

# ComplexNumber

0.1

Generated by Doxygen 1.8.11

MIT License

Copyright (c) 2017 Nathan Graule

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Packages . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Namespace Documentation</b>	<b>7</b>
4.1	SolarLiner Namespace Reference . . . . .	7
4.2	SolarLiner.ComplexNumber Namespace Reference . . . . .	7
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	SolarLiner.ComplexNumber.Complex Struct Reference . . . . .	9
5.1.1	Detailed Description . . . . .	11
5.1.2	Member Enumeration Documentation . . . . .	11
5.1.2.1	ComplexStyle . . . . .	11
5.1.3	Constructor & Destructor Documentation . . . . .	11
5.1.3.1	Complex(double a, double b, bool isTrigonometric=false) . . . . .	11
5.1.3.2	Complex(double theta) . . . . .	11
5.1.3.3	Complex(PointF point) . . . . .	11
5.1.4	Member Function Documentation . . . . .	12
5.1.4.1	Equals(object obj) . . . . .	12
5.1.4.2	GetHashCode() . . . . .	12

5.1.4.3	operator Complex(double value)	12
5.1.4.4	operator double(Complex val)	12
5.1.4.5	operator int(Complex val)	12
5.1.4.6	operator!=(Complex a, Complex b)	12
5.1.4.7	operator!=(Complex a, object b)	12
5.1.4.8	operator!=(object a, Complex b)	12
5.1.4.9	operator*(Complex a, double b)	12
5.1.4.10	operator*(double a, Complex b)	13
5.1.4.11	operator*(Complex a, Complex b)	13
5.1.4.12	operator+(Complex a, Complex b)	13
5.1.4.13	operator+(Complex a, double b)	13
5.1.4.14	operator-(Complex a)	13
5.1.4.15	operator-(Complex a, double b)	13
5.1.4.16	operator-(Complex a, Complex b)	13
5.1.4.17	operator/(Complex a, Complex b)	13
5.1.4.18	operator/(Complex a, double b)	13
5.1.4.19	operator/(double a, Complex b)	13
5.1.4.20	operator==(Complex a, Complex b)	13
5.1.4.21	operator==(Complex a, object b)	13
5.1.4.22	operator==(object a, Complex b)	13
5.1.4.23	SetRTheta(double r, double theta)	13
5.1.4.24	ToPoint()	14
5.1.4.25	ToString()	14
5.1.4.26	ToString(ComplexStyle Style)	14
5.1.5	Member Data Documentation	14
5.1.5.1	Epsilon	14
5.1.5.2	I	14
5.1.6	Property Documentation	14
5.1.6.1	Conjugate	14
5.1.6.2	Imaginary	15
5.1.6.3	IsImaginary	15
5.1.6.4	IsReal	15
5.1.6.5	Normalized	15
5.1.6.6	R	15
5.1.6.7	Real	15
5.1.6.8	Theta	15
<b>6</b>	<b>File Documentation</b>	<b>17</b>
6.1	ComplexNumber/Complex.cs File Reference	17
6.2	ComplexNumber/ComplexMath.cs File Reference	17

# Chapter 1

## Namespace Index

### 1.1 Packages

Here are the packages with brief descriptions (if available):

<a href="#">SolarLiner</a>	7
<a href="#">SolarLiner.ComplexNumber</a>	7



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">SolarLiner.ComplexNumber.Complex</a>	
Class handling complex numbers. . . . .	9





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

ComplexNumber/ <a href="#">Complex.cs</a> . . . . .	17
ComplexNumber/ <a href="#">ComplexMath.cs</a> . . . . .	17



## Chapter 4

# Namespace Documentation

### 4.1 SolarLiner Namespace Reference

#### Namespaces

- namespace [ComplexNumber](#)

### 4.2 SolarLiner.ComplexNumber Namespace Reference

#### Classes

- struct [Complex](#)  
*Class handling complex numbers.*
- class **ComplexMath**  
*Handles math in the [Complex](#) plane.*



## Chapter 5

# Class Documentation

### 5.1 SolarLiner.ComplexNumber.Complex Struct Reference

Class handling complex numbers.

#### Public Types

- enum [ComplexStyle](#) { [ComplexStyle.TRIGONOMETRIC](#), [ComplexStyle.CARTESIAN](#), [ComplexStyle.COMPLEX](#), [ComplexStyle.CO↵MPONENT](#), [ComplexStyle.PHASOR](#) }
- Enumerates the different kinds of string output used in [ToString\(\)](#).*

#### Public Member Functions

- [Complex](#) (double a, double b, bool isTrigonometric=false)  
*Initializes a new instance of the [SolarLiner.ComplexNumber.Complex](#) struct.*
- [Complex](#) (double theta)  
*Initializes a new instance of the [SolarLiner.ComplexNumber.Complex](#) struct.*
- [Complex](#) (PointF point)  
*Initializes a new instance of the [SolarLiner.ComplexNumber.Complex](#) struct.*
- void [SetRTheta](#) (double r, double theta)  
*Sets the trigonometric components of the complex number*
- override string [ToString](#) ()  
*Returns a System.String that represents the current [SolarLiner.ComplexNumber.Complex](#) using the Cartesian form.*
- string [ToString](#) ([ComplexStyle](#) Style)  
*Returns a System.String that represents the current [SolarLiner.ComplexNumber.Complex](#) using the given style.*
- PointF [ToPoint](#) ()  
*Converts the current complex number into a System.Drawing.PointF.*
- override bool [Equals](#) (object obj)  
*Determines whether the specified System.Object is equal to the current [SolarLiner.ComplexNumber.Complex](#).*
- override int [GetHashCode](#) ()  
*Serves as a hash function for a [SolarLiner.ComplexNumber.Complex](#) object.*

## Static Public Member Functions

- static implicit `operator Complex` (double value)
- static implicit `operator double` (`Complex` val)
- static implicit `operator int` (`Complex` val)
- static `Complex operator+` (`Complex` a, `Complex` b)
- static `Complex operator+` (`Complex` a, double b)
- static `Complex operator-` (`Complex` a)
- static `Complex operator-` (`Complex` a, double b)
- static `Complex operator-` (`Complex` a, `Complex` b)
- static `Complex operator*` (`Complex` a, double b)
- static `Complex operator*` (double a, `Complex` b)
- static `Complex operator*` (`Complex` a, `Complex` b)
- static `Complex operator/` (`Complex` a, `Complex` b)
- static `Complex operator/` (`Complex` a, double b)
- static `Complex operator/` (double a, `Complex` b)
- static bool `operator==` (`Complex` a, `Complex` b)
- static bool `operator==` (`Complex` a, object b)
- static bool `operator==` (object a, `Complex` b)
- static bool `operator!=` (`Complex` a, `Complex` b)
- static bool `operator!=` (`Complex` a, object b)
- static bool `operator!=` (object a, `Complex` b)

## Static Public Attributes

- static readonly `Complex I` = new `Complex`(0, 1)  
*The Imaginary number i.*
- static readonly `Complex Epsilon` = new `Complex`(double.Epsilon, double.Epsilon)  
*Smallest positive complex number.*

## Properties

- double `Real` [get, set]  
*Gets or sets the real part of the complex number.*
- double `Imaginary` [get, set]  
*Gets or sets the imaginary part of the complex number.*
- double `R` [get]  
*Gets the radius, or length, or magnitude, of the complex number.*
- double `Theta` [get]  
*Gets the angle of the complex number.*
- bool `IsImaginary` [get]  
*Returns whether the complex number is imaginary or not (no real part).*
- bool `IsReal` [get]  
*Returns whether the complex number is real or not (no imaginary part).*
- `Complex Conjugate` [get]  
*Returns the complex conjugate (a-bi).*
- `Complex Normalized` [get]  
*Returns the normalized complex number.*

### 5.1.1 Detailed Description

Class handling complex numbers.

### 5.1.2 Member Enumeration Documentation

#### 5.1.2.1 enum **SolarLiner.ComplexNumber.Complex.ComplexStyle** [strong]

Enumerates the different kinds of string output used in [ToString\(\)](#).

Enumerator

**TRIGONOMETRIC** Trigonometric form ( $r \cdot e^{i \cdot \theta \cdot \pi}$ ).

**CARTESIAN** Cartesian form ( $a + bi$ ).

**COMPONENT** Component form ( $[r, \theta]$ ).

**PHASOR** Prints complex number as a sum of sin and cos functions ( $r \cdot (\cos(\theta) + i \cdot \sin(\theta))$ ).

### 5.1.3 Constructor & Destructor Documentation

#### 5.1.3.1 **SolarLiner.ComplexNumber.Complex.Complex** ( double a, double b, bool isTrigonometric = false )

Initializes a new instance of the [SolarLiner.ComplexNumber.Complex](#) struct.

Parameters

<i>a</i>	The real or radius component.
<i>b</i>	The imaginary or angle component.
<i>isTrigonometric</i>	If set to <code>true</code> , a and b are trigonometric inputs.

#### 5.1.3.2 **SolarLiner.ComplexNumber.Complex.Complex** ( double theta )

Initializes a new instance of the [SolarLiner.ComplexNumber.Complex](#) struct.

Parameters

<i>theta</i>	Trigonimetric angle component.
--------------	--------------------------------

#### 5.1.3.3 **SolarLiner.ComplexNumber.Complex.Complex** ( PointF point )

Initializes a new instance of the [SolarLiner.ComplexNumber.Complex](#) struct.

Parameters

<i>point</i>	Point to convert to a complex.
--------------	--------------------------------

### 5.1.4 Member Function Documentation

#### 5.1.4.1 override bool SolarLiner.ComplexNumber.Complex.Equals ( object obj )

Determines whether the specified System.Object is equal to the current [SolarLiner.ComplexNumber.Complex](#).

Parameters

<i>obj</i>	The System.Object to compare with the current <a href="#">SolarLiner.ComplexNumber.Complex</a> .
------------	--

Returns

true if the specified System.Object is equal to the current [SolarLiner.ComplexNumber.Complex](#); otherwise, false.

#### 5.1.4.2 override int SolarLiner.ComplexNumber.Complex.GetHashCode ( )

Serves as a hash function for a [SolarLiner.ComplexNumber.Complex](#) object.

Returns

A hash code for this instance that is suitable for use in hashing algorithms and data structures such as a hash table.

#### 5.1.4.3 static implicit SolarLiner.ComplexNumber.Complex.operator **Complex** ( double value ) [static]

#### 5.1.4.4 static implicit SolarLiner.ComplexNumber.Complex.operator double ( **Complex** val ) [static]

May produce a loss of data when [SolarLiner.ComplexNumber.Complex](#) has an imarinary part.

#### 5.1.4.5 static implicit SolarLiner.ComplexNumber.Complex.operator int ( **Complex** val ) [static]

May produce a loss of data when [SolarLiner.ComplexNumber.Complex](#) has an imarinary part.

#### 5.1.4.6 static bool SolarLiner.ComplexNumber.Complex.operator!= ( **Complex** a, **Complex** b ) [static]

#### 5.1.4.7 static bool SolarLiner.ComplexNumber.Complex.operator!= ( **Complex** a, object b ) [static]

#### 5.1.4.8 static bool SolarLiner.ComplexNumber.Complex.operator!= ( object a, **Complex** b ) [static]

#### 5.1.4.9 static **Complex** SolarLiner.ComplexNumber.Complex.operator\* ( **Complex** a, double b ) [static]



- 5.1.4.10 static **Complex** SolarLiner.ComplexNumber.Complex.operator\* ( double a, **Complex** b )  
[static]
- 5.1.4.11 static **Complex** SolarLiner.ComplexNumber.Complex.operator\* ( **Complex** a, **Complex** b )  
[static]
- 5.1.4.12 static **Complex** SolarLiner.ComplexNumber.Complex.operator+ ( **Complex** a, **Complex** b )  
[static]
- 5.1.4.13 static **Complex** SolarLiner.ComplexNumber.Complex.operator+ ( **Complex** a, double b )  
[static]
- 5.1.4.14 static **Complex** SolarLiner.ComplexNumber.Complex.operator- ( **Complex** a ) [static]
- 5.1.4.15 static **Complex** SolarLiner.ComplexNumber.Complex.operator- ( **Complex** a, double b )  
[static]
- 5.1.4.16 static **Complex** SolarLiner.ComplexNumber.Complex.operator- ( **Complex** a, **Complex** b )  
[static]
- 5.1.4.17 static **Complex** SolarLiner.ComplexNumber.Complex.operator/ ( **Complex** a, **Complex** b )  
[static]
- 5.1.4.18 static **Complex** SolarLiner.ComplexNumber.Complex.operator/ ( **Complex** a, double b )  
[static]
- 5.1.4.19 static **Complex** SolarLiner.ComplexNumber.Complex.operator/ ( double a, **Complex** b )  
[static]
- 5.1.4.20 static bool SolarLiner.ComplexNumber.Complex.operator== ( **Complex** a, **Complex** b )  
[static]
- 5.1.4.21 static bool SolarLiner.ComplexNumber.Complex.operator== ( **Complex** a, object b ) [static]
- 5.1.4.22 static bool SolarLiner.ComplexNumber.Complex.operator== ( object a, **Complex** b ) [static]
- 5.1.4.23 void SolarLiner.ComplexNumber.Complex.SetRTheta ( double r, double theta )

Sets the trigonometric components of the complex number

Parameters

<i>r</i>	The radius component.
<i>theta</i>	The angle component.

#### 5.1.4.24 `PointF` `SolarLiner.ComplexNumber.Complex.ToPoint ( )`

Converts the current complex number into a `System.Drawing.PointF`.

Returns

A new `System.Drawing.PointF` corresponding to the complex number.

#### 5.1.4.25 `override string` `SolarLiner.ComplexNumber.Complex.ToString ( )`

Returns a `System.String` that represents the current [SolarLiner.ComplexNumber.Complex](#) using the Cartesian form.

Returns

A `System.String` that represents the current [SolarLiner.ComplexNumber.Complex](#).

#### 5.1.4.26 `string` `SolarLiner.ComplexNumber.Complex.ToString ( ComplexStyle Style )`

Returns a `System.String` that represents the current [SolarLiner.ComplexNumber.Complex](#) using the given style.

Returns

A `System.String` that represents the current [SolarLiner.ComplexNumber.Complex](#).

Parameters

<i>Style</i>	Chosen complex styling.
--------------	-------------------------

### 5.1.5 Member Data Documentation

#### 5.1.5.1 `readonly Complex` `SolarLiner.ComplexNumber.Complex.Epsilon = new Complex(double.Epsilon, double.Epsilon) [static]`

Smallest positive complex number.

#### 5.1.5.2 `readonly Complex` `SolarLiner.ComplexNumber.Complex.I = new Complex(0, 1) [static]`

The Imaginary number  $i$ .

### 5.1.6 Property Documentation

#### 5.1.6.1 `Complex` `SolarLiner.ComplexNumber.Complex.Conjugate [get]`

Returns the complex conjugate ( $a-bi$ ).

The complex conjugate.

5.1.6.2 double SolarLiner.ComplexNumber.Complex.Imaginary [get], [set]

Gets or sets the imaginary part of the complex number.

The imaginary part.

5.1.6.3 bool SolarLiner.ComplexNumber.Complex.IsImaginary [get]

Returns whether the complex number is imaginary or not (no real part).

true if this instance is imaginary; otherwise, false.

5.1.6.4 bool SolarLiner.ComplexNumber.Complex.IsReal [get]

Returns whether the complex number is real or not (no imaginary part).

true if this instance is real; otherwise, false.

5.1.6.5 **Complex** SolarLiner.ComplexNumber.Complex.Normalized [get]

Returns the normalized complex number.

The normalized complex number with radius=1.

5.1.6.6 double SolarLiner.ComplexNumber.Complex.R [get]

Gets the radius, or length, or magnitude, of the complex number.

5.1.6.7 double SolarLiner.ComplexNumber.Complex.Real [get], [set]

Gets or sets the real part of the complex number.

The real part.

5.1.6.8 double SolarLiner.ComplexNumber.Complex.Theta [get]

Gets the angle of the complex number.

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

- ComplexNumber/[Complex.cs](#)



## Chapter 6

# File Documentation

### 6.1 ComplexNumber/Complex.cs File Reference

#### Classes

- struct [SolarLiner.ComplexNumber.Complex](#)  
*Class handling complex numbers.*

#### Namespaces

- namespace [SolarLiner.ComplexNumber](#)

### 6.2 ComplexNumber/ComplexMath.cs File Reference

#### Classes

- class **SolarLiner.ComplexNumber.ComplexMath**  
*Handles math in the [Complex](#) plane.*

#### Namespaces

- namespace [SolarLiner.ComplexNumber](#)

