

Approximation de $\sqrt{2}$ par balayage.

Compétences visées

- **chercher**, expérimenter – en particulier à l'aide d'outils logiciels ;
- **modéliser**, faire une simulation, valider ou invalider un modèle ;
- **représenter**, choisir un cadre (numérique, algébrique, géométrique...), changer de registre ;
- **calculer**, appliquer des techniques et mettre en œuvre des algorithmes.

Ces compétences sont mises en œuvre dans le cadre de l'extrait du programme de 2^{nde} GT ci-dessous :

" Déterminer par balayage un encadrement de $\sqrt{2}$ d'amplitude inférieure ou égale à 10^{-n} ."

Situation déclenchante

Le nombre $\sqrt{2}$ n'est pas un nombre rationnel : il ne peut pas s'écrire sous la forme d'une fraction. Comment déterminer une valeur approchée de ce nombre ? Comme $1 \leq 2 \leq 4$, par croissance de la fonction racine carrée sur son domaine de définition, on peut encadrer $\sqrt{2}$, par les entiers 1 et 2. Comment obtenir en encadrement plus précis ?

Problématique

Ecrire un programme qui permet de déterminer un encadrement d'amplitude 10^{-n} de $\sqrt{2}$. Les bornes de l'intervalle seront arrondies à 10^{-n} .

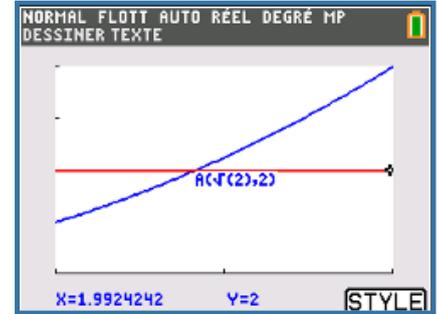
Fiche méthode

Proposition de résolution

On pose la fonction $f(x) = x^2$.

En traçant le tableau de variation de la fonction f sur $[1; 2]$ on "voit" (par le théorème des valeurs intermédiaires) que l'équation $f(x) = 2$ possède une unique solution.

Ainsi, on crée une fonction **racine(n)** qui donne un encadrement de la valeur $\sqrt{2}$ avec n décimales. Le principe de la fonction est d'incrémenter les abscisses de 10^{-n} tant que les images restent inférieures à 2.



Première idée non satisfaisante

On pourrait naïvement proposer le programme ci contre.

L'instruction `from math import *` permet d'importer la bibliothèque `math` et toutes les fonctions associées. Nous en avons ici besoin pour utiliser la fonction carré.

La fonction **racine(pas)** permet de retourner un intervalle contenant $\sqrt{2}$ avec la longueur du pas .

La variable a représente la valeur des abscisses. Tant que l'image de a par la fonction carrée est inférieure à 2 on augmente l'abscisse de la valeur du pas. La boucle `while` s'arrête donc lorsqu'on vient juste de dépasser le point recherché.

```
EDITEUR : RACINE
LIGNE DU SCRIPT 0003
from math import *

def racine(pas):
    a=1
    while a**2<2:
        a=a+pas
    return (a-pas,a)
```

Observons l'exécution du programme pour différentes valeurs de pas.

Lors de l'exécution du programme, avec un pas de 0.1 ou de 0.01 , les résultats restent cohérents. Mais avec un pas de 0.001 ou un pas plus précis, les résultats ne correspondent pas aux attentes. D'où provient ce phénomène ?

```
PYTHON SHELL
>>> racine(0.1)
(1.4, 1.5)
>>> racine(0.01)
(1.41, 1.42)
>>> racine(0.001)
(1.4139999999999954, 1.4149999999999954)
>>> |
```

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus



Fiche méthode

Problèmes liés aux nombres à virgule

Les nombres à virgule flottante sont représentés, au niveau matériel, en fractions de nombres binaires (base 2). Malheureusement, la plupart des fractions décimales ne peuvent pas avoir de représentation exacte en fractions binaires. Par conséquent, en général, les nombres à virgule flottante qui sont saisis **sont seulement approximatés** en fractions binaires pour être stockés dans la machine.

Pour éviter ce type de problème nous aurons recours dans le programme à la commande `round(a,b)` qui arrondi le nombre `a` avec `b` chiffres après la virgule.

```
PYTHON SHELL
>>> round(1.3333,2)
1.33
>>> |
```

```
EDITEUR : RACINE2
LIGNE DU SCRIPT 0008
from math import *

def racine(n):
    a=1
    while a**2 < 2:
        a=a+10**(-n)
    return (round(a-10**(-n),n),round(a,n))
```

```
PYTHON SHELL
>>> racine(3)
(1.414, 1.415)
>>> racine(4)
(1.4142, 1.4143)
>>> |
```

```
EDITEUR : RACINE2
LIGNE DU SCRIPT 0001
from math import *

def racine(n):
    a=1.4
    while a**2 < 2:
        a=a+10**(-n)
    return (round(a-10**(-n),n),round(a,n))
```

Etapes de résolution

L'instruction `from math import *` permet d'importer la bibliothèque `math` et toutes les fonctions associées. Nous en avons ici besoin pour utiliser la fonction carré.

La fonction `racine(n)` permet de retourner un encadrement de $\sqrt{2}$ avec `n` chiffres après la virgule.

La variable `a` représente la valeur des abscisses. Tant que l'image de `a` par la fonction carrée est inférieure à 2 on augmente l'abscisse de la valeur du pas (10^{-n}). La boucle `while` s'arrête donc lorsqu'on vient juste de dépasser le point recherché.

On obtient les résultats ci-contre lors de l'exécution de la fonction .

Pour obtenir un temps de réponse satisfaisant si l'on souhaite une précision plus grande (à partir de `n=6`) il est préférable de changer la valeur de départ dans le programme : on peut par exemple saisir `a=1.4` (résultat obtenu précédemment à l'aide du programme)

Pour profiter de tutoriels vidéos, Flasher le QRCode ou cliquer dessus

