Séance 3 : Simuler l'ouverture de canaux ioniques grâce aux chaînes de Markov

Cours de L3 : Mathématiques I - ce qu'un biologiste ne peut ignorer

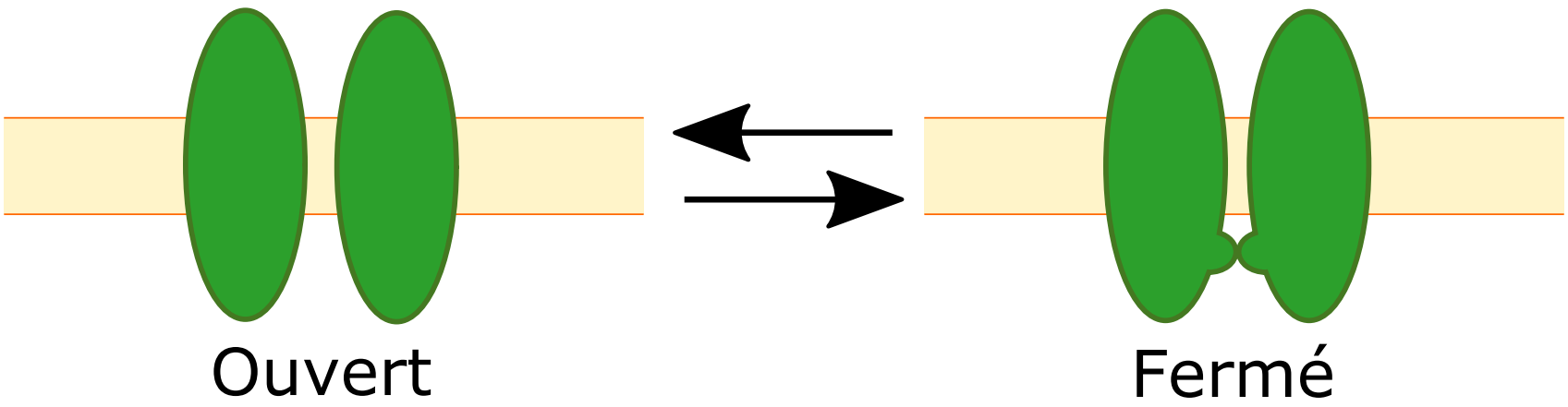
Benoît Perez-Lamarque & Pierre Vincens

ENS Paris - Département de Biologie

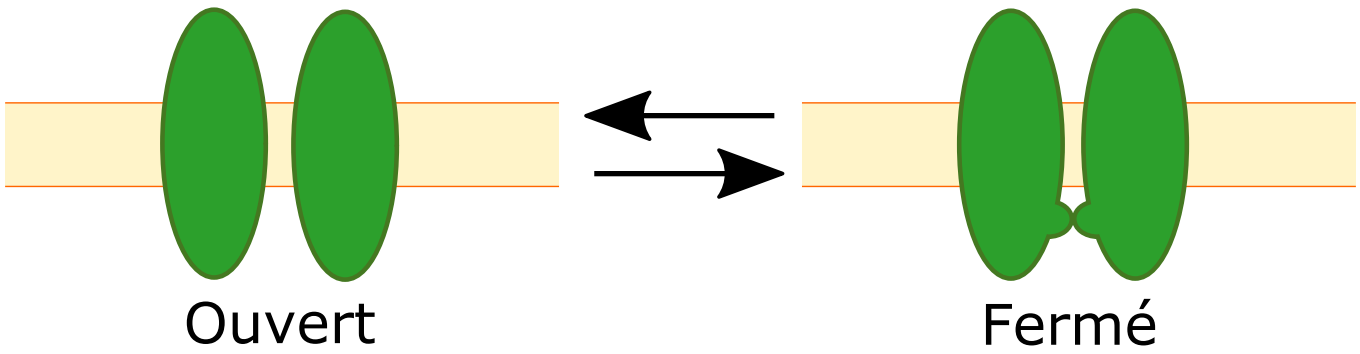
17 janvier 2020

Canal ionique membranaire

Soit un transporteur transmembranaire que l'on trouve dans deux conformations : ouvert ou fermé.
On sait que la protéine change de conformation de manière **aléatoire** selon un certain taux que l'on peut mesurer.



Canal ionique membranaire



On souhaite modéliser ce système pour répondre à un certain nombre de questions :

- Quel est le comportement d'un tel transporteur sur le long terme ?
- Quel est le comportement d'un grand nombre de ces transporteurs à la surface de la membrane plasmique ?

Les outils de la modélisation :

Caractériser un modèle en 6 étapes :

- Déterministe *versus* stochastique : entièrement déterminée par paramètres et conditions initiales *versus* composantes aléatoires ;
- Nature de l'espace d'état : ensemble discret, espace vectoriel...
- Temps discret *versus* continu : *e.g.* espace du temps $\in \mathbb{N}_+$ *versus* $\mathbb{R}$;
- Autonome *versus* non-autonome : le temps n'apparaît pas comme variable dans l'équation de la dynamique *versus* apparaît.
- Équation de la dynamique : $u_t = f(u_{t-1})$, $\dot{u} = f(u)$...
- Conditions initiales : u_0

Modélisation d'un canal ionique

Nous allons utiliser une **Chaîne de Markov** :

- à temps discret.
- avec comme **espace d'état** (de cardinal fini) l'ensemble 'Fermé, Ouvert' ou $\{0, 1\}$.
- avec comme **condition initiale** X_0, qui peut être une variable aléatoire (sa loi est appelée **mesure initiale** de la chaîne) ou une valeur déterministe (par exemple 'Ouvert', ce qui revient à avoir comme mesure initiale $P(X_0 = 1) = 1$).
- avec les **probabilités de transitions** : $a_{ij} = \mathbb{P}(X_{n+1} = j \mid X_n = i)$

On note la **matrice de transition** suivante :

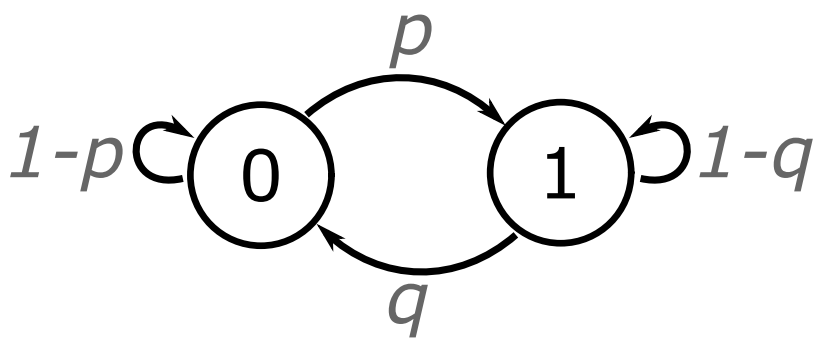
$$A = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}$$

Modélisation d'un canal ionique

Nous allons utiliser une **Chaîne de Markov** :

On note la **matrice de transition** suivante :

$$A = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}$$



Exercice 1

Comparaison de nombre à virgule flottante

On ne teste jamais la somme de nombres à virgule flottante avec une égalité stricte. En effet, la représentation des nombres à virgule flottante pose de nombreux problèmes.

Par exemple : assert $0.1 + 0.2 == 0.3$

Comparaison de nombre à virgule flottante

On ne teste jamais la somme de nombres à virgule flottante avec une égalité stricte. En effet, la représentation des nombres à virgule flottante pose de nombreux problèmes.

Par exemple : assert $0.1 + 0.2 == 0.3$

S'il est impossible de représenter $\frac{1}{3}$ dans l'écriture décimale sans utiliser une infinité de nombres après la virgule (0.33333...), c'est aussi le cas de 0.1 en écriture binaire (0.0001100110011...)

Comparaison de nombre à virgule flottante

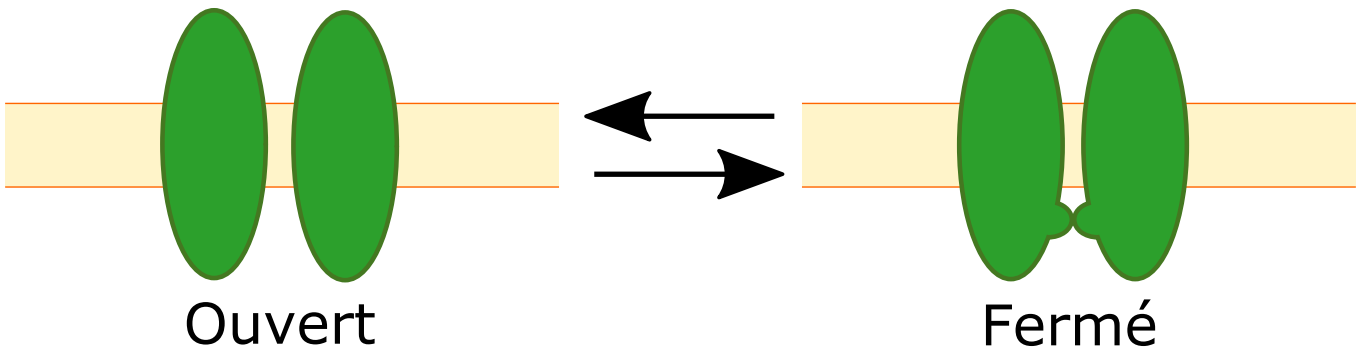
En général, on teste l'égalité des nombre à virgule flottante à une certaine précision près. Par exemple, on testera :

assert $abs(((0.1 + 0.2) - 0.3)/0.3) < 1e - 6$

La fonction numpy.testing.assert_allclose permet de faire de telles comparaisons élément par élément sur un array.

Partie I : Simulation d'une chaîne de Markov

A) Chaînes à 2 états :



Comment simuler la **trajectoire** d'un tel objet ?

La trajectoire est la donnée $(X_n)_{n=0,1,2,3\ldots}$, c'est à dire une suite à valeur dans l'espace d'état et indexée sur les nombre entiers.

Partie I : Simulation d'une chaîne de Markov

Réaliser des tirages de nombres aléatoires à l'aide d'un ordinateur :

Il est impossible de réaliser des tirages de nombre aléatoire avec la majorité des ordinateurs : à la place on utilise des algorithmes qui génèrent des **suites de nombres pseudo-aléatoires**.

Partie I : Simulation d'une chaîne de Markov

Réaliser des tirages de nombres aléatoires à l'aide d'un ordinateur :

Il est impossible de réaliser des tirages de nombre aléatoire avec la majorité des ordinateurs : à la place on utilise des algorithmes qui génèrent des **suites de nombres pseudo-aléatoires**.

Par exemple, un générateur congruentiel linéaire fonctionne ainsi :

$$X_{n+1} = (aX_n + b) \bmod m$$

Partie I : Simulation d'une chaîne de Markov

Réaliser des tirages de nombres aléatoires à l'aide d'un ordinateur :

Il est impossible de réaliser des tirages de nombre aléatoire avec la majorité des ordinateurs : à la place on utilise des algorithmes qui génèrent des **suites de nombres pseudo-aléatoires**.

Par exemple, un générateur congruentiel linéaire fonctionne ainsi :

$$X_{n+1} = (aX_n + b) \bmod m$$

Cette suite est périodique et si on prend $a = 25$, $b = 16$, $m = 256$, $X_0 = 50$ la période est de 5 !
$>$ 50, 242, 178, 114, 50, 242, 178, 114, 50, 242, ...

Partie I : Simulation d'une chaîne de Markov

Réaliser des tirages de nombres aléatoires à l'aide d'un ordinateur :

Numpy implémente un générateur de nombre pseudo-aléatoire appelé **Mersenne Twister**. Chaque appel à la fonction np.random.random() donne le prochain élément dans la suite de nombres pseudo aléatoires. La période de cette suite est de $2^{19937} - 1$.

Partie I : Simulation d'une chaîne de Markov

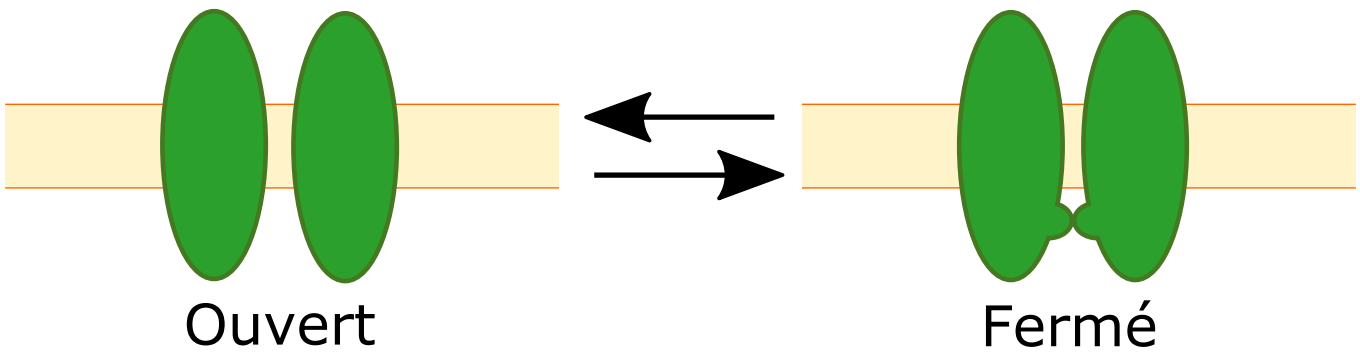
Réaliser des tirages de nombres aléatoires à l'aide d'un ordinateur :

Numpy implémente un générateur de nombre pseudo-aléatoire appelé **Mersenne Twister**. Chaque appel à la fonction np.random.random() donne le prochain élément dans la suite de nombres pseudo aléatoires. La période de cette suite est de $2^{19937} - 1$.

Les nombres pseudo aléatoires dépendent de la condition initiale du générateur, appelée **graine**. Vous pouvez fixer la graine de numpy à l'aide de la commande np.random.seed.

Partie I : Simulation d'une chaîne de Markov

A) Chaînes à 2 états :



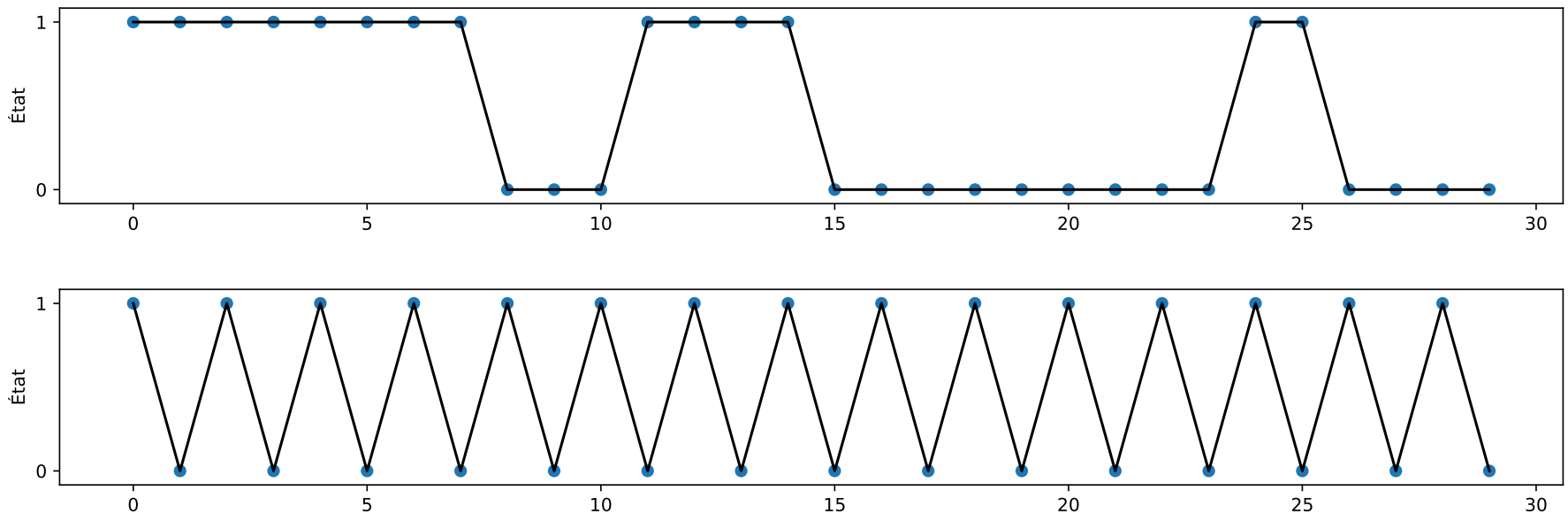
Comment simuler la **trajectoire** d'un tel objet ?

La trajectoire est la donnée $(X_n)_{n=0,1,2,3\ldots}$, c'est à dire une suite à valeur dans l'espace d'état et indexée sur les nombre entiers.

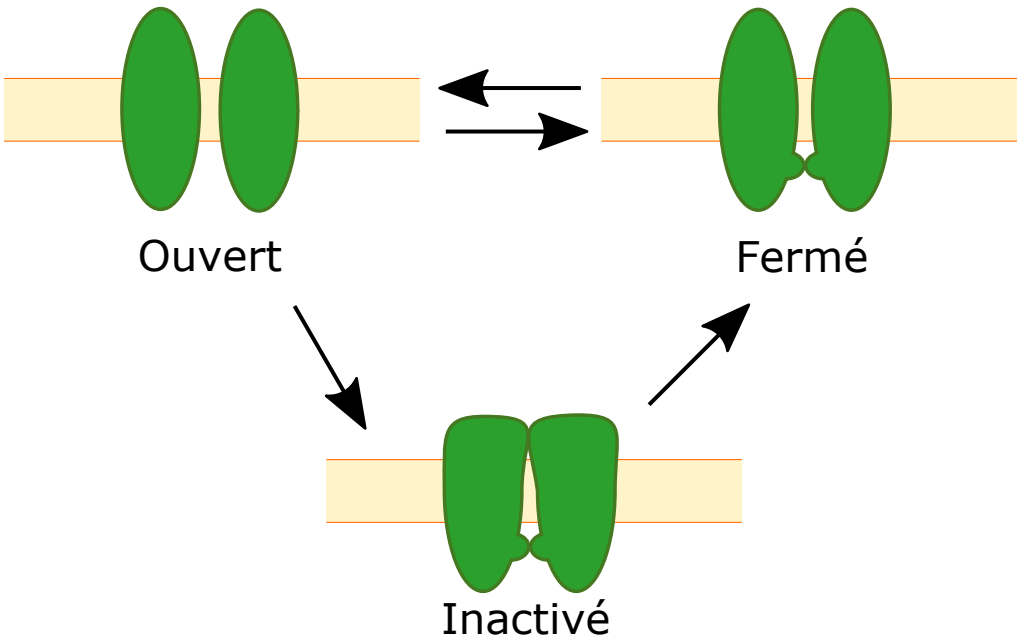
Exercice 2

Partie I : Simulation d'une chaîne de Markov

A) Chaînes à 2 états :



B) Rajout d'un troisième état :



Si on prend une chaîne légèrement plus compliquée, par exemple en ajoutant un troisième état, il faut changer notre algorithme pour simuler sa trajectoire.

Partie I : Simulation d'une chaîne de Markov

Comment échantillonner une densité de probabilité discrète ?

Le problème principal de cet algorithme est de choisir l'état au temps $n+1$ sachant que la chaîne était dans l'état i au temps n. Cela revient à échantillonner la densité de probabilité discrète définie sur la i-ème ligne de la matrice de transition.

Le problème c'est qu'un ordinateur par défaut ne peux générer que des nombres pseudo-aléatoires entre 0 et 1 (np.random.random()).

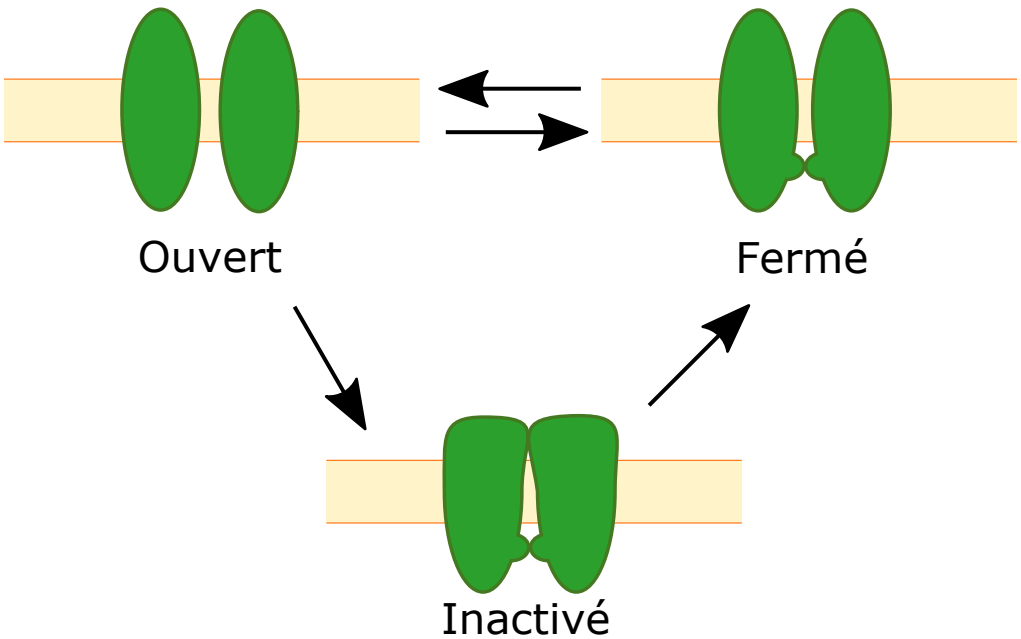
Partie I : Simulation d'une chaîne de Markov

Comment échantillonner une densité de probabilité discrète ?

Méthode de la fonction de répartition inverse :
Cette méthode fait appel à l'inverse de fonction de répartition. La fonction de répartition F donne $F(k) = P(X \leq k)$. Elle est constante par morceaux pour une variable discrète. Il suffit de l'inverser, de tirer un nombre entre 0 et 1 et de regarder sur quel "palier" on tombe pour avoir la valeur recherchée.

En pratique, on peut utiliser la fonction np.random.choice(<espace d'etat>, <nombre de tirage>, p=<probabilités>), qui utilise la méthode de la fonction de répartition.

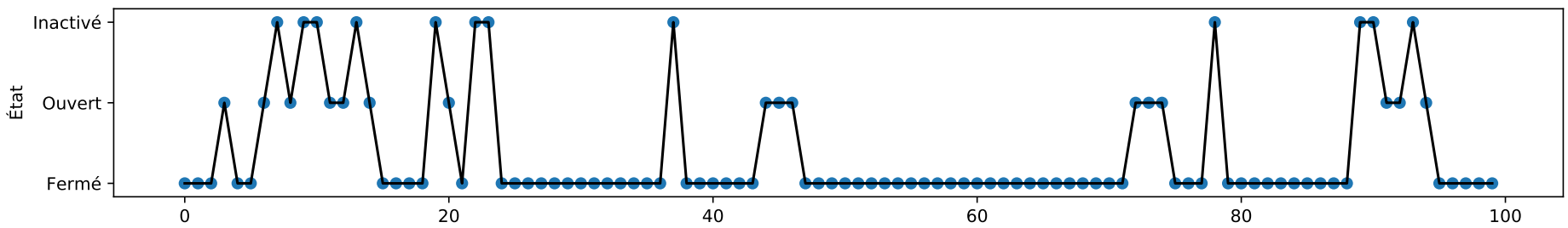
B) Rajout d'un troisième état :



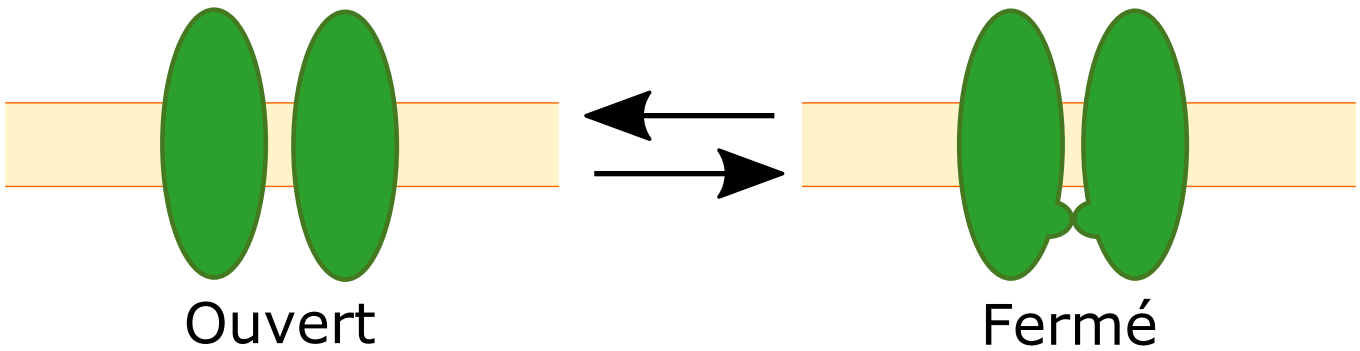
Exercice 3

Partie I : Simulation d'une chaîne de Markov

B) Rajout d'un troisième état :



Partie II : Mesure invariante et théorème ergodique



On souhaite modéliser ce système pour répondre à un certain nombre de questions :

- Quel est le comportement d'un tel transporteur sur le long terme ?
- Quel est le comportement d'un grand nombre de ces transporteurs à la surface de la membrane plasmique ?

Partie II : Mesure invariante et théorème ergodique

La **mesure invariante** d'une chaîne de Markov récurrente positive est donnée par le **vecteur propre dominant** de sa matrice de transition.

Si on a une trajectoire, la **proportion du temps passé en** i est :

$$\tau_i = \frac{1}{T} \sum_{k=0}^{T} 1_{(X_k = i)}$$

Si on a N trajectoires notées $X^{(1)}, X^{(2)}, ...X^{(N)}$, l'**estimateur de la probabilité d'un évènement** :

$$\mathbb{P}(X_t = i) \approx \rho_{t,i} = \frac{1}{N} \sum_{j=0}^{N} 1_{(X_t^{(j)} = i)}$$

Exercice 4

Partie II : Mesure invariante et théorème ergodique

A) Proportion du temps passé en un état donné :

On cherche à mesurer la proportion du temps passé dans un état donné et à l'afficher sous forme graphique.

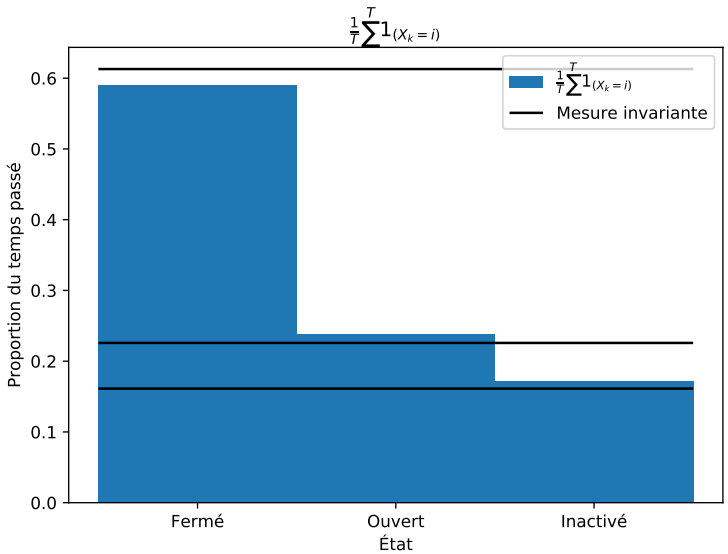
Exercice 5

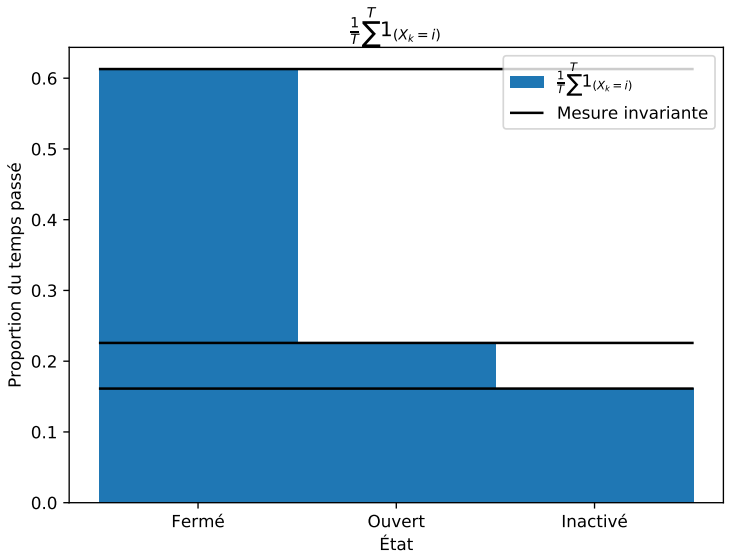
Partie II : Mesure invariante et théorème ergodique

A) Proportion du temps passé en un état donné :

On cherche à mesurer la proportion du temps passé dans un état donné et à l'afficher sous forme graphique.

Exercice 5



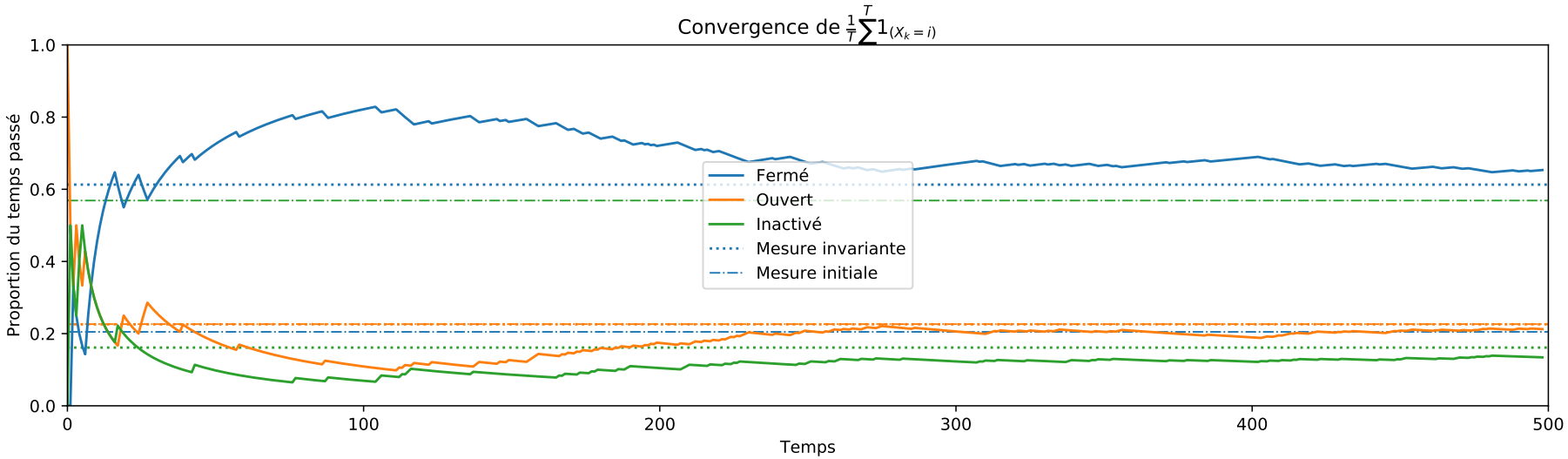


Partie II : Mesure invariante et théorème ergodique

A) Proportion du temps passé en un état donné :

On cherche à mesurer la proportion du temps passé dans un état donné et à l'afficher sous forme graphique.

Exercice 5



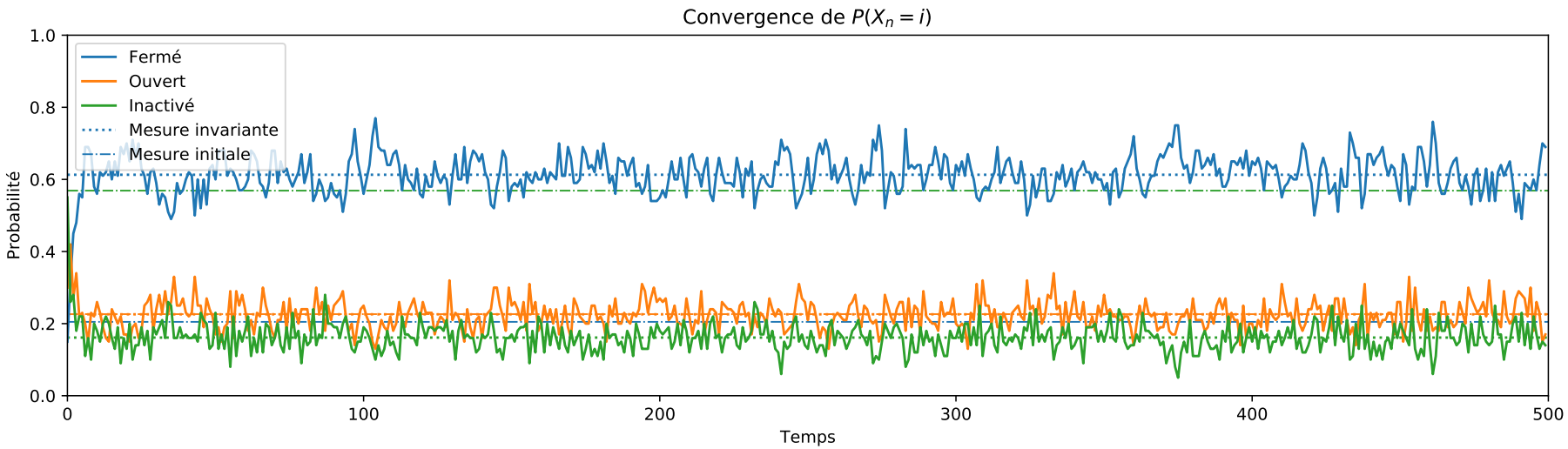
B) Probabilité d'être dans un état donné à un temps donné :

Exercice 6

Partie II : Mesure invariante et théorème ergodique

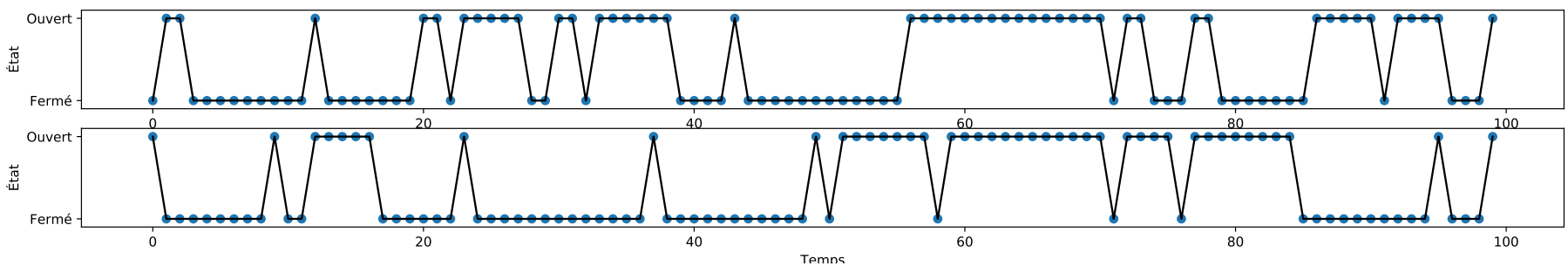
B) Probabilité d'être dans un état donné à un temps donné :

Exercice 6



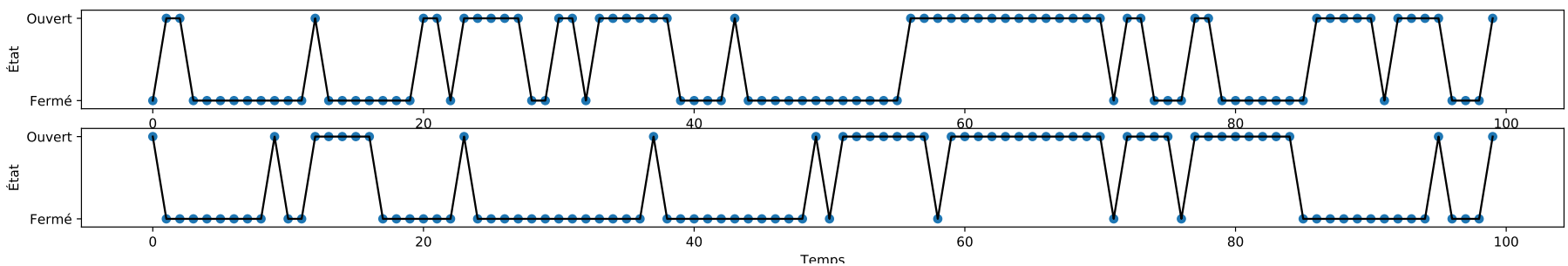
Partie III : Inférence

L'ouverture de deux types de canaux ioniques a été suivie au cours du temps et nous avons enregistré des séries temporelles de mesure de patch clamp qui donnent l'état ouvert ou fermé des canaux au cours du temps.



Partie III : Inférence

L'ouverture de deux types de canaux ioniques a été suivie au cours du temps et nous avons enregistré des séries temporelles de mesure de patch clamp qui donnent l'état ouvert ou fermé des canaux au cours du temps.



On cherche à estimer les matrices de transition de ces deux canaux à partir des trajectoires mesurées.

Exercice 7

Partie III : Inférence

Comment savoir si les matrices de transition sont justes ?

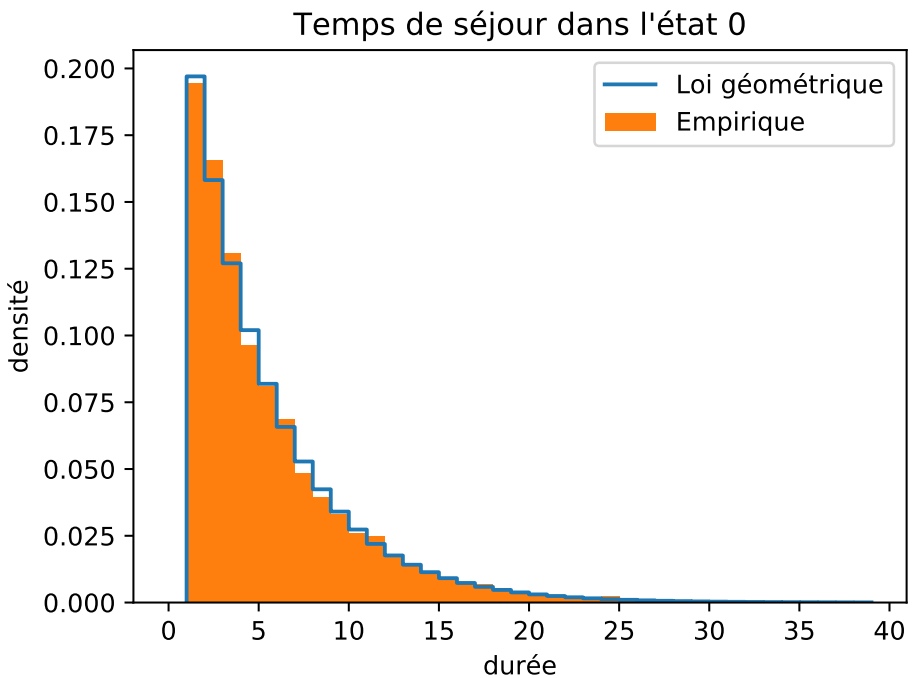
Dans un modèle à deux états, les temps d'attente dans l'état ouvert ou fermé suivent des lois géométriques. Regardons si c'est le cas.

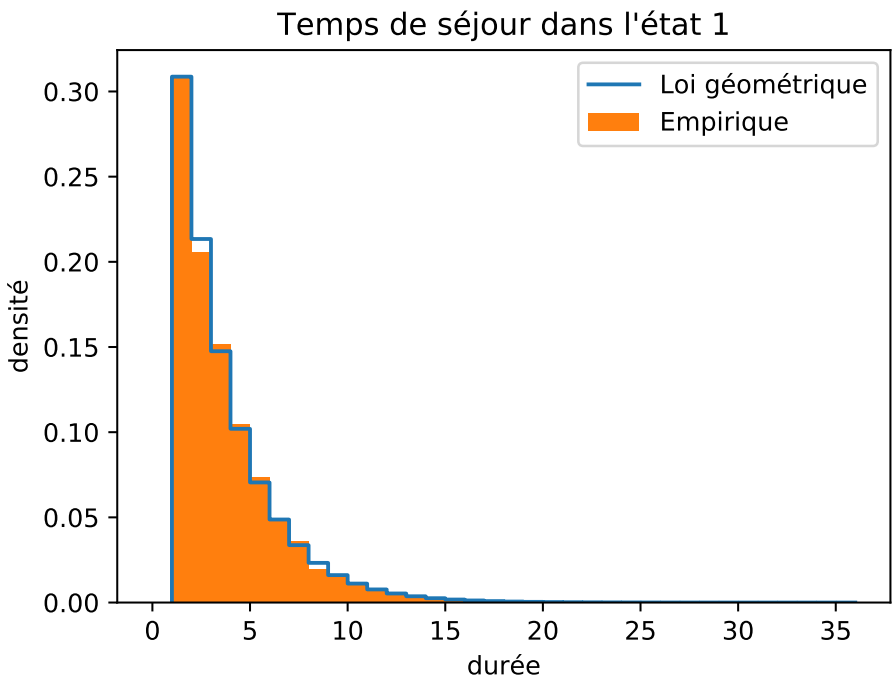
Exercice 8

Partie III : Inférence

Comment savoir si les matrices de transition sont justes ?

Canal 1

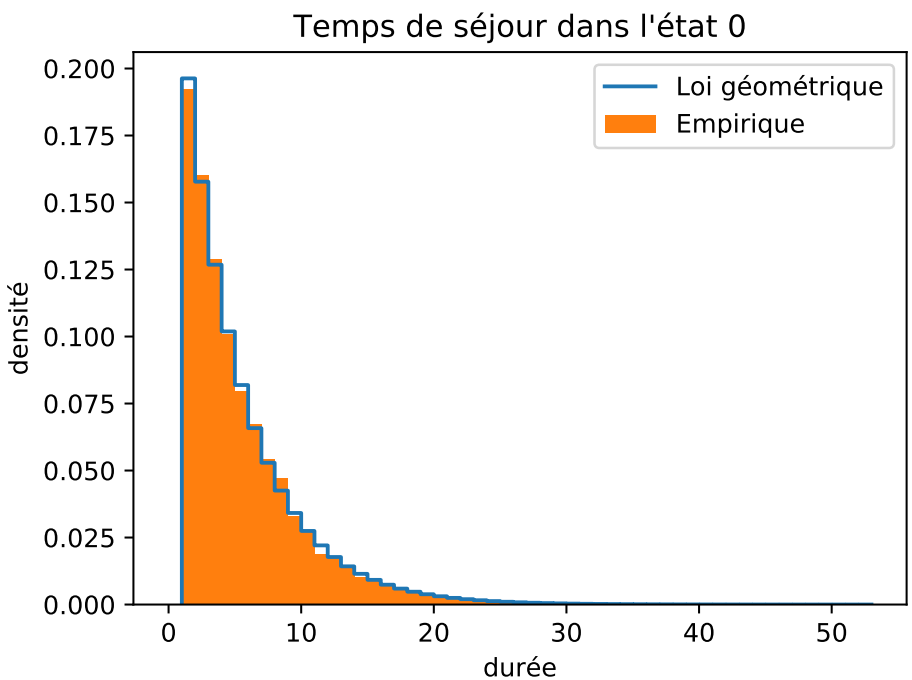


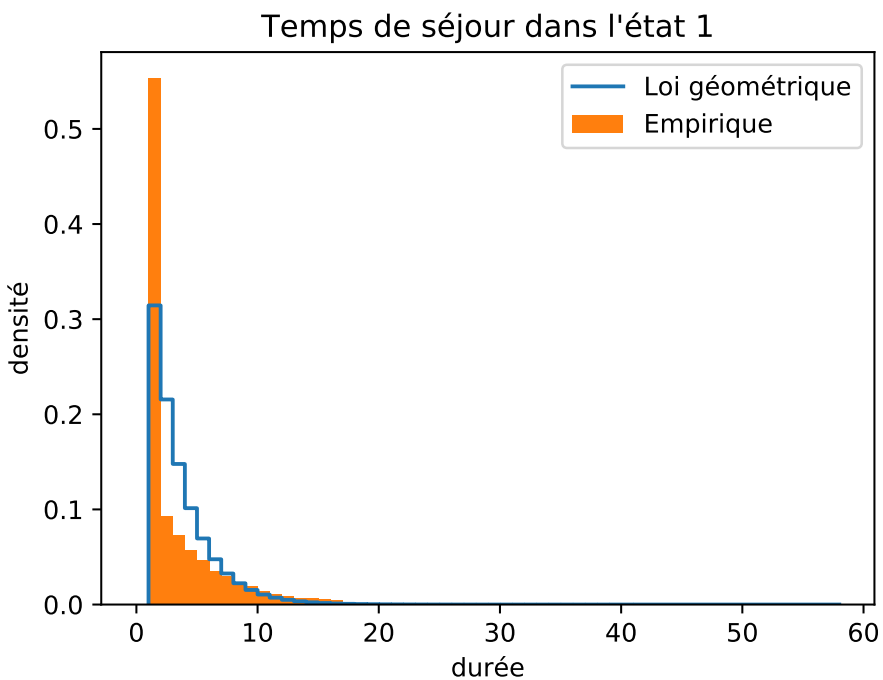


Partie III : Inférence

Comment savoir si les matrices de transition sont justes ?

Canal 2





Partie III : Inférence

Comment savoir si les matrices de transition sont justes ?

Les deux jeux de données donnent la même matrice de transition mais le second n'est pas issu d'une chaîne de Markov à deux états ! La méthode pour étudier ce genre de sytème est de faire un modèle de mélange (statistique) pour voir si le temps de séjour n'est pas la composition de plusieurs loi géométriques, ce qui donne un indice sur le nombre d'états cachés qui existent. On peut ensuite paramétriser un modèle de Markov Caché.