# Simulations de chaînes de Markov en langage C.

**Exercice 1.** Nous souhaitons simuler la marche aléatoire qui se déplace de $+1$ avec probabilité $p$ et de $-1$ avec probabilité $1-p$. Pour cela, nous introduisons :

```c
typedef struct {
        int current_state;
        int init_state;
        unsigned n;
        double p;
} RW;

RW * RW_allocate(int init_state n0, double p0);
void RW_free(RW * X);
int RW_update(RW * X, gsl_rng * G);
void RW_reset(RW * X);
```

1. Écrire le code des quatre fonctions.
2. Écrire un programme qui écrit dans un fichier "traj1.txt" une trajectoire de longueur 10000 partant de 0 pour le paramètre $p = 0.5$. Refaire la même chose dans un fichier "traj2.txt" pour $p = 0.7$.
3. Pour $p = 0.6$, quel est le temps moyen mis par une marche aléatoire pour atteindre le point $x = 600$ ?
4. Écrire une autre structure `RW2` et ses fonctions associées qui permette de simuler la marche qui se déplace de $+1$ avec probabilité $a$, de $-1$ avec probabilité $b$ et reste sur place avec probabilité $1-a-b$. Représenter une trajectoire de longueur 1000 et comprendre à quoi elle ressemble.

**Exercice 2.** (chaîne à deux états)

1. Écrire, sur le modèle précédent, une structure `TwoStatesMarkov` et ses quatre fonctions associées qui permettent de simuler une chaîne de Markov de matrice de transition
$$Q = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}.$$
2. Vérifier numériquement le théorème ergodique.
3. Quel est le temps moyen de retour en chacun des deux points ?

**Exercice 3.** (chaîne avec un nombre fini d'états) Soit $Q$ une matrice de transition de taille $M$. Soit $Q_c$ la matrice définie par :

$$Q_c(i,j) = \sum_{k=1}^{j} Q(i,j)$$

Nous considérons la structure en `C` définie par :

```
typedef struct {
        int current_state;
        int init_state;
        unsigned n;
        unsigned M;
        double * Qc;
} QMC;
```

où `Qc` représente un tableau dynamique de taille $M^2$ tel que `Qc[N*i+j]` vaille $Q_c(i,j)$.

1. Écrire les quatre fonctions d'allocation, de libération, de redémarrage et d'itérations. La fonction d'allocation doit être du type :
   ```
   RW * RW_allocate(int init_state n0, int M0,FILE * matrix_data);
   ```
   où `matrix_data` est un fichier contenant la matrice $Q$ écrite ligne par ligne.

2. Pourquoi préfère-t-on stocker la matrice $Q_c$ plutôt que la matrice $Q$ ?

3. Reprendre des exercices de TD avec des matrices de transition finies, et répondre numériquement aux questions posées.

4. Si la matrice $Q$ contient beaucoup de 0, pourquoi la solution proposée n'est-elle pas optimale ? Proposer une autre solution, la programmer et mesurer la différence de performances.

**Exercice 4.** Nous nous intéressons à la chaîne de Markov sur $\mathbb{N}$ dont les probabilités de transitions non nulles sont données par :

$$\forall k \in \mathbb{N}, \qquad Q(k,k+1) = 1 - p_k \qquad\qquad Q(k,0) = p_k \tag{1}$$

1. Programmer une structure `MC1` et ses quatre fonctions associées dans le cas $p_k = 1/10$. Représenter une trajectoire typique (assez longue).

2. Programmer une structure `MC2` et ses quatre fonctions associées dans le cas $p_k = 1/(k+2)$. Représenter une trajectoire typique (assez longue).

3. Programmer une structure `MC3` et ses quatre fonctions associées dans le cas $p_k = e^{-k}$. Représenter une trajectoire typique (assez longue).

4. Dans chacun des cas, mesurer le temps moyen de retour à l'origine et interpréter les résultats.

## Écrire et lire dans un fichier en langage `C`

Pour écrire ou lire dans un fichier, on adaptera l'exemple suivant :

```c
#include <stdio.h>
int main(void) {
        double x=3.14, y=2.17;
        int n=2.16;
        // Ouverture du fichier (w pour write)
        FILE *  fichier_pour_ecrire = fopen("ecriture.txt","w");
        // Ecriture du dans le fichier
        // %f pour les float, %lf pour les doubles, %d pour les entiers
        // \n pour aller à la ligne
        fprintf(fichier_pour_ecrire,"%f %d toto \n", x,n);
        fprintf(fichier_pour_ecrire,"%f %d \n", y,n+1);
        // Fermeture du fichier
        fclose(fichier_pour_ecrire);

        //Ouverture du fichier (r pour read)
        FILE  * fichier_pour_lire = fopen("lire.txt","r");
        // Stockage d'un double puis d'un entier dans x et n
        fscanf(fichier_pour_lire,"%lf %d", &x,&n);
        //Fermeture du fichier
        fclose(fichier_pour_lire);
        return 0;
}
```

## Utilisation de Gnuplot

- Dans un terminal, il faut aller dans le répertoire où sont les fichiers à visualiser puis entrer la commande `gnuplot` .
- Une fois dans `gnuplot` , représenter les données d'un fichier `exemple.txt` avec *un nombre par ligne* se fait par :

  ```
  > plot "exemple.txt"
  ```

  Les données seront représentées par des points. Pour avoir une ligne brisée, il faut taper :

  ```
  > plot "exemple.txt" with lines
  ```

- Supposons qu'un fichier ait, sur toutes ses lignes, une suite de 4 valeurs séparées par des espaces. Pour représenter la quatrième valeur en fonction de la deuxième, on utilise :

  ```
  > plot "exemple.txt" using 2:4 with lines
  ```

- On sort de `gnuplot` avec l'instruction `quit` .
- Une fois un graphique représenté, on peut zoomer en dessinant un rectangle avec le bouton *droit* de la souris enfoncé. Une icône de la barre de menu du graphique permet de revenir à la taille initiale.