Les exercices d'algorithmiques au baccalauréat série S

Renée De Graeve

7 octobre 2012

Site web, page pédagogique et forum de Xcas :
– `www-fourier.ujf-grenoble.fr/~parisse/giac_fr.html`
– `www-fourier.ujf-grenoble.fr/~parisse/irem.html`
– `http://xcas.e.ujf-grenoble.fr/XCAS`
Se reporter en **section 7** pour écrire, compiler et exécuter un programme `Xcas`.

1 Trois exercices classiques

1.1 La série harmonique

On considère la suite $u_n = \sum_{j=1}^{n} \frac{1}{j}$.

1. Montrer que $u_{2n} - u_n$ est une somme de n termes et que chaque terme est supérieur ou égal à $\frac{1}{2n}$. En déduire que pour tout n, on a
$$u_{2n} - u_n \geq \frac{1}{2}$$
2. En déduire que u_n tend vers $+\infty$ quand n tend vers $+\infty$.
3. On écrit u_{16} en faisant apparaitre $u_{2n} - u_n$ ($n = 1, 2, 4, 8$:
$$u_{16} = \sum_{j=1}^{16} \frac{1}{j} = 1 + \frac{1}{2} + (\frac{1}{3} + \frac{1}{4}) + (\frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8}) + \sum_{j=9}^{16} \frac{1}{j}$$
Montrer que $u_{16} > 3$.
4. Écrire un algorithme permettant de trouver la plus petite valeur de n pour laquelle $3 < u_n$.
5. Écrire un algorithme permettant de trouver la plus petite valeur de n pour laquelle $p \leq u_n$ avec $p \in \mathbb{R}$. Tester ce programme pour $p = 3, 5, 10, 11$.

La solution
$u_{2n} - u_n = \frac{1}{n+1} + .. + \frac{1}{n+n}$ et on a : $\frac{1}{n+n} \leq \frac{1}{n+k}$ pour $k = 1..n$.
$u_{2n} - u_n$ a donc n termes et chaque terme est supérieur ou égal à $\frac{1}{2n}$ donc
$u_{2n} - u_n \geq \frac{n}{2n} = \frac{1}{2}$.
Donc $u_{2n} - u_n$ ne tend pas vers zéro quand n tend vers $+\infty$ donc u_n n'est pas convergente. Comme u_n est croissante et non convergente, u_n n'est pas bornée donc u_n tend vers $+\infty$ quand n tend vers $+\infty$

L'algorithme en langage naturel

```
Entree :         p entier
Variables :      j entier,S reel
Initialisation : Affecter a S la valeur 0
                 Affecter a j la valeur 0
Traitement :     Tant que S<p faire
                   Affecter a j la valeur j+1
                   Affecter a S la valeur S+1/j
                 FinTantque
Sortie :         j
```

La solution en langage Xcas

```
harmonique(p):={
  local j, S;
  S:=0;
  j:=0;
  tantque S<p faire
    j:=j+1;
    S:=S+1/j;
  ftantque;
  retoune j;
};;
```

On tape : `harmonique(3)` On obtient : `11`
On tape : `harmonique(5)` On obtient : `83`
On tape : `harmonique(10)` On obtient : `12367`
On tape : `harmonique(11)` On obtient : `33617`

1.2 Le compte bancaire

Lors de la naissance de Pierre son grand-père dépose sur un compte bancaire 100 euros. À chaque anniversaire de Pierre, il dépose sur ce compte 100 euros auquel il ajoute le double de l'age de Pierre.

- Écrire un algorithme permettant de trouver le montant se trouvant sur son compte le lendemain des 10 ans de Pierre
- Modifier l'algorithme précédent pour déterminer à quel age Pierre pourra acheter un objet de 2000 euros ? de P euros ?
- Modifier les algorithmes précédents lorsque le compte où est déposé l'argent rapporte 2.5% l'an (net d'impots, de taxes et de droit de succession !).

La solution en langage naturel
Montant S du compte lorsque Pierre a n ans :

```
Entree :         n entier
Variables :      S reel, j entier
Initialisation : Affecter a S la valeur 100
Traitement :     Pour j allant de 1 a n faire
                     Affecter a S la valeur S+100+2*j
                 FinPour
Sortie :         S
```

Somme disponibleS>=P et age correspondant de Pierre

```
Entree :         P reel
Variables :      S reel, j entier
Initialisation : Affecter a S la valeur 100
                 Affecter a j la valeur 0
Traitement :     Tantque S<P
                   Affecter a j la valeur j+1
                   Affecter a S la valeur S+100+2*j
                 FinTantque
Sortie :         S,j
```

Si le compte rapporte 2.5% par an, on écrira dans le traitement des deux algorithmes précédents : `Affecter à S la valeur S*1.025+100+2*j`
(au lieu de `Affecter à S la valeur S+100+2*j`)
En langage `Xcas` on renverra éventuellement la valeur de `S` arrondie avec seulement 2 chiffres après la virgule (`evalf(S,2)`).
La solution en langage `Xcas`

```
banque(n):={
  // Montant du compte lorsque Pierre a n ans
  local S,j;
  S:=100;
  pour j de 1 jusque n faire
    S:=S+100+2*j;
  fpour;
  retourne S;
};;
banques(P):={
  // Somme disponibleS>=P et age correspondant de Pierre
  local S,j;
  S:=100;
  j:=0;
  tantque S<P faire
    j:=j+1;
    S:=S+100+2*j;
  ftantque;
  retourne S,j;
};;
banquier(n):={
  // On applique un interet de 2.5 pour cent
  // Montant du compte lorsque Pierre a n ans
  local S,j;
  S:=100;
  pour j de 1 jusque n faire
    S:=S*1.025+100+2*j;
  fpour;
  retourne evalf(S,2);
};;
```

```
banquiers(P):={
  // On applique un interet de 2.5 pour cent
  // Somme disponible>=P et age correspondant de Pierre
  local S,j;
  S:=100;
  j:=0;
  tantque S<P faire
    j:=j+1;
    S:=S*1.025+100+2*j;
  ftantque;
  retourne evalf(S,2),j;
};;
```

On tape : `banque(10)` On obtient : `1210`
On tape : `banques(2000)` On obtient : `2106,17`
On tape : `banquier(10)` On obtient : `1367.02`
On tape : `banquiers(2000)` On obtient : `2027.75,14`

1.3 La suite de Syracuse

Soit a un entier positif. On veut étudier la suite de Syracuse définie par :

$$\begin{cases} u_0 &= a \\ u_n &= u_{n-1}/2 \qquad \text{si } u_{n-1} \text{ est pair} \\ u_n &= 3u_{n-1}+1 \quad \text{si } u_{n-1} \text{ est impair.} \end{cases}$$

Cette suite se termine toujours par 1, 4, 2, 1... mais on ne sait pas le démontrer ! ! !

- Écrire un algorithme permettant de trouver la première valeur n de k pour laquelle $u_k = 1$.
- Modifier cet algorithme afin de connaitre en plus la plus grande valeur m prise par u_k lorsque $k = 0..n$.
- Tester ce programme pour $a = 3$, $a = 5$, $a = 7$, $a = 75$ et $a = 97$.
- Tracer dans un repère orthogonal, pour $a = 1...100$, les points de coordonnées (a, n) (en rouge) et les points de coordonnées (a, m) (en noir).

La solution en langage naturel

Algorithme qui renvoie la première valeur de k pour laquelle $u_k = 1$.

```
Entree :         a
Variables :      k
Initialisation : Affecter a k la valeur 0
Traitement :     Tant que a different de 1 faire
                   Si a est pair alors
                     Affecter a a la valeur a/2
                   Sinon
                     Affecter a a la valeur 3a+1
                   FinSi
                   Affecter a k la valeur k+1
                 FinTantque
Sortie :         k
```

Algorithme qui renvoie n est la première valeur de k pour laquelle $u_k = 1$ et m la plus grande valeur prise par u_k lorsque $k = 0..n$.

```
Entree :          a
Variables :       k,m
Initialisation :  Affecter a m la valeur a
                  Affecter a k la valeur 0
Traitement :      Tant que a different de 1 faire
                    Si a est pair alors
                      Affecter a a la valeur a/2
                    Sinon
                      Affecter a a la valeur 3a+1
                      Si a>m alors Affecter a m la valeur a
                    FinSi
                    Affecter a k la valeur k+1
                  FinTantque
Sortie :          k,m
```

La solution en langage `Xcas`

Voici le programme qui renvoie n ($u_n = 1$) et $m = \max_{k=1..n} u_k$ quand $u_0 = a$:

```
syracuse(a):={
  local k,m;
  k:=0;
  m:=a;
  tantque a!=1 faire
    si irem(a,2)==0 alors
      a:=iquo(a,2);
    sinon
      a:=a*3+1;
      si a>m alors m:=a; fsi;
    fsi;
    k:=k+1;
  ftantque;
retourne k,m;
};;
```

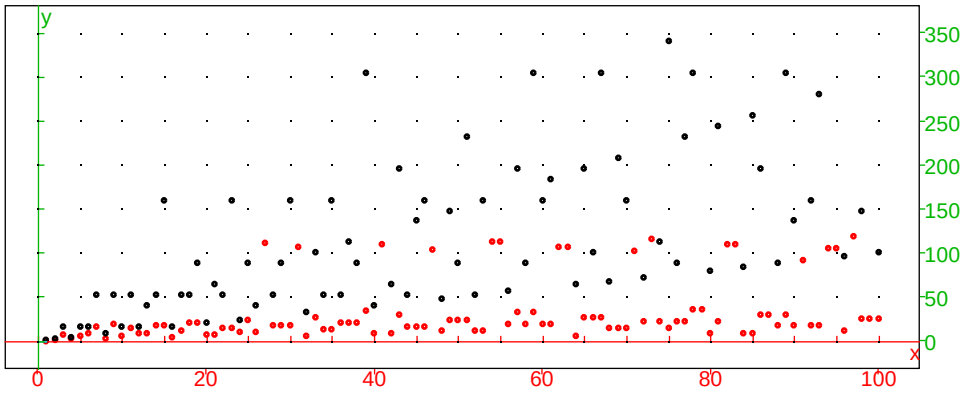
On tape : `syracuse(3)` On obtient : `7, 16`
On tape : `syracuse(5)` On obtient : `5, 16`
On tape : `syracuse(7)` On obtient : `16, 52`
On tape : `syracuse(62)` On obtient : `107, 9232`
On tape : `syracuse(63)` On obtient : `107, 9232`
On tape : `syracuse(75)` On obtient : `14,340`
On tape : `syracuse(97)` On obtient : `118,9232`

Voici le programme qui affiche en rouge les points (a, n) et en noir les points (a, m) lorsque $a = 1..100$.

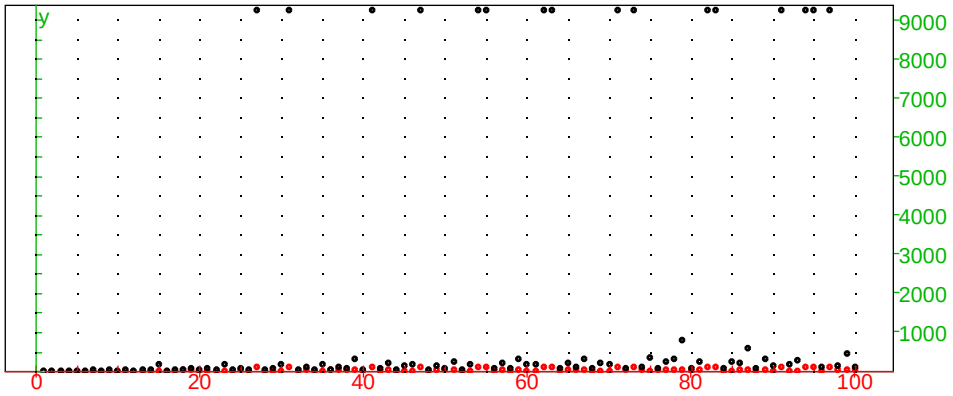
```
syracuse100():={
  local a,n,m,L;
  L:=NULL;
```

```
  pour a de 1 jusque 100 faire
    n,m:=syracuse(a);
    L:=L,point(a,n,affichage=rouge),point(a,m,affichage=noir);
  fpour;
  retourne L;
};;
```

On tape : `syracuse100()` et on obtient :



En changeant le repère, on peut voir des points tels que point(97,9232)



2 En 2009 Centre étrangers

On considère la fonction $f(x) = xe^x - 1$ sur $\mathbb{R}$.

– Calculer $f(0$ et $f(1)$.
En étudiant les variations de f sur $\mathbb{R}$, montrer que $f(x) = 0$ admet une solution unique dans [0 ;1].

– On considère l'algorithme :

```
Entree :          f la fonction precedente
                  n un entier
Variables :       a,b,m,p
Initialisation :  Affecter a a la valeur 0
                  Affecter a b la valeur 1
Traitement :      Tant que b-a>10^-n faire
                    Affecter a m la valeur (a+b)/2
                    Affecter a p la valeur f(a)*f(m)
```

```
                    Si p>0
                      Affecter a a la valeur m
                    Sinon
                      Affecter a b la valeur m
                    FinSi
                  FinTantque
Sortie :          a,b
```

On fait fonctionner cet algorithme avec $n = 1$.

Donner les valeurs que prennent successivement les différents paramètres.

– Que détermine cet algorithme ?

Quelle influence le nombre n a-t-il sur le résultat obtenu ?

La solution

On a $f(0) = -1$ et $f(1) \simeq 1.71828182846$

Sur $]-\infty; 0]$ $f(x) = xe^x - 1 \leq -1 < 0$ donc f ne s'annule pas.

La fonction f est continue et est strictement croissante sur $[0; +\infty[$ car sa derivée $f'(x) = e^x(x + 1)$ est positive sur $[0; +\infty[$.

Donc d'après le théorème des valeurs intermédiaires $f(x) = 0$ a une solution unique comprise entre 0 et 1 puisque $f(0) < 0$ et $f(1) > 0$.

L'algorithme trouve un encadrement de longueur inférieure à 10^{-n} de cette solution : à chaque étape on partage l'intervalle $[a; b]$ en deux (dichotomie). Si m est le milieu de $[a; b]$, on regarde si $f(a)$ et $f(m)$ sont de même signe : si oui, m peut remplacer a et sinon m peut remplacer b et le zéro se trouve toujours entre a et b.

Lorsque $n = 1$ cet encadrement est de longueur 0.1

Initialisation : a=0 et b=1

Etape 1 : m=0.5, p=$f(0)$*$f(0.5)$=0.17563936465, a=0.5, b=1

Etape 2 : m=0.75, p=$f(0.5)$*$f(0.75)$=-0.103232038761, a=0.5, b=0.75

Etape 3 : m=0.625, p=$f(0.5)$*$f(0.625)$=-0.02944659346, a=0.5, b=0.625

Etape 4 : m=0.5625, p=$f(0.5)$*$f(0.5625)$=0.002244979408, a=0.5625, b=0.625

Arrêt du tantque car ($b - a$)<0.1 donc **Résultat :** 0.5625, 0.625

La traduction en langage `Xcas`

Avec `Xcas`, on peut mettre `f` en paramètre. L' algorithme est alors valable pour toutes les fonctions continues `f` qui vérifient si `f(0)*f(1)<0`.

L'algorithme calcule alors une valeur approchée d'une racine de `f`.

```
dichotomie(f,n):={
  local a,b,m,p;
  a:=0.; b:=1.;
  tantque b-a>10^-n faire
    m:=(a+b)/2;
    p:=f(a)*f(m);
    si p>0 alors
      a:=m;
    sinon
      b:=m;
    fsi;
  ftantque;
  retourne a,b;
}:;
```

On tape : `f(x):=x*e^x-1 :; dichotomie(f,1)`
On obtient : `x->x*exp(x)-1; 0.5625,0.6250`
Pour visualiser les étapes intermédiaires indiquées ci-dessus, on tape :
`debug(dichotomie(f,1))`. Puis on clique plusieurs fois sur le bouton `sst`.
On a :
`dichotomie(f,2)` renvoie `0.5625,0.5703125`
`dichotomie(f,5)` renvoie `0.567138671875,0.56714630127`
On a ainsi un encadrement du zéro de `f` plus précis.

3 En 2010 Amérique du sud

- On considère l'algorithme :

```
Entree :          n un entier
Variables :       u,S,j
Initialisation :  Affecter a u la valeur 1
                  Affecter a S la valeur 1
                  Affecter a j la valeur 0
Traitement :      Tant que j<n faire
                    Affecter a u la valeur 2u+1-j
                    Affecter a S la valeur S+u
                    Affecter a j la valeur j+1
                  FinTantque
Sortie :          u,S
```

  Justifier que pour $n = 3$, le résultat de cet algorithme est `11, 21`.
- On considère les suites u_n et S_n définies pour $n \in N$ par :
  $u_0 = 1$ et $u_{n+1} = 2u_n + 1 - n$ et $S_n = u_0 + u_+ ... + u_n$.
  Que représente les valeurs données par cet algorithme ?
- Le but de cette question est de trouver u_n en fonction de n Modifier l'algorithme pour qu'il renvoie aussi $u_n - n$. Montrer que $u_n - n = 2^n$.
- Calculer $1 + 2 + ... + n$ et $1 + 2 + 2^2 + ... + 2^n$.
  En déduire S_n en fonction de n.

La solution
Pour $n = 3$, on a :
Initialisation : u=1, S=1, j=0
Etape 1 : u=3, S=4, j=1
Etape 2 : u=6, S=10, j=2
Etape 3 : u=11, S=21, j=3
Arrêt du tantque car j>=3 donc **Résultat :** 11, 21.
Dans le corps du tantque on calcule u, S et j et on a $u = u_j$ et $S = S_j$.
Lorsque $j = n$ le tantque s'arrête et renvoie u_n, S_n.
Faire attention à l'ordre des instructions : le nouveau u (qui est $u_{j+1} = 2u_j + 1 - j$) se calcule à partir de l'ancien u et de l'ancien j.

La traduction en langage `Xcas`

```
suiteserie(n):= {
  local u,S,j;
```

```
  u:=1;
  S:=1;
  j:=0;
  tantque j<n faire
    u:=2u+1-j;
    S:=S+u;
    j:=j+1;
  ftantque;
  retourne u,S;
};;
```

On veut trouver u_n en fonction de n, on modifie l'algorithme pour qu'il renvoie aussi $u_n - n$, on modifie seulement l'avant-dernière ligne en `retourne u,S-n,S;`

- Pour $n = 0$, on a :
  Initialisation : u=1, S=1, j=0
  Résultat : 1,1,1
- Pour $n = 1$, on a :
  Initialisation : u=1, S=1, j=0
  Etape 1 : u=3, S=4, j=1
  Résultat : 3,2,4
- Pour $n = 2$, on a :
  Initialisation : u=1, S=1, j=0
  Etape 1 : u=3, S=4, j=1
  Etape 2 : u=6, S=10, j=2
  Résultat : 6,4,10
- Pour $n = 3$, on a :
  Initialisation : u=1, S=1, j=0
  Etape 1 : u=3, S=4, j=1
  Etape 2 : u=6, S=10, j=2
  Etape 3 : u=11, S=21, j=3
  Résultat : 11,8,21

Il semble que $u_n - n = 2^n$. Montrons le par récurrence :
pour $n = 0$ $u_0 - 0 = 1 = 2^0$ et si $u_n - n = 2^n$ alors
$u_{n+1} - (n+1) = 2u_n + 1 - n - (n+1) = 2(u_n - n) = 2^{n+1}$.
De plus $1 + 2 + .. + n = \frac{n(n+1)}{2}$ et $1 + 2 + 2^2 + .. + 2^n = 2^{n+1} - 1$, donc

$$S_n = \frac{n(n+1)}{2} + 2^{n+1} - 1$$

On vérifie pour $n = 3$ $u_3 = 2^3 + 3 = 8 + 3 = 11$ et $S_3 = 6 + 2^4 - 1 = 16 + 5 = 21$
On a bien $u_5 = 2^5 + 5 = 32 + 5 = 37$ et $S_5 = 5 * 3 + 64 - 1 = 78$
On tape : `suiteserie(3)` On obtient : `11, 21`
On tape : `suiteserie(5)` On obtient : `37, 78`

Remarque Il me semble préférable d'écrire cet algorithme avec un `pour`. Mais attention aux indices ! ! ! On doit utiliser la relation de récurrence sous la forme :

$$u_n = 2u_{n-1} + 2 - n \text{ ou encore } u_{n+1} = 2u_n + 2 - (n+1)$$

car dans le corps du `pour` on calcule successivement : u_1, S_1 lorsque $j = 1$, u_2, S_2, lorsque $j = 2$,... et u_n, S_n lorsque $j = n$.
Alors que précédement avec `tantque`, on utilise la relation $u_{n+1} = 2u_n + 1 - n$ car on calcule successivement u_1, S_1 lorsque $j = 0$, u_2, S_2 lorsque $j = 1$ et u_n, S_n lorsque $j = n - 1$ et c'est pourquoi la condition d'arrêt du `tantque` est `j<n`.
On tape :

```
suiteserie1(n):= {
  local u,S,j;
  u:=1;
  S:=1;
  pour j de 1 jusque n faire
    u:=2u+2-j;
    S:=S+u;
  fpour;
  retourne u,u-n,S;
};;
```

4 En 2011 La Réunion

On considère la fonction $f(x) = 4e^{x/2} - 5$ sur $\mathbb{R}$.
On note C_f le graphe de f dans un repère orthogonal

- Calculer $f(0)$ et $f(1)$.
  En étudiant les variations de f sur $\mathbb{R}$, montrer que $f(x) = 0$ admet une solution unique dans [0 ;1].
- On considère l'algorithme :

```
Entree :          f la fonction precedente
                  p un reel >0
Variables :       a,b
Initialisation :  Affecter a a la valeur 0
                  Affecter a b la valeur -1
Traitement :      Tant que b<0 faire
                    Affecter a a la valeur a+p
                    Affecter a b la valeur f(a)
                  FinTantque
Sortie :          a-p,a
```

  Que fait cet algorithme ?
  Que renvoie ce programme lorsque $p = 1$, $p = 0.1$, $p = 0.01$, $p = 0.001$?

La solution et traduction en langage `Xcas`
On a $f(0) = -1$ et $f(1) \simeq 1.5948850828$
La fonction f est continue et est strictement croissante sur $\mathbb{R}$ puisque sa derivée qui vaut $f'(x) = 2e^{x/2}$ est positive.
Donc d'après le théorème des valeurs intermédiaires $f(x) = 0$ a une solution unique comprise entre 0 et 1 puisque $f(0) < 0$ et $f(1) > 0$.
L'algorithme trouve un encadrement de longueur p de cette solution.
Lorsque $p = 1$ cet encadrement est $0, 1$
Lorsque $p = 0.1$ cet encadrement est $0.4, 0.5$ car $f(0.4) \simeq -0.114388967359$<0 et $f(0.5) \simeq 0.136101666751$>0

Lorsque $p = 0.01$ cet encadrement est $0.44, 0.45$ car $f(0.44) \simeq -0.0156930776507$<0 et $f(0.45) \simeq 0.00929086476731$>0
Avec `Xcas`, on tape pour définir la fonction f :
`f(x):=4*exp(x/2)-5`
Avec `Xcas`, on peut mettre `f` en paramètre. L' algorithme est alors valable pour toutes les fonctions continues `f` qui vérifient `f(0)<0` `f(1)>0` et `p<=1`.
L'algorithme calcule alors une valeur approchée d'une racine de `f` entre 0 et 1.

```
zeroapprox(f,p):={
  local a,b;
  a:=0;
  b:=f(a);
  tantque b<0 faire
    a:=a+p;
    b:=f(a);
  ftantque;
  retourne a-p,a
};;
```

On tape : `zeroapprox(f,0.1)` On obtient : `0.4,0.5`
On tape : `zeroapprox(f,0.01)` On obtient : `0.44,0.45`
On tape : `zeroapprox(f,0.001)`
On obtient : `0.445999999998,0.446999999998`
On tape : `zeroapprox(f,0.0001)`
On obtient : `0.446199999974,0.446299999974`
Remarque Ce programme est moins performant que le programme de dichotomie vu précédemment.

5 En juin 2012 France

Soit (u_n) la suite définie pour tout entier strictement positif par :

$$u_n = 1 + \frac{1}{2} + \frac{1}{3} ... + \frac{1}{n} - \ln(n)$$

1. On considère l'algorithme suivant :

```
Entree :          l'entier n
Variables :       j est un entier
                  u est un reel
Initialisation :  Affecter a u la valeur 0
Traitement :      Pour j variant de 1 a n
                    Affecter a u la valeur u +1/j
                  fPour
Sortie :          Afficher u
```

   Donner la valeur exacte affichée par cet algorithme lorsque l'utilisateur entre la valeur $n = 3$.

2. Recopier et compléter l'algorithme précédent afin qu'il affiche la valeur de u_n lorsque l'utilisateur entre la valeur de n.

3. Voici les résultats fournis par l'algorithme modifié, arrondis à 10^{-3}.

n	6	7	8	9	10	100	1000	1500	2000
u_n	0.658	0.647	0.638	0.632	0.626	0.582	0.578	0.578	0.577

À l'aide de ce tableau, formuler des conjectures sur le sens de variation de la suite (u_n) et son éventuelle convergence.

La solution et la traduction avec `Xcas`

Le but de l'exercice est de trouver une approximation de la constante d'Euler :
$\gamma = \lim_{n\to+\infty}(1+\frac{1}{2}+\frac{1}{3}...+\frac{1}{n}-\ln(n))$.
On montre dans un premier temps que cette limite existe car

- u_n est décroissante, en effet $u_{n+1}-u_n = \dfrac{1}{n+1}+\ln(\dfrac{n}{n+1})$ et l'étude de $f(x) = \dfrac{1}{x+1}+\ln(\dfrac{x}{x+1})$ montre que f est négative sur $[1;+\infty[$
- de plus u_n est minorée par 0. En effet pour $p \in \mathbb{N}^*$,on a :

$$\int_p^{p+1}\frac{dx}{x} \le \int_p^{p+1}\frac{dx}{p} \;\Rightarrow\; \ln(p+1)-\ln(p) \le \frac{1}{p}$$

En sommant cette inégalité pour $p = 1..n-1$ on a :

$$\ln(n) \le 1+\frac{1}{2}+\frac{1}{3}...+\frac{1}{n-1}$$

donc :

$$0 < \frac{1}{n} <\le 1+\frac{1}{2}+\frac{1}{3}...+\frac{1}{n-1}+\frac{1}{n}-\ln(n)$$

Ainsi u_n est décroissante et minorée par 0. Donc u_n a une limite positive appelée "constante d'Euler".

L'algorithme calcule $1+\frac{1}{2}+\frac{1}{3}...+\frac{1}{n}$
Pour $n = 3$
Initialisation : u=0
Etape 1 : j=1, u=1
Etape 2 : j=2, u=3/2
Etape 3 : j=3, u=11/6
Resultat : 11/6 ou 1.83333333333

On modifie l'algorithme pour qu'il affiche la suite $u_n = 1+\dfrac{1}{2}+...+\dfrac{1}{n}-\ln(n)$, pour cela, il suffit de modifier la sortie :

```
Entree :          l'entier n
Variables :       j et n sont des entiers naturels
                  u est un reel
Initialisation :  Affecter a u la valeur 0
Traitement :      Pour j variant de 1 a n
                    Affecter a u la valeur u +1/j
                  fPour
Sortie :         Afficher u-ln(n)
```

La traduction avec `Xcas`

On retourne une valeur numérique grâce à `evalf(u)` qui transforme le rationnel u en un nombre décimal.

```
csteuler(n):={
  local j, u;
  u:=0;
  pour j de 1 jusque n faire
    u:=u+1/j;
  fpour;
  retourne evalf(u)-ln(n);
};;
```

On tape : `csteuler(10)` On obtient : `0.626383160974`
On tape : `csteuler(100)` On obtient : `0.582207331651`
On tape : `csteuler(1000)` On obtient : `0.577715581568`
On tape : `csteuler(2000)` On obtient : `0.577465644068`
On tape : `csteuler(20000)` On obtient : `0.577240664693`
Cela montre que la constante d'Euler est proche de 0.577240664693.
On tape pour vérifier (`Xcas` connait cette constante) :
`evalf(euler_gamma)`
On obtient : `0.5772156649018`
Remarque Les deux variantes de `csteuler` écrites ci-dessous font à chaque étape un calcul numérique car on a mis `1./j` au lieu de `1/j`. La fonction `csteuler0` calcule la somme des $\frac{1}{k}$ pour k allant de 1 à n, alors que `csteuler1` calcule la somme des $\frac{1}{k}$ pour k allant de n à 1

```
csteuler0(n):={
  local j,u;
  u:=0;
  pour j de 1 jusque n faire
    u:=u+1./j;
  fpour;
  retourne u-ln(n);
} :;
csteuler1(n):={
  local j,u;
  u:=0;
  pour j de n jusque 1 pas -1 faire
    u:=u+1./j;
  fpour;
  retourne u-ln(n);
};;
```

On tape : `csteuler0(2000)` On obtient : `0.577465643831`
On tape : `csteuler1(2000)` On obtient : `0.577465644032`
On tape : `csteuler(2000)` On obtient : `0.577465644068`
Il faut comprendre pourquoi les résultats obtenus avec `csteuler0`, `csteuler1` et `csteuler` sont différents :
`csteuler1` donne un résultat meilleur que `csteuler0` car il commence par ajouter des petits nombres donc la somme conserve plus de décimales.
Le résultat de `csteuler` est encore meilleur car il ne fait l'approximation décimale qu'à la fin du programme.

6 En septembre 2012 France

Étude de la suite (u_n) définie sur $\mathbb{N}$ par :

$$u_0 = 3 \text{et pour tout entier naturel} n,\ u_{n+1} = \frac{1}{2}(u_n + \frac{7}{u_n})$$

1. La convergence de u_n
   On pourra utiliser sans démonstration que pour tout entier naturel n, $u_n > 0$.
   - On désigne par f la fonction définie sur l'intervalle $]0;+\infty[$ par :
     $f(x) = \frac{1}{2}(x + \frac{7}{x})$.
     Démontrer que la fonction f admet un minimum.
     En déduire que pour tout entier naturel n, $u_n \geq \sqrt{7}$
   - Soit n un entier naturel quelconque. Étudier le signe de $u_{n+1} - u_n$.
     Pourquoi peut-on en déduire que la suite (u_n) est convergente ?
     Déterminer l la limite de u_n.
   - Démontrer que pour tout entier naturel n, $u_{n+1} - \sqrt{7} = \frac{(u_n - \sqrt{7})^2}{2u_n}$
   - On définit la suite (d_n) par :
     $d_0 = 1$ et pour tout entier naturel n, $d_{n+1} = \frac{1}{2}d_n^2$.
     Démontrer par récurrence que pour tout entier naturel n, $u_n - \sqrt{7} \leq d_n$

   Avec `Xcas`, on tape :
   `f(x):=1/2*(x+7/x)` qui renvoie `(x)->1/2*(x+7/x)`
   `D:=normal(f'(x))` qui renvoie `(x^2-7)/(2*x^2)`
   `solve(D)` qui renvoie `[-(sqrt(7)),sqrt(7)]`
   `plotfunc(f(x))` qui renvoie `le graphe de f`
   `normal(f(sqrt(7)))` qui renvoie `sqrt(7)`
   Puisque $u_{n+1} = f(u_n)$, on tape :
   `normal(f(x)-x)` qui renvoie `(-x^2+7)/(2*x)`
   Comme $u_n \geq \sqrt{7}$ on en déduit que $u_{n+1} - u_n$ est négatif donc que la suite (u_n) est décroissante et minorée donc convergente vers une limite $l \geq \sqrt{7}$ qui vérifie $f(l) = l$.
   On tape `solve(f(x)-x)` qui renvoie `[-(sqrt(7)),sqrt(7)]`
   Donc $l = \sqrt{7}$.
   `factor(f(x)-sqrt(7))` renvoie `(x-(sqrt(7)))^2/(2*x)`
   $u_n - \sqrt{7} \leq d_n$ car $u_0 - d_0 = 2 < \sqrt{7}$ (puisque 4<7) et si $u_n - \sqrt{7} \leq d_n$ alors $u_{n+1} - \sqrt{7} = \frac{1}{2}\frac{(u_n - \sqrt{7})^2}{u_n} \leq \frac{1}{2}(u_n - \sqrt{7})^2$ car $u_n > 1$ donc $u_{n+1} - \sqrt{7} \leq d_n^2/2 = d_{n+1}$.
   Remarques
   On a aussi si $a_0 = 1$ et $a_{n+1} = a_n^2/5$ pour $n \in \mathbb{N}$, $u_n - \sqrt{7} \leq a_n$. En effet $u_0 < a_0 = d_0$ et si $u_n - \sqrt{7} \leq a_n$ alors puisque $5/2 < \sqrt{7} < u_n$ alors $u_{n+1} - \sqrt{7} \leq \frac{(u_n - \sqrt{7})^2}{5} = a_{n+1}$.
   On a aussi : $0 < u_n - \sqrt{7} = \frac{u_n^2 - 7}{u_n + \sqrt{7}} < \frac{u_n^2 - 7}{5}$.
2. On considère l'algorithme suivant :

   `Entree :  l'entier p`

```
Variables :      n  est un entier naturel
                 d est un reel.
Initialisations : Affecter a d la valeur 1.
                  Affecter a n la valeur 0
Traitement :     Tantque d > 10^(-p) .
                  Affecter a d la valeur 0.5*d^2
                  Affecter a n la valeur n + 1
                 fTantque
Sortie :         Afficher n
```

En entrant la valeur 9, l'algorithme affiche le nombre 5.
Quelle inégalité peut-on en déduire pour d_5 ?
Justifier que u_5 est une valeur approchée de $\sqrt{7}$ à 10^{-9} près.

La Réponse

La variable d va calculer successivement les valeurs de d_n. La boucle s'arrête quand $d_n \leq 10^{-p}$ et le programme affiche le valeur n tel que $d_n \leq 10^{-p}$.
Donc si $p = 9$ et si le programme affiche 5 c'est que $d_5 \leq 10^{-9}$.
Comme $u_n - \sqrt{7} \leq d_n$ on en déduit que $u_5 - \sqrt{7} \leq 10^{-9}$.
Avec `Xcas`, on tape :

```
herond(p):={
local n,d;
n:=0;
d:=1;
tantque d>10^-p faire
  d:=d^2/2;
  n:=n+1;
ftantque;
retourne n;
}:;
```

Mais c'est dommage de ne pas calculer la valeur de u_n. On tape :

```
heron(p):={
local n,d,u;
n:=0;
d:=1;
u:=3;
tantque d>10^-p faire
  d:=d^2/2;
  u:=(u+7/u)/2;
  n:=n+1;
ftantque;
retourne n,u;
}:;
```

On a `n,u:=heron(9):;n,evalf(u)` renvoie `(Done,5,2.64575131106)`
On vérifie et on tape `evalf(sqrt(7))` et on obtient `2.64575131106`
On a remarqué que $0 < u_5 - \sqrt{7} < \dfrac{u_5^2 - 7}{5}$. On calcule $\dfrac{u_5^2 - 7}{5}$, on a :
`evalf(u^2-7)/5)=1.86619176571e-38`
donc on peut dire que $0 < u_5 - \sqrt{7} < 2 * 10^{-38}$

On peut aussi modifier la fonction `heron` en `herona` et utiliser la suite a_n ($a_0 = 1$, $a_{n+1} = a_n^2/5$) qui comme d_n vérifie $u_n - \sqrt{7} \leq a_n$ pour $n \in \mathbb{N}$.
On tape `herona(21)` et on obtient `5,2.645751311064590590502`

7 D'autres algorithmes sur ce modèle

7.1 Calcul de $1 + 2 + .. + n^2$

Soit la suite $u_n = 1 + 4 + .. + n^2$.
Écrire un algorithme qui calcule u_n en fonction de n.
Puis pour $n = 1..10$, calculer u_n et $\dfrac{6u_n}{n(2n+1)}$
Montrer que $u_n = \dfrac{n(n+1)(2n+1)}{6}$

La solution
On écrit l'algorithme :

```
Entree :          l'entier n
Variables :       j est un entier
                  S est un reel
Initialisation :  Affecter a S la valeur 0
Traitement :      Pour j variant de 1 a n
                    Affecter a S la valeur S+j^2
                  fPour
Sortie :         Afficher S
```

Avec `Xcas` :

```
Scarre(n):={
  local j,S;
  S:=0;
  pour j de 1 jusque n faire
    S:=S+j^2;
  fpour;
  retourne S;
}:;
```

Pour avoir la séquence des valeurs de u_n pour $n = 1..10$, on tape :
`seq(Scarre(n),n=1..10)`
On obtient :
`1,5,14,30,55,91,140,204,285,385`
Pour avoir la séquence des valeurs de $\dfrac{6u_n}{n(2n+1)}$ pour $n = 1..10$, on tape :
`seq(6*Scarre(n)/(n*(2n+1)),n=1..10)`
On obtient :
`2,3,4,5,6,7,8,9,10,11`
Pour avoir la séquence des valeurs de $\dfrac{n(n+1)(2n+1)}{6}$ pour $n = 1..10$, on tape :
`seq(n*(n+1)*(2n+1)/6,n=1..10)`
et on obtient `1,5,14,30,55,91,140,204,285,385`
Il reste donc à démontrer par récurrence que :
$u_n = 1 + 2^2 + .. + n^2 = \dfrac{n(n+1)(2n+1)}{6}$

7.2 Calcul de $1+\frac{1}{2^2}+..+\frac{1}{n^2}$

Écrire un algorithme qui calcule $u_n = 1+\frac{1}{2^2}+..+\frac{1}{n^2}$ en fonction de n.
La solution : on écrit l'algorithme :

```
Entree :          l'entier n>=1
Variables :       j un entier, S un reel
Initialisation :  Affecter a S la valeur 0
Traitement :      Pour j variant de 1 a n
                    Affecter a S la valeur S+1/j^2
                  fPour
Sortie :          Afficher S
```

Avec Xcas

```
Sinvcarre(n):={
  local j,S;
  S:=0;
  pour j de 1 jusque n faire
    S:=S+1/j^2;
  fpour
  retourne S;
}:;
```

On tape : `seq(Sinvcarre(p),p=1..9)`
On obtient : `1,5/4,49/36,205/144,5269/3600,5369/3600,`
`266681/176400,1077749/705600,9778141/6350400`
On tape : `evalf(seq(Sinvcarre(p)),p=0..9)`
On obtient : `1.0,1.25,1.36111111111,1.42361111111,1.46361111111,`
`1.49138888889,1.51179705215,1.52742205215,1.53976773117`
On tape (on admet que u_n tend vers $\pi^2/6$ quand n tend vers $+\infty$) :
`sqrt(6.*Sinvcarre(1000))`
On obtient : `3.14063805621`

7.3 Calcul des termes de la suite de Fibonacci

La suite de Fibonacci u_n est définie par :
$u_0 = 1,\ u_1 = 1,\ u_{n+2} = u_{n+1} + u_n$ pour $n \in \mathbb{N}$
Écrire un algorithme qui calcule u_n en fonction de n.
La solution : on écrit l'algorithme :

```
Entree :          l'entier n.
Variables :       j,a,b,c sont des entiers
Initialisation :  Affecter a a la valeur 1
                  Affecter a b la valeur 1
Traitement :      Pour j variant de 2 a n
                    Affecter a c la valeur a+b
                    Affecter a a la valeur b
                    Affecter a b la valeur c
                  fPour
Sortie :          Afficher b
```

Avec Xcas

```
fibo(n):={
  local j,a,b,c;
  a:=1;
  b:=1;
  pour j de 2 jusque n faire
    c:=a+b;
    a:=b;
    b:=c;
  fpour;
  retourne b;
};;
```

On tape : `seq(fibo(p),p=0..10)`
On obtient les 11 premiers termes de la suite de Fibonacci :
`1,1,2,3,5,8,13,21,34,55,89`
On tape : `fibo(101)/fibo(100)`
On obtient :
`927372692193078999176/573147844013817084101`
On tape : `evalf(fibo(101)/fibo(100)),(1+sqrt(5))/2.`
On obtient : `1.61803398875, 1.61803398875`
Il reste à montrer que $v_n = \dfrac{u_{n+1}}{u_n}$ tend vers à $\dfrac{1+\sqrt{5}}{2}$ qui est le nombre d'or.

8 Écrire, compiler et exécuter un programme Xcas

- Ouvrir un niveau éditeur de programme soit en tapant `Alt+p`, soit avec le menu `Prg`►`Nouveau programme`. Une boite de dialogue s'ouvre pour faciliter la définition d'une nouvelle fonction.
- Taper le programme en utilisant les boutons `Fonctions`, `Test` et `Boucles` (assistant de création d'une fonction, d'un test ou d'une boucle) ou le menu `Ajouter` de l'éditeur. Le nom du programme et de ses arguments ne doit pas être un mot-clef ou une commande de `Xcas`, on les repère facilement car ces mots reservés apparaissent en bleu ou brun.
- Cliquer sur `OK` ou appuyer sur `F9`, pour compiler le programme.
- Dans une ligne de commandes, taper le nom du programme suivi entre parenthèses par les valeurs des paramètres séparées par des virgules et appuyer sur `Enter`