Liste des savoir-faire en Informatique et Algorithmique (1)

En Informatique et Algorithmique en ECS1, je dois savoir :

I Commandes de base

1. effectuer des opérations arithmétiques sur des réels avec `+, -, *, /, **,` $\wedge$ (et sur les complexes en utilisant `%i`) sans oublier l'ordre des opérations (et donc en mettant des parenthèses où il faut).
2. utiliser des approximations d'irrationnels bien connus : `%e` et `%pi`.
3. utiliser les fonctions de référence : `log, exp, floor, abs, sqrt, sin, cos, tan`.
4. calculer la racine $n^{\text{ième}}$ d'un réel `x` positif avec `x`$\wedge$`(1/n)`
5. créer des variables par affectation avec la commande = et les manipuler.
6. utiliser `ans`, `clear`.
7. mettre à jour la valeur stockée dans une variable via des opérations la faisant intervenir (exemple : `x=x+2` ajoute 2 au réel stocké dans `x`).
8. échanger le contenu deux variables.
9. différencier le point, la virgule et le point virgule.
10. commenter une commande avec `//`.
11. manipuler les variables booléennes `%t` (pour `T`, vrai) et `%f` (pour `F`, faux) avec les opérateurs logiques `&,` | ,$\sim$ et les stocker dans des variables.
12. effectuer des tests logiques avec les opérateurs `==, <>, >, <, >=, <=` (du type tester si $x \in [a, b[$ ou $x \in]-\infty, a] \cup]b, +\infty[$, etc.)
13. faire la différence entre = et ==.
14. créer une chaine de caractère.
15. concaténer deux chaines de caractère avec + (et différencier cette opération de l'addition de réels) et utiliser `string`.
16. créer un vecteur ligne (ou liste) en énumérant dans l'ordre ses éléments (pouvant être des nombres, des booléens, des chaines de caractères, etc.) entre crochets et séparés par des virgules et stocker le vecteur dans une variable (exemple : `v=[1,2,3,4,5,6]`).
17. créer un vecteur vide avec la commande `[]`.
18. obtenir la taille (i.e. le nombre d'éléments qu'il contient) d'un vecteur avec la fonction `length`.
19. ajouter `x` au vecteur `v` avec `v=[v,x]` (resp. `v=[x,v]`) pour le placer en dernière (resp. première) position.
20. accéder à la `i`$^{\text{ième}}$ coordonnée du vecteur `v` avec la commande `v(i)`, et modifier ou supprimer sa valeur.
21. créer un vecteur constitué des entiers de `m` à `n` avec la commande `m:n` (et `m:p:n` si on désire les espacer par pas de `p`).
22. créer un vecteur constitué des nombres entre `a` à `b` régulièrement espacés avec la commande `linspace(a,b,n)`.
23. utiliser les commandes `rand(1,n), ones(1,n), zeros(1,n)`.
24. effectuer des opérations arithmétiques (coordonnées par coordonnées) sur des vecteurs de même taille constitués de nombres avec `+, -, .*, ./, .**, .`$\wedge$ ainsi que des tests de comparaison.
25. appliquer une fonction de référence à un vecteur de nombres coordonnées par coordonnées (exemple `cos([%pi,0])`).
26. utiliser l'aide Scilab avec la commande `help`.

II Programmation et fonctions

27. utiliser SciNotes : ouvrir un script, enregistrer un script avec l'extension `.sce` et l'exécuter dans la console.
28. utiliser les commandes `input` et `disp`.
29. créer une structure conditionnelle avec `if/then/else` et éventuellement `elseif` (sans oublier le `end` à la fin).
30. créer une structure répétitive avec une boucle `for` ou une boucle `while` (sans oublier le `end` à la fin).
31. s'aider d'un tableau pour comprendre une boucle `for` ou une boucle `while`.
32. créer une fonction avec la syntaxe `function/endfunction`, l'enregistrer avec l'extension `.sci`, l'exécuter dans la console et l'utiliser (notamment la fonction peut prendre en entrée plusieurs variables et renvoyer plusieurs sorties).
33. rendre compatible les fonctions avec un traitement vectoriel en utilisant les opérations `.*`, `./`, `.^`. Si la fonction contient des structures conditionnelles, alors il va y avoir des bugs en appliquant cette fonction à un vecteur. Il n'est pas exigible de savoir régler ce problème.
34. calculer `n`! avec une boucle `for` :
    - via un script :

```
n=input('Entrer un entier positif :')
y=1;
for  k=1:n
     y=y*k;
end
disp(string(n)+'!='+string(y))
```

    - via une fonction :

```
function y=factorielle(n)
y=1;
for  k=1:n
     y=y*k;
end
endfunction
```

    On peut éventuellement ajouter un message d'erreur au cas où n n'est pas un entier naturel.
35. calculer $\binom{n}{p}$ avec une boucle `for` :
    - via un script :

```
p=input('Entrer un entier p>0 :')
n=input('Entrer un entier n>=p :')
y=1;
for  k=1:p
     y=y*(n-p+k)/k;
end
disp(string(p)+' parmi '+string(n)
          +' est égal à '+string(y)+'.')
```

    - via une fonction :

```
function y=coeffbinom(n,p)
x=1;
for  k=1:p
     y=y*(n-p+k)/k;
end
endfunction
```

III Représentation graphique

36. effacer les fenêtres graphiques en cours avec `clf();`
37. ouvrir une nouvelle fenêtre graphique avec `scf();`
38. relier des points du plan dont les abscisses (resp. les ordonnées) sont stockées dans un vecteur `x` (resp. `y`) avec `plot(x,y,'opt')` ou `plot2d(x,y,options)`. Il n'est pas exigible de connaitre par cœur les options (à part éventuellement les couleurs pour `plot`).
39. tracer une fonction avec la commande `plot(x,f(x),'opt')` ou `plot2d(x,f(x),options)`. Si la fonction n'est pas compatible avec un traitement vectoriel, on peut utiliser également `plot(x,f,'opt')` ou `fplot2d(x,f,options)`.
40. superposer plusieurs courbes avec `plot(x1,y1,'opt1',x2,y2,'opt1',...,xn,yn,'optn')`. Il y a d'autres variantes, notamment si les vecteurs d'abscisses sont communs, mais il n'est pas nécessaire de les connaitre par cœur.
41. tracer un diagramme à barres avec `bar(x,y)`, où `x` est un vecteur contenant les abscisses des catégories et `y` un vecteur contenant les effectifs respectifs de ces catégories.
42. tracer un histogramme avec `histplot(n,x)`, où `x` est un vecteur contenant des données réelles et `n` un entier naturel strictement positif correspondant au nombre de classes (de même amplitude par défaut).

43. ajouter une légende à un graphique (nécessaire lorsqu'on a superposé plusieurs courbes ou graphiques) avec la commande `legend('leg1','leg2',...,'legn')` et un titre avec `title`.
44. tracer plusieurs graphiques sur une même fenêtre avec `subplot`.

IV Calcul des termes d'une suite

45. calculer les termes successifs d'une suite définie par une relation de récurrence. Soit $(u_n)_{n\in\mathbb{N}}$ une suite telle $u_{n+1} = \mathtt{F}(u_n, n)$ pour tout $n \in \mathbb{N}$. Le script suivant calcule le $\mathtt{n}^{\text{ième}}$ terme de la suite de terme initial `x` et le stocke dans `u` :

```
u=x;
for   k=1:n
      u=F(u,k);
end
```

Quelques exemples :

u_{100} lorsque $u_0 = 3$ et $\forall n \in \mathbb{N}, \quad u_{n+1} = n + e^{-u_n}$.

```
u=3;
for   k=1:100
      u=k+exp(-u);
end
```

u_{50} lorsque $u_0 = \pi$ et $\forall n \in \mathbb{N}, \quad u_{n+1} = 2u_n \cos(u_n)$.

```
u=%pi;
for   k=1:50
      u=2*u*cos(u);
end
```

u_{200} lorsque $u_0 = 1$, $u_1 = 2$ et $\forall n \in \mathbb{N}, \quad u_{n+2} = 3u_{n+1} + 5u_n$.

```
u=1; v=0;
for   k=1:199
      w=v;
      v=3*v+5*u;
      u=w;
end
```

Si la suite converge vers $\ell \in \mathbb{R}$ alors, lorsque n est grand, on obtient une approximation de ℓ à ε-près, avec $\varepsilon > 0$ petit (généralement on a déterminé au préalable un entier n afin que $|u_n - \ell| \leqslant \varepsilon$).

46. représenter graphiquement les termes d'une suite (pratique pour conjecturer une éventuelle convergence). Le script suivant représente graphiquement les $\mathtt{n}^{\text{ième}}$ premiers termes de la suite de terme initial `x` :

```
u=x; U=[u];
for   k=1:n
      u=F(u,k);
      U=[U,u]//A chaque itération, on ajoute le terme suivant au vecteur U
end
plot(0:n,U)//On trace les premiers termes
```

47. sommer les termes d'une suite. Soit $(u_n)_{n\in\mathbb{N}}$ une suite telle $u_{n+1} = \mathtt{F}(u_n, n)$ pour tout $n \in \mathbb{N}$. Le script suivant calcule la somme des `n` premiers termes de la suite de terme initial `x` et la stocke dans `s` :

```
u=x;s=u;
for   k=1:n
      u=F(u,k);
      s=s+u;//A chaque itération, on ajoute le terme suivant
            //à la somme des précédents
end
```

Quelques exemples :

$\displaystyle\sum_{k=0}^{100} \frac{1}{k!}$

```
u=1;s=u;
for   k=1:100
      u=u/k;
      s=s+u;
end
```

$\displaystyle\sum_{k=2}^{50} u_k$ lorsque $u_2 = 5$ et $\forall n \geqslant 2 \in \mathbb{N}\backslash\{0,1\}, \quad u_{n+1} = \frac{u_n^2}{n-1}$.

```
u=5;s=u;
for   k=2:50
      u=u^2/(k-1);
      s=s+u;
end
```

$\displaystyle\sum_{k=1}^{200} \frac{1}{k^2}$

```
s=0;
for   k=1:200
      s=s+1/k^2;
end
```

Si la suite $\left(\displaystyle\sum_{k=0}^{n} u_n\right)_{n\in\mathbb{N}}$ converge vers $S \in \mathbb{R}$ alors, lorsque n est grand, on obtient une approximation de S à ε-près.

48. calculer le nombre d'itérations nécessaires pour obtenir une approximation de la limite (la connaissant) d'une suite avec une précision donnée. Si on connait la limite de la suite et qu'on la stocke dans une variable L, alors le script suivant calcule n tel que la $|u_n - L| \leqslant eps$:

```
u=x; n=0;
while  abs(u-L)>eps
       u=F(u,k);
       n=n+1;//Tant que la condition est fausse, on calcule le terme suivant.
end
```

Par exemple, que fait le script suivant ?

```
u=1; s=u; n=1;
while  abs(s-%e)>0.001
       u=u/k;
       s=s+u;
       n=n+1;
end
```

Dans le prochain épisode : Matrices, résolution de systèmes, calcul approché de la racine d'une équation du type $f(x) = 0$, calcul des valeurs approchées d'une intégrale par la méthode des rectangles, probabilité.