# Simulations de chaînes de Markov en langage `C` — 2.

On veillera à toujours bien libérer la mémoire allouée, fermer les fichiers ouverts, etc.

**Exercice 1.**

1. Écrire dans un fichier `markov.h` la déclaration de la structure `MarkovChain` avec les champs `current_state`, `init_state`, `time`, `nb_states` de type `int`, et `Q_cumul` de type `double *` pour stocker les probabilités cumulées de la matrice de transition de notre chaîne de Markov.
2. Écrire dans ce même fichier le prototype des fonctions `MC_allocate`, `MC_free`, `MC_reset` et `MC_iterate`.
3. Écrire dans le fichier `markov.c` le code de ces fonctions.

**Exercice 2.** Soit la chaîne de Markov sur trois états (**0**, **1**, **2**) définie par la matrice de transition suivante :

$$Q = \begin{pmatrix} 0,3 & 0,2 & 0,5 \\ 0,2 & 0,4 & 0,4 \\ 0,3 & 0 & 0,7 \end{pmatrix}$$

1. Que peut-on dire de cette chaîne de Markov ? (classes, récurrence, transience, périodicité,...)
2. Créer un fichier `create_traj.c` dont la fonction `main` écrira dans un fichier `traj1.txt` une trajectoire aléatoire de la chaine de Markov, partant de l'état **0**, et de longueur $L = 100000$. On n'écrira pas la valeur de départ pour $n = 0$, mais juste les valeurs renvoyées par `MC_iterate`.
3. Écrire dans un fichier `stat_traj.c` une fonction `main` qui
   — définit une matrice $A$ $3 \times 3$ remplie avec des 0
   — lit la trajectoire depuis le fichier `traj1.txt`
   — au cours de la lecture modifie la matrice $A$ de telle sorte qu'à la fin, $A_{i,j}$ soit égal au nombre de fois qu'une transition de $i$ à $j$ a eu lieu au cours de la trajectoire. (N'oubliez pas que la première transition a lieu entre **0** et la première position enregistrée dans le fichier.
4. Construire à partir de $A$ un tableau contenant le nombre de visites à chaque site.
5. À partir du théorème ergodique, construire un estimateur de la mesure de probabilité invariante $\pi$ pour cette chaîne de Markov.
6. Construire à partir de $A$ un estimateur de la matrice de transition $Q$. Justifiez pourquoi cela marche (on vérifiera que la suite des couples $((X_n, X_{n+1}))_n$ est aussi une chaîne de Markov à laquelle on peut appliquer le théorème ergodique.
7. Afficher une valeur approchée $\tilde{Q}$ de $Q$, une valeur approchée $\tilde{\pi}$ de $\pi$ et vérifier numériquement que $\tilde{pi}$ est presque un vecteur propre à gauche pour $\tilde{Q}$.

**Exercice 3.**

1. Dans des fichiers `rw_circle.h` et `rw_circle.c`, adapter la structure et les fonctions utiles pour les chaines de Markov pour représenter une marche aléatoire sur un cercle de longueur $n$ (pour laquelle la transition depuis chaque état **0**, ..., $\mathbf{n-1}$ vers le précédent ou le suivant *modulo* $n$, se fait avec probabilité $\frac{1}{2}$). On notera qu'on a pas besoin de stocker la matrice de transition, car la règle de mise à jour est très simple.

2. Dans un fichier `rwc_hitting_times.c` , on veut estimer le temps moyen de premier retour depuis **0** de tous les autres sites, en fonction de $n$. Pour cela, pour différentes valeurs de $n$, créer une marche sur le cercle de taille $n$, et un tableau dynamique `ht` de taille $n$. Pour un nombre $K$ de trajectoires, on initialise le tableay `ht` avec des nombres négatifs. Tant qu'il y a un coefficient négatif, on fait progresser la trajectoire, et on enregistre au temps $k$ la valeur de $k$ dans `ht[` $X_k$ `]` si la valeur de cette case était encore strictement négative. Une fois toutes les cases remplies avec des nombres positifs, le tableau `hk` contient les temps d'atteinte/premier retour pour la trajectoire en question. Moyenner les résultats sur les $K$ trajectoires et comparer avec le résultat théorique obtenu en classe.
3. Dessiner avec gnuplot ce temps moyen pour $n = 30$.
4. Estimer la mesure de probabilité invariante pour ce modèle.

**Exercice 4.** Dynamique de Glauber pour le modèle d'Ising à une dimension.

Le modèle d'Ising, introduit dans les années 1920, est un modèle mathématique pour le ferromagnétisme. Les atomes d'un cristal sont représentés par les sommets d'un graphe $G$ . Chaque atome se comporte comme un petit aimant (*spin*), qui peut être dirigé vers le haut (+1) ou vers le bas (-1). Pour une valeur fixée de la température inverse $\beta$, la distribution des spins est aléatoire : la probabilité que le spin à chaque sommet de $G$ soient donné par la fonction $\sigma : G \to \{-1, +1\}$ est

$$\mathbb{P}_\beta(\sigma) = \frac{1}{Z_\beta} \exp\left( \sum_{e:v\sim w} \beta\sigma(v)\sigma(w) \right)$$

où $Z_\beta$ est un facteur de normalisation

$$Z_\beta = \sum_\sigma \exp\left( \sum_{e:v\sim w} \beta\sigma(v)\sigma(w) \right)$$

qui est appelé *fonction de partition*. La fonction de partition est difficile à calculer à cause du grand nombre de termes à sommer, ce qui rend l'estimation de la probabilité d'une configuration compliquée.

Afin de simuler une configuration aléatoire $\sigma$, l'idée est d'utiliser une chaîne de Markov irréductible apériodique, dont $\mathbb{P}_\beta$ sera la mesure invariante, et d'utiliser le théorème de convergence vers la mesure invariante pour obtenir que la distribution de la chaîne au bout d'un temps très long s'approche de $\mathbb{P}_\beta$.

Une telle chaîne de Markov est la dynamique de Glauber. Si la configuration de spin à l'instant $n$ est $\sigma$, alors :

— on choisit uniformément un sommet $v$ du graphe
— la configuration $\sigma'$ à l'instant $n+1$ est identique à $\sigma$ en tout sommet $w \neq v$, et au sommet $v$, $\sigma'(v)$ vaut +1 avec probabilité

$$\frac{e^{\beta S(\sigma,v)}}{e^{\beta S(\sigma,v)} + e^{-\beta S(\sigma,v)}}, \quad \text{avec} \quad S(\sigma, v) = \sum_{w\sim v} \sigma(w).$$

1. Montrer que la chaîne ainsi construite est irréductible, récurrente, apériodique, et que $\mathbb{P}_\beta$ est l'unique mesure invariante.
2. On prend pour le graphe $G$ une ligne, dont les sommets sont 0,1,. . . ,N. On supposera que les spins en 0 et en N sont gelés et resteront fixés à +1 (le choix aléatoire du spin sera donc entre 1 et $N-1$). Dans des fichiers `ising.h` et `ising.c` , déclarer et définir les structures et les fonctions pour simuler ces chaînes de Markov. La structure devra avoir un champs pour stocker $\beta$ ainsi que la taille $N$ du système.

3. Pour $N = 10$, 20, 50, et pour $\beta = 1$, 2, 5 calculer la moyenne sur 100 trajectoire de la magnétisation (valeur moyenne du spin) au milieu du système au bout de 10, 100, 1000, 10000, itérations, à partir du point de départ où tous les spins sont $+1$. Évaluer empiriquement le temps qu'il faut pour s'approcher de la mesure invariante.
4. Pour $N = 50$, dessiner en fonction de la température la magnétisation moyenne au milieu du système.

**Compilation :** le compilateur `gcc` doit être invoqué avec les options `-lgsl` et `-lgslcblas` . Quelques exemples :

```
//pour produire un fichier objet prog.o avec optimisation O2
gcc proc.c -O2 -lgsl -lgslcblas -I /usr/lib/include -lm
//pour produire un exécutable prog.exe
gcc -o prog.exe prog.c -lgsl -lgslcblas -I /usr/lib/include -lm
```

```
#include<gsl/gsl_rng.h> // pour les générateurs
#include<gsl/gsl_randist.h> // pour les lois usuelles
#include <stdio.h> //pour l'affichage
#include <time.h> //pour lire l'heure
int main(void) {
        // CREATION DU GENERATEUR:
        gsl_rng * Gege= gsl_rng_alloc(gsl_rng_mt19937);
        // CHOIX DE LA GRAINE
        gsl_rng_set(Gege,2016);
        //ou
        gsl_rng_set(Gege,time(NULL));

        // GENERATION D'UNE UNIFORME SUR [0,1]
        double x=gsl_rng_uniform(Gege);

        // GENERATION D'UNE UNIFORME SUR 0,1,...,N-1:
        int N=6;
        int k=gsl_rng_uniform_int(Gege,N);//uniforme sur 0,1,...,5

        // GENERATION D'UNE V.A. DE LOI BINOMIALE(N,p)
        N=8; double p=0.25;
        int u=gsl_ran_binomial(Gege,p,N);
        // AFFICHAGE DE QUELQUES RESULTATS:
        printf("x=%f\n y=%f\n n=%d\n u=%d",x,y,n,u);

        // EFFACEMENT DU GENERATEUR (NE PAS OUBLIER !!!)
        gsl_rng_free(Gege);
        return 0;
}
```