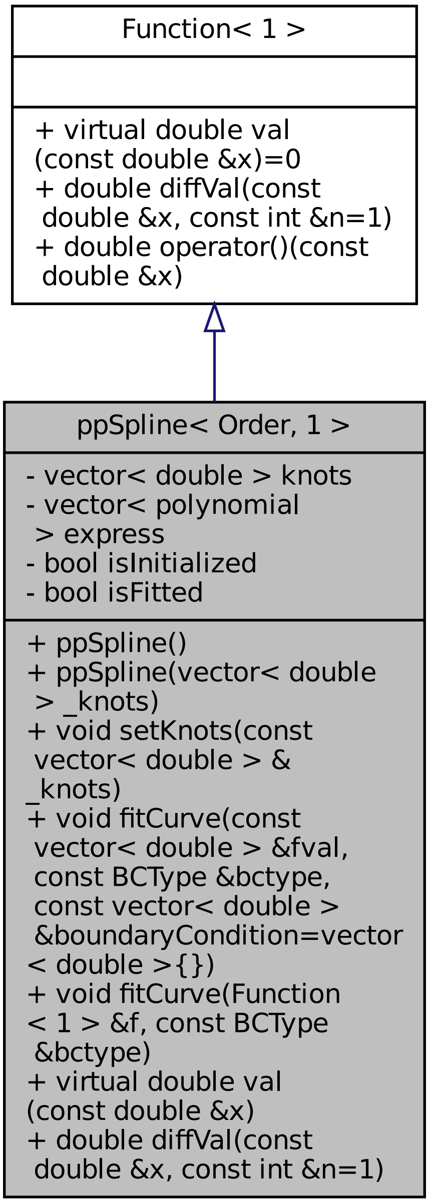
- splines.h

4.9 ppSpline< Order, 1 > Class Template Reference

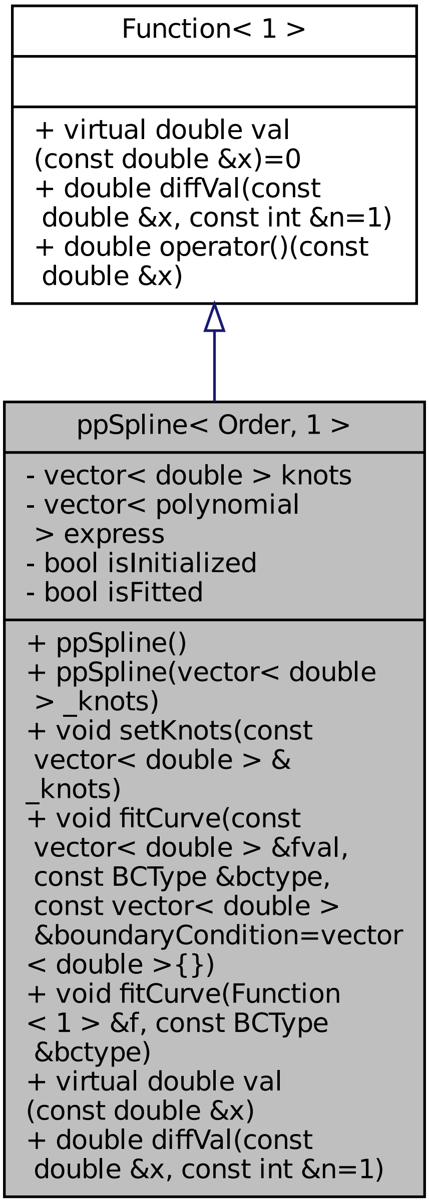
specialization for one dimension piecewise polynomial splines

```
#include <splines.h>
```

Inheritance diagram for ppSpline< Order, 1 >:



Collaboration diagram for ppSpline< Order, 1 >:


Function< 1 >
+ virtual double val (const double &x)=0
+ double diffVal(const double &x, const int &n=1)
+ double operator()(const double &x)
ppSpline< Order, 1 >
- vector< double > knots
- vector< polynomial > express
- bool isInitialized
- bool isFitted
+ ppSpline()
+ ppSpline(vector< double > _knots)
+ void setKnots(const vector< double > & _knots)
+ void fitCurve(const vector< double > &fval, const BCType &bctype, const vector< double > &boundaryCondition=vector < double >{})
+ void fitCurve(Function < 1 > &f, const BCType &bctype)
+ virtual double val (const double &x)
+ double diffVal(const double &x, const int &n=1)

Public Member Functions

- ppSpline ()

  default construct a new pp Spline object
- ppSpline (vector< double > _knots)

  Construct a PP Spline which have setted interpolation knots.
- void setKnots (const vector< double > &_knots)

  Set interpolation knots for splines.

- void fitCurve (const vector< double > &fval, const BCType &bctype, const vector< double > &boundary↩ Condition=vector< double >{})

  compute polynomial on each interpolation subinterval to interpolate a series of points
- void fitCurve (Function< 1 > &f, const BCType &bctype)

  compute polynomial on each interpolation subinterval to interpolate a function
- virtual double val (const double &x)

  pure virtual function to return the value of function at x
- double diffVal (const double &x, const int &n=1)

Private Attributes

- vector< double > knots

  interpolation knots
- vector< polynomial > express

  expression of splines as a piecewise polynomials
- bool isInitialized

  is the spline set the knots, if it's not, users can't fit the curve
- bool isFitted

  is the spline fitting some curve, if it's not, users can't get the value at any points of splines

4.9.1 Detailed Description

template<int Order>
class ppSpline< Order, 1 >

specialization for one dimension piecewise polynomial splines

Template Parameters

Order	order of splines

4.9.2 Constructor & Destructor Documentation

4.9.2.1 ppSpline() [1/2]

```
template<int Order>
ppSpline< Order, 1 >::ppSpline ( )  [inline]
```

default construct a new pp Spline object

4.9.2.2 ppSpline() [2/2]

```
template<int Order>
ppSpline< Order, 1 >::ppSpline (
            vector< double > _knots )  [inline]
```

Construct a PP Spline which have setted interpolation knots.

Parameters

_knots	interpolation knots

4.9.3 Member Function Documentation

4.9.3.1 diffVal()

```
template<int Order>
double ppSpline< Order, 1 >::diffVal (
             const double & x,
             const int & n = 1 )  [inline]
```

4.9.3.2 fitCurve() [1/2]

```
template<int Order>
void ppSpline< Order, 1 >::fitCurve (
             const vector< double > & fval,
             const BCType & bctype,
             const vector< double > & boundaryCondition = vector<double>{} )  [inline]
```

compute polynomial on each interpolation subinterval to interpolate a series of points

Parameters

fval	the value of curve at the interpolation knots
bctype	boundary condtion type
bondaryCondition	inputted extra condition

4.9.3.3 fitCurve() [2/2]

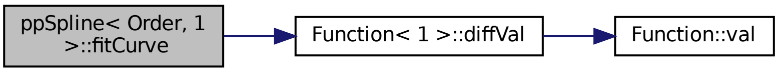
```
template<int Order>
void ppSpline< Order, 1 >::fitCurve (
             Function< 1 > & f,
             const BCType & bctype )  [inline]
```

compute polynomial on each interpolation subinterval to interpolate a function

Parameters

f	function that you want to interpolate
bctype	boundary condtion type

Here is the call graph for this function:



4.9.3.4 setKnots()

```
template<int Order>
void ppSpline< Order, 1 >::setKnots (
             const vector< double > & _knots )  [inline]
```

Set interpolation knots for splines.

Parameters

_knots	interpolation knots

4.9.3.5 val()

```
template<int Order>
virtual double ppSpline< Order, 1 >::val (
             const double & x )  [inline], [virtual]
```

pure virtual function to return the value of function at x

Parameters

x	independent variable of function

Returns

double a real number

Implements Function< 1 >.

4.9.4 Member Data Documentation

4.9.4.1 express

```
template<int Order>
vector<polynomial> ppSpline< Order, 1 >::express  [private]
```

expression of splines as a piecewise polynomials

4.9.4.2 isFitted

```
template<int Order>
bool ppSpline< Order, 1 >::isFitted  [private]
```

is the spline fitting some curve, if it's not, users can't get the value at any points of splines

4.9.4.3 isInitialized

```
template<int Order>
bool ppSpline< Order, 1 >::isInitialized  [private]
```

is the spline set the knots, if it's not, users can't fit the curve

4.9.4.4 knots

```
template<int Order>
vector<double> ppSpline< Order, 1 >::knots  [private]
```

interpolation knots

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

- splines.h

Chapter 5

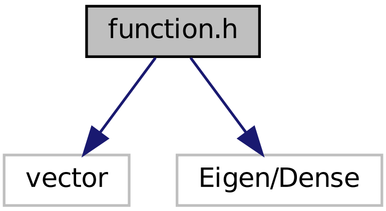
File Documentation

5.1 function.h File Reference

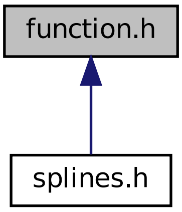
implement a function class and a polynomial class

```
#include <vector>
#include "Eigen/Dense"
```

Include dependency graph for function.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Function< Dim >

  A function(math) abstract class.
- class Function< 1 >

  specialization for one dimension function, definition is as same as high dimension
- class polynomial

  polynomial inherited from Function<1>

Functions

- polynomial operator+ (const polynomial &a, const polynomial &b)
- polynomial operator- (const polynomial &a, const polynomial &b)
- polynomial operator* (const polynomial &a, const polynomial &b)
- polynomial operator/ (const polynomial &a, const double &b)

5.1.1 Detailed Description

implement a function class and a polynomial class

Author

czx 3210103924

Version

1.0

Date

2024-01-13

Copyright

Copyright (c) 2024

5.1.2 Function Documentation

5.1.2.1 operator*()

```
polynomial operator* (
            const polynomial & a,
            const polynomial & b )
```

Parameters

a	one of polynomial to operator
b	the other polynomial to operator

Returns

polynomial result of multiplication of two polynomials

5.1.2.2 operator+()

```
polynomial operator+ (
             const polynomial & a,
             const polynomial & b )
```

Parameters

a	one of polynomial to operator
b	the other polynomial to operator

Returns

polynomial result of addition of two polynomials

5.1.2.3 operator-()

```
polynomial operator- (
             const polynomial & a,
             const polynomial & b )
```

Parameters

a	minuend
b	subtrahend

Returns

polynomial result of subtraction of two polynomials

5.1.2.4 operator/()

```
polynomial operator/ (
             const polynomial & a,
             const double & b )
```

Parameters

a	dividend, a polynomial
b	divisor, a real number

Returns

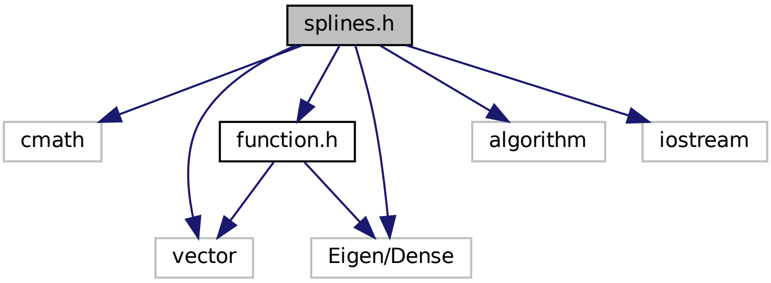
polynomial result of that a polynomial divide a real number

5.2 splines.h File Reference

implement arbitrary dimension liner and cubic piecewise polynomial splines and arbitrary order B-form splines and one dimension cardinal B splines

```
#include <cmath>
#include "function.h"
#include <vector>
#include <algorithm>
#include "Eigen/Dense"
#include <iostream>
```

Include dependency graph for splines.h:



Classes

- class BSpline< Order, Dim, t >

  arbitrary order BSplines for curve in arbitrary dimension
- class BSpline< Order, 1, t >

  specialization for one dimension B-form splines
- class BSpline< Order, 1, t >::BBasis

  the Basis function of B-form splines

- class CardinalBSpline< Order >

    one dimension cardinal B-form splines
- class ppSpline< Order, Dim >

    arbitrary order BSplines for curve in arbitrary dimension
- class ppSpline< Order, 1 >

    specialization for one dimension piecewise polynomial splines

Enumerations

- enum BSplineType { myDefault1 , cardinal }

    type of B-form splines
- enum BCType {
  myDefault2 , complete , nature , second ,
  notAKnot , periodic }

    boundary condition for cubic splines

Functions

- template<int Dim>
  double l2Nrom (const Vector< double, Dim > &p)

    compute 2-norm of points

5.2.1 Detailed Description

implement arbitrary dimension liner and cubic piecewise polynomial splines and arbitrary order B-form splines and one dimension cardinal B splines

Author

czx 3210103924

Version

1.0

Date

2024-01-13

Copyright

Copyright (c) 2024

5.2.2 Enumeration Type Documentation

5.2.2.1 BCType

```
enum BCType
```

boundary condition for cubic splines

Enumerator

myDefault2	default boundary condtion type for liner pp-spline, cardinal B-spline and arbitrary order B-splines
complete	complete cubic splines
nature	natural cubic splines
second	cubic splines with specified second derivative
notAKnot	not-a-knot cubic splines
periodic	periodic cubic splines

5.2.2.2 BSplineType

```
enum BSplineType
```

type of B-form splines

Enumerator

myDefault1	default type of B-form splines
cardinal	cardinal B-form splines

5.2.3 Function Documentation

5.2.3.1 l2Nrom()

```
template<int Dim>
double l2Nrom (
             const Vector< double, Dim > & p )
```

compute 2-norm of points

Template Parameters

Dim	dimension

Parameters

p	a point in the Dim dimension space

Returns

double: 2-norm of points

Index