S2 Analyse : T.P. 1

I. Résolution d'une équation par dichotomie.

Ecrire une procédure *dic* qui prend en entrée une fonction continue f, deux réels a et b tels que $f(a) < 0$ et $f(b) > 0$, un entier n et qui donne une solution approchée par défaut de $f(x) = 0$ dans $[a, b]$ avec une erreur inférieure 10^{-n}, par la méthode de dichotomie. On affichera le résultat sous forme d'un nombre décimal avec n chiffres après la virgule.

Exécuter cette procédure avec différentes précisions n pour la fonction :
$f : \ x \mapsto f(x) = x^7 - 7x^3 + 8x^2 - 3, \quad a = 1, \quad b = 1,5.$

Exécuter cette procédure avec les précisions $n = 8, 9, 10$ pour la fonction :
$f : \ x \mapsto x - \sin(x) - 1/4$ avec $a = 0$, $b = 1.5$. Que remarquez-vous ? Où est le problème ?

Rappel : *la procédure par dichotomie (voir le chapitre 2, page 3) construit une suite d'intervalles $[a_n, b_n]$ encadrant la solution. La valeur a_n est la valeur approchée par défaut de la solution à l'étape n, et la longueur $b_n - a_n$ est une majoration de l'erreur commise.*

Commandes utiles *: proc, while, if, evalf. Voir l'aide dans les pages suivantes.*

II. Une suite récurrente dépendant d'un paramètre.

On considère la suite de Feigenbaum définie pour $n \geq 0$ par

$$u_0 = 0.5, \quad u_{n+1} = a u_n (1 - u_n)$$

où a est un paramètre réel pouvant varier entre 1 et 4.

1. Ecrire une procédure *Fei* qui prend en entrée a et n et qui calcule le terme u_n pour cette valeur a du paramètre.
2. Dessiner les graphes des suites obtenues (on dessinera les 100 premiers termes) pour les valeurs de a suivantes : $a = 1,\ 2.5,\ 3.1,\ 3.5,\ 3.56,\ 3.6,\ 3.9$.
   Commenter les résultats obtenus (monotonie ? convergence ? comportement à l'infini.)
3. Afin de comprendre l'effet du paramètre a sur le comportement à l'infini de la suite, on s'intéresse à la fonction qui à a appartenant à $[1, 4]$ associe un terme u_n "lointain" de la suite (par exemple u_{200}).

   - Tracer le graphe de la fonction : $f200 : \ a \mapsto u_{200}$.

   - Recommencer pour la fonction $f201 : \ a \mapsto u_{201}$. Que remarquez-vous ?

   - Puisque, au-delà de $a = 3$ les comportements de la suite semblent pouvoir sauter entre différents choix, on va superposer sur un même graphe tous les comportements possibles : dessiner sur une même figure les différents graphes des fonctions $fn : \ a \mapsto u_n$ pour n allant de 200 à 205.

Procédure de connexion.

L'IUT ne disposant pas du logiciel Maple, nous travaillons par une connexion avec le C.I.C.T. en suivant la procédure suivante :
1- démarrer le poste sous LINUX ;
2- se connecter *en local*, à l'aide du login et mot de passe *personnels* ;
3- se connecter au C.I.C.T. en ouvrant une fenêtre Terminal et en utilisant la commande :

ssh -X *login*@telline.cict.fr + *mot de passe*

Le *login* et *mot de passe* de cette étape sont distribués en séance par binôme, et devront être conservés pour les séances suivantes :

Le login CICT de mon binôme :
Le mot de passe CICT de mon binôme :

4- Ouvrir le logiciel Maple par la commande : *xmaple&*

Aide-mémoire Maple.

Syntaxe d'une procédure Maple.

<nom> := proc(<arg1>,<arg2>,<arg3>...)
local <var1>,<var2>,<var3>... ;
<instructions> ;
return <resultat> ;
end proc ;

où <nom> désigne le nom de la procédure, <arg1> le ou les arguments d'entrée (sur l'exemple n) auxquels l'utilisateur de la procédure devra donner une valeur particulière (par exemple : n=352), et <var1>... les variables "locales" c'est-à-dire les variables utilisées à l'intérieur du programme (par exemple ci-dessous : m, p,...). Une d'entre elles peut être le <resultat> à afficher (ci-dessous p).

Boucles : if, for, while.

• L'instruction conditionnelle **if** est utilisée pour exécuter une suite d'instructions lorsqu'une condition est vraie, et (éventuellement) une autre lorsqu'elle est fausse. Sa syntaxe est :

if ... then ... else ... end if ;

- La présence de la commande "else" n'est pas obligatoire.
- On peut remplacer "end if" par "fi" (if à l'envers).

• La commande **for** est utilisée pour répéter une suite d'instructions un certain nombre fois. Sa syntaxe est :

for ... from ... to ... by ... do ... end do ;

• La commande **while** permet aussi de créer une boucle d'instructions qui sont exécutées tant qu'une condition est satisfaite et se termine lorsqu'elle ne l'est plus. Sa syntaxe est :

while ... do ... end do ;

Autres commandes.

• evalf(...,n) : evalue numériquement un résultat avec n chiffres après la virgule.

• Construction d'une fonction : f := x -> expression dépendant de x ;

• Graphe d'une fonction au dessus de l'intervalle $[a, b]$: plot(f,a..b) ;
On peut superposer plusieurs graphes en plaçant les fonctions dans une liste.

• Graphe d'une suite : chaque point $[n, u_n]$ est formé d'une liste à deux termes. On construit ensuite une liste L formée de ces points pour n allant de 1 à N (utiliser la commande seq, et encadrer cette séquence par des crochets).
On dessine le graphe de la suite par : plot(L,style=point,view=[1..N,a..b]) ;
où $[a, b]$ est l'intervalle sur l'axe des ordonnées dans lequel on choisit de représenter la suite.