DS4

Correction du devoir Surveillé du 18/11/2019

Exercice 1

1. On peut proposer le script suivant :

```
function u = suite(n)
    u = 0
    for k = 1:n
        u = cos(u)
    end
endfunction
```

2. Il nous faut déjà créer un vecteur `y` contenant tous les termes de la suite (u_n) pour $n = 0, \dots, 20$. Il restera alors à tracer les points de coordonnées (k, u_k) pour $k = 0, \dots, 20$, ce qu'on peut faire à l'aide de la commande `plot2d`. On peut utiliser le script suivant.

```
y = zeros(1,21)
for k=1:20
    y(k) = suite(k)
end
x = 0:20
plot2d(x,y,-1)
```

Le terme -1 permet de ne pas relier les points.

3. On passe à la limite dans l'expression $u_{n+1} = \cos(u_n)$. Comme tout converge et que cos est une fonction continue, on obtient $\ell = \cos(\ell)$. En particulier, ℓ est un point fixe de cos.

> **Déjà Vu ?**
>
> Il s'agit d'une suite récurrente linéaire. Nous avons déjà rencontré de telles suites à de nombreuses reprises (voir par exemple le DS1 ou le TD0). Les méthodes d'études de telles suites sont détaillées dans le **Complément 0. Méthodes d'étude d'une suite récurrente d'ordre 1.**

La limite ℓ est l'abscisse du point d'intersection des courbes de $x \mapsto x$ et de $x \mapsto \cos(x)$ sur l'intervalle $[0, 1]$ (qui est un intervalle stable par cos, et qui contient donc tous les termes de la suite). Par lecture graphique, on obtient donc $\ell \approx 0,74$.

4. On peut utiliser le script suivant :

```
function n = premier_entier(p)
    u = 0
    v = 1
    n = 0
```

```
    while abs(u-v)>10^(-p)
        u = v
        v = cos(v)
        n = n+1
    end
endfunction
```

`u` et `v` contiennent respectivement les termes u_n et u_{n+1}, et on augmente n tant que $|u_n - u_{n+1}| > 10^{-p}$.

5. En utilisant les fonctions définies plus haut, ainsi que l'inégalité :

$$|u_n - \ell| \leq |u_n - u_{n+1}| \leq 10^{-p},$$

le script suivant fait l'affaire :

```
function l = approx(p)
    n = premier_entier(p)
    l = suite(n)
endfunction
```

Exercice 2

1. On écrit donc :

```
P = grand(1,1000,'nor',72,6)
t = grand(1,1000,'nor',1.78,0.04)
```

2. En utilisant les opérations pointées, pour que les opérations se fassent composantes par composantes, on a :

```
I = P./(t.^2)
```

3. En se souvenant des commandes correspondantes :

```
disp(mean(I)), disp(stdev(I))
```

On obtient par exemple l'un des échantillon généré une moyenne de 22.669 et un écart-type de 2.225

4. On peut faire une boucle `for` comme suit :

```
N = 0
for k = 1:1000
    if I(k)>25
        N = N+1
```

```
    end
end
disp(N/1000)
```

ou en utilisant la fonction `find` :

```
disp(length(find(I>25))/1000)
```

On obtient avec notre série statistique par exemple 0.145. Il y a donc 14,5% des personnes en surpoids.

5. Par un tri par modalité, on peut s'attendre à avoir des effectifs de 1 pour chaque modalité. Il n'est donc pas pertinent de procéder ainsi. On utilisera donc par tri par classes.

6. On prend les plus petites m et plus grandes M valeurs de la série I. On divise alors l'intervalle $[m, M]$ en 4 sous-intervalles de même taille afin de constituer nos classes (qui seront donc chacune de même longueur... un autre choix aurait pu être plus pertinent étant donné le diagramme circulaire obtenu). La commande `dsearch` parcourt alors la série I et indique dans `b` l'effectif de chaque classe. Rappelons que `a` est un vecteur qui nous intéresse moins : il contient pour chaque élément de I, sa classe correspondante. La commande `pie` enfin trace le diagramme circulaire correspondant à ce tri par classes, et légende chaque classe par un numéro de 1 à 4.

   Par lecture graphique, on a immédiatement que la classe modale est la numéro 2.