# Entiers_Gauss

February 17, 2019

## 1 Entiers de Gauss

L'ensemble $\mathbb{G} = \{x + iy, x, y \in \mathbb{Z}\}$ est un sous-anneau du corps $\mathbb{C}$ des nombres complexes. Il est appelé l'anneau des entiers de Gauss. Dans ce qui suit nous représenterons en Python les éléments de l'anneau $\mathbb{G}$ par des couples $(x, y)$ d'entiers relatifs. Très précisément, le couple $(x, y)$ représente l'élément $x + iy \in \mathbb{G}$.

Après avoir défini les opérations de l'anneau, on écrit une division euclidienne dans $\mathbb{G}$. À partir de là, il est facile de décrire l'algorithme d'Euclide qui calcule le pgcd de deux entiers de Gauss.

### 1.1 1. Opérations dans $\mathbb{G}$

#### 1.1.1 1.1 Addition, soustraction, opposé, conjugué

Ces opérations s'écrivent sans difficultés.

```
In [4]: def add(z1, z2):
            return (z1[0] + z2[0], z1[1] + z2[1])

In [5]: add((1, 2), (3, -1))

Out[5]: (4, 1)

In [6]: def sub(z1, z2):
            return (z1[0] - z2[0], z1[1] - z2[1])

In [7]: sub((1, 2), (3, -1))

Out[7]: (-2, 3)

In [8]: def oppose(z):
            return (-z[0], -z[1])

In [9]: oppose((2, -3))

Out[9]: (-2, 3)

In [10]: def conjugue(z):
             return (z[0], -z[1])

In [11]: conjugue((2, 1))

Out[11]: (2, -1)
```

### 1.1.2 1.2 Produit, carré du module, inverse

Ici encore, rien à dire.

```
In [12]: def mult(z1, z2):
             a = z1[0] * z2[0] - z1[1] * z2[1]
             b = z1[0] * z2[1] + z1[1] * z2[0]
             return (a, b)

In [13]: mult((1, 2), (3, 4))

Out[13]: (-5, 10)

In [14]: def norme(z):
             return z[0] ** 2 + z[1] ** 2

In [15]: norme((1, 2))

Out[15]: 5

In [16]: def inverse(z):
             if norme(z) == 1: return conjugue(z)
             else: raise Exception('non inversible')

In [17]: inverse((0, 1))

Out[17]: (0, -1)
```

### 1.1.3 1.3 Divisibilité

Soient $a, b \in \mathbb{G}$. $b$ divise $a$ lorque le nombre complexe $\frac{a}{b} \in \mathbb{C}$ est en fait dans $\mathbb{G}$.

On définit donc le quotient dans $\mathbb{C}$ de deux entiers de Gauss. La fonction `quot_C` ci-dessous prend en paramètres $z_1, z_2 \in \mathbb{G}$, $z_2 \neq 0$, et renvoie un couple $(z, n)$ où $z \in \mathbb{G}$ et $n \in \mathbb{N}^*$ sont tels que $\frac{z_1}{z_2} = \frac{z}{n}$. Rien de magique à cela, il suffit de remarquer que $\frac{a+ib}{c+id} = \frac{(ac+bd)+i(bc-ad)}{c^2+d^2}$.

On a alors $z_2 | z_1$ si et seulement si $n$ divise dans $\mathbb{Z}$ la partie réelle et la partie imaginaire de $z$.

```
In [18]: def quot_C(z1, z2):
             a, b = z1
             c, d = z2
             return ((a * c + b * d, b * c - a * d), c * c + d * d)

In [19]: quot_C((16, 2), (3, 11))

Out[19]: ((70, -170), 130)
```

Il est maintenant simple d'écrire la relation "divise".

```
In [20]: def divise (b, a):
             z, n = quot_C(a, b)
             u, v = z
             return (u % n == 0) and (v % n == 0)
```

```
In [21]: divise((1, 2), (5, 5))
```

```
Out[21]: True
```

```
In [22]: quot_C((5, 5), (1, 2))
```

```
Out[22]: ((15, -5), 5)
```

## 1.2 2. Une division euclidienne dans $\mathbb{G}$

La fonction `quot_G`prend deux entiers de Gauss $a$ et $b$ et paramètres. Elle renvoie un entier de Gauss $q$ tel que $|\frac{a}{b} - q| < 1$. Les entiers $x$ et $y$ qui apparaissent dans le corps de la fonction sont en fait $\lfloor \frac{u}{n} + \frac{1}{2} \rfloor$ et $\lfloor \frac{v}{n} + \frac{1}{2} \rfloor$, c'est à dire deux entiers proches de $\frac{u}{n}$ et $\frac{v}{n}$ de moins de $\frac{1}{2}$.

```
In [23]: def quot_G(a, b):
             z, n = quot_C(a, b)
             u, v = z
             x = (2 * u + n) // (2 * n)
             y = (2 * v + n) // (2 * n)
             return (x, y)
```

```
In [24]: quot_G((16, 2), (3, 11))
```

```
Out[24]: (1, -1)
```

La fonction `div_G` effectue la division euclidienne des entiers de Gauss $a$ et $b$, $b \neq 0$. Elle renvoie un couple $(q, r) \in \mathbb{G}^2$ tel que $a = bq + r$ et $|r| < |b|$.

```
In [25]: def div_G(a, b):
             q = quot_G(a, b)
             r = sub(a, mult(q, b))
             return (q, r)
```

```
In [26]: div_G((16, 2), (3, 11))
```

```
Out[26]: ((1, -1), (2, -6))
```

## 1.3 3. Algorithme d'Euclide

Muni d'une division euclidienne, $\mathbb{G}$ entre dans la famille très sélect des anneaux euclidiens. Dans un tel anneau on peut définir la notion de pgcd. L'algorithme d'Euclide, bien connu pour les entiers relatifs, y reste valide.

### 1.3.1 3.1 Calcul du pgcd

```
In [27]: def gcd(a, b):
             while norme(b) != 0:
                 q, r = div_G(a, b)
                 a, b = b, r
             return a
```

```
In [28]: gcd((16, 2), (3, 11))
```

```
Out[28]: (-1, 3)
```

### 1.3.2 3.2 Affichage formatté

La même fonction, mais on renvoie la liste de tous les couples (quotient, reste) calculés par l'algorithme d'Euclide. On écrit d'abord une fonction qui renvoie une représentation de l'entier de Gauss $z$ sous la forme '$a + ib$'.

```
In [29]: def to_str(z):
             x, y = z
             if x == 0 and y == 0: return '0'
             elif y == 0: return str(x)
             elif x == 0: return str(y) + 'i'
             elif y > 0:
                 if y == 1: return str(x) + '+i'
                 else: return str(x) + '+' + str(y) + 'i'
             else:
                 if y == -1: return str(x) + '-i'
                 else: return str(x) + str(y) + 'i'
```

```
In [30]: to_str((-1,1))
```

```
Out[30]: '-1+i'
```

```
In [31]: def euclid(a, b):
             s = []
             while norme(b) != 0:
                 q, r = div_G(a, b)
                 s.append((to_str(q), to_str(r)))
                 a, b = b, r
             return s
```

```
In [32]: euclid((16, 2), (3, 11))
```

```
Out[32]: [('1-i', '2-6i'), ('-1+i', '-1+3i'), ('-2', '0')]
```

### 1.3.3 3.3 Vérification : coefficients de Bézout

Tout cela est bien, mais nos fonctions renvoient-elles des résultats corrects ? Une bonne façon de s'en persuader est d'écrire une fonction qui calcule les coefficients de Bézout. La fonction ci-dessous prend en paramètres deux entiers de Gauss $a$ et $b$. Elle renvoie un triplet $(u, v, \delta)$ d'entiers de Gauss tels que $ua + vb = \delta$ et $\delta$ est un pgcd de $a$ et $b$.

```
In [33]: def bezout(a, b):
             u1, v1, r1 = (1, 0), (0, 0), a
             u2, v2, r2 = (0, 0), (1, 0), b
             while norme(r2) != 0:
                 q, r = div_G(r1, r2)
                 u = sub(u1, mult(q, u2))
                 v = sub(v1, mult(q, v2))
                 u1, v1, r1 = u2, v2, r2
                 u2, v2, r2 = u, v, r
             return (u1, v1, r1)
```

```
In [34]: a = (1678665, 2443543)
         b = (-35543211, 1178967)
         u, v, r = bezout(a, b)
         print(u, v, r)
```

```
(-550285, 9584249) (-697884, 391673) (1, -1)
```

Maintenant, calculons $au + bv - \delta$ !

```
In [35]: sub(add(mult(u, a), mult(v, b)),r)
```

```
Out[35]: (0, 0)
```

Nous avons bon espoir que tout n'est pas faux dans ce qui précède :-).