Chapitre 12

Exercices sur ordinateur

L'ordinateur obéit à vos ordres,
pas à vos intentions.

Anonyme

L'homme est sujet à l'erreur.
Mais s'il veut vraiment
commettre la gaffe absolue, alors
là, il lui faut un ordinateur.

Dan Rather

12.1 Calcul de π

Dans cet exercice, nous allons calculer la valeur de π. Pour ce faire, nous allons tenir compte du fait que la surface d'un cercle dont le rayon est de longueur C est πC^2, alors que le carré qui le contient, de côté $2C$, a une surface de $4C^2$. Par conséquent, le rapport entre les 2 surfaces est de $\pi/4$. Pour estimer ce rapport, on tire un grand nombre de paires de nombres entre $-C$ et C, qui seront les coordonnées X et Y. Si $X^2 + Y^2 < C^2$, alors le point (X, Y) appartient au disque, sinon, il n'appartient pas. On fait le rapport entre le nombre de paire de nombre qui appartienne au disque et le nombre qui n'y sont pas, et le résultat est multiplié par 4, ce qui donne une estimation de π. Quel résultat obtient-on avec 10, 100, 1000, 10000, 100000 pairs de nombres ?

12.2 Dessiner une fractale

On va dessiner une fractale, tiens, pour changer...
Choisissez une constante complexe c par exemple c=0.21+0.21i.
Prenez z0=x+iy ou x et y sont l'abscisse et l'ordonnee d'un point du plan donné. Appliquez la transformation suivante :

$$z_{n+1} = z_n^2 + c$$

, de nouveau et de nouveau. Vous allez voir une distinction fondamentale : pour certaines valeurs de z, l'application fuit rapidement vers l'infini, et pour d'autres non. Comme MATLAB ne gère pas trop bien l'infini, on va saturer à ± 100.

La méthode est la suivante :

- On se donne un nombre complexe (j'ai pris $c = -0.181 - 0.667$).
- On crée une grille régulière entre -1.3 et 1.3 (par pas de 0.002).

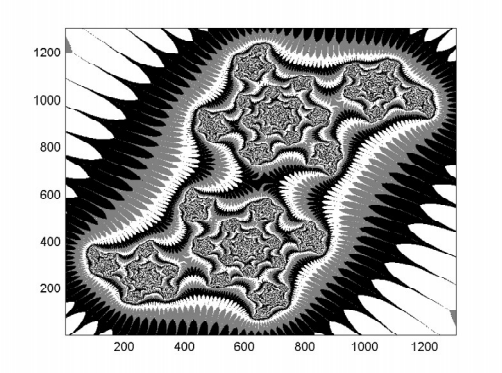

FIG. 12.1 – QU'EST-CE QUE C'EST JOLI, ÇA, ALORS...

- On fait un calcul itératif de la valeur en chaque point, en éliminant les valeurs plus grandes que 100 ou plus petites que -100.
- On dessine le résultat.

J'ai obtenu ce qui est représenté sur la Figure 12.1

12.3 Additionner des fractions

Le problème semble bête comme chou, mais en pratique, ce qu'on s'amuse!!!

Bon, je résume. On suppose deux fractions, données en termes de deux numérateurs et deux dénominateurs. On veut calculer leur somme et la simplifier.

12.3.1 L'initialisation

On va créer une fonction. Elle aura quatre paramètres d'entrées (num1, num2, den1, den2). Elle sort deux nombres, num et den.

12.3.2 Le calcul simple

On sait que

$$\begin{aligned} \frac{\text{num}}{\text{den}} &= \frac{\text{num}_1\text{den}_2 + \text{num}_2\text{den}_1}{\text{den}_1\text{den}_2} \\ \text{num} &= \text{num}_1\text{den}_2 + \text{num}_2\text{den}_1 \\ \text{den} &= \text{den}_1\text{den}_2 \end{aligned}$$

Calculons déjà cela. Maintenant, il y a lieu de simplifier, et c'est là où l'on rit.

12.3.3 La simplification

Pour ce faire, il faut calculer le PGCD (plus grand commun diviseur). Commençons par nous doter d'une fonction qui décompose le nombre en facteurs premiers (voir fin du chapitre 6, on ne va pas réinventer la roue à chaque fois...). Par cette fonction, on crée les tableaux NUM et DEN des facteurs premiers décomposant les numérateurs et dénominateurs.

Cela fait, il faut simplifier. Là, je vous laisse réfléchir à la méthode. Si vous le faites pédestrement, vous n'êtes pas arrivés. En revanche, il existe une solution simple et assez élégante.

12.3.4 Les cas particuliers

On est presqu'au bout. Il faut encore traiter les cas particuliers :

1. si le numérateur vaut zéro,
2. si le dénominateur vaut zéro,
3. si les deux sont nuls,
4. si l'un ou les deux sont négatifs.

12.4 Le jeu de pendu

Le fichier liste_francais.txt contient une liste de plus de 20 000 mots de français. Tirer un mot au hasard, et faite le deviner lettre par lettre, avec un maximum de 10 erreurs. Les étapes suivantes peuvent être programmées indépendamment et testées de même.

12.4.1 Fonction 1 : choisir le mot

- En entrée, rien
- Tirer un nombre au sort entre 1 et 22732 (nombre de mots dans la liste).
- La fonction renvoie, dans une variable caractère, le mot choisi.

12.4.2 Fonction 2 : la lettre a-t-elle déjà été proposée

- En entrée : les lettres restantes (dans l'ordre alphabétique) et la lettre proposée.
- Parcourir les lettres pour voir si la lettre proposée est restante.
- Si oui, on met un flag à 1 et on la retire de la liste, sinon, on laisse le flag à zéro.
- La fonction renvoie la nouvelle liste de lettre restante et le flag.

12.4.3 Fonction 3 : la lettre est-elle dans le mot

- En entrée, la lettre, le mot, un tableau de qui vaut zéro là pour les lettres non trouvées, et un pour les lettres trouvées.
- On passe en revue les lettres une à une. Lorsque la lettre apparait, on met 1 dans le tableau des lettres trouvées.
- On utilise un flag pour savoir si la lettre apparaissait au moins une fois.
- La fonction renvoie le flag et le tableau de zéros et uns mis à jour.

12.4.4 Le programme principale

- On lance la fonction 1. On stocke le résultat dans la variable mot.
- On crée un tableau avec tout l'alphabet, un autre, de zéros, de la même taille que le mot.
- On met à zéro le nombre d'essai déjà effectés.
- On crée une variable chaine de caractères qui contient autant de '_' qu'il n'y a de lettre dans le mot.
- On demande une lettre à l'utilisateur. On la teste avec la fonction 2.
- Si elle n'a pas encore été proposée, on lance la fonction 3.
- Si le flag indique que la lettre n'existe pas dans le mot, on ajoute 1 à la variable qui compte les essais non fructueux.
- Si le flag indique qu'elle y est présente, on remplace les '_' du mot à la bonne place par la bonne lettre.
- Et on revient à la demande de lettre jusqu'à ce que soit on ai fait plus de 10 essais, soit que tout le mot soit trouvé.
- Si il y a plus de 10 essais, on indique que c'est perdu et on donne la solution. Si le mot est trouvé, on dit que c'est gagné, et on confirme la solution.

12.5 Le jeu de la vie

Cet algoritme simule l'évolution d'une colonie bactérienne. Le problème est simple. On prend une grille, genre 100 sur 100. On l'initialise à zéro : toutes les bactéries sont mortes. On tire au sort un certain nombre de points de la grille, et on les mets à 1. Après celà, on observe l'évolution avec les lois suivantes : si le nombre de bactéries vivantes voisines d'une bactérie est compris entre K1 et K2, la bactérie sera vivante au tour suivant. Sinon, elle sera morte.

12.5.1 L'initialisation

On définit un tableau de 100 sur 100, le nombre N de points que l'on va "allumer", le nombre nt de pas de temps, et les constantes K1 et K2 (ça marche bien avec K2=2). On tire au sort (fonction rand) les N points à allumer, et on change leur valeur pour 1.

12.5.2 L'évolution

On doit faire une boucle pour calculer un pas de temps après l'autre. Comme il y a des instructions graphiques, celle-là, je vous la donne :

```
% ouvre un fichier game_of_live.avi, qui contiendra le film.
aviobj=avifile('game_of_live.avi','fps',25,'Compression','Indeo5');
% Ici, c'est un peu tordu, je veux que les points de valeurs 1 soient noirs et les points de valeurs
colormap gray
map=colormap;
mm=flipud(map);
colormap(mm)
% ouf, c'est fini
for i=1:nt
% La magie est dans la fonction 'tour' qui calcule la valeur de la grille au pas suivant.
    grid=tour(grid,K1,K2);
% ici, on dessine le résultat
    pcolor(grid); shading flat
% et ici, on le met dans le film
    frame=getframe(gcf);
    aviobj=addframe(aviobj,frame);
    clf
% voila, la boucle est bouclée
end
% Et on ferme le fichier du film
aviobj=close(aviobj)
```

Bon, là, on a déjà bien bossé... mais le plus dur reste à faire...

12.5.3 La fonction tour

Que doit faire cette fonction ?
(1) Calculer le nombre de voisins,
(2) En fonction de ce nombre, calculer pour chaque case de la grille ce qu'il en advient au tour suivant.

Pour calculer le nombre de voisin, il y a un problème au bord. Pour ne pas m'embéter, j'ai pris une géométrie torique, c'est-à-dire que la ligne au-dessus de la première ligne est la dernière ligne, que la ligne après la dernière ligne est la première ligne (ce qui nous fait un cylindre), et que la colonne qui suit la dernière colonne est la première et inversément (ce qui achève le tore).

Le choix le plus judicieux, ici, est de créer un halo, c'est-à-dire de rajouter une colonne devant et une colonne derrière, une ligne au-dessus et une ligne en dessous, où l'on recopie la ligne/colonne adéquate.

Comme cela, tant que l'on ne traite que les case de la grille, elles ont toutes tous leurs voisins, donc pas besoin de faire de teste.

Bon, il ne reste plus qu'à faire la somme de la valeur des cases voisines, et de voir si cette somme est entre K1 et K2. Pour ces points-là, on met un, pour les autres, on met zéro. Et c'est fini. Ici, la programmation vectorielle est infinement plus rapide et élégante que la programmation par boucle. Donc...

12.6 Compteur des points au tennis

Bon... En voila un exercie bête et méchant... Le problème est simple, on rentrera en temps réel qui a gagné chaque échange, et l'ordinateur nous donne les points (y compris les jeux et les sets), nous dit s'il y a tie-break et tout. L'unité élémentaire, c'est le jeu, nous commencerons là.

12.6.1 Les règles

La plupart du temps, il est nécessaire de remporter deux sets afin de gagner la partie. La seule exception est celle des matches du tableau masculin lors de rencontres dans des tournois du Grand Chelem, ou en Coupe Davis. Pour gagner une manche, il faut être le premier à marquer six jeux avec au moins deux jeux d'écart, dans le cas contraire la manche se poursuit. Les scores possibles pour remporter un set sont ainsi : 6/0, 6/1, 6/2, 6/3, 6/4 et 7/5 (si les deux joueurs n'ont pu se départager au bout de dix jeux). Si les deux joueurs n'ont pas été en mesure de se départager au cours des douze premiers jeux (donc à égalité à 6/6), ils disputent un jeu décisif, qui vaut un jeu, et permet donc de remporter la manche 7/6. En revanche, dans les tournois du grand chelem, exception faite de l'US Open, chez les hommes comme chez les femmes, il n'y a pas de tie-break dans la manche décisive (la cinquième chez les hommes, la troisième chez les femmes), et le match n'est remporté que lorsque l'on parvient à avoir deux jeux d'avance sur l'adversaire ; par exemple 8/6, 9/7, 10/8, etc.

L'invention du jeu décisif date de 1970, soit deux ans après le début de l'ère open. La finalité de ce jeu était d'empêcher des matches interminables, car il arrivait à l'époque que des sets soient gagnés sur le score de 29/27 par exemple. Le principe du jeu décisif est assez simple. Les joueurs servent à tour de rôle. Celui qui débute ne sert qu'une fois de droite à gauche, puis son adversaire sert deux fois de suite, de gauche à droite, puis de droite à gauche, et ainsi de suite. Le gagnant de la manche est le premier joueur à atteindre sept points avec au moins deux points d'écart (Ex : 7/2, 7/5, 9/7...) La manche est alors gagnée sur le score de 7-6.

Une manche se remporte donc en marquant un certain nombre de jeux. Afin de remporter un jeu, il est nécessaire de marquer au moins quatre points, soit sur son service lorsque l'on sert, soit sur le service adverse lorsque l'on reçoit. Il est donc possible, soit pour le serveur, soit pour le receveur de remporter un jeu, même si théoriquement, le serveur est avantagé par rapport au receveur. Si les deux adversaires marquent trois points, on a une situation d'égalité, expliquée ci-après. Lors d'un jeu, voici la manière dont les points sont décomptés : zéro, quinze pour un point marqué, trente pour deux points marqués, quarante pour trois points marqués. Lorsque les deux joueurs ont marqué trois points, (donc à 40/40), il y a égalité. Celui qui marque le point suivant obtient un avantage. Pour marquer le jeu, un joueur qui a l'avantage doit marquer un autre point. Si c'est le joueur qui n'a pas l'avantage qui marque le point suivant, on revient à égalité, et ainsi de suite jusqu'à ce que l'un des deux joueurs remporte le jeu.

Au niveau de l'arbitrage, on donne toujours le score du serveur en premier. Par exemple, si le serveur marque trois points contre deux à son adversaire, le score est 40/30. Dans le cas contraire, le score est 30/40. Il en est de même au niveau des avantages, lorsqu'il y a égalité dans un jeu. Lorsque c'est le serveur qui a l'avantage, l'arbitre annoncera avantage service, et si c'est le receveur qui a l'avantage, on aura avantage dehors.

Je propose de faire cet algorithme par boucles while imbriquées, l'une pour le gain de la manche, l'autre pour le gain du match.

12.6.2 L'initialisation

Il faut demander à l'utilisateur en combien de sets gagnant est la partie, et si on applique la règle des tie-breaks tout le temps, jamais ou tout le temps sauf en cas de manche décisive.

12.6.3 Le jeu

D'abbord, il faut compter avec la façon bizarre de compter les points (15-30-40) et puis avec les avantages et égalité. On ne s'occupera pas du service, et on aura un joueur 1 et un joueur 2, en donnant toujours les points du joueur 1 d'abord.

12.6.4 La manche

Il faut compter les jeux, et voir s'il y a lieu de faire une manche décisive. Dans ce cas, la gérer.

12.6.5 Le match

Bon, ben là, il suffit de voir si l'une des deux parties à remporter le bon nombre de manches.