Détermination d'une valeur approchée de la racine carrée d'un nombre

Stéphane Clément

IREM Aix Marseille

1 Algorithme de Héron

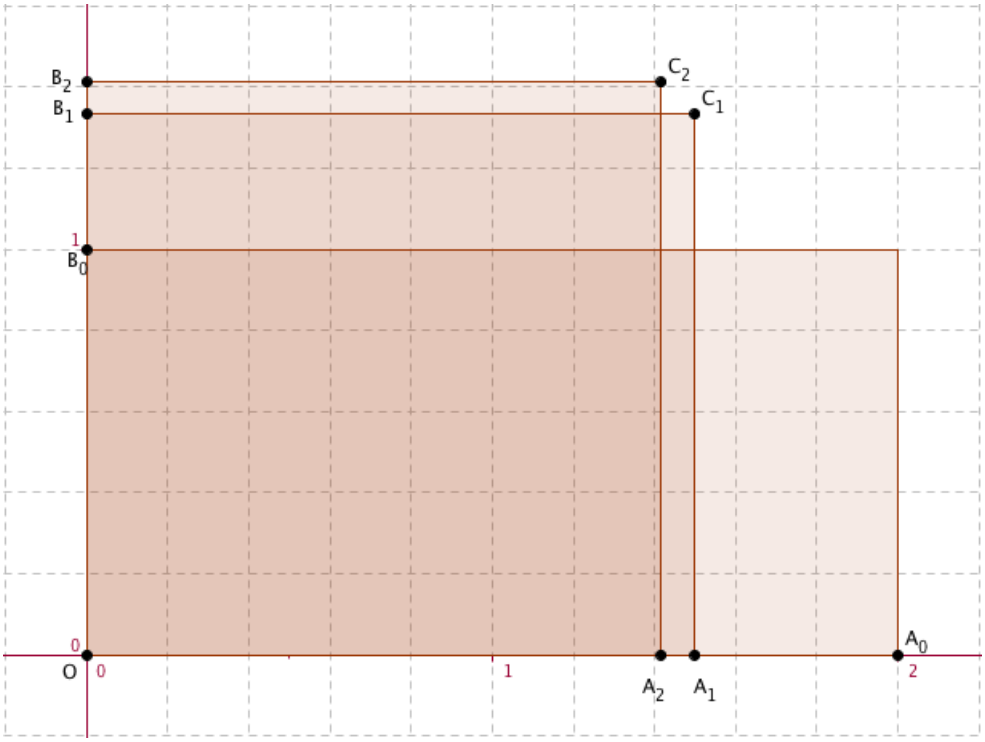
La calcul de valeurs approchées de nombres irrationnels est un type de problème qui peut être une raison d'être à de nombreux contenus mathématiques, de la classe de première S en particulier. Ce qui est présenté ci-dessous a été testé en classe ; la programmation par les élèves a été faite avec l'environnement Scilab.

1.1 Le principe mathématique

Pour les mathématiques actuelles, rechercher la racine carrée d'un nombre A revient à résoudre l'équation $x^2 - A = 0$.
Chez les mathématiciens grecs, extraire la racine carré de A c'est trouver un carré dont l'aire est A. En prenant un rectangle de côté arbitraire a_0 et de même aire, il est nécessaire que la longueur de l'autre côté soit $\frac{A}{a_o}$. Mais ce rectangle n'est pas carré (en général). Pour le rendre "plus carré", il suffit de prendre un rectangle dont la longueur est la moyenne arithmétique des deux côtés précédents soit

$$\frac{a_o + \frac{A}{a_o}}{2}$$

et dont l'aire reste A. En itérant indéfiniment le processus, on transforme petit à petit le rectangle en carré de même aire.
Dans la suite, on va supposer que $A > 1$. Si la racine carré cherchée et inférieure à 1, on pourra toujours se ramener à $A > 1$.
La figure suivante, réalisée avec le logiciel GeoGebra et suivie de son protocole de construction, illustre cette technique pour la détermination d'une valeur approchée de $A = \sqrt{2}$.



No.	Nom	Définition	Valeur
1	Point O	Point d'intersection de axeX et	O = (0, 0)
2	Point A_0	Point sur axeX	A_0 = (2, 0)
3	Point C_0		C_0 = (2, 1)
4	Point A_1	((y(B_0) + x(A_0)) / 2, 0)	A_1 = (1.5, 0)
5	Point B_1	(0, 2 / x(A_1))	B_1 = (0, 1.3333333333)
6	Point C_1	(x(A_1), y(B_1))	C_1 = (1.5, 1.3333333333)
7	Point A_2	((y(B_1) + x(A_1)) / 2, 0)	A_2 = (1.4166666667, 0)
8	Point B_2	(0, 2 / x(A_2))	B_2 = (0, 1.4117647059)
9	Point C_2	(x(A_2), y(B_2))	C_2 = (1.4166666667,

1.2 Exemple à la main : approximations successives de racine de 2

Partons d'un rectangle de côtés de longueurs 1 et 2 et utilisons la technique.
Première itération : prenons la demi-somme pour l'un des côtés $\frac{1+2}{2} = \frac{3}{2}$. Pour que l'aire du rectangle soit 2, nécessairement la longueur du deuxième côté est $\frac{2}{\frac{3}{2}} = \frac{4}{3}$.

Deuxième itération : la demi-somme est $\frac{\frac{3}{2}+\frac{4}{3}}{2} = \frac{17}{12}$ et la mesure du deuxième côté est : $\frac{2}{\frac{17}{12}} = \frac{24}{17}$
Ainsi de suite, on obtient successivement les valeurs rangées dans la tableau suivant :

Rang de l'itération	Longueur du premier côté	Longueur du deuxième côté
0	2	1
1	$\frac{3}{2} \approx 1.5$	$\frac{4}{3} \approx 1.33333$
2	$\frac{17}{12} \approx 1.41667$	$\frac{24}{17} \approx 1.411776$
3	$\frac{577}{408} \approx 1.414215$	$\frac{816}{577} \approx 1.414211$
4	$\frac{665\,857}{470\,832} \approx 1.41421356$	$\frac{941\,664}{665\,857} \approx 1.41421356$

On remarque qu'à la troisième itération, on a un encadrement à 10^{-5} et qu'à la quatrième une très bonne valeur approchée.
Ces calculs se réalisent facilement avec un logiciel de calcul formel, comme ci-dessous avec wxMaxima :

```
(%i1) a:2;
      b:1;
(%o1) 2
(%o2) 1
```

```
(%i3) a:(a+b)/2;
      b:2/a;
(%o3) 3/2
(%o4) 4/3
```

```
(%i5) a:(a+b)/2;
      b:2/a;
(%o5) 17/12
(%o6) 24/17
```

```
(%i7) a:(a+b)/2;
      b:2/a;
(%o7) 577/408
(%o8) 816/577
```

```
(%i9) a:(a+b)/2;
      b:2/a;
(%o9) 665857/470832
(%o10) 941664/665857
```

1.3 L'algorithme

L'algorithme permet de donner un encadrement de la solution et peut s'écrire de deux façons :

1. en calculant simultanément deux suites (technique proche du principe) ;

$a \leftarrow 2$
$b \leftarrow 1$
$e \leftarrow$ précision souhaitée
tant que $a - b \leqslant e$ **faire**
  $a \leftarrow (a + b)/2$
  $b \leftarrow 2/a$
résultat a et b

Algorithme écrit sous le logiciel AlgoBox

CODE DE L'ALGORITHME :

```
1    VARIABLES
2      a EST_DU_TYPE NOMBRE
3      b EST_DU_TYPE NOMBRE
4      e EST_DU_TYPE NOMBRE
5    DEBUT_ALGORITHME
6      AFFICHER "Précision souhaitée"
7      LIRE e
8      a PREND_LA_VALEUR 2
9      b PREND_LA_VALEUR 1
10     TANT_QUE (a-b>e) FAIRE
11       DEBUT_TANT_QUE
12       a PREND_LA_VALEUR (a+b)/2
13       b PREND_LA_VALEUR 2/a
14       FIN_TANT_QUE
15     AFFICHER b
16     AFFICHER "< sqrt(2) <"
17     AFFICHER a
18   FIN_ALGORITHME
```

Généré par AlgoBox

2. en ne calculant qu'une seule suite [1].

$x \leftarrow 0$
$y \leftarrow 1$
$e \leftarrow$ précision **tant que** $|x - y| \leqslant e$ **faire**
  $x \leftarrow y$
  $y \leftarrow \dfrac{x + \frac{2}{x}}{2}$
résultat y

1.4 Programmation

En Python

1. Cf APMEP Bulletin 486 *Étude d'un très vieil algorithme* par Catherine Combelles

```
# -*- coding:Utf-8 -*-
from math import *
a=2.
b=1.
e=0.0000001
while(a-b>e):
        a=(a+b)/2
        b=2 / a
        print(a,b)
```

La programmation en langage Python ci-dessus donne le résultat ci-dessous :

```
>>>
(1.5, 1.3333333333333333)
(1.4166666666666665, 1.411764705882353)
(1.4142156862745097, 1.4142114384748701)
(1.4142135623746899, 1.4142135623715002)
>>>
```

Avec Scilab

```
1 a=2
2 b=1
3 e=0.000000000001
4 while a-b>e
5     a=(a+b)/2;
6     b=2/a;
7     disp([b,a]);
8 end
```

2 Comparaison de rapidité de convergence par rapport à d'autres algorithmes

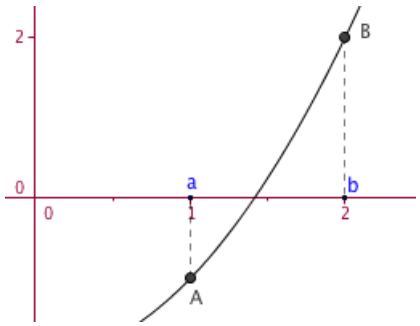
Sensibiliser les élèves à la comparaison de vitesse de convergence

2.1 Comparaison à l'algorithme de dichotomie

L'algorithme de dichotomie est abordé dès la seconde, mais la mise en œuvre avec des élèves peut s'avérer quelques fois difficile à ce niveau. On peut décider de le retravailler en première.

Entrées : a réel, b réel, $a < b$, f fonction continue sur $[a, b]$ telle que $f(a) * f(b) \leqslant 0$, ϵ la précision est un nombre réel arbitrairement petit.
tant que $b - a > \epsilon$ **faire**
    $c \leftarrow \dfrac{a+b}{2}$
    si $f(a) * f(c) \leqslant 0$ **alors** $b \leftarrow c$ **sinon** $a \leftarrow c$
Sortie a est un nombre réel qui approche une racine de f à ϵ près.

Cet algorithme peut être utilisé pour la recherche d'une valeur approchée d'une éventuelle solution à une équation dont les élèves ne connaissent pas de solution algébrique ; mais aussi dans le cadre d'une la recherche d'une valeur approchée d'un nombre (par exemple $\sqrt{2}$).
Il est facile de démontrer que la fonction $x \longmapsto x^2 - 2$ est croissante sur $[1; 2]$ que $f(1) = -1$ et que $f(2) = 2$. Il y a donc bien une solution à $f(x) = 0$ dans l'intervalle $[1; 2]$.



Code de l'algorithme :

```
1     VARIABLES
2       a EST_DU_TYPE NOMBRE
3       b EST_DU_TYPE NOMBRE
4       e EST_DU_TYPE NOMBRE
5       c EST_DU_TYPE NOMBRE
6     DEBUT_ALGORITHME
7       AFFICHER "f est une fonction définie et strictement monotone sur [a;b]. "
8       AFFICHER "On cherche une solution à f(x)=0."
9       AFFICHER "f(a) et f(b) sont de signes différents."
10      AFFICHER "Donner les bornes de l'intervalle [a,b]"
11      LIRE a
12      LIRE b
13      AFFICHER "Précision souhaitée"
14      LIRE e
15      TANT_QUE (b-a>=e) FAIRE
16        DEBUT_TANT_QUE
17        c PREND_LA_VALEUR (a+b)/2
18        SI (F1(a)*F1(c)<=0) ALORS
19          DEBUT_SI
20          b PREND_LA_VALEUR c
21          FIN_SI
22          SINON
23            DEBUT_SINON
24            a PREND_LA_VALEUR c
25            FIN_SINON
26        FIN_TANT_QUE
27      AFFICHER a
28      AFFICHER "<x_0<"
29      AFFICHER b
30    FIN_ALGORITHME
31
32  Fonction numérique utilisée :
33  F1(x)=x*x-2
```

Généré par AlgoBox

```
# -*- coding:Utf-8 -*-
from math import *
def f(x):
        return x*x-2
a=1.
b=2.
e=0.0000001
while(b-a>e):
        c=(a+b)/2
        if f(a)*f(c)>0:
                a=c
        else:
                b=c
        print(a,b)
```

La programmation en langage Python ci-dessus donne le résultat ci-dessous :

```
>>>
(1.0, 1.5)
(1.25, 1.5)
(1.375, 1.5)
(1.375, 1.4375)
(1.40625, 1.4375)
(1.40625, 1.421875)
(1.4140625, 1.421875)
(1.4140625, 1.41796875)
(1.4140625, 1.416015625)
(1.4140625, 1.4150390625)
(1.4140625, 1.41455078125)
(1.4140625, 1.414306640625)
(1.41418457031250, 1.414306640625)
(1.41418457031250, 1.41424560546875)
(1.41418457031250, 1.414215087890625)
(1.41419982910156250, 1.414215087890625)
(1.41420745849609380, 1.414215087890625)
(1.41421127319335940, 1.414215087890625)
(1.41421318054199220, 1.414215087890625)
(1.41421318054199220, 1.41421413421630860)
(1.41421318054199220, 1.41421365737915040)
(1.41421341896057130, 1.41421365737915040)
(1.41421353816986080, 1.41421365737915040)
(1.41421353816986080, 1.41421359777450560)
>>>
```

On remarque que l'algorithme de Héron converge beaucoup plus rapidement que l'algorithme de dichotomie.

2.2 Lien entre la méthode de Newton et l'algorithme de Héron

La méthode de Newton aussi appelé méthode de la tangente, est en fait l'algorithme de Héron [cf IREM Aix marseille : méthode de newton.pdf]. C'est ce que montre la ligne de calculs ci-dessous

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)} \text{ avec } f(x) = x^2 - 2 \text{ et } f'(x) = 2x.$$

$$u_{n+1} = u_n - \frac{u_n^2 - 2}{2u_n} = \frac{2u_n^2 - u_n^2 + 2}{2u_n} = \frac{u_n + \dfrac{2}{u_n}}{2}$$