

Rapport du projet de FAT

MODÉLISATION D'UN RÉSEAU DE VÉLIB

Adrien BLASSIAU
Corentin JUVIGNY

SOMMAIRE

1	Modélisation Markovienne du problème	1
1.1	Rappel des consignes	1
1.2	Modélisation	1
1.2.1	Espace d'états	1
1.2.2	Générateur infinitésimal	2
1.2.3	Probabilité stationnaire	3
2	Calibration des paramètres	5
2.1	λ_i	5
2.2	λ_{ij}	5
2.3	p_{ij}	5
2.4	Paramètres calculés à partir du modèle réduit à 5 stations	6
3	Étude de la modélisation	7
3.1	Étude avec 1 vélo	7
3.2	Étude avec 100 vélos	9
4	Étude de l'impact des paramètres	11
4.1	Impact du nombre de vélos dans le système sur la probabilité stationnaire	11
4.2	Impact de la durée du processus sur l'obtention de la probabilité stationnaire . .	12

```
In [1]: from migration_process import *
import numpy as np
import matplotlib.pyplot as plt
from parameters import *
```

Ce rapport a été généré à partir de `rapport.ipynb`

1. MODÉLISATION MARKOVIENNE DU PROBLÈME

1.1. Rappel des consignes

Tout d'abord rappelons les consignes du problème avec les notations utilisées :

- On s'intéresse à N stations Vélib qui contiennent respectivement $v_i, i \in \{1, \dots, N\}$ places. Il y a $K \leq \sum_{i=1}^N v_i$ vélos dans le système.
- Des vélos se déplacent entre ces stations. Aucun vélo ne vient de l'extérieur, le système est donc **fermé**.
- On dispose des **paramètres** suivants :
 - $\forall i \in \{1, \dots, N\}, \lambda_i$ **correspond à l'intensité des occurrences de demande de vélos à la station i .**
 - $\forall i, j \in \{1, \dots, N\}$ tels que $i \neq j$, **le temps de trajet en vélo de la station i à la station $j \sim \mathcal{E}(\lambda_{ij})$.** Attention, $\lambda_{ij} \neq \lambda_{ji}$ en général.
 - une matrice de routage P carrée de taille N , avec $\forall i, \sum_{j=1}^N p_{ij} = 1$ (elle est donc stochastique) où $\forall i, j, p_{ij}$ **est interprété comme la probabilité d'aller vers j pour un vélo qui part de la station i .**
- On fait **hypothèses** suivantes :
 - Il n'y a **pas de boucle**, donc $\forall i, p_{ii} = 0$
 - Si un vélo trouve la station d'arrivée j pleine, on tire au sort une nouvelle station d'arrivée k suivant p_{jk} où $k \in \{1, \dots, N\}$ avec $k \neq j$
 - Si une requête à la station i ne trouve pas de vélo, elle est perdue.

1.2. Modélisation

1.2.1 Espace d'états

On représente notre système par un ensemble de N^2 noeuds de deux types :

- N noeuds représentent les stations de Vélib, ils sont de la forme $n_{ii}, \forall i \in \{1, \dots, N\}$.
- $N(N - 1)$ noeuds représentant les itinéraires d'une station vers une autre, ils sont de la forme $n_{ij}, \forall i, j \in \{1, \dots, N\}$ tels que $i \neq j$.

On s'inspire du chapitre *sur les colonies* issu du livre de Kelly pour la modélisation Markovienne. Une **colonie** est ici un noeud de notre graphe, donc une station ou un itinéraire, et un **individu** est un vélo.

L'espace des états est :

$$S = \{(n_{ij})_{i,j \in \{1, \dots, N\}} \in \{0, \dots, K\}^{N^2} : \sum_{i=1}^N \sum_{j=1}^N n_{ij} = K \wedge \forall i \in \{1, \dots, N\}, n_{ii} \leq v_i\}$$

Ainsi :

- chaque état est noté $n = (n_{11}, n_{12}, \dots, n_{NN})$ où n_{ij} correspond au nombre de vélos dans l'état correspondant
- $\sum_{i=1}^N \sum_{j=1}^N n_{ij} = K \wedge \forall i \in \{1, \dots, N\}, n_{ii} \leq v_i$ sont les deux **contraintes de capacité** qui formalisent le fait que le nombre de vélos total est K et que chaque station ne doit pas stocker plus de vélos qu'elle n'a de place.

Enfin, pour résumer, on quitte une station n_{ii} en suivant une loi exponentielle de paramètre λ_i , on choisit une station d'arrivée n_{jj} avec une probabilité p_{ij} puis on se retrouve dans l'itinéraire vers la station n_{jj} que l'on quitte selon une loi exponentielle de paramètre λ_{ij} .

1.2.2 Générateur infinitésimal

Soit n_k et n_l deux noeuds distincts, $i, j \in \{1, \dots, N\}$ avec $i \neq j$. Tout comme Kelly, on définit un opérateur T_{kl} qui formalise soit :

- le **départ d'un vélo d'une station n_{ii} vers un itinéraire n_{ij}** . Dans ce cas, n_k est une station et $k = (ii)$; n_l est un itinéraire et $l = (ij)$.
- l'**arrivée d'un vélo dans une station n_{jj} venant d'un itinéraire n_{ij}** , si $n_{jj} < v_j$. Dans ce cas, n_k est un itinéraire et $k = (ij)$; n_l est une station et $l = (jj)$.
- l'**arrivée d'un vélo dans un itinéraire n_{jm} venant d'un itinéraire n_{ij}** , si $n_{jj} = v_j$. Dans ce cas, n_k est un itinéraire et $k = (ij)$; n_l est un itinéraire et $l = (jm)$.

$$T_{kl}(n_{11}, n_{12}, \dots, n_{NN}) = \begin{cases} T_{kl}(n_{11}, n_{12}, \dots, n_k - 1, \dots, n_l + 1, \dots, n_{NN}), & \text{si } k = (ii), l = (ij) \text{ et } i < j \\ T_{kl}(n_{11}, n_{12}, \dots, n_l + 1, \dots, n_k - 1, \dots, n_{NN}), & \text{si } k = (ii), l = (ij) \text{ et } i > j \\ T_{kl}(n_{11}, n_{12}, \dots, n_k - 1, \dots, n_l + 1, \dots, n_{NN}), & \text{si } k = (ij), l = (jj) \text{ et } i < j \text{ et } n_{jj} < v_j \\ T_{kl}(n_{11}, n_{12}, \dots, n_l + 1, \dots, n_k - 1, \dots, n_{NN}), & \text{si } k = (ij), l = (jj) \text{ et } i > j \text{ et } n_{jj} < v_j \\ T_{kl}(n_{11}, n_{12}, \dots, n_k - 1, \dots, n_l + 1, \dots, n_{NN}), & \text{si } k = (ij), l = (jm) \text{ et } i > j \text{ et } n_{jj} = v_j \\ T_{kl}(n_{11}, n_{12}, \dots, n_l + 1, \dots, n_k - 1, \dots, n_{NN}), & \text{si } k = (ij), l = (jm) \text{ et } i < j \text{ et } n_{jj} = v_j \end{cases}$$

Définissons maintenant les taux de transition qui forment les coefficients du générateur infinitésimal, en utilisant la méthode proposée par Kelly. Une transition ne peut avoir lieu qu'entre un état n et un état $T_{kl}(n)$, où n_k et n_l sont deux noeuds distincts. On considère que les taux de transition ont la forme suivante : $q(n, T_{kl}(n)) = \Lambda_{kl} \phi_k(n_k)$. On distingue les trois cas vus plus haut :

- le **départ d'un vélo d'une station n_{ii} vers un itinéraire n_{ij}** . Dans ce cas, n_k est une station et $k = (ii)$; n_l est un itinéraire et $l = (ij)$. On a :

- $\Lambda_{kl} = \lambda_i * p_{ij}$ où λ_i et p_{ij} sont donnés dans l'énoncé
- $\phi_k(n_k) = \min(n_{ii}, 1)$ (1 car une station a un serveur)
- **donc**

$$q(n, T_{kl}(n)) = \lambda_i * p_{ij} * \min(n_{ii}, 1)$$

- **l'arrivée d'un vélo dans une station n_{jj} venant d'un itinéraire n_{ij}** , si $n_{jj} < v_j$. Dans ce cas, n_k est un itinéraire et $k = (ij)$; n_l est une station et $l = (jj)$. On a :

- $\Lambda_{kl} = \lambda_{ij} * p_{ij} = \lambda_{ij}$ où λ_{ij} est donné dans l'énoncé et $p_{ij} = 1$ car on sait déjà où l'on va.
- $\phi_k(n_k) = \min(n_{ij}, K) = n_{ij}$ car on ne peut pas avoir plus de vélos sur un itinéraire que le nombre de vélos total (K car un itinéraire à une infinité de serveur, on prend K le nombre total de vélos)
- **donc**

$$q(n, T_{kl}(n)) = \lambda_{ij} * n_{ij}$$

- **l'arrivée d'un vélo dans un itinéraire n_{jm} venant d'un itinéraire n_{ij}** , si $n_{jj} = v_j$. Dans ce cas, n_k est un itinéraire et $k = (ij)$; n_l est un itinéraire et $l = (jm)$. On a :

- $\Lambda_{kl} = \lambda_{ij} * p_{ij} * p_{jm} = \lambda_{ij} * p_{jm}$ où λ_{ij} est donné dans l'énoncé, $p_{ij} = 1$ car on sait déjà où l'on va et la station n_{jj} étant remplie, on tire une nouvelle station m avec une probabilité p_{jm} .
- $\phi_k(n_k) = \min(n_{ij}, K) = n_{ij}$ car on ne peut pas avoir plus de vélos sur un itinéraire que le nombre de vélos total (K car un itinéraire à une infinité de serveur, on prend K le nombre total de vélos)
- **donc**

$$q(n, T_{kl}(n)) = \lambda_{ij} * n_{ij} * p_{jm}.$$

1.2.3 Probabilité stationnaire

Existence :

L'unique probabilité stationnaire π du processus X sur S , si elle existe, vérifie le système matriciel $\pi A = 0$ où A est le générateur infinitésimal. L'existence est assurée si :

- Le processus est irréductible, c'est-à-dire si tout état est accessible à partir de n'importe quel autre état en un nombre fini d'itérations. **C'est trivialement le cas ici.**
- L'espace d'états doit être fini. On rappelle que l'on a :

$$S = \{(n_{ij})_{i,j \in \{1, \dots, N\}} \in \{0, \dots, K\}^{N^2} : \sum_{i=1}^N \sum_{j=1}^N n_{ij} = K \wedge \forall i \in \{1, \dots, N\}, n_{ii} \leq v_i\}$$

Le cardinal de cet ensemble est borné par $(K + 1)^{N^2}$, il est donc fini.

Expression :

Pour déterminer l'expression de la probabilité stationnaire, on utilise les équations de trafic données par Kelly pour tout d'abord déterminer les α_{ij} (on se place dans le cas particulier d'un système à un individu, comme Kelly). On obtient les deux équations suivantes :

$$\forall i, \quad \alpha_{(ii)} \sum_{j \neq i} \Lambda_{(ii),(ij)} = \sum_{j \neq i} \alpha_{(ji)} \Lambda_{(ji),(ii)} \quad (1)$$

$$\forall i \neq j, \quad \alpha_{(ij)} \Lambda_{(ij),(ji)} = \alpha_{(ii)} \Lambda_{(ii),(ij)} \quad (2)$$

qui donnent :

$$\forall i, \quad \alpha_{(ii)} \sum_{j \neq i} \lambda_i p_{ij} = \sum_{j \neq i} \alpha_{(ji)} \lambda_{ji} \quad (3)$$

$$\forall i \neq j, \quad \alpha_{(ij)} \lambda_{ij} = \alpha_{(ii)} \lambda_i p_{ij} \quad (4)$$

en enfin, en considérant que $\sum_{j \neq i} p_{ij} = 1$ et en inversant i et j dans (4) :

$$\forall i, \quad \alpha_{(ii)} \lambda_i = \sum_{j \neq i} \alpha_{(ji)} \lambda_{ji} \quad (5)$$

$$\forall i \neq j, \quad \alpha_{(ji)} \lambda_{ji} = \alpha_{(jj)} \lambda_j p_{ji} \quad (6)$$

En injectant (6) dans (5) et en posant $\alpha_{ii} = \alpha_i$ on obtient :

$$\forall i, \quad \alpha_i = \frac{1}{\lambda_i} \sum_{j \neq i} \alpha_j \lambda_j p_{ji} \quad (7)$$

De plus, on a la contrainte suivante, sachant que $\forall i, \alpha_i > 0$ et avec $i \neq j, \alpha_{(ij)} > 0$:

$$\sum_i \alpha_i + \sum_i \sum_{j \neq i} \alpha_{(ij)} = 1 \quad (8)$$

qui devient, en utilisant (4)

$$\sum_i \alpha_i + \sum_i \sum_{j \neq i} \frac{\alpha_i \lambda_i p_{ij}}{\lambda_{ij}} = 1 \quad (9)$$

donc, on obtient, après inversion de i et j :

$$\sum_j \left(1 + \lambda_j \sum_{i \neq j} \frac{p_{ji}}{\lambda_{ji}} \right) \alpha_j = 1 \quad (10)$$

On peut maintenant réécrire (7) matriciellement, en utilisant (10) :

$$(P - I)\alpha = X \quad (11)$$

où P et la matrice dont les coefficients sont les $\left(\frac{\lambda_j p_{ji}}{\lambda_i} \right)_{i,j \in \{1, \dots, N\}}$.

Pour éviter d'obtenir la solution nulle, on remplace la première ligne de $(P - I)$ par $\left(1 + \lambda_j \sum_{i \neq j} \frac{p_{ji}}{\lambda_{ji}} \right)_{j \in \{1, \dots, N\}}$ et la première ligne de X par 1.

On obtient les α_i en résolvant ce système. Pour obtenir les α_{ij} , on utilise (4).

Notre probabilité stationnaire est finalement obtenue en utilisant la formule du Théorème 2.4 de Kelly :

$$\pi(n) = G_K^{-1} \prod_{i=1}^N \prod_{j=1}^N \frac{\alpha_{ij}^{n_{ij}}}{\prod_{r=1}^N \phi_{ij}(r)}, \quad n \in S \quad (12)$$

où G_N peut être obtenue en considérant que $\sum_{n \in S} \pi(n) = 1$.

2. CALIBRATION DES PARAMÈTRES

Nous calculons maintenant les valeurs des paramètres λ_i , λ_{ij} et q_{ij} à l'aide des données du réseau de Rouen.

Pour cela on dispose de deux fichiers contenant respectivement le nombre moyen de départs et d'arrivées à chacune des stations de Rouen et le temps de trajet moyen entre ces stations.

2.1. λ_i

Les λ_i correspondent à l'intensité des occurrences de la demande de vélos à la station i . On suppose qu'elles suivent une loi de Poisson, d'où $\mathbb{E}[X_i] = \lambda_i$. On les trouve à partir du nombre moyen de départs en une heure. Les valeurs obtenues sont disponibles dans le fichier **Calibration données Vlib - Feuille 1.pdf**.

2.2. λ_{ij}

Les λ_{ij} sont directement reliés au nombre de vélos qui quittent la station i pour se rendre à la station j par unité de temps. On les obtient à partir du tableau contenant l'ensemble des temps de trajets entre chacune des stations. On utilise le fait que les temps de trajets suivent une loi exponentielle $e(\lambda_{ij})$. On en déduit donc que $\mathbb{E}[X] = \frac{1}{\lambda_{ij}}$. D'où $\lambda_{ij} = \frac{1}{\mathbb{E}[X_{ij}]} = \frac{1}{\text{temps moyen pour aller de } i \text{ en } j}$ (Les λ_{ii} n'ont pas de valeur car il ne peut y avoir un trajet entre une station et elle-même). Les valeurs obtenues sont disponibles dans le fichier **Calibration données Vlib - Feuille 2.pdf**.

2.3. p_{ij}

Afin d'obtenir les valeurs des p_{ij} nous sommes partis du tableau contenant le nombre moyen de départs et d'arrivées dans chaque station du réseau. Nous avons construit le programme linéaire suivant :

$$\max \sum_{i=1}^N \sum_{j=1}^N p_{i,j}$$

sous les contraintes :

$$\left\{ \begin{array}{ll} \forall j \in \{1..N\}, & k_{min}Arr_j \leq \sum_{i=1}^N p_{i,j}d_i \leq k_{max}Arr_j \\ \forall i \in \{1..N\}, & \sum_{j=1}^N p_{i,j} = 1 \\ \forall i \in \{1..N\}, & p_{i,i} = 0 \\ \forall i, j \in \{1..N\} \text{ avec } i \neq j, & 0.01 \leq p_{i,j} \leq 1 \end{array} \right.$$

où :

- Arr_j est le nombre moyen d'arrivée à la station j et d_i est le nombre moyen de départ en i .
- La fonction objectif a été choisie de façon à maximiser les valeurs possibles de p_{ij} .
- k_{min} et k_{max} sont des constantes permettant de prendre en compte le fait qu'étant donné que les valeurs données ne sont pas des valeurs exactes mais des moyennes, l'équation $\forall j \in \{1..N\} \sum_{i=1}^N p_{i,j}d_i = Arr_j$ est fautive car la somme de tous les départs ne coïncident pas avec celles de toutes les arrivées.

Les résultats obtenus ne sont pas satisfaisant, en effet le modèle PL renvoie des solutions mettant la plupart des valeurs à 0.01 et seulement deux ou trois à des valeurs acceptables. Il aurait fallu avoir des valeurs plus précises sur les nombres de départs et d'arrivées afin d'obtenir des résultats plus probants.

2.4. Paramètres calculés à partir du modèle réduit à 5 stations

À partir du modèle à 5 stations, on calcule les différents paramètres dont notre modèle aura besoin pour simuler des trajectoires. Tous ces paramètres sont stockés dans un objet de type **MigrationProcess**. Voici son contenu lors de son initialisation :

```
In [2]: process = MigrationProcess(number_of_stations,
                                   number_of_bikes,
                                   initial_time,
                                   lambda_station_matrix,
                                   lambda_station_list_per_mins,
                                   lambda_itinerary_matrix,
                                   routing_matrix,
                                   station_size_list)

process.print_process()
```

Nombre de stations : 5

Nombre de vélos dans le réseau : 80

Temps initial : 0.0

Intensité des occurrences de demande de vélos à la station i (1/min) :

```
[ 0.04666667  0.06166667  0.09166667  0.05833333  0.07666667]
```

Intensité des trajets en vélo de la station i à la station j (1/min) :

```
[[ 0.          0.33333333  0.2          0.14285714  0.14285714]
 [ 0.5         0.          0.5         0.2         0.2         ]]
```



```

[ 0.25      0.5      0.      0.33333333  0.33333333]
[ 0.125     0.16666667 0.25    0.      0.5      ]
[ 0.14285714 0.14285714 0.2     0.5     0.      ]]
Matrice de routage :
[[ 0.   0.2  0.3  0.2  0.3 ]
 [ 0.2  0.   0.3  0.2  0.3 ]
 [ 0.2  0.25 0.   0.25 0.3 ]
 [ 0.15 0.2  0.3  0.   0.35]
 [ 0.2  0.25 0.35 0.2  0.  ]]
Nombre de places par station :
[24 20 20 15 20]
État courant (nombre de vélos par colonie),
généré aléatoirement à l'initialisation:
[[ 3.  3.  3.  5.  5.]
 [ 5.  3.  5.  3.  2.]
 [ 3.  3.  0.  2.  5.]
 [ 1.  7.  1.  2.  2.]
 [ 3.  1.  6.  2.  5.]]
Temps durant lequel chaque station est vide (nul au démarrage ...) :
[ 0.  0.  0.  0.  0.]

```

3. ÉTUDE DE LA MODÉLISATION

Avant de commencer l'étude, on remarque que les états initiaux sont générés aléatoirement par la fonction **init_state_matrix**, tout en respectant les différentes contraintes de capacité.

3.1. Étude avec 1 vélo

On commence par le cas trivial de 1 vélo en partage sur les 5 stations. L'espace des états est ici :

$$S = \{(n_{ij})_{i,j \in \{1, \dots, N\}} \in \{0, \dots, K\}^{N^2} : \sum_{i=1}^N \sum_{j=1}^N n_{ij} = 1 \wedge \forall i \in \{1, \dots, N\}, n_{ii} \leq v_i\}$$

Le vélo circule entre les stations et les itinéraires. Ainsi, il y a autant d'états que de possibilité de remplir une des colonies par un vélo donc N^2 états. Ici, on s'intéresse à une modélisation à 5 stations donc 25 états : $\{(1, 0, 0, \dots), (0, 1, 0, \dots), (0, 0, 1, \dots), \dots\}$. L'expression des α_{ij} a été obtenue dans la section **Probabilité stationnaire** plus haut.

Exprimons maintenant la probabilité stationnaire dans ce cas précis, en utilisant la formule donnée par Kelly.

Déterminons en premier le facteur G_1 . Soit $n = (n_{11}, \dots, n_{kl}, \dots, n_{NN})$ un état quelconque tel que le vélo se trouve dans la colonie n_{kl} . On trouve, après simplification de l'expression de la probabilité stationnaire de Kelly, $\pi(n) = G_1^{-1} \alpha_{kl}$.

$$\text{Ainsi, } \forall n \in S : \sum_{i=1}^N \sum_{j=1}^N \pi(n) = \sum_{i=1}^N \sum_{j=1}^N G_1^{-1} \alpha_{ij} = 1, \text{ donc } G_1 = \sum_{i=1}^N \sum_{j=1}^N \alpha_{ij}.$$

On sait que la probabilité qu'une colonie n_{kl} soit **pleine** est donnée par : $\pi(n) = G_1^{-1} \alpha_{kl}$, donc la probabilité qu'une colonie n_{kl} soit **vide** est donnée par $\pi(n) = 1 - G_1^{-1} \alpha_{kl}$

En utilisant cette formule, on obtient les **probabilités théoriques que chaque station soit vide** suivantes :

```
In [3]: number_of_bikes = 1
        process = MigrationProcess(number_of_stations,
                                   number_of_bikes,
                                   initial_time,
                                   lambda_station_matrix,
                                   lambda_station_list_per_mins,
                                   lambda_itinerary_matrix,
                                   routing_matrix,
                                   station_size_list)

        theory = process.compute_theoretical_proba_emptyness()
        theory
```

```
Out[3]: array([[ 0.82453272],
               [ 0.84449565],
               [ 0.86547538],
               [ 0.84433184],
               [ 0.84072577]])
```

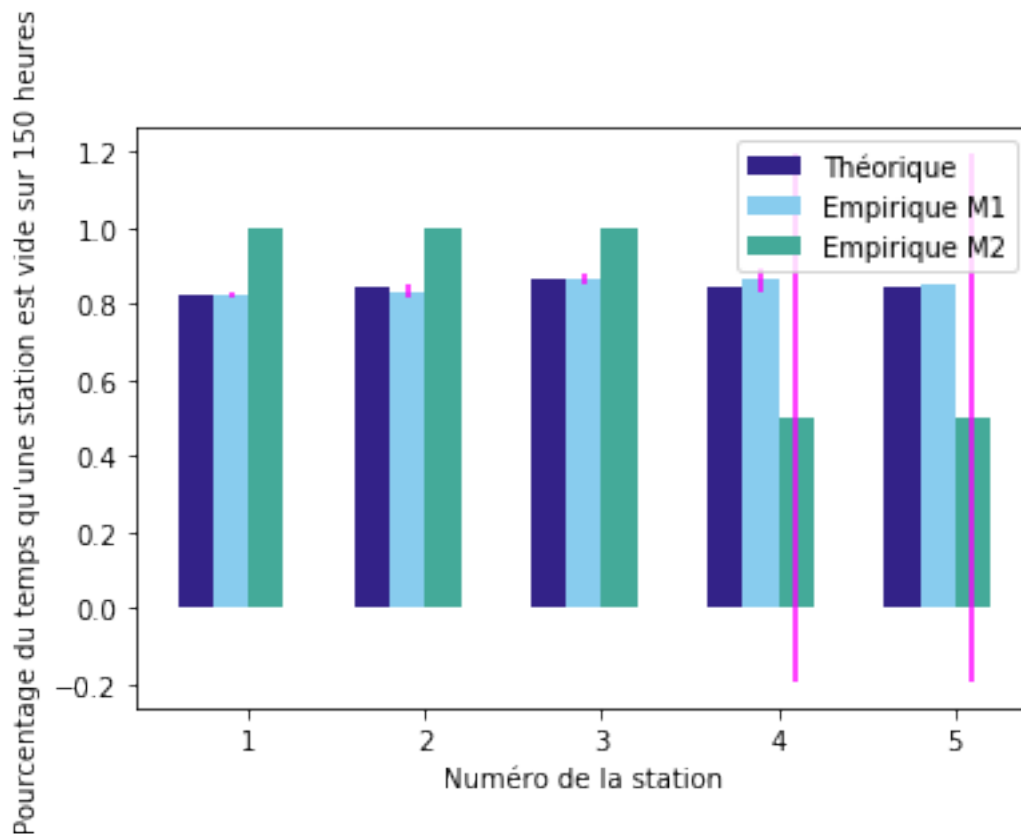
On calcule maintenant de manière empirique la probabilité que chaque station soit vide. On lance 1000 simulations. Cette probabilité est calculée de deux manières différentes :

- En calculant la moyenne du pourcentage d'itérations où chaque station est vide durant les 150 heures de chaque simulation (méthode M1)
- En faisant la moyenne du nombre de fois où chaque station se retrouve vide à la fin des 150 heures du processus, donc après un temps long (méthode M2)

On obtient les **probabilités empiriques que chaque station soit vide** suivantes pour chaque méthode, avec les écarts de l'intervalle de confiance à 95% associés. Les intervalles de confiance sont en rose sur le graphique.

```
In [4]: _ = MigrationProcess.estimate_time(1000,T_max,process,theory,True)
```

St°	P Théo	P Emp M1	IC 95% M1	P Emp M2	IC 95% M2
1	0.824533	0.832089	[-0.016055;+0.016055]	1.000000	[-0.000000;+0.000000]
2	0.844496	0.840269	[-0.002507;+0.002507]	0.800000	[-0.350615;+0.350615]
3	0.865475	0.867937	[-0.008783;+0.008783]	0.800000	[-0.350615;+0.350615]
4	0.844332	0.849095	[-0.009952;+0.009952]	0.800000	[-0.350615;+0.350615]
5	0.840726	0.833136	[-0.015467;+0.015467]	1.000000	[-0.000000;+0.000000]



On remarque que l'on obtient de **très bons résultats** pour la **méthode 1**, bien que les intervalles de confiance soient très **resserrés**. Nos probabilités obtenues empiriquement coïncident avec les probabilités obtenues théoriquement.

La **méthode 2** est de **moins bonne qualité**, ce qui était prévisible. Cependant, les intervalles de confiance sont **plus larges** ce qui permet de **coïncider** là encore avec les résultats obtenus **théoriquement** et de valider l'**ergodicité** du processus.

On peut remarquer que la durée indiquée de **150 heures** semblent être **convenable** pour atteindre la **probabilité stationnaire**.

3.2. Étude avec 100 vélos

On garde la même matrice de routage, les mêmes temps de transit, etc. En revanche, on suppose que l'on a 100 vélos dans le système.

L'espace des états est ici :

$$S = \{(n_{ij})_{i,j \in \{1, \dots, N\}} \in \{0, \dots, K\}^{N^2} : \sum_{i=1}^N \sum_{j=1}^N n_{ij} = 100 \wedge \forall i \in \{1, \dots, N\}, n_{ii} \leq v_i\}$$

Cette fois ci, on remarque qu'il va être très difficile d'obtenir des résultats théoriques car **l'espace d'états est immense**, l'étude de la complexité va nous le confirmer.

On calculer par simulation la probabilité stationnaire pour chaque station d'être vide, en précisant les intervalles de confiance :

```
In [5]: number_of_bikes = 100
        process = MigrationProcess(number_of_stations,
                                   number_of_bikes,
                                   initial_time,
                                   lambda_station_matrix,
                                   lambda_station_list_per_mins,
                                   lambda_itinerary_matrix,
                                   routing_matrix,
                                   station_size_list)

        _ = MigrationProcess.estimate_time(1, T_max, process, None, True)
```

St°	P	Emp	M1	IC 95% M1	P	Emp	M2	IC 95% M2
1	0.000000			[-0.000000; +0.000000]	0.000000			[-0.000000; +0.000000]
2	0.000000			[-0.000000; +0.000000]	0.000000			[-0.000000; +0.000000]
3	0.000000			[-0.000000; +0.000000]	0.000000			[-0.000000; +0.000000]
4	0.000000			[-0.000000; +0.000000]	0.000000			[-0.000000; +0.000000]
5	0.000000			[-0.000000; +0.000000]	0.000000			[-0.000000; +0.000000]

La capacité totale de nos stations est de $24 + 20 + 20 + 15 + 20 = 99$ vélos. Avec 100 vélos dans le système, les stations sont très souvent remplies et quasiment jamais vide, car il est compliqué de vider une colonie avec une population initiale très importante et une matrice de routage avec des coefficients assez homogènes. Il est donc logique que la probabilité stationnaire recherchée soit presque nulle pour toutes les stations.

Étude de la complexité :

Notons V_i l'ensemble des états où la station i est vide. Pour 100 vélos et 5 stations, on cherche quelle serait la complexité du calcul de $\pi(V_1)$ en termes de nombres d'opérations. Pour simplifier le calcul, on suppose que les stations ont une capacité infinie, tout comme les itinéraires. On peut donc placer tout les vélos sur une station ou un itinéraire.

On sait que le nombre de façon de répartir k éléments indiscernables dans n ensembles discernables, est donné par :

$$C_k^{n+k-1} = \frac{(n+k-1)!}{k!(n-1)!}$$

Dans notre cas, $k = 100$ et $n = 25$, cela donne : $C_{100}^{124} = 26010968307696038491182501 = 2.6010968 * 10^{25}$.

On rappelle :

$$\pi(n) = G_{100}^{-1} \prod_{i=1}^N \prod_{j=1}^N \frac{\alpha_{ij}^{n_{ij}}}{\prod_{r=1}^{n_{ij}} \phi_{ij}(r)}, \quad n \in S \quad (13)$$

L'objectif est de calculer $\pi(V_1)$. Pour cela, il faut déterminer la constante de normalisation G_{100}^{-1} qui nécessite de faire la somme des probabilités de tous les états, donc C_{100}^{124} sommes.

Chaque somme contient N^2 produits d'au moins 100 produits au dénominateur car $\sum_{i=1}^N \sum_{j=1}^N n_{ij} = 100$.

On peut donc estimer que le nombre d'opérations à réaliser est majoré par $100 * 25 * C_{100}^{124}$. On peut trivialement minorer le nombre d'opérations par $100 * 25 * C_{100}^{119}$ en considérant que les stations doivent rester vide. L'énoncé demande une estimation logarithmique en base 10 de ce nombre pour avoir une idée du nombre de chiffre. On obtient que le nombre d'opérations a entre 25 et 28 chiffres, ce qui est très conséquent !

On considère une machine ayant une cadence théorique de 10 milliards de FLOPS (donc réalisant 10^{10} opérations par secondes), ce qui est classique pour un PC de nos jours.

Pour réaliser 10^{24} opérations (nombre à 25 chiffres), elle mettra approximativement $T = \frac{10^{24}}{(10^{10} * 60 * 60 * 24 * 365)} \approx 3$ millions d'années ...

4. ÉTUDE DE L'IMPACT DES PARAMÈTRES

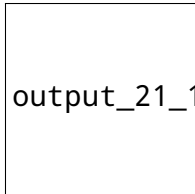
Il pourrait être intéressant de faire varier certains paramètres afin d'observer leurs impacts sur la probabilité stationnaire qu'une station soit vide.

4.1. Impact du nombre de vélos dans le système sur la probabilité stationnaire

Regardons tout d'abord l'impact du nombre de vélos initial sur le système. On considère un répartition aléatoire des vélos au lancement du processus. On simule une expérience par nombre de vélos, en utilisant la méthode 1 pour estimer la probabilité qu'une station soit vide. Chaque expérience est lancée une fois. Dans l'idéal, il faudrait les lancer bien plus de fois mais cela prend beaucoup de temps de calcul.

In [6]: `MigrationProcess.bike_number_impact(10)`

Progress : 100 %



output_21_1.png

On remarque que plus le nombre de vélos est important dans le système, plus la probabilité qu'une station soit vide est faible. De plus, plus il y a de vélos dans le système, plus les probabilités stationnaires entre les différentes stations se creuse. On observe ainsi que c'est la station 3 qui a le plus de chance d'être vide et que cette remarque devient de plus en plus flagrante plus le nombre de vélos dans le système est important.

4.2. Impact de la durée du processus sur l'obtention de la probabilité stationnaire

Regardons maintenant l'impact de la durée du processus sur l'obtention de la probabilité stationnaire avec 1 vélo. On rappelle qu'après 150 heures, la probabilité stationnaire semblaient être atteinte vu les résultats précédents. Chaque expérience est lancée 1000 fois. On regarde l'écart à la probabilité stationnaire en fonction du temps.

```
In [ ]: MigrationProcess.process_duration_impact(50,theory)
```

Progress : 31 %

On remarque que la probabilité stationnaire est atteinte assez rapidement. On observe qu'autour de 20 heures de processus, les différentes probabilités ne varient presque plus .

```
In [ ]:
```