Le calcul des logarithmes

Présentation et objectifs

Dans le programme (spécialité Terminale)

Contenus	**Capacités attendues**
Fonction logarithme népérien. Propriétés algébriques du logarithme. Fonction dérivée du logarithme, variations.	Utiliser l'équation fonctionnelle de l'exponentielle ou du logarithme … **Exemple d'algorithme** Algorithme de Briggs pour le calcul du logarithme.

L'histoire du calcul des logarithmes

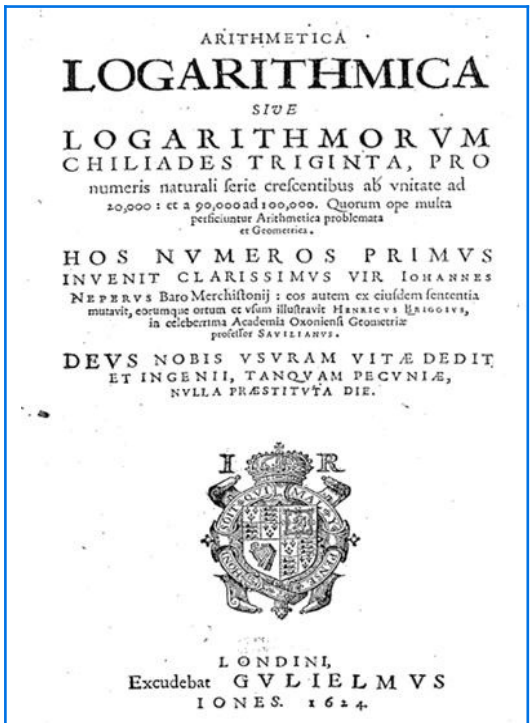
C'est l'Écossais John Napier (ou Neper) (1550-1617) qui inventa les logarithmes, initialement dans le but de faciliter les calculs propres aux activités de banque et de commerce, puis, assez rapidement, pour les très longs calculs trigonométriques nécessaires pour la navigation. Très vite s'est posée la question du calcul effectif (mais approché) en vue de construire des « tables de logarithmes ». Différentes méthodes furent aussitôt proposées, dont celle que nous allons étudier ici.
Henry Briggs (1561-1630) qui fut un mathématicien, astronome et géographe anglais très influent en son époque, s'attela au problème ; il inventa aussi les logarithmes décimaux.

Le principe

Imaginons un texte que Briggs aurait pu écrire, où il expliquerait comment calculer des logarithmes « naturels » (népériens)…

« Pour calculer le logarithme d'un certain nombre positif et non nul, je prends la racine carrée de ce nombre, de manière répétée jusqu'à obtenir un résultat suffisamment proche de 1. À la suite, je soustrais 1 et je double le résultat autant de fois que j'ai fait de racines auparavant. »

L'idée de l'**algorithme de Briggs** est donc la suivante : pour approcher $\ln(x)$ (pour $x>0$), on remplace x par $\sqrt{x}$ (n fois), l'entier n étant « grand » (on y reviendra plus loin). L'approximation à donner est alors $2^n(x-1)$.


ARITHMETICA
LOGARITHMICA
SIVE
LOGARITHMORVM
CHILIADES TRIGINTA, PRO
numeris naturali ſerie creſcentibus ab vnitate ad
20,000 : et a 90,000 ad 100,000. Quorum ope multa
perficiuntur Arithmetica problemata
et Geometrica.
HOS NVMEROS PRIMVS
INVENIT CLARISSIMVS VIR IOHANNES
NEPERVS Baro Merchiſtonij : eos autem ex eiuſdem ſententia
mutavit, eorumque ortum et vſum illuſtravit HENRICVS BRIGGIVS,
in celeberrima Academia Oxonienſi Geometriæ
profeſſor SAVILIANVS.
DEVS NOBIS VSVRAM VITÆ DEDIT
ET INGENII, TANQVAM PECVNIÆ,
NVLLA PRÆSTITVTA DIE.
I R
LONDINI,
Excudebat GVLIELMVS
IONES. 1624

Frontispice de l'ouvrage publié par Briggs (crédit : Jacques Laporte)

Quelques explications

On peut formaliser l'algorithme de Briggs à l'aide d'une suite récurrente définie par $u_0=x$ et $u_{n+1}=\sqrt{u_n}$ pour tout entier naturel n, dont on démontre qu'elle converge vers 1.

On utilise alors l'approximation de la courbe représentative du logarithme par sa tangente au point d'abscisse 1, soit : $\ln(x)\approx x-1$ pour des nombres réels x proches de 1. Dans le cas présent : si u_n

est proche de 1 alors $\ln(u_n)$ est proche de u_n-1 ; sachant que $\ln(u_n)=\frac{1}{2^n}\ln(x)$, on en vient à considérer que $\ln(x)$ est « à peu près égal » à $v_n=2^n(u_n-1)$.

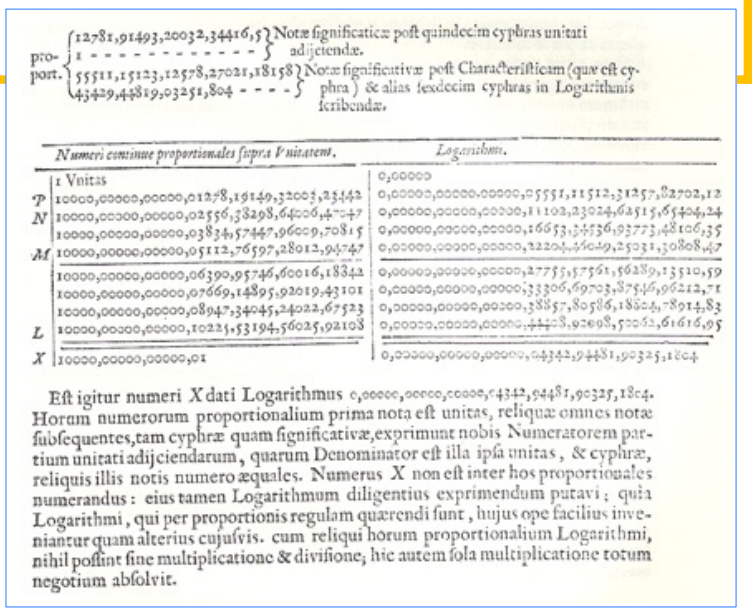
Il reste à choisir n de manière adéquate pour garantir la précision souhaitée. Vient alors une seconde idée : ce qui précède peut aussi bien être conduit avec $\frac{1}{x}$ à la place de x, aboutissant cette fois à une approximation de $\ln\left(\frac{1}{x}\right)=-\ln(x)$.

On définit donc la suite (U_n) par $U_0=\frac{1}{x}$ pour $x>0$ puis $U_{n+1}=\sqrt{U_n}$ pour tout entier naturel n, et enfin $V_n=2^n(1-U_n)$ qui converge aussi vers $\ln(x)$.

Nous admettrons que les suites (v_n) et (V_n) sont monotones et de sens contraires ; elles vont donc fournir des encadrements de $\ln(x)$, d'autant plus resserrés que n sera grand.

Objectifs

1. Écrire une fonction en langage Python donnant les approximations du logarithme de x pour un nombre d'itérations n donné, et comparer ces résultats avec la valeur fournie par la calculatrice, pour diverses valeurs de x : $x=2, x=20, x=404, x=0.05$ et diverses valeurs de n. Qu'observe-t-on quand n dépasse 40 ?
2. Écrire une fonction en langage Python donnant une approximation du logarithme de x pour une précision p donnée, et tester cette fonction pour $x=2, x=20, x=404, x=0.05$ et pour une précision $p=10^{-6}$. Pour ce faire, engager simultanément les algorithmes de Briggs pour le calcul de $\ln(x)$ et de $-\ln\left(\frac{1}{x}\right)$, et stopper dès que l'écart entre les deux approximations est suffisamment faible (par rapport à p).



Numeri continuæ proportionales ſupra Vnitatem. | *Logarithmi.*

Eſt igitur numeri X dati Logarithmus 0,00000,00000,00000,64342,94481,90325,18c4. Horum numerorum proportionalium prima nota eſt unitas, reliquæ omnes notæ ſubſequentes, tam cyphræ quam ſignificativæ, exprimunt nobis Numeratorem partium unitati adjiciendarum, quarum Denominator eſt illa ipſa unitas, & cyphræ, reliquis illis notis numero æquales. Numerus X non eſt inter hos proportionales numerandus : eius tamen Logarithmum diligentius exprimendum putavi ; quia Logarithmi, qui per proportionis regulam quærendi ſunt, hujus ope facilius inveniantur quam alterius cujuſvis. cum reliqui horum proportionalium Logarithmi, nihil poſſint ſine multiplicatione & diviſione, hic autem ſola multiplicatione totum negotium abſolvit.

Ci-contre : une table de calcul de Briggs (crédit : Jacques Laporte)

L. DIDIER & R. CABANE

Fiche méthode

Objectif 1 : codage en Python

Il est très simple du moment que n est donné (voir ci-contre). Les résultats sont acceptables tant qu'on ne fait pas trop dépasser à n la valeur 30.

```
ÉDITEUR : BRIG
LIGNE DU SCRIPT 0001
from math import *

def br(x,n):
  for i in range(n):
    x=sqrt(x)
  return (x-1)*2**n
```

Pour le codage des nombres et des puissances, voir l'annexe 1.

Une anomalie ?

Quand on prend $n=52$, on obtient une réponse aberrante.

Explication : les racines répétées finissent par produire des nombres extrêmement proches de 1, tellement que la machine les confond avec le nombre 1. En effet, les nombres réels « de la machine » (dits « à virgule flottante ») n'ont qu'un nombre limité de décimales (ou plutôt de bits) !

```
PYTHON SHELL
>>> br(2,30)
0.6931471824645996
>>> br(2,40)
0.693115234375
>>> br(2,50)
0.5
>>> br(2,52)
0.0
>>> 1+2**-52
1.0
>>>
Fns… | a A # | Outils | Éditer | Script
```

Dans l'environnement Python, les nombres 1.0 et $1+2^{-52}$ semblent égaux (annexe 1) .

Objectif 2 : codage en Python

L'algorithme va s'inspirer du précédent, en conduisant simultanément les itérations pour l'approximation du logarithme de x (variable y), et celles pour l'opposé du logarithme de $\frac{1}{x}$ (variable z).

Comme ici n n'est pas fixé à l'avance, on continue tant que (boucle `while`) l'écart entre les variables y et z (variable e, valeur absolue de $y-z$) dépasse le double de la précision p donnée (on prend le double parce que la réponse finale se fait avec la moyenne des deux valeurs, divisant l'écart par 2). L'initialisation de l'écart e doit se faire à une valeur suffisamment grande pour que la boucle puisse « démarrer », on a ici choisi le triple[1] de la précision p.

```
ÉDITEUR : BRIGGS
LIGNE DU SCRIPT 0015
def b(x,p):
  n=0
  e=3*p
  while e>2*p:
    x=sqrt(x)
    n=n+1
    u=(x-1)*2**n
    v=(1-1/x)*2**n
    e=abs(u-v)
  return (u+v)/2
```

On renvoie la moyenne des deux valeurs encadrantes u et v, ce qui assure un écart maximum de e/2 par rapport à la valeur cible.

```
PYTHON SHELL
>>> b(20,1e-4)-log(20)
1.043694464186729e-09
>>> b(20,1e-5)-log(20)
6.144018627196601e-11
>>> b(20,1e-6)-log(20)
-1.131827964684362e-10
>>> b(20,1e-5)
2.995732273615431
>>> b(404,1e-5)
6.001414877944626
```

On a choisi 20 et 404, qui sont des nombres entiers donc le logarithme est proche d'un entier !

1 Toute valeur supérieure à $2p$ conviendrait.

Pour aller plus loin

Objectif 3 : calculer des logarithmes ?

Briggs s'est surtout occupé de « calculer » les logarithmes des nombres premiers. Pourquoi ?

Calculer efficacement des logarithmes !

Il convient ici de se souvenir du fait que tout nombre entier est décomposable en un produit de nombres premiers (éventuellement répétés). Considérons par exemple un nombre entier s'écrivant $n=p^{\alpha}q^{\beta}r^{\gamma}$ avec p, q, r premiers et α,β,γ entiers positifs : on sait calculer son logarithme sous la forme $\ln(n)=\alpha\ln(p)+\beta\ln(q)+\gamma\ln(r)$, pour peu qu'on sache « calculer » les logarithmes des nombres premiers. Et à partir de ce point on sait aussi calculer les logarithmes des nombres rationnels (une fraction étant un quotient de deux nombres entiers), en particulier des nombres décimaux avec un dénominateur puissance de 10.

Prolongements

Quelques lectures pour aller un peu plus loin :

- Vies de John Napier et Henry Briggs (pages Wikipedia en français).
- « Histoire de la fonction logarithme », site de l'académie de Limoges
- « Histoires de logarithmes », publié chez Ellipses par la commission Inter-Irem d'Épistémologie et d'Histoire des Mathématiques (https://www.univ-irem.fr/spip.php?article667).