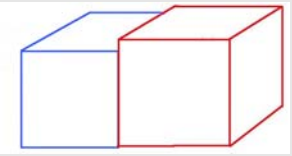
Résolution : les deux cubes

Le problème

Dans la figure ci-contre, les côtés des deux cubes ont une différence de 4 cm, tandis que la somme de leurs volumes est égale à 1 000 cm^2.
Déterminer les dimensions de chacun des cubes.



Complément culturel

Les problèmes qui relèvent de la résolution d'équations sont un champ propice à l'utilisation de l'algorithmique. On parle alors de résolution « numérique » d'une équation, dans le sens où l'on obtient une valeur approchée des solutions (lorsqu'il y en a). Il existe trois méthodes de résolution de ce type de problèmes :

– par balayage ;
– par la méthode de Newton ;
– par dichotomie.

Les programmes

Python

• L'équation s'écrit $x^3 + (x + 4)^3 = 1\,000$. Pour la résoudre, la méthode de balayage consiste à encadrer x par un « pas » qui est à chaque étape divisé par 10.
La boucle est parcourue quinze fois, le pas final est donc égal à $10 / 10^{15} = 10^{-14}$.
On notera la rapidité d'exécution du programme « les2cubes ».

le programme

```
def les2cubes():
    x=0
    Pas=10
    i=0
    while i<15:
            while x**3+(x+4)**3<1000:
                    x=x+Pas
            x=x-Pas
            Pas=Pas/10
            i=i+1
    return x
```

le résultat

```
>>> les2cubes()
5.4337568869100989
```

• La méthode de Newton consiste, elle, à ramener l'équation à la forme $f(x) = 0$, où $f(x) = x^3 + (x + 4)^3 - 1\,000$. On considère alors la suite définie par u_0 fixé et la relation de récurrence $u_{n+1} = u_n - f(u_n) / f'(u_n)$.

Cela conduit dans le cas présent à :

$$\begin{cases} u_0 = 5, \\ u_{n+1} = \left(2u_n^3 + 6u_n^2 + 468\right) / \left(3u_n^2 + 12u_n + 24\right). \end{cases}$$

le programme

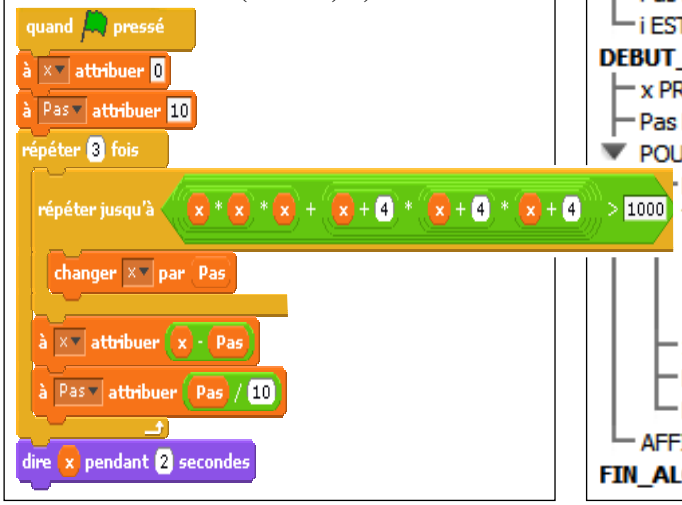
```
def newton(n):
    A=5
    i=0
    while i<n:
            A=(2*A**3+6*A**2+468)/(3*A**2+12*A+24
            i=i+1
    return A
```

le résultat

```
>>> newton(10)
5.4337568869101096
```

Scratch

On note x la longueur du côté du plus petit des deux cubes. On effectue un balayage de la droite réelle, en partant de la valeur $x = 0$ et en prenant un pas de plus en plus petit.
Le test d'arrêt de boucle est que la somme des volumes dépasse strictement 1 000. Ceci oblige, préalablement au changement de pas, à effectuer un pas en arrière (d'où l'instruction « à x attribuer (x – Pas) »).



AlgoBox

Le test est le suivant :
$x^3 + (x + 4)^3 \leqslant 1\,000$?

Il se traduit par la séquence :
pow(x,3) + *pow*(x + 4,3) <= 1 000.
Dans ce programme, la boucle est parcourue dix fois.

```
VARIABLES
    x EST_DU_TYPE NOMBRE
    Pas EST_DU_TYPE NOMBRE
    i EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    x PREND_LA_VALEUR 0
    Pas PREND_LA_VALEUR 10
    POUR i ALLANT_DE 1 A 10
    DEBUT_POUR
    TANT_QUE (pow(x,3)+pow(x+4,3)<=1000) FAIRE
        DEBUT_TANT_QUE
        x PREND_LA_VALEUR x+Pas
        FIN_TANT_QUE
    x PREND_LA_VALEUR x-Pas
    Pas PREND_LA_VALEUR Pas/10
    FIN_POUR
    AFFICHER x
FIN_ALGORITHME
```

Prolongements

Une dernière méthode, la méthode de dichotomie, consiste à partir d'un intervalle [a ; b] contenant la valeur qui annule f et à réduire progressivement cet intervalle :

le programme

```
>>> def dicho(a,b):
        if b-a<10**(-10):
            return a
        c=(a+b)/2
        if f(a)*f(c)>=0:
            return dicho(c,b)
        else :
            return dicho(a,c)
```

– on prend le centre c de l'intervalle ;

– si $f(c) \times f(a) > 0$, on prend $a = c$, et sinon on prend $b = c$;

– la condition d'arrêt de l'algorithme porte sur la longueur de l'intervalle [a ; b]. Le programme est récursif (voir en page 25). On notera là aussi la rapidité d'obtention d'une valeur approchée de la solution.

le résultat

```
>>> dicho(4,6)
5.4337568868650123
```