Constructions Géométriques

Rationnel, nombre d'or et fractions continues

Alexis NEDELEC

LISYC EA 3883 UBO-ENIB-ENSIETA
Centre Européen de Réalité Virtuelle
Ecole Nationale d'Ingénieurs de Brest

Objectifs du cours

Construction de package python

- tests unitaires de fonctions
- notion de packages, modules
- programmes de tests des fonctions de modules
- installation, distribution (`setup.py, MANIFEST`)

Projet d'étude : rationnels et irrationnels

- définition des rationnels
- représentations géométriques
- arbre de construction des rationnels
- fractions continues et irrationnels
- construction géométrique du nombre d'or

Construction de packages

Arborescence de répertoires

```
{logname@hostname} tree nombres
nombres
|-- MANIFEST
|-- README.txt
|-- doc
|-- setup.py
|-- src
|   `-- rationnel
|       |-- __init__.py
|       |-- figures.py
|       |-- geometrie.py
|       |-- nombre.py
|       `-- utils.py
```

Programmes de test

Arborescence de répertoires

```
‘-- test
    |-- calkinWilf.py
    |-- constructionOr.py
    |-- euclide.py
    |-- ford.py
    |-- geometrie.py
    ‘-- spiraleOr.py
{logname@hostname}
```

Projet d'étude : les rationnels

Nombre rationnel : Définition

Quotient de 2 entiers relatifs

$$\mathbb{Q} = \{\frac{m}{n} | (m,n) \in \mathbb{Z} \times \mathbb{N} - \{0\}\}$$

$\mathbb{Z}$: ensemble des entiers
$\mathbb{N}$: ensemble des entiers naturels

Remarques

- plusieurs représentations : $\frac{1}{2}, \frac{2}{4}, \frac{3}{6}$
- forme privilégiée : (m,n) : nombres premiers entre eux
- fraction : nombre rationnel non-entier
- fraction irréductible : (m,n) premiers entre eux
- irrationnel : nombre réel qui n'est pas un rationnel

Nombre rationnel : définition

Arithmétique des rationnels

Egalité :

$$\frac{a}{b} = \frac{c}{d} \Longleftrightarrow a.d = b.c$$

Addition :

$$\frac{a}{b} + \frac{c}{d} = \frac{a.d + b.c}{b.d}$$

Multiplication :

$$\frac{a}{b}.\frac{c}{d} = \frac{a.c}{b.d}$$

Opposé,inverse :

$$-(\frac{a}{b}) = \frac{-a}{b} = \frac{a}{-b} \ , \ (\frac{a}{b})^{-1} = \frac{b}{a} \ (a \neq 0)$$

Nombre rationnel : construction

Suites de Farey (1766-1826)

"Si, après avoir rangé dans leur ordre de grandeur les fractions irréductibles ... on en prend trois de suite ... en additionant les numérateurs et dénominateurs de la première et de la troisème on obtient une fraction non-nécessairement irréductible, égale à la fraction intermédiaire (fraction médiane)"

Suites de Farey : Exemples

$$F_1 = \{0, 1\}\ ,\ F_2 = \{0, \frac{1}{2}, 1\}\ ,\ F_3 = \{0, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, 1\}$$

$$F_4 = \{0, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, 1\}\ ,\ F_5 = \{0, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, 1\}$$

Suites de Farey

Propriétés

Si $\frac{a}{b}$ et $\frac{c}{d}$ sont deux fractions consécutives de F_n alors :

$$b.c - a.d = 1$$

Si $\frac{a}{b}, \frac{c}{d}, \frac{e}{f}$ sont trois fractions successives de F_n alors :

$$\frac{c}{d} = \frac{a+e}{b+f} = \frac{a}{b} \oplus \frac{e}{f}$$

Récurrence : F_{n+1} à partir de F_n

- pour 2 fractions consécutives $\frac{a}{b}$ et $\frac{c}{d}$ de F_n
- insérer la fraction médiane irréductible si $(b+d) \leq n+1$

Suites de Farey : Récurrence

Récurrence : F_{n+1} à partir de F_n

$$F_6 = \{0, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{5}{6}, 1\}$$

On obtient F_7 en insérant :

$$F_7 = \{0, \frac{1}{7}, \frac{1}{6}, \frac{1}{5}, \frac{1}{4}, \frac{2}{7}, \frac{1}{3}, \frac{2}{5}, \frac{3}{7}, \frac{1}{2}, \frac{4}{7}, \frac{3}{5}, \frac{2}{3}, \frac{5}{7}, \frac{3}{4}, \frac{4}{5}, \frac{5}{6}, \frac{6}{7}, 1\}$$

Suites de Farey : implémentation

Module : rationnel.nombre

```
def farey(n) :
    """
    suite de Farey d'ordre (n)  -> list (Fn)
    >>> farey(1)
    [(0, 1), (1, 1)]
    >>> farey(3)
    [(0, 1), (1, 3), (1, 2), (2, 3), (1, 1)]
    ...
    """
    assert type(n) is int and n > 0
    Fn = [(0,1),(1,1)]
```

Suites de Farey : implémentation

Module : rationnel.nombre

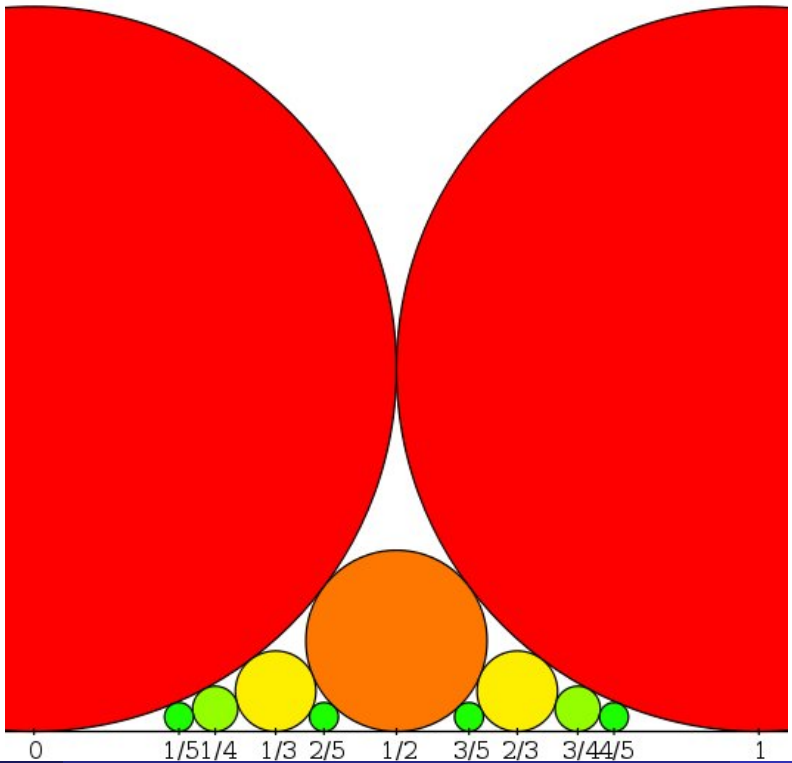
```
    if n > 1 :
        i=0
        while Fn[i][0]!=1 or Fn[i][1]!=1:
            if Fn[i][1]+Fn[i+1][1] <= n :
                Fn.insert(i+1,\
                          (Fn[i][0]+Fn[i+1][0],\
                           Fn[i][1]+Fn[i+1][1]))
            else :
                i=i+1
    return Fn
```

Suites de Farey : représentation

Cercles de Ford (1886-1975)

Représentation géométrique associé à la fraction irréductible $\frac{p}{q}$

- cercle de centre $(\frac{p}{q}, \frac{1}{2q^2})$
- de rayon $\frac{1}{2q^2}$



Cercle de Ford : implémentation

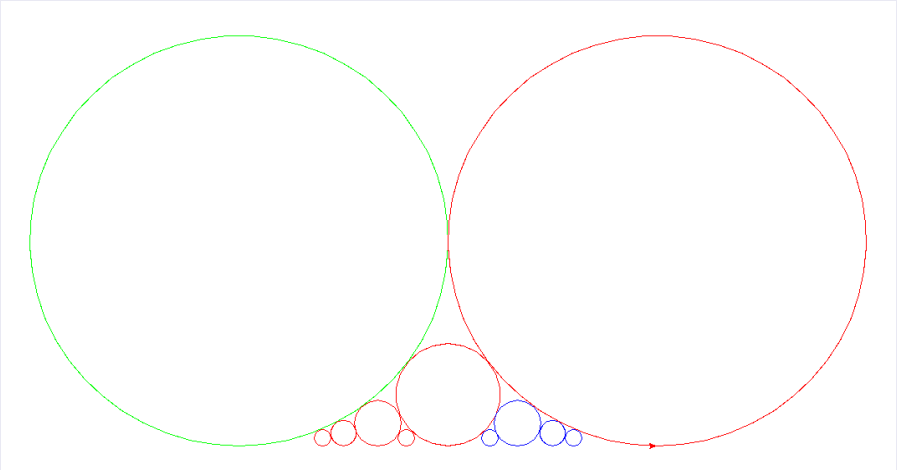
Module : rationnel.figures

```
def fordCircle(rational,scale=500.0, x=0, y=0) :
    """
    representation de (rational) par un cercle de Ford
    ...
    """
    assert type(rational) is tuple
    assert len(rational) == 2
    assert type(scale) is float
    a=(scale*rational[0])/rational[1]
    b=scale/(2*rational[1]**2)
    up()
    goto(x+int(a),y)
    down()
    circle(b)
```

Cercle de Ford : Programme de test

Répertoire de test : `ford.py`

```
import sys
sys.path.append("../src")
from rationnel.nombre import farey
from rationnel.figures import fordCircle
for rational in farey(5) :
    fordCircle(rational)
```



Nombre rationnel : construction

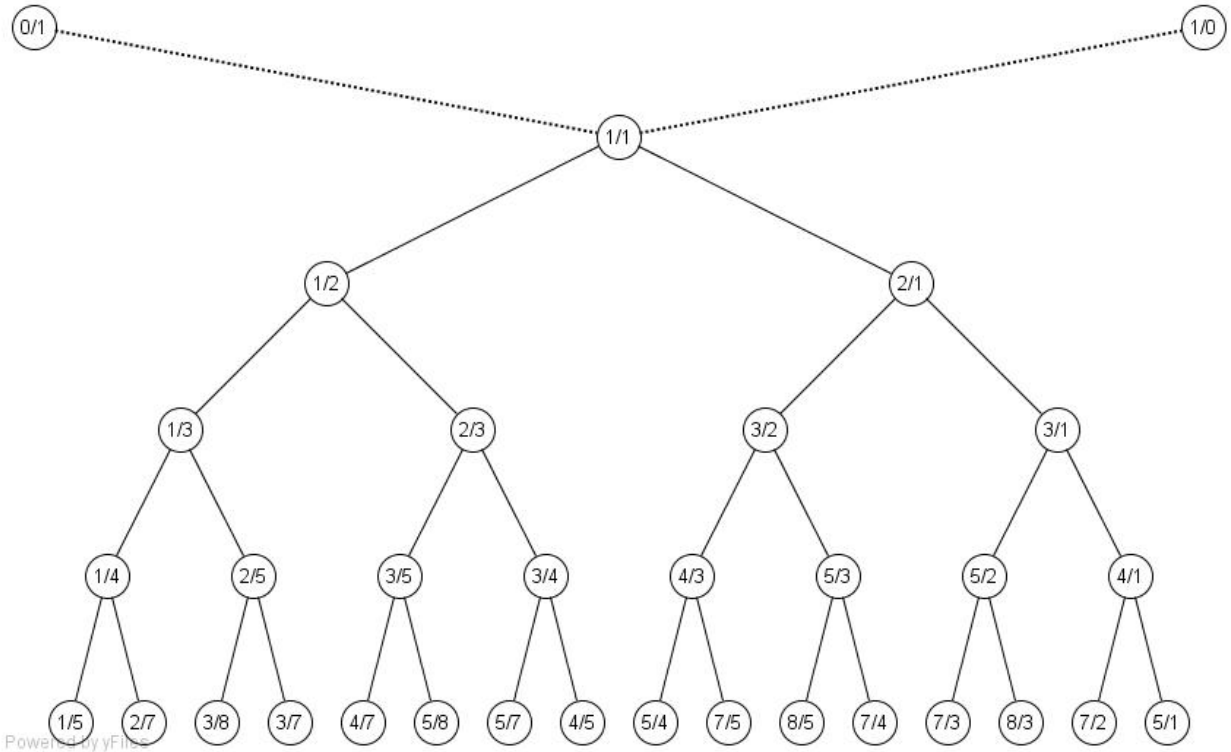
Arbre de Stern-Brocot

- Moritz Stern (1807-1894) en 1858 : *Über eine zahlentheoretische Funktion. Crelle's Journal 55 :193-220*
- Achille Brocot (1817-1878) en 1861 : *Calcul des rouages par approximation, nouvelle méthode. Revue Chronométrique 3 :186-194*

Définition

- construction de l'ensemble des rationnels $\mathbb{Q}$.
- on part du couple de fractions irréductibles (0/1, 1/0)
- on insère entre $(\frac{p}{q}, \frac{p'}{q'})$ la fraction médiante $(\frac{p+p'}{q+q'})$

Arbre de Stern-Brocot



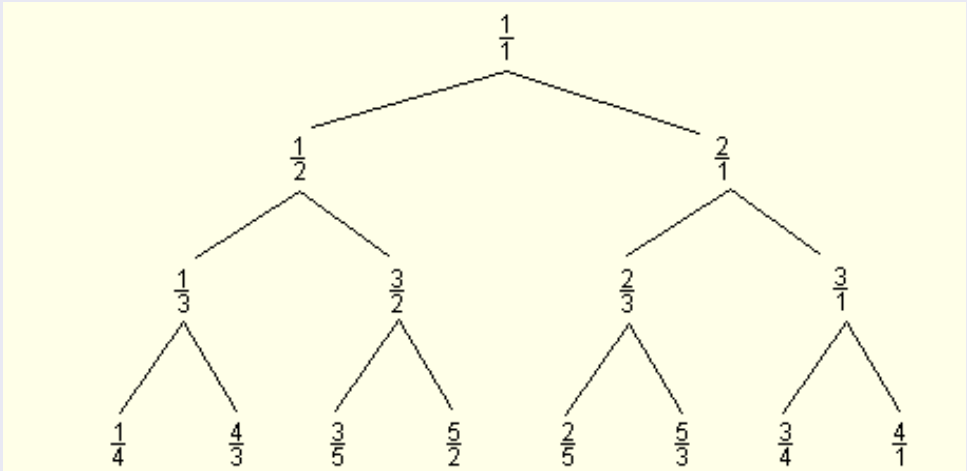
Remarques

- extrême-gauche se trouvent les fractions unitaires
- extrême-droite, les nombres entiers sous forme rationnelle

Arbre de Calkin-Wilf

Variation autour de l'arbre de Stern-Brocot

- on part d'un rationnel (a/b)
- on génère deux enfants : gauche $(\frac{a}{a+b})$, droite $(\frac{a+b}{b})$



- l'arbre contient tous les rationels positifs une seule fois
- sur chaque niveau : le dénominateur du rationel gauche coincide avec le numérateur de son voisin
- même remarque en developpant ligne à ligne

Arbre de Stern-Brocot

Trouver le rationnel positif correspondant à un nombre

- soit x ce nombre, $l = \frac{0}{1}$, $r = \frac{1}{0}$ $(r : \infty)$
- si $l \leq x \leq r$ alors trois cas possibles :
  - $l \leq x < l \oplus r$
  - $x = l \oplus r$
  - $l \oplus r < x \leq r$

Algorithme

- initialisation : $x, l = \frac{0}{1}$, $r = \frac{1}{0}$
- traitements
  - premier cas : $r \leftarrow (l \oplus r)$
  - deuxième cas : on a trouvé le rationnel correspondant
  - troisième cas : $l \leftarrow (l \oplus r)$
- arrêt : si x non-rationnel, définir un seuil de précision

Arbre de Stern-Brocot

Représentation matricielle

Un rationnel (p/q) peut s'exprimer par un mot constitué d'une succession de (L, R) représentant les matrices :

$$L = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} R = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

définissant le chemin de la racine jusqu'au noeud représentant le rationnel de l'arbre de Stern-Brocot.

Exemple : (5/7)

$$LRRL \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 7 \end{pmatrix}$$

Arbre de Stern-Brocot

Module : rationnel.nombre

```
from math import sqrt
from numpy import matrix
def code2rational(mot) :
    """
    conversion,codage LR,de (mot) -> nuplet (rational)
    >>> code2rational('LLRR')
    (3, 7)
    >>> code2rational('RLRLRLRLRLRLRLRLRLRLRLRLRL')
    (121393, 75025)
    """
    assert codeLR(mot)
    active = matrix('[1,0;0,1]')
    vector = matrix([[1],[1]])
```

Arbre de Stern-Brocot

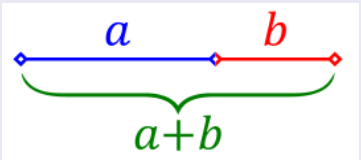
Module : rationnel.nombre

```
    L = matrix('[1,0;1,1]')
    R = matrix('[1,1;0,1]')
    for c in mot :
        if c == 'L' :
            active = active*L
        elif c =='R' :
            active = active*R
    result =  active*vector
    rational = (int(result[0][0]),int(result[1][0]))
    return rational
r=code2rational('RLRLRLRLRLRLRLRLRLRLRLRLRL')
print 1.*r[0]/r[1]      # 1.61803398867
print (1 + sqrt(5))/2   # 1.61803398875
```

Nombre d'or : $\varphi = \frac{1+\sqrt{5}}{2} = 1.61803398875...$

Définition Euclidienne

"Une droite est dite coupée en extrême et moyenne raison lorsque la droite entière est au plus grand segment comme le plus grand segment est au plus petit."

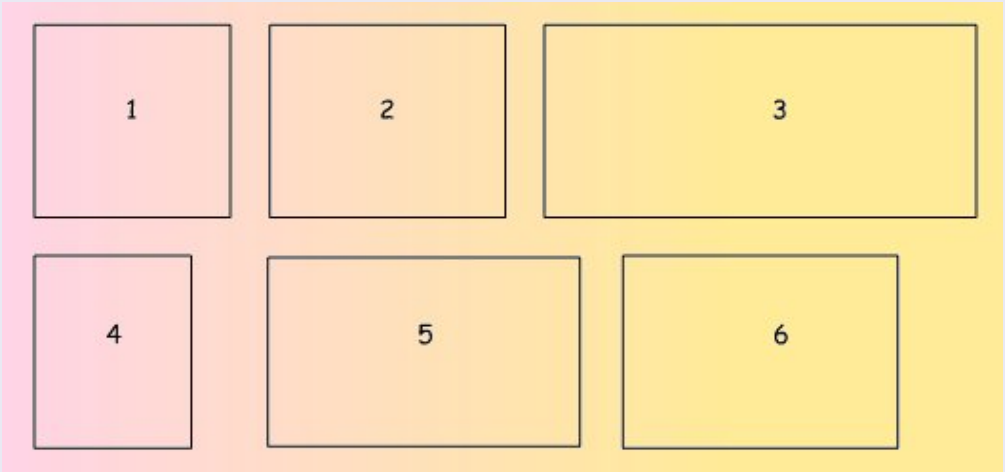


$$\frac{a}{b} = \frac{a+b}{a}$$

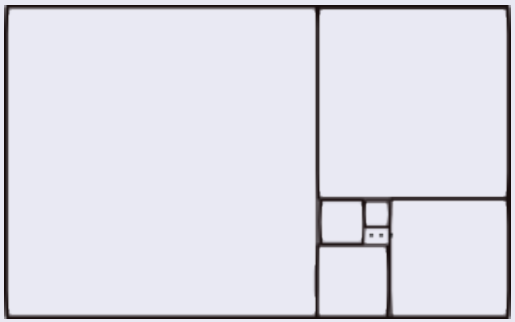
Le rapport $\frac{a}{b}$ est alors égal au nombre d'or ($\varphi = \frac{a}{b}$)

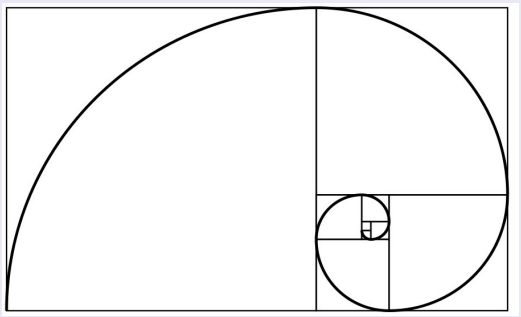
Nombre d'or : $\varphi = \frac{1+\sqrt{5}}{2} = 1.61803398875...$

Test de proportion d'or



Spirale d'Or

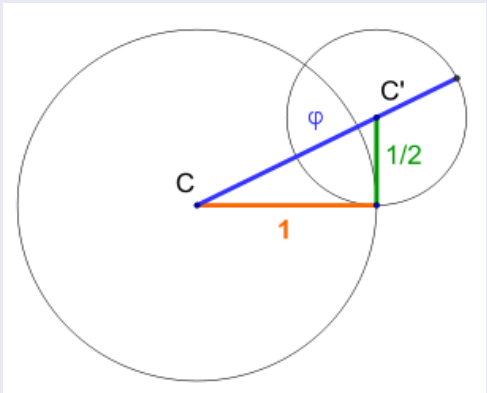




Construction du nombre d'Or

A la règle et au compas

- tracer un segment de longueur unité.
- tracer un segment de longueur 1/2, perpendiculaire au précédent
- tracer un cercle de centre C' (extrémité du segment précédent) et de rayon 1/2
- tracer le segment passant par les points (C,C') intersectant le cercle de centre (C') en dehors de (CC')



A la règle et au compas

Calculs à effectuer

Etant donné deux points `(p1,p2)` du plan

- trouver la direction d'un vecteur perpendiculaire :
  `direct=perpendiculaire(p1,p2)`
- calculer le centre C' (`center`) du cercle :

```
rayon = distance(p1,p2)/2.0
center = scale(rayon/norme(direct),direct)
center = somme(center,p2)
```

A la règle et au compas

Calculs à effectuer

- trouver les coefficients de la droite (`p1,center`) :
  `da,db = coeffsDroite(p1,center)`
- calculer l'intersection droite (`p1,center`) avec le cercle de centre (`C'`) et de rayon (`distance(p1,p2)/2.0`)
  `a,b,c= coeffsDroiteCercle(da,db,center,rayon)`
  `rac = racines(a,b,c)`
  `pi1,pi2 = pointsIntersection(da,db,rac)`
- récupérer le point d'intersection extérieur au segment (CC')

Coefficients de droite : $y = a.x + b$

Module : rationnel.geometrie

```
from linearSystems.directMethods.gauss import solve
def coeffsDroite(p1,p2) :
    """
    resoudre le systeme :
    a*x1 + b*1 = y1
    a*x2 + b*1 = y2
    (p1,p2) -> list : (a,b) coefficients de la droite
    >>> coeffsDroite([-100,-100],[100,100])
    [1.0, 0.0]
    >>> coeffsDroite([-100,100],[100,-100])
    [-1.0, 0.0]
    >>> coeffsDroite([0,0],[100,0])
    [0.0, 0.0]
```

Coefficients de droite : $y = a.x + b$

Module : rationnel.geometrie

```
>>> coeffsDroite([0,0],[0,100])
[]
"""
assert type(p1) is list and len(p1) > 1
assert type(p2) is list and len(p2) > 1
A = [[p1[0],1], [p2[0],1]]
b=[p1[1],p2[1]]
return solve(A,b)
```

Intersection droite/cercle

Module : rationnel.geometrie

```
def coeffsDroiteCercle(da,db,center,rayon=1.0) :
    """
    droite : Y=da*X+db,
    cercle : X**2+Y**2 = rayon**2
    intersection : (X-Xc)**2+((da*X+db-Yc)**2 = rayon**2
    trinome : (1+da**2)*X**2-2*(Xc+da*(Yc-db))*X
                            +Xc**2+ (db-Yc)**2-r**2 = 0
    -> tuple : (a,b,c) coefficients du trinome
    """
```

Intersection droite/cercle

Module : rationnel.geometrie

```
    assert type(da) is int or type(da) is float
    assert type(db) is int or type(db) is float
    assert type(rayon) is int or type(rayon) is float
    assert type(center) is list and len(center) > 1
    a = 1+da**2
    b = -2*(center[0]+ da*(center[1]-db))
    c = center[0]**2 + (db-center[1])**2 -rayon**2
    return a,b,c
```

Intersection droite/cercle

Module : rationnel.geometrie

```
def pointsIntersection(da,db,racines) :
    """
    da,db : coefficients de droite (Y=da*X+db)
    intersection droite/cercle (a*X**2 + b*X + c = 0)
    racines : de l'equation d'intersection
    -> list ([r1,da*r1+db],[r2,da*r2+db])
    """
    assert type(da) is int or type(da) is float
    assert type(db) is int or type(db) is float
    assert type(racines) is list and len(racines) == 2
```

Intersection droite/cercle

Module : rationnel.geometrie

```
    p1 =[]
    p2 =[]
    p1.append(racines[0])
    p1.append(da*racines[0]+db)
    p2.append(racines[1])
    p2.append(da*racines[1]+db)
    return p1,p2
```

Calcul du nombre d'or

Module : rationnel.geometrie

```
def constructionOr(p1,p2,seuil=1.e-6) :
    """Trouver le point (p) tel que:
    (1+sqrt(5))/2 == distance(p1,p)/distance(p1,p2)
    au (seuil) près  -> list (p)
    >>> constructionOr([0.0,0.0],[100.0,0.0])
    [144.72135954999581, 72.360679774997905]
    >>> constructionOr([0.0,100.0],[0.0,0.0])
    [72.360679774997905, -44.72135954999581]
    >>> constructionOr([-100.0,100.0],[100.0,-100.0])
    [334.1640786499874, -44.72135954999581]
    """
    assert type(p1) is list and len(p1) > 1
    assert type(p2) is list and len(p2) > 1
```

Calcul du nombre d'or

Module : rationnel.geometrie

```
nombreOr=(1+sqrt(5))/2
direction=perpendiculaire(p1,p2)
rayon=distance(p1,p2)/2.0
center=scale(rayon/norme(direction),direction)
center = somme(center,p2)io
da,db = coeffsDroite(p1,center)
a,b,c= coeffsDroiteCercle(da,db,center,rayon)
rac = racines(a,b,c)
pi1,pi2 = pointsIntersection(da,db,rac)
if fabs(distance(p1,pi1)/distance(p1,p2)-nombreOr)\
                <= seuil : p=pi1
else : p=pi2
return p
```

Représentation géométrique

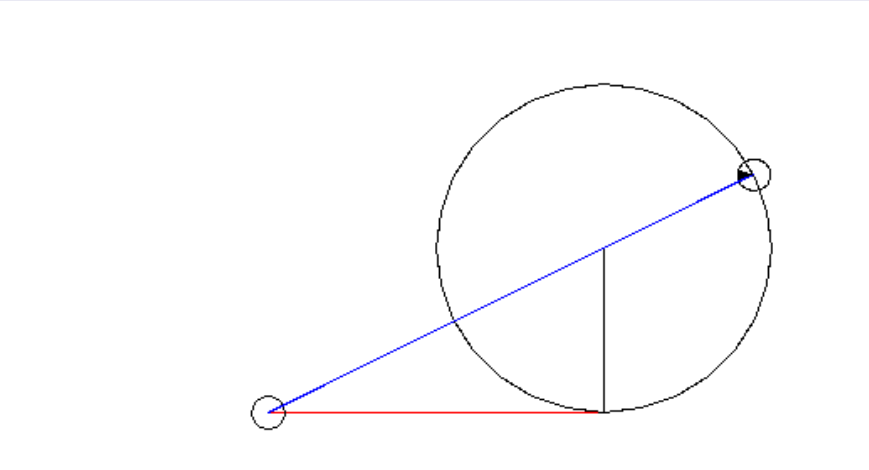
Module : rationnel.figures

```
def figureOr(p1,p2) :
    """
    Representation geometrique du nombre d'Or.
    Proportion d'Or (ligne bleu,ligne rouge)
    """
    p=constructionOr(p1,p2)
    color("red")
    ligne(p1,p2)
    center = perpendiculaire(p1,p2)
    rayon = distance(p1,p2)/2.0
    center = scale(rayon/norme(center),center)
    center = somme(center,p2)
    color("black")
    ligne(p2,center)
```

Représentation géométrique

Module : rationnel.figures

```
cercle(center,distance(p2,center))
cercle(p,10)
cercle(p1,10)
color("blue")
ligne(p1,p)
color("black")
return
```



Représentation géométrique

Répertoire de test : constructionOr.py

```
import sys
sys.path.append("../src")
from math import sqrt
from rationnel.geometrie import constructionOr,distance
from rationnel.figures import figureOr
p1=[0.0,200.0]
p2=[0.0,0.0]
print "nombredor",(1 + sqrt(5))/2
p=constructionOr(p1,p2)
print "regle et compas:",distance(p1,p)/distance(p1,p2)
figureOr(p1,p2)
```

Calcul du nombre d'or

Proportion d'or

$$\frac{a+b}{a} = \frac{a}{b} \Leftrightarrow 1 + \frac{b}{a} = \frac{a}{b} \Leftrightarrow \frac{a}{b} + 1 = \left(\frac{a}{b}\right)^2 \Leftrightarrow \left(\frac{a}{b}\right)^2 - \frac{a}{b} - 1 = 0$$

Equation du second degré suivante :

$$x^2 - x - 1 = 0$$

Une seule solution positive :

$$\varphi = \frac{1+\sqrt{5}}{2}$$

Nombre d'or et fraction continue

φ : Fraction continue

$\varphi^2 - \varphi - 1 = 0$

$\varphi^2 = \varphi + 1 \Rightarrow \varphi = 1 + \frac{1}{\varphi} \Rightarrow \varphi = 1 + \frac{1}{1+\frac{1}{\varphi}} \Rightarrow ...$

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cdots}}}}$$

φ : Radicaux infiniment itérés

$\varphi^2 = \varphi + 1 \Rightarrow \varphi = \sqrt{1+\varphi} \Rightarrow \sqrt{1+\sqrt{1+\varphi}} \Rightarrow ...$

$$\varphi = \sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+...}}}}$$

Fractions continues

Fractions continues généralisées

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \frac{b_3}{a_3 + \ldots}}}$$

Fractions continues simples : $b_1 = 1, b_2 = 1, b_3 = 1 \ldots$

Exemple de fraction continue simple

$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \frac{1}{2 + \ldots}}}$$

Fractions continues

Fractions continues et nombre réel

Tout nombre réel a une représentation sous forme :

- de fraction continue finie ou infinie
- avec a_0 : entier relatif
- et les a_j : entiers strictement positifs.

Intérêt des fractions

- approximations de nombres réels par des rationnels
- algorithmes pour l approximation de racines carrées
- démonstrations d irrationalité voire de transcendance

D'Euclide aux fractions continues

pgcd(30,13)

$$\begin{aligned} 30 &= 2*13 + 4 \\ 13 &= 3*4 + 1 \\ 1 &= 1*1 + 0 \end{aligned}$$

Fraction continue du rationnel (30,13)

$$\frac{30}{13} = 2 + \frac{4}{13} = 2 + \frac{1}{\frac{13}{4}}$$

$$\frac{13}{4} = 3 + \frac{1}{4} = 3 + \frac{1}{\frac{4}{1}}$$

$$\frac{4}{1} = 4 + \frac{0}{1} = 4$$

Fraction continue : $[2, 3, 4]$

D'Euclide aux fractions continues

Représentation en fraction continue

- rationnel $\frac{30}{13}$
- fraction continue : $[2, 3, 4]$

$$\frac{30}{13} = 2 + \cfrac{1}{3 + \cfrac{1}{4}}$$

Pavage du rectangle (30,13)

Pavage d'Euclide

Module : rationnel.figures

```
def pavageEuclide(longueur,largeur,x=0,y=0,scale=10.0):
    # """..."""" + assert
    setheading(0)
    while largeur != 0 :
        if heading()==0 :
            for i in range(int(longueur/largeur)) :
                up()
                goto(x+scale*largeur*i,y)
                down()
                quadrilatere(x+scale*largeur*i, y,\
                             scale*largeur, scale*largeur)
            x=x+scale*largeur*(i+1)
            setheading(90)
        else :
```

Pavage d'Euclide

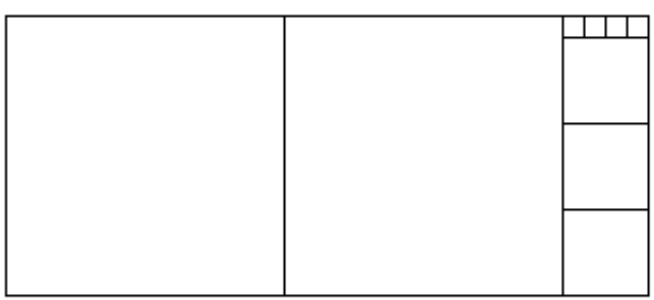
Module : rationnel.figures

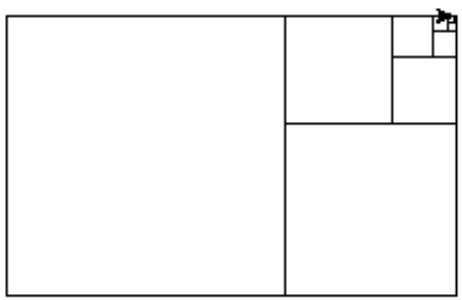
```
            for i in range(int(longueur/largeur)) :
                up()
                goto(x,y+scale*largeur*i)
                down()
                quadrilatere(x, y+scale*largeur*i,
                        scale*largeur, scale*largeur)
            y=y+scale*largeur*(i+1)
            setheading(0)
        up()
        goto(x,y)
        reste= longueur%largeur
        longueur=largeur
        largeur=reste
    return longueur
```

Répertoire de test : euclide.py

Module : rationnel.figures

```
import sys
sys.path.append("../src")
from math import sqrt
from rationnel.figures import pavageEuclide
pgcd = pavageEuclide(30,13,-300)
largeur=13
longueur = largeur*(1 + sqrt(5))/2
pavageEuclide(longueur,largeur,100)
```





De Stern-Brocot aux fractions continues

Représentation matricielle d'un nombre

$$\begin{pmatrix}27\\35\end{pmatrix} = L^1R^3L^2R^1L^1\begin{pmatrix}1\\1\end{pmatrix}$$

$$\varphi = L^1R^1L^1R^1L^1R^1L^1R^1L^1R^1L^1R^1....\begin{pmatrix}1\\1\end{pmatrix}$$

Développement en fraction continue d'un nombre

$$\frac{27}{35} = [0,1,3,2,1,1]$$

$$\varphi = [1,1,1,1,1,1,1...]$$

De Fibonacci aux fractions continues

Nombre d'or : $\varphi = [1, 1, 1, 1, 1, 1, 1...]$

$$
\begin{array}{rcccc}
[1,1] & = 1 + & \frac{1}{1} & = & 2 \\
[1,1,1] & = 1 + & \frac{1}{1+\frac{1}{1}} & = & 3/2 \\
[1,1,1,1] & = 1 + & \frac{1}{1+\frac{1}{1+\frac{1}{1}}} & = & 5/3 \\
[1,1,1,1,1] & = 1 + & \ldots & = & 8/5
\end{array}
$$

Fibonacci

$\mathcal{F}_1 = \mathcal{F}_2 = 1$

$\mathcal{F}_{n+2} = \mathcal{F}_{n+1} + \mathcal{F}_n$

$\mathcal{F}_n : 1, 1, 2, 3, 5, 8, 13, 21, 34...$

$$\frac{\mathcal{F}_3}{\mathcal{F}_2} = 2, \frac{\mathcal{F}_4}{\mathcal{F}_3} = 3, \frac{\mathcal{F}_5}{\mathcal{F}_4} = 5/3, \frac{\mathcal{F}_6}{\mathcal{F}_5} = 8/5 \ldots$$

Installation

Fichier : setup.py

```
from distutils.core import setup
setup (
        name = 'Nombres' ,
        description = 'Nombres rationnels' ,
        version = '1.0' ,
        url = 'iroise.enib.fr/Moodle' ,
        author = 'info1a' ,
        author_email = 'info1a@enib.fr' ,
        packages =[ 'rationnel'
             ] ,
        package_dir ={ 'rationnel': 'src/rationnel',
                       }
      )
{logname@hostname} python setup.py install
```

Distribution

Fichier : MANIFEST

```
README.txt
setup.py
doc/nombres-1.0.pdf
src/rationnel/__init__.py
src/rationnel/figures.py
src/rationnel/geometrie.py
src/rationnel/nombre.py
src/rationnel/utils.py
test/calkinWilf.py
test/constructionOr.py
test/euclide.py
test/...
{logname@hostname} python setup.py sdist
```

Distribution

Archive

```
{logname@hostname} python setup.py sdist
running sdist
reading manifest file 'MANIFEST'
creating Nombres-1.0
creating Nombres-1.0/doc
...
making hard links in Nombres-1.0...
...
creating dist
tar -cf dist/Nombres-1.0.tar Nombres-1.0
gzip -f9 dist/Nombres-1.0.tar
removing 'Nombres-1.0' (and everything under it)
{logname@hostname} ls dist
Nombres-1.0.tar.gz
```

Bibliographie

Ouvrages

- **Tangente** : "Les nombres"
  HS numéro 33 Editions Pole (2008)

Adresses "au Net"

- un peu de tout : `fr.wikipedia.org/wiki`
- maths et Nombre d'Or :
  `pagesperso-orange.fr/therese.eveilleau/pages/truc_mat`
- maths et Stern-Brocot :
  `pagesperso-orange.fr/jean-paul.davalan/arit/stern/index.html`
- Stern-Brocot, Calkin-Wilf ... :
  `www.cut-the-knot.org/blue/Fusc.shtml`