+ QUIZ

Ressourcesinformatiques

Version en ligne
OFFERTE !
pendant 1 an

Python

Introduction au calcul numérique

En téléchargement

scripts

Michel ROUSSELET

eni

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence ENI de l'ouvrage **RIPYTCN** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Avant-propos

Chapitre 1
Nombres, opérations et fonctions dans Python

Chapitre 2
Suites de nombres réels

Chapitre 3
Fonction exponentielle et fonctions logarithmes

Chapitre 4
Dérivation numérique et équations différentielles

Chapitre 5
Résolution approchée des équations

Chapitre 6
Calcul infinitésimal et intégration numérique

Chapitre 7
Nombres complexes

Chapitre 8
Éléments de statistiques

Chapitre 11
Arithmétique et cryptographie

Chapitre 12
Matrices 2x2 et matrices 3x3

Chapitre 13
Géométrie analytique

Annexes

Chapitre 2
Suites de nombres réels

1. Suites et racines carrées

L'idée de répéter un calcul en changeant les nombres utilisés à chaque étape est très ancienne puisqu'on en trouve la trace à Babylone, 1800 ans avant J.-C. En Grèce, on a souvent eu recours aux suites pour calculer des racines carrées.

1.1 La méthode d'Archytas de Tarente

Archytas de Tarente (vers 435 av. J.-C. ; 347 av. J.-C) était un disciple de Pythagore. Ses travaux ont concerné la notion de moyenne. Étant donnés deux nombres a et b, leur moyenne arithmétique m est égale à $\frac{a+b}{2}$, leur moyenne géométrique g est définie par $g^2 = ab$ et leur moyenne harmonique h est définie par $\frac{2}{h}=\frac{1}{a}+\frac{1}{b}$. On démontre que $h<g<m$. Archytas de Tarente a utilisé cette relation pour calculer des racines carrées.

Par exemple, pour trouver une valeur approchée de $\sqrt{3}$, il commence par écrire $3=2\times\frac{3}{2}$ puis, comme le montre le tableau suivant, il déroule ses calculs :

Étape n°	x	y	Moyenne arithmétique de x et de y	Moyenne harmonique de x et de y
1	2	$\frac{3}{2}$	$\frac{7}{4}$	$\frac{12}{7}$
2	$\frac{7}{4}$	$\frac{12}{7}$	$\frac{97}{56}$	$\frac{168}{97}$
3	$\frac{97}{56}$	$\frac{168}{97}$	$\frac{18817}{10864}$	$\frac{32592}{18817}$

Avec les notations actuelles, on peut écrire $\frac{18817}{10864}=1{,}73205081\ldots$ et $\frac{32592}{18817}=1{,}732050805\ldots$. . Le nombre $\sqrt{3}$ est connu avec une erreur qui porte sur la 9^{e} décimale seulement. On peut généraliser la méthode d'Archytas de Tarente à un nombre réel positif A quelconque en utilisant le programme qui suit :

```
# Calcul d'une racine carrée avec la méthode d'Archytas de Tarente
A=eval(input("Valeur de A : "))
x,y=2,A/2
for i in range(1,6):
    m=(x+y)/2
    h=x*y/m
    print("x=",x," et y=",y)
    x,y=m,h
```

On a choisi de faire le calcul en cinq étapes car l'algorithme est très performant.

Voici par exemple le calcul de $\sqrt{2}$:

```
Valeur de A : 2
x= 2  et y= 1.0
x= 1.5  et y= 1.3333333333333333
x= 1.4166666666666665  et y= 1.411764705882353
x= 1.4142156862745097  et y= 1.41421143847487
x= 1.4142135623746899  et y= 1.4142135623715
```

1.2 La méthode de Héron d'Alexandrie

Au I[er] siècle après J.-C., le mathématicien et ingénieur grec Héron d'Alexandrie (75-150) a exposé une méthode[2] très rapide et très simple pour calculer la racine carrée d'un nombre réel A positif. On choisit une valeur approchée quelconque u de $\sqrt{A}$ et on calcule $v = \frac{1}{2}(u + \frac{A}{u})$. Il est facile de voir que $\sqrt{A}$ est compris entre u et v. On recommence le calcul en remplaçant u par v et on continue ainsi jusqu'à atteindre la précision désirée. Ainsi, le tableau qui suit montre les cinq premières étapes du calcul de $\sqrt{5}$.

Étape n°	u	v
1	3.0	2.3333333333333335
2	2.3333333333333335	2.238095238095238
3	2.238095238095238	2.2360688956433634
4	2.2360688956433634	2.236067977499978
5	2.236067977499978	2.23606797749979

On peut généraliser la méthode de Héron d'Alexandrie à un nombre réel positif A quelconque avec ce programme :

```
# Calcul d'une racine carrée avec la méthode de Héron d'Alexandrie
A=eval(input("Valeur du nombre A ? "))
n=eval(input("Valeur de n ? "))
u=A/2
for i in range(1,n+1):
    v=(u+A/u)/2
    print(v)
    u=v
```

Pour calculer $\sqrt{A}$ quand A est positif, on peut choisir un nombre positif quelconque comme première approximation de $\sqrt{A}$, y compris le nombre A lui-même. Calculons par exemple les 8 premières valeurs approchées de $\sqrt{10}$ avec ce programme :

```
from math import*
a=10
n=8
u=a
for i in range(1,n+1):
    v=(u+a/u)/2
    print(v)
    u=v
```

On obtient ces résultats :

```
3.659090909090909
3.196005081874647
3.16245562280389
3.162277665175675
3.162277660168379
3.162277660168379
3.162277660168379
3.162277660168379
```

Dès la cinquième approximation, on a obtenu 15 décimales exactes. On démontre que la suite des approximations de $\sqrt{10}$ est illimitée. Le développement décimal de $\sqrt{10}$ possède donc une infinité de décimales. En effet, $\sqrt{10}$ est un nombre irrationnel qui ne peut pas être représenté exactement par une fraction. C'est aussi un nombre algébrique qui vérifie l'équation $x^2=10$.

1.3 Le calcul d'une racine cubique

On peut généraliser la méthode de Héron et calculer la racine cubique d'un nombre positif a. Si x_{n-1} est une valeur approchée de cette racine, la valeur approchée suivante x_n est donnée par le calcul $x_n = \frac{1}{2}\left(x_{n-1} + \frac{a}{x_{n-1}^2}\right)$. Le programme qui suit calcule des valeurs approchées successives u et v de $\sqrt[3]{a}$. Le calcul est arrêté quand $|v\text{-}u|<10^{-6}$. Comme Python peut calculer $\sqrt[3]{a}$, on pourra comparer la valeur exacte à la valeur calculée.

```
# Calcul d'une racine cubique. Méthode de Héron d'Alexandrie
from math import*
a=eval(input("Valeur du nombre positif a ? "))
u=a/3
v=u/2+a/(2*u*u)
e=abs(u-v)
while e>0.000001:
    u=v/2+a/(2*v*v)
    e=abs(v-u)
    v=u
print("Valeur approchée de la racine cubique de ",a," =",v)
print("Valeur exacte selon Python = ", a**(1/3))
```

Voici le résultat obtenu pour $a=100$:

```
Valeur du nombre positif a ? 100
Valeur approchée de la racine cubique de 100  = 4.641589088997662
Valeur exacte selon Python =  4.641588833612778
```