Algorithme de Briggs-Vlacq

Différents algorithmes ont été utilisés pour calculer les logarithmes. Celui qui va être étudié a été proposé par le mathématicien anglais **Henry Briggs** (1561-1630) et le mathématicien hollandais **Adrien Vlacq** (1600-1667)

L'objectif est de ce TP est de comprendre les mécanismes de l'**algorithme de Briggs-Vlacq**, en l'étudiant d'abord sur le papier, puis de le faire « tourner » à l'aide du tableur, et enfin de le programmer en Python.

PARTIE 1

Voici la présentation de cet algorithme par le mathématicien français **Jacques Ozanam** (1640-1718).

PROPOSITION IX.

PROBLEME.

Trouver le Logarithme d'un nombre proposé.

POur trouver le Logarithme d'un nombre donné, comme de 9, qui eſt entre 1 & 10, dont on connoit les Logarithmes 0.0000000, 1.0000000 ou 0.00000000, 1.00000000, en les augmentant chacun d'un zero, pour avoir plus exactement le Logarithme qu'on cherche, à cauſe des Fractions qui reſtent aprés la derniere figure; augmentez auſſi les deux nombres 1, 10, & tous les autres de la Progreſſion geometrique, d'autant de zeros que leurs Logarithmes en contiennent, comme icy de ſept zeros, pour avoir exactement dans le même nombre de figures le Logarithme du nombre proposé 9, qui alors vaudra autant que 9.0000000, comme 1 vaut autant que 1.0000000, que nous appellerons A, & 10 autant que 10.0000000, que nous appellerons B: & faites ainſi.

Cherchez *par Prop.* 6. entre A & B un moyen geometrique proportionnel C qui eſt moindre que le nombre proposé 9.0000000, c'eſt pourquoy pour approcher davantage de ce nombre 9, il faudra chercher entre les deux plus proches B & C, un ſecond moyen proportionnel D, qui étant encore moindre que le nombre proposé 9.0000000, & plus proche que le nombre trouvé C, on cherchera entre ce plus proche C, & le plus grand B, un troiſiéme moyen proportionnel D, qui étant encore moindre que le nombre proposé 9.0000000, on cherchera pareillement entre ce plus proche D, & le plus grand B, un quatriéme moyen proportionnel E, qui eſt

D 4 enco-

DE LA CONSTRUCT. DES TAB. CHAP. IV. 57

encore moindre que le proposé 9.0000000: c'eſt pourquoy on cherchera de nouveau entre ce prochainement moindre E, & le plus grand B, un quatriéme moyen proportionnel F, qui quoique moindre que 9.0000000, en approche plus que le precedent D, c'eſt pourquoy on cherchera entre ce prochainement moindre F, & le plus grand B, un cinquié-

Voir le tableau d'Ozanam en page 4 de ce document

Quelques questions possibles :

1) Qu'e

Justi

Si or

2) Expli

3) Pour

4) Pour

Cette

PARTIE 2

Voici la traduction moderne de cet algorithme à l'aide de suites permettant de donner une valeur approchée du logarithme décimal de 9.

On considère les suites récurrentes suivantes :

$$(u_n)_{n\geq 0}:\begin{cases} u_0=1 \\ u_{n+1}=\sqrt{u_n v_n} \ si \ \sqrt{u_n v_n}\leq 9 \\ u_{n+1}=u_n \ si \ \sqrt{u_n v_n}>9 \end{cases} \qquad (v_n)_{n\geq 0}\begin{cases} v_0=10 \\ v_{n+1}=v_n \ si \ \sqrt{u_n v_n}\leq 9 \\ v_{n+1}=\sqrt{u_n v_n} \ si \ \sqrt{u_n v_n}>9 \end{cases}$$

$$(s_n)_{n\geq 0}:\begin{cases} s_0=0 \\ s_{n+1}=\dfrac{s_n+t_n}{2} \ si \ \sqrt{u_n v_n}\leq 9 \\ s_{n+1}=s_n \ si \ \sqrt{u_n v_n}>9 \end{cases} \qquad (t_n)_{n\geq 0}:\begin{cases} t_0=1 \\ t_{n+1}=t_n \ si \ \sqrt{u_n v_n}\leq 9 \\ t_{n+1}=\dfrac{s_n+t_n}{2} \ si \ \sqrt{u_n v_n}>9 \end{cases}$$

1) Compléter le tableau suivant :

n	un	vn	racine(un * vn)	sn	tn

2) Ce tableau montre que l'antécédent de $\frac{1}{2}$ par la fonction log est environ égal à 3,1622.

Démontrer que cet antécédent α vérifie effectivement $\log(\alpha)=\frac{1}{2}$.

Donner à l'aide du tableau une valeur approchée de l'antécédent β de $\frac{3}{4}$ et démontrer que cet antécédent β vérifie bien $\log(\beta)=\frac{3}{4}$. Que donnent en réalité les colonnes s_n et t_n ?

3) Entourer en bleu les valeurs du tableau correspondantes aux réels A,B,C,D,E et F du tableau d'Ozanam.

4) Démontrer que si $0<a\leq b$ alors $0<a\leq\sqrt{ab}\leq b$

Que peut-on en déduire sur l'ordre des réels u_n , v_n et de leur moyenne géométrique ?

Partie 3

1) Construire la feuille de calcul suivante en la complétant de manière à obtenir la valeur approchée de $\log(9)$ obtenue par Ozanam.

Vous formaterez les cellules de manière à afficher 7 décimales, comme dans le tableau d'Ozanam.

	A	B	C	D	E	F	G
1	Algorithme de BRIGGS-VLACQ						
2							
3	On cherche le logarithme de	9					
4							
5		*n*	*un*	*vn*	*racine(un * vn)*	*log(un)*	*log(vn)*
6		0	1,0000000	10,0000000	3,1622777	0,0000000	1,0000000
7		1	3,1622777	10,0000000	5,6234133	0,5000000	1,0000000

La cellule C7 contient la formule **=SI(E6<=B3;E6;C6)**

Rappel : Syntaxe de la fonction **SI**

= SI (condition ; valeur de la cellule si la condition est vraie ; valeur de la cellule si la condition est fausse)

2) Donner une valeur approchée à 10^{-7} près de $\log(9)$. Comparer le résultat obtenu avec la calculatrice. Le tableur permet-il d'obtenir une meilleure valeur approchée de $\log(9)$? Si oui, laquelle ?

Partie 4

1) Saisir et **compléter** le programme suivant :

Attention

Le programme proposé aux élèves n'est pas le programme ci-contre.
Les questions demandent en effet d'ajouter quelques lignes et d'en compléter certaines autres.

Ci-contre, on donne le programme complété.

```
1  # Créé par jpvan, le 21/10/2019 avec EduPython
2  from math import sqrt
3
4  # approximation du logarithme décimal de x - ALGO de Briggs-Vlacq
5  def briggs(x):
6      n = 0
7      u = 1
8      v = 10
9      log_u = 0
10     log_v = 1
11     racine_u_v = sqrt(u*v)
12     while abs(u-x)>0.00000000001:
13         if (racine_u_v <= x):
14             u = racine_u_v
15             log_u = (log_u+log_v)/2
16             print(u," ; ",log_u)
17             resultat = log_u
18         else :
19             v = racine_u_v
20             log_v = (log_u+log_v)/2
21             print(v," ; ",log_v)
22             resultat = log_v
23         racine_u_v = sqrt(u*v)
24     return 'log de '+str(x)+' est environ égal à '+str(resultat)
```

2) Expliquer la ligne de code

```
while abs(u-x)>0 .000000000001 :
```

3) Ce programme Python permet-il d'obtenir une meilleure valeur approchée de log(9) qu'avec le tableur ? Si oui, laquelle ?

4) Comment compléter ce programme afin d'afficher dans la console les valeurs des lignes C,D,E,F… du tableau d'Ozanam ?

Le tableau d'Ozanam

Nombr. Propor.		Logarithmes.
A	1.0000000	0.00000000
C	3.1612777	0.50000000
B	10.0000000	1.00000000
B	10.0000000	1.00000000
D	5.6234132	0.75000000
C	3.1612777	0.50000000
B	10.0000000	1.00000000
E	7.4989421	0.87500000
D	5.6234132	0.75000000
B	10.0000000	1.00000000
F	8.6596432	0.93750000
E	7.4989421	0.87500000
B	10.0000000	1.00000000
G	9.3057204	0.96875000
F	8.6596432	0.93750000
G	9.3057204	0.96875000
H	8.9768713	0.95312500
F	8.6596432	0.93750000
G	9.3057204	0.96875000
I	9.1398179	0.96093750
H	8.9768713	0.95312500
I	9.1398179	0.96093750
K	9.0579777	0.95703125
H	8.9768913	0.95312500
K	9.0579777	0.95703125
L	9.0173333	0.95507812
H	8.9768713	0.95312500
L	9.0173333	0.95507812
M	8.9970796	0.95410156
H	8.9768713	0.95312500
L	9.0173333	0.95507812
N	9.0072008	0.95458984
M	8.9970796	0.95410156
N	9.0072008	0.95458984
O	9.0021388	0.95434570
M	8.9970796	0.95410156
O	9.0021388	0.95434570
P	8.9996088	0.95422363
M	8.9970796	0.95410156

Nombr. Propor.		Logarithmes.
O	9.0021388	0.95434570
Q	9.0008737	0.95428467
P	8.9996088	0.95422363
Q	9.0008737	0.95428467
R	9.0002412	0.95425415
P	8.9996088	0.95422363
R	9.0002412	0.95428467
S	8.9999250	0.95421889
P	8.9996088	0.95422363
R	9.0002412	0.95428467
T	9.0000831	0.95424652
S	8.9999250	0.95423889
T	9.0000831	0.95424652
V	9.0000041	0.95424171
S	8.9999250	0.95423889
V	9.0000041	0.95424171
X	8.9999650	0.95424080
S	8.9999250	0.95423889
V	9.0000041	0.95424171
Y	8.9999845	0.95424117
X	8.9999650	0.95424080
V	9.0000041	0.95424171
Z	8.9999943	0.95424123
Y	8.9999845	0.95424117
V	9.0000041	0.95424171
&	8.9999992	0.95424147
Z	8.9999943	0.95424123
V	9.0000041	0.95424171
AA	9.0000016	0.95424159
&	8.9999992	0.95424147
AA	9.0000016	0.95424159
BB	9.0000004	0.95424153
&	8.9999992	0.95424147
BB	9.0000004	0.95424153
CC	8.9999998	0.95424150
&	8.9999992	0.95424147
BB	9.0000004	0.95424153
DD	9.0000000	0.95424151
CC	8.9999998	0.95424147

L'algorithme de Briggs

Idée de l'algorithme :

Pour $x > 0$, on veut trouver une valeur approchée de $\ln(x)$.

Pour cela, on crée la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = x$ et pour tout entier naturel $n, u_{n+1} = \sqrt{u_n}$ dont le terme général tend vers 1.

Ainsi, à partir d'un certain rang, u_n est proche de 1 donc $\ln(u_n) \simeq u_n - 1$. Or, $u_n = x^{\frac{1}{2^n}}$ donc $\ln(u_n) = \frac{1}{2^n}\ln(x)$

D'où $\ln(x) \simeq 2^n(u_n - 1)$

Démonstration :

Soit $x > 0$

- Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_0 = x$ et pour tout entier naturel $n, u_{n+1} = \sqrt{u_n}$. Montrons que $(u_n)_{n\in\mathbb{N}}$ est convergente de limite 1.

Par récurrence immédiate, on prouve que pour tout entier naturel n, $u_n = x^{\frac{1}{2^n}} = e^{\frac{\ln(x)}{2^n}}$

On a $\lim\limits_{n\to+\infty} \frac{\ln(x)}{2^n} = 0$ donc $\lim\limits_{n\to+\infty} e^{\frac{\ln(x)}{2^n}} = e^0 = 1$ par composition de limites.

Ainsi :

$$(u_n)_{n\in\mathbb{N}} \text{ est convergente et } \lim_{n\to+\infty} u_n = 1 \text{ (*)}$$

- Montrons que $t - 1$ est une approximation de $\ln(t)$ lorsque t est proche de 1

On sait que la fonction ln est dérivable sur $]0\ ;\ +\infty[$ (de dérivée la fonction inverse sur cet intervalle) donc en particulier en 1.

D'une part, $ln'(1) = \frac{1}{1} = 1$

D'autre part, en étudiant la limite du taux d'accroissement, on a $ln'(1) = \lim\limits_{t\to1} \frac{\ln(t)-\ln(1)}{t-1} = \lim\limits_{t\to1} \frac{\ln(t)}{t-1}$

D'où :

$$\lim_{t\to1} \frac{\ln(t)}{t-1} = 1 \text{ (**)}$$

- D'après (*) et(**), on a

$$\lim_{n\to+\infty} \frac{\ln(u_n)}{u_n - 1} = 1$$

Ainsi, pour n assez grand, $\ln(u_n) \simeq u_n - 1$ et comme $\ln(u_n) = \ln\left(x^{\frac{1}{2^n}}\right) = \frac{1}{2^n}\ln(x)$, on trouve

$$\ln(x) \simeq 2^n(u_n - 1)$$

Programmation sur PYTHON :

On prend par exemple comme condition de la boucle While que le terme u_n se situe dans l'intervalle $[0{,}999999\ ;\ 1{,}000001]$

```
def ln(x):
    assert x>0
    n=0
    u=x
    while not(0.999999<u<1.000001):
        u=u**0.5
        n=n+1
    return(2**n*(u-1))
```

Ce programme donne $\ln(2) \simeq 0{,}693147409$ et la calculatrice donne $\ln(2) \simeq 0{,}693147806$. On a donc une bonne approximation.