Procédures classiques en scilab

I Liste des procédures classiques à savoir refaire.

Ces procédures, les plus simples ou les plus classiques, sont les plus susceptibles de tomber dans une procédure à écrire soi-même aux écrits.

1 Définition d'une fonction.

Il faut connaître, afin de l'utiliser notamment dans les programmes sur les suites récurrentes, la syntaxe d'une définition de fonction : deux exemples ici selon qu'il y ait une ou plusieurs définitions :

$$f(x) = \frac{2x}{\sqrt{1+x^2}} \quad \text{et} \quad g(x) = \begin{cases} \frac{e^x-1}{x} & \text{si } x \neq 0 \\ 1 & \text{sinon} \end{cases}$$

```
function y=f(x)
   y=2*x/(sqrt(1+x^2))
endfunction

function y=g(x)
   if x<>0 then
      y=(exp(x)-1)/x
   else y=1
   end
endfunction
```

On remarquera la présence de la variable informatique muette y dans laquelle on doit affecter le résultat de la fonction.

2 Calcul du terme d'ordre n d'une suite.

Question : Soit (u_n) une suite définie par $u_0 = 1$ et $u_{n+1} = f(u_n)$. Ecrire une procédure qui calcule et affiche le terme d'ordre n de la suite.

Réponse : il faut demander la valeur de n à l'utilisateur et affecter u_0 à u. Ensuite il faut effectuer l'opération $u = f(u)$ pour calculer le terme suivant, autant de fois que nécessaire pour arriver à u_n : il faut donc le faire exactement n fois. On choisit donc une boucle for. Enfin on fait bien attention d'afficher la valeur de u et pas celle de n, puisque c'est u_n qui est demandé et pas n.

```
function y=f(x)
   (On définit la fonction f selon la syntaxe donnée ci-dessus)
endfunction
n=input('Donner la valeur de n : ')
u=1
for i=1:n do
   u=f(u)
end
disp(u)
```

Remarques : évidemment si u_0 vaut autre chose que 1, on change l'affectation. De même si c'est u_1 et pas u_0 qui est connu, on change le contenu du for par for i=2 to n (puisque le premier passage dans la boucle est le calcul de u_2).

3 Calcul du premier terme d'une suite vérifiant une certaine condition, et/ou du rang de ce terme.

Question : Soit (u_n) une suite définie par $u_0 = 1$ et $u_{n+1} = f(u_n)$. Ecrire une procédure qui calcule et affiche le rang du premier terme de la suite tel que $u_n \geq 10^6$.

Réponse : il faut affecter u_0 à u. Ensuite il faut effectuer l'opération $u = f(u)$ pour calculer le terme suivant, autant de fois que nécessaire pour obtenir $u_n \geq 10^6$: comme on ne sait pas combien d'itérations cela prendra, il faut utiliser une boucle while. Attention, on reste dans la boucle TANT QUE ..... donc il faut y mettre la condition contraire de celle qu'on veut obtenir (celle-ci devant correspondre à la sortie de la boucle).

Enfin à la sortie de la boucle, on est sûrs que u_n vérifie la condition cherchée (ici $u_n \geq 10^6$).

Il reste le problème du calcul de n, qui correspond au nombre de passages effectués dans la boucle. On doit savoir le calculer à l'aide d'un compteur : on initialise n à sa valeur initiale (0 si c'est u_0 qui est connu, 1 si c'est u_1, etc...) puis DANS LA BOUCLE, à chaque calcul u=f(u) calculant le terme d'après, on ajoute 1 à n pour qu'il corresponde au rang de ce nouveau terme (n=n+1).
Enfin on fait attention d'afficher n qui correspond au rang demandé (si l'énoncé demande la valeur de u_n, on affiche u à la place.

```
function y=f(x)
   (On définit la fonction f selon la syntaxe donnée ci-dessus)
endfunction
n=0
u=1
while u<10^6 do
   u=f(u)
   n=n+1
end
disp(n)
```

4 Calcul d'une somme.

Question : soit u_n une suite (on va donner deux types de définitions ci-dessous, explicite ou récurrente), calculer la somme $\sum_{k=0}^{n} u_k$.

$$\forall n \in \mathbb{N},\ u_n = \frac{1}{2(n+1)^2} \quad \text{et} \quad v_0 = 1, \forall n \in \mathbb{N},\ v_{n+1} = f(v_n).$$

Réponse : le calcul d'une somme consiste à effectuer une boucle dans laquelle, à chaque passage, on ajoute le terme suivant de la somme. Comme on connaît le nombre de termes à ajouter, on connaît le nombre d'itérations : on utilise donc une boucle for.

Il y a ensuite le problème du calcul des termes de u_k qu'il faudra sommer :

- Si la définition est explicite, on sait écrire le terme à ajouter à chaque passage : il n'y a donc aucune difficulté.
- Si elle est récurrente, le terme d'ordre n n'est pas connu et ne peut être calculé qu'avec une boucle for. Cependant cela ne garde pas en mémoire les différentes valeurs de la suite (elles sont remplacées au fur et à mesure par la suivante) : il est donc IMPERATIF de calculer u_n DANS LA MEME BOUCLE que le calcul de la somme :

```
n=input('Donner la valeur de n : ')
S=0
for i=0:n do
   S=S+1/(2*(i+1)^2)
end
disp(S)
```

```
function y=f(x)
   (On définit la fonction f selon la syntaxe donnée ci-dessus)
endfunction
n=input('Donner la valeur de n : ')
v=1
S=v
for i=1:n do
   v=f(v)
   S=S+u
end
disp(S)
```

On remarquera que dans le deuxième cas, comme le premier terme de la suite est défini en dehors de la boucle, il faut penser à initialiser la somme avec ce premier terme déjà ajouté.

D'autre part on fera très attention à l'ordre des affectations v=f(v) et S=S+v : il faut obligatoirement que le calcul du nouveau terme précède son ajout, donc il doivent être écrits dans ce sens obligatoirement.

Dans le premier cas, on remarquera l'importance de commencer la valeur de i à 0 et de la terminer à n : en effet i représente le k de $\sum_{k=0}^{n}$ et dit donc aller de 0 à n, sans quoi le calcul de $(i+1)^2$ ne correspondra pas à la bonne valeur.

5 Approximation ou encadrement de la valeur d'un nombre défini implicitement : méthode de dichotomie.

De très loin la plus complexe des procédures à connaître, la dichotomie repose sur des encadrements successifs et de plus en plus précis de la valeur d'un α défini implicitement (cad solution d'une équation, mais obtenu mais bijection continue).

Cette méthode repose obligatoirement sue l'obtention mathématique, en préliminaire, de l'existence de α et d'un premier encadrement de α.

Exemple : on suppose qu'on a prouvé l'existence de α unique tel que $f(\alpha) = 0$, et que $1 < \alpha < 2$: (supposons que f soit croissante)

x	0	1	α	2	$+\infty$
$f(x)$	$-\infty$	$\nearrow$ $m<0$	$\nearrow$ 0	$\nearrow$ $M>0$	$\nearrow$ $+\infty$

Question : Donner un encadrement de α avec une précision de 10^{-3}.

Réponse : on va alors affiner l'encadrement par la méthode suivante : on calcule la valeur moyenne entre les extrémité de l'encadrement (ici 1 et 2), et on essaie de savoir si elle est inférieure ou supérieure ou α : pour cela on calcule son image par f et on regarde son signe : si elle est positive, on sait alors que :

$$1 < \alpha < 1/2$$

et si elle est négative, que

$$1/2 < \alpha < 1$$

et on a obtenu un encadrement deux fois plus précis. Ensuite on réitère en prenant la nouvelle valeur moyenne, et on obtient de nouveau un encadrement deux fois plus précis, etc....

En scilab, pour coder cette méthode, il faut d'abord définir la fonction f et prendre deux variables a et b qu'on initialise aux deux bornes de l'encadrement (ici 1 et 2).

Ensuite il faut réitérer la méthode expliquée précédemment autant de fois que nécessaire pour que l'encadrement soit aussi précis que le demande l'énoncé. C'est donc une condition, et on utilise une boucle while : enfin comme il faut continuer TANT QUE $b-a \leq 10^{-3}$ (qu'on veut obtenir) EST FAUX, donc TANT QUE $b-a > 10^{-3}$.

A chaque passage dans la boucle, on réapplique la méthode expliquée : on calcule la moyenne de a et b (il faut donc l'affecter dans une variable c) puis :

Si $f(c)$ est négatif, on sait que $c < \alpha < b$ donc on peut remplacer a par c (a=c) pour obtenir le nouvel encadrement plus précis.

Sinon, c'est que $f(c)$ est positif, donc $a < \alpha < c$ et on remplace donc b par c (b=c).

Enfin on affiche les deux bornes de l'encadrement de α obtenu à la sortie de la boucle while, qui sont obligatoirement écartés de moins de 10^{-3} (sinon on ne serait pas sorti de la boucle).

```
function y=f(x)
   (On définit la fonction f selon la syntaxe donnée ci-dessus)
endfunction
a=1 , b=2
while b-a>10^(-3) do
   c=(a+b)/2
   if f(c)>0 then
      b=c
   else a=c
   end
end
disp(a,b)
```

ATTENTION : Il faut adapter la méthode lorsque la fonction est décroissante : on effectue les mêmes opérations, mais les conclusions selon le signe de $f(x)$ son inversées :

x	0	1	α	2	$+\infty$
$f(x)$	$+\infty$	$\searrow$ $M>0$	$\searrow$ 0	$\searrow$ $m<0$	$\searrow$ $-\infty$

donc si $f(c) > 0$, c'est que $c < \alpha < b$, donc on doit remplacer a par c (a=c) et dans le cas contraire $a < \alpha < c$, et on remplace b par c (b=c).