Initiation à Matlab

Disponible en ligne depuis

https://niess.github.io/matlab-instru/

Réponses aux exercices de la 1ère session

V. Niess

lundi 21/11/2016

Exercice 1 : les nombres heureux

- Les nombres `1`, `7`, `10`, `13`, `19`, `23`, `28` et `31` sont heureux.

- Un nombre malheureux génère une séquence périodique de nombres, par conséquent également malheureux. Par exemple en partant du nombre 2 on obtient la séquence :

  ```
  2, 4, 16, 37, 58, 89, 145, 42, 20, 4, ...
  ```

  La séquence de nombres ainsi obtenue se répète donc à l'infini. On en déduit également que les nombres `4`, `16`, `37`, `58`, `89`, `145`, `42` et `20` sont malheureux.

Exercice 2 : le nombre d'or

- Vous pouvez utiliser le code suivant, par exemple :

```
>> phi = 1

>> phi = 1 + 1 / phi % itération 1

...

>> phi = 1 + 1 / phi % itération n

>> (1 + sqrt(5)) / 2
```

- Les touches ↑ et ↓ du clavier permettent de naviguer dans l'historique des commandes. Cela vous permet, notamment, de rappeler la dernière instruction sans avoir à la retaper.

- L'initialisation de `phi` à la valeur `1` est arbitraire. Vous pouvez utiliser n'importe qu'elle autre valeur. La convergence de la série étant d'autant plus rapide que vous partez d'une valeur proche de la valeur vraie de φ.

- Pour `phi` initialisé à `1`, on obtient le 3$^{\text{ème}}$ chiffre significatif de φ en 9 itérations.

Exercice 2b: retour sur le nombre d'or

- La séquence de commandes suivantes construit un tableau `phi` de taille croissante, contenant les différents termes de la série, selon :

```
>> phi = 1

>> phi = [phi, 1 + 1 / phi(end)] % itération 1

...

>> phi = [phi, 1 + 1 / phi(end)] % itération n

>> epsilon = phi / ((1 + sqrt(5)) / 2) - 1

>> 100 *epsilon(7)

>> 100 *epsilon(8)
```

- On atteint une précision relative de `1/1000` en 7 itérations. On notera que le 1er élément du tableau `phi` est une initialisation et n'est pas compté comme une itération.

Exercice 3: calcul de π

- A partir du vecteur d'indices `k` on construit un vecteur contenant les 5 premiers termes des séries. Les fonction `sum` et `prod` permettent de sommer ou multiplier ces termes. Soit par exemple :

```
>> k = [0, 1, 2, 3, 4]

>> Leibnitz = 4 * sum((- 1) .^ k ./ (2 * k + 1))

>> Madhava = sqrt(12) * sum((-1 / 3) .^ k ./ (2 * k + 1))

>> k = k+ 1

>> Wallis = 2 * prod((2 * k) .^ 2 ./ ((2 * k) .^ 2 - 1))

>> 100 * (Leibnitz / pi - 1)

>> 100 * (Madhava / pi - 1)

>> 100 * (Wallis / pi - 1)
```

- La série de Madhava a la convergence la plus rapide. En 5 itérations on approche π avec une précision relative de `3 / 10,000`. Pour la série de Leibniz et le produit de Wallis les erreurs relative sont de `6%` et `4%`.

Exercice 1b : retour sur les nombres heureux

- Le résultat de `'0123456789' - '0'` est un tableau contenant les nombres réels de 0 à 9 :

```
>> '0123456789' - '0'

ans =
  0 1 2 3 4 5 6 7 8 9
```

- Pour calculer la somme des carrés des chiffres d'un nombre on commence par le convertir en chaîne de caractères, avec la fonction `num2str`. Puis on soustrait le caractère `'0'` pour obtenir un tableau de réels contenant les chiffres. Il suffit ensuite d'utiliser l'opérateur `.^` et la fonction `sum` vue précédemment. Ainsi :

```
>> x = 14071789 % Initialisation

>> x = sum((num2str(x) - '0') .^ 2) % Iteration
```

- On obtient la séquence suivante : `261`, `41` et `17`. Or `17` est malheureux d'après les résultats de l'exercice 1. Par conséquent `14071789` est aussi malheureux.

Exercice 3b: calcul de π par Monte-Carlo #1

- Pour générer des couples de valeurs `(x, y)` dans `[0,1]x[0,1]` l'on utilise la fonction`rand`. On définit ensuite une variable logique qui teste si un point est inscrit ou non dans le cercle de rayon unité. Soit, par exemple :

```
>> N = 10^4

>> x = 1 - rand(1 , N);

>> y = 1 - rand(1, N);

>> r = x .^ 2 + y .^ 2;

>> dedans = (r <= 1);

>> m = length(find(dedans))

>> pi_MC = 4 * m/N

>> sigma_MC = 2 * sqrt(m*(N - m)/(N ^ 3))
```

Notez l'utilisation du point virgule pour éviter d'afficher les grands tableaux à l'écran, ce qui prend beaucoup de temps pour `N = 10^5` ou plus.

Exercice 3b: calcul de π par Monte-Carlo #2

- On obtient les résultats suivants :
    - `N = 10^4` : `pi = 3.16 ± 0.008`
    - `N = 10^5` : `pi = 3.15 ± 0.003`
    - `N = 10^6` : `pi = 3.143 ± 0.0008`

  Soit une précision relative de `2/10,000` pour `N = 10^6`, comparable à ce que l'on obtient avec la série de Madhava mais en seulement 5 itérations pour cette dernière! Aussi, dans ce cas de figure la méthode Monte-Carlo est bien moins performante que l'approximation par une série.