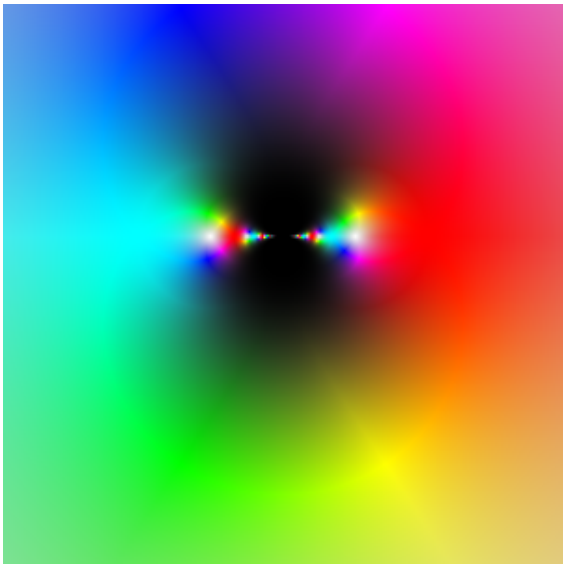
Mathematics with Python and Ruby/Complex numbers in Ruby



As real numbers can be strictly ordered while complex numbers can't be, an object oriented language such as *Ruby* doesn't really consider the complex numbers as numbers. Yet *Ruby* has an object to handle these numbers, and it is called *Complex*.

1 Example

For example, to create the complex number *4+3i* it suffices to write

```
a=Complex(4,3) puts(a)
```

This needs two numbers which can be fractions or integers. In this last case, the complex number is a gaussian integer. This information could be useful at a dinner, who knows?

2 Operations

2.1 Basic operations

Addition, subtraction, multiplication and division are still denoted by +, -, * and /:

```
a=Complex(2,3) b=Complex(4,3) puts(a+b) puts(a-b) puts(a*b) puts(a/b)
```

These examples show that the sum, the difference and the product of two gaussian integers are gaussian too, but their quotient is not necessarily gaussian. The example with the subtraction shows that even when the result of an operation is real, *Ruby* does consider it as complex anyway.

2.2 Exponentiation

To raise a complex to a power, the double asterisk is still used:

```
i=Complex(0,1) puts(i**2) puts(i**i)
```

Here $i^2 = 0i-1$ and not *really* -1! Also, i^i is a real number! By the way an exponent need not be a real number.

But with 0.5 as an exponent one can compute *the* square root of a complex number. But any complex number (except zero) has *two* square roots. How does *Ruby* choose between them? For example the square roots of *7+24i* are *4+3i* and *−4-3i*. *Ruby* chooses the first one. Other examples:

```
puts((−1)**0.5) puts(Complex(−1,0)**0.5)
```

If -1 is seen as a real number, it has no square root at all, whereas considered as a complex number, it has two square roots, the one which is nearest to *i* is displayed (but it is not exactly equal to *i* because of roundup errors)

3 Properties

3.1 Example

The two (real) numbers which are used to create a complex number are its *real part* and *imaginary part*. They can be obtained with *real* and *imag*:

```
a=Complex(4,3) puts(a.real) puts(a.imag) puts(a.conj)
```

This example shows an othe property, the *conjugate* of a complex number, which, contrary to the other ones, is a complex number too.

4 Geometrical properties

The main properties of a complex number which are useful to geometry are its *modulus* and *argument*:

```
a=Complex(4,3) puts(a.abs) puts(a.arg) puts(a.polar)
```

The last property, *polar*, gives at the same time the modulus and the argument, which allows one to solve the problem from the last chapter:

```
a=12 b=5 z=Complex(a,b) puts(z.polar)
```

with *require 'cmath'* one can access to functions on complex numbers.

5 Exponential

```
require 'cmath' t=Complex(0,Math::PI/3) w=CMath.exp(t) puts(w.real==0.5) puts(w.real-0.5) puts(w.imag==Math.sqrt(3)/2)
```

This example shows how legitimate it is to write $e^{-\frac{\pi}{3}i}$ for $\frac{1}{2}+i\frac{\sqrt{3}}{2}$.

The hyperbolic functions are also available for the complex numbers:

```
require 'cmath' a=Complex(4,3) puts(CMath.cosh(a)) puts(CMath.sinh(a)) puts(CMath.tanh(a))
```

6 Logarithms

The inverses of the preceding function can also be applied to complex numbers:

```
require 'cmath' a=Complex(4,3) puts(CMath.log(a)) puts(CMath.log10(a)) puts(CMath.acosh(a)) puts(CMath.asinh(a)) puts(CMath.atanh(a))
```

7 Circular functions

7.1 Direct

Every complex number has a cosine, a sine and a tangent:

```
require 'cmath' z=Complex(4,3) puts(CMath.cos(z)) puts(CMath.sin(z)) puts(CMath.tan(z))
```

7.2 Inverse

```
require 'cmath' z=Complex(4,3) puts(CMath.acos(z)) puts(CMath.asin(z)) puts(CMath.atan(z))
```

It is even possible to apply *atan2* to a pair of complex numbers:

```
require 'cmath' a=Complex(4,3) b=Complex(2,1) puts(CMath.atan2(a,b))
```

This complex version of the atan2 function is a good example of a complex surface: atan2(−1,−1) = 5π/4, whereas atan2(1,1) = atan(1) = π/4:

```
require 'cmath' puts(CMath.atan2(−1,−1)) puts(CMath.atan2(1,1))
```

8 Text and image sources, contributors, and licenses

8.1 Text

- **Mathematics with Python and Ruby/Complex numbers in Ruby** *Source:* https://en.wikibooks.org/wiki/Mathematics_with_Python_and_Ruby/Complex_numbers_in_Ruby?oldid=2577593 *Contributors:* Dcljr, Hackbinary, Alain Busser and Collingwood

8.2 Images

- **File:Sin1perz.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/a5/Sin1perz.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Kovzol

8.3 Content license

- Creative Commons Attribution-Share Alike 3.0