# My Project

Generated by Doxygen 1.8.13

# Contents


**1 Cubic Spline Class** 1

**2 Class Index** 3

2.1 Class List . . . 3

**3 File Index** 5

3.1 File List . . . 5

**4 Class Documentation** 7

4.1 CubicSpline Class Reference . . . 7

4.1.1 Constructor & Destructor Documentation . . . 7

4.1.1.1 CubicSpline() [1/4] . . . 8

4.1.1.2 CubicSpline() [2/4] . . . 8

4.1.1.3 CubicSpline() [3/4] . . . 8

4.1.1.4 CubicSpline() [4/4] . . . 8

4.1.2 Member Function Documentation . . . 9

4.1.2.1 operator()() . . . 9

4.1.2.2 SetPoints() [1/3] . . . 9

4.1.2.3 SetPoints() [2/3] . . . 9

4.1.2.4 SetPoints() [3/3] . . . 10

**5 File Documentation** 11

5.1 CubicSpline.h File Reference . . . 11

5.1.1 Detailed Description . . . 11

5.1.2 Macro Definition Documentation . . . 12

5.1.2.1 ASSERT_WITH_MESSAGE . . . 12

5.2 CubicSpline_nonC++11.h File Reference . . . 12

5.2.1 Detailed Description . . . 12

5.2.2 Macro Definition Documentation . . . 13

5.2.2.1 ASSERT_WITH_MESSAGE . . . 13

**Index** 15

# Chapter 1

# Cubic Spline Class

Cubic spline class that is not dependent on other software.

### Usage

Two versions are present: a C++11 version which includes support for std::array and one that doesn't require C++11.

CubicSpline accepts std::vector, std::array (C++11) or standard arrays (i.e. double array[10]).

There are two ways to initiate the CubicSpline class:

1. You can begin with an empty constructor such as:

   ```
   {C++}
   CubicSpline f;
   ```

   and set the two arrays via:

   ```
   {C++}
   f.SetPoints(x, y);
   ```

2. Can input the two arrays in the constructor:

   ```
   {C++}
   CubicSpline f(x, y);
   ```

You are required to have the x array/vector in order. In a future release, it can sort the vectors if they are not sorted.

To get the cubic spline interpolation at position xi, call:

```
{C++}
double result = f(xi)
```

xi can be an integer, float or double but will always return a double and f is the CubicSpline class.

The spline does output values for values outside of your initial range but it cannot be well trusted and is not recommended.

main.cpp included has an example on how to run the CubicSpline class with a small benchmark.

## License

# Chapter 2

# Class Index

## 2.1 Class List

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

CubicSpline . . . . . 7

# Chapter 3

# File Index

## 3.1 File List

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


CubicSpline.h
Cubic spline implimentation with generic arrays, std::array and std::vector . . . . . 11
CubicSpline_nonC++11.h
Cubic spline implimentation with generic arrays and std::vector . . . . . 12

# Chapter 4

# Class Documentation

## 4.1 CubicSpline Class Reference

### Public Member Functions

- CubicSpline ()
- template<typename T >
  CubicSpline (const std::vector< T > &x, const std::vector< T > &y)
- template<typename T , int N, int M>
  CubicSpline (const T(&x) [N], const T(&y) [M])
- template<typename T , std::size_t N, std::size_t M>
  CubicSpline (const std::array< T, N > &x, const std::array< T, M > &y)
- template<typename T >
  void SetPoints (const std::vector< T > &x, const std::vector< T > &y)
- template<typename T , int N, int M>
  void SetPoints (const T(&x) [N], const T(&y) [M])
- template<typename T , std::size_t N, std::size_t M>
  void SetPoints (const std::array< T, N > &x, const std::array< T, M > &y)
- template<typename T >
  double operator() (T x) const
- template<typename T >
  **CubicSpline** (const std::vector< T > &x, const std::vector< T > &y)
- template<typename T , int N, int M>
  **CubicSpline** (const T(&x) [N], const T(&y) [M])
- template<typename T >
  void **SetPoints** (const std::vector< T > &x, const std::vector< T > &y)
- template<typename T , int N, int M>
  void **SetPoints** (const T(&x) [N], const T(&y) [M])
- template<typename T >
  double **operator()** (T x) const

### 4.1.1 Constructor & Destructor Documentation

#### 4.1.1.1 CubicSpline() [1/4]

```
CubicSpline::CubicSpline ( )  [inline]
```

Basic CubicSpline constructor

#### 4.1.1.2 CubicSpline() [2/4]

```
template<typename T >
CubicSpline::CubicSpline (
            const std::vector< T > & x,
            const std::vector< T > & y )  [inline]
```

CubicSpline constructor using std::vectors

**Parameters**

| | |
|---|---|
| *x.* | x-vector of the spline. Needs to be in increasing order and same type as y-vector |
| *y.* | y-vector of the spline. Needs to be the same type as x-vector |

#### 4.1.1.3 CubicSpline() [3/4]

```
template<typename T , int N, int M>
CubicSpline::CubicSpline (
            const T(&) x[N],
            const T(&) y[M] )  [inline]
```

CubicSpline constructor using arrays

**Parameters**

| | |
|---|---|
| *x.* | x-array of the spline. Needs to be in increasing order and same type as y-array |
| *y.* | y-array of the spline. Needs to be the same type as x-array |

#### 4.1.1.4 CubicSpline() [4/4]

```
template<typename T , std::size_t N, std::size_t M>
CubicSpline::CubicSpline (
            const std::array< T, N > & x,
            const std::array< T, M > & y )  [inline]
```

CubicSpline constructor using std::array

**Parameters**

| | |
|---|---|
| *x.* | x-array of the spline. Needs to be in increasing order and same type as y-array |
| *y.* | y-array of the spline. Needs to be the same type as x-array |

### 4.1.2 Member Function Documentation

#### 4.1.2.1 operator()()

```
template<typename T >
double CubicSpline::operator() (
            T x ) const  [inline]
```

CubicSpline Operator ().

**Parameters**

| | |
|---|---|
| *x.* | Position where you want to evaluate the spline |

**Returns**

the evaluation of the spline as a double

#### 4.1.2.2 SetPoints() [1/3]

```
template<typename T >
void CubicSpline::SetPoints (
            const std::vector< T > & x,
            const std::vector< T > & y )  [inline]
```

CubicSpline SetPoints using std::vector. To be used when using the default constructor

**Parameters**

| | |
|---|---|
| *x.* | x-array of the spline. Needs to be in increasing order and same type as y-array |
| *y.* | y-array of the spline. Needs to be the same type as x-array |

#### 4.1.2.3 SetPoints() [2/3]

```
template<typename T , int N, int M>
void CubicSpline::SetPoints (
```

```
            const T(&) x[N],
            const T(&) y[M] )  [inline]
```

CubicSpline SetPoints using arrays. To be used when using the default constructor

**Parameters**

| | |
|---|---|
| *x.* | x-array of the spline. Needs to be in increasing order and same type as y-array |
| *y.* | y-array of the spline. Needs to be the same type as x-array |

#### 4.1.2.4 SetPoints() [3/3]

```
template<typename T , std::size_t N, std::size_t M>
void CubicSpline::SetPoints (
            const std::array< T, N > & x,
            const std::array< T, M > & y )  [inline]
```

CubicSpline SetPoints using std::array. To be used when using the default constructor

**Parameters**

| | |
|---|---|
| *x.* | x-array of the spline. Needs to be in increasing order and same type as y-array |
| *y.* | y-array of the spline. Needs to be the same type as x-array |

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

- CubicSpline.h
- CubicSpline_nonC++11.h

# Chapter 5

# File Documentation

## 5.1 CubicSpline.h File Reference

Cubic spline implimentation with generic arrays, std::array and std::vector.

```
#include <array>
#include <assert.h>
#include <vector>
```

### Classes

- class CubicSpline

### Macros

- #define **ASSERT_WITH_MESSAGE**(condition, message)

### 5.1.1 Detailed Description

Cubic spline implimentation with generic arrays, std::array and std::vector.

**Author**

Josh Hooker

**Date**

07/30/2017

**Version**

1.0

### 5.1.2 Macro Definition Documentation

#### 5.1.2.1 ASSERT_WITH_MESSAGE

```
#define ASSERT_WITH_MESSAGE(
              condition,
              message )
```

**Value:**

```
do {                                                                \
    if (!(condition)) {                                              \
      printf((message));                                             \
    }                                                                \
    assert((condition));                                             \
  } while (false)
```

## 5.2 CubicSpline_nonC++11.h File Reference

Cubic spline implimentation with generic arrays and std::vector.

```
#include <assert.h>
#include <vector>
```

### Classes

- class CubicSpline

### Macros

- #define **ASSERT_WITH_MESSAGE**(condition, message)

### 5.2.1 Detailed Description

Cubic spline implimentation with generic arrays and std::vector.

#### Author

Josh Hooker

#### Date

07/30/2017

#### Version

1.0

### 5.2.2 Macro Definition Documentation

#### 5.2.2.1 ASSERT_WITH_MESSAGE

```
#define ASSERT_WITH_MESSAGE(
              condition,
              message )
```

**Value:**

```
do {                                                                \
    if (!(condition)) {                                              \
      printf((message));                                             \
    }                                                                \
    assert((condition));                                             \
  } while (false)
```

# Index

ASSERT_WITH_MESSAGE
    CubicSpline.h, 12
    CubicSpline_nonC++11.h, 13

CubicSpline, 7
    CubicSpline, 7, 8
    operator(), 9
    SetPoints, 9, 10
CubicSpline.h, 11
    ASSERT_WITH_MESSAGE, 12
CubicSpline_nonC++11.h, 12
    ASSERT_WITH_MESSAGE, 13

operator()
    CubicSpline, 9

SetPoints
    CubicSpline, 9, 10