Partie_2

January 21, 2025

[1]:
```
## Imports

import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as resol
```

1 Projet mathématique - Partie II

1.1 Chaînes de Markov et temps d'arrêt

Lien du sujet : https://drive.google.com/file/d/1EPzGk_UQ5y2bjMI8iTgFYmfPHWRwlQMf/view

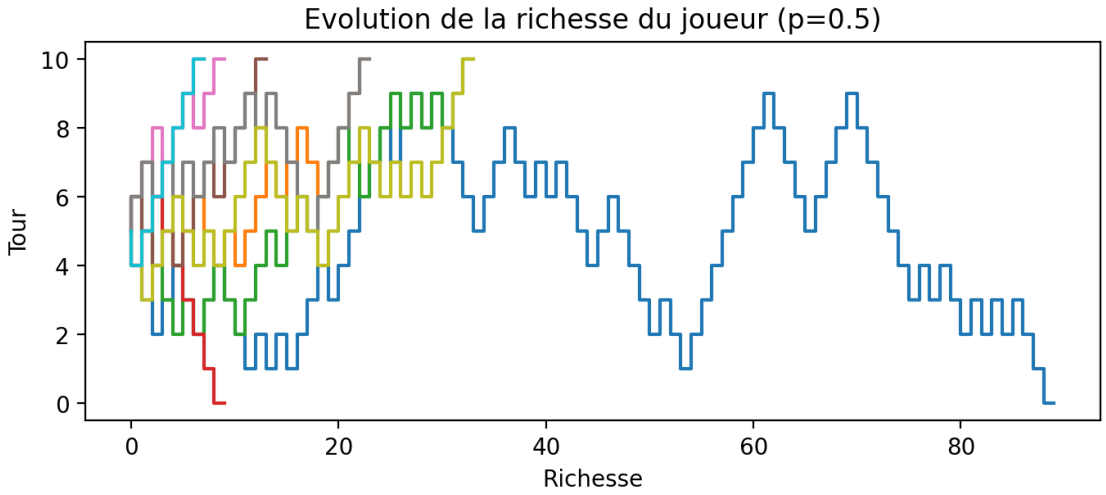
[2]:
```
def sim_dis(p, x, M):
    y=np.zeros(M, dtype=int)
    for k in range(M):
        u=np.random.uniform(0,1)
        S,i=0,0
        while S<u:
            S+=p[i]
            i+=1
        j=i-1
        y[k] = x[j]
    return y
```

1.1.1 Question 1

1.a

[3]:
```
N = 100
X0 = 5
a = 10
prob = 0.5
M = 1000

fig = plt.figure(figsize=(8,3),dpi=200)

def ruine_du_joueur(N, X0, a, p, to_print=False) :
  # Initialisation
  n = 0 #nombre de tours
```

```
  Xn = X0 #argent du joueur à l'étape n
  L_Xn = [X0]
  # Tours
  while (Xn!=0 and Xn!=a and n<N) :
    n+=1
    Xn += sim_dis([1-p,p], [-1,1], 1)[0]
    L_Xn.append(Xn)
  if to_print :
    plt.step(range(n+1),L_Xn)
  return [n,Xn]

L = []
sum = 0
for i in range (M) :
  L.append(ruine_du_joueur(N, X0, a, prob, i<10))
  sum+=(L[i][1]==a)
plt.xlabel("Richesse")
plt.ylabel("Tour")
plt.title("Evolution de la richesse du joueur (p="+str(prob)+")")
plt.show()
p = sum / M
print("p =",p)
```




```
p = 0.469
```

Soit $(X_n)_{n\in\mathbb{N}}$ la chaîne de Markov décrivant la richesse du joueur dans le jeu la ruine du joueur .

Soit le temps d'arrêt $\tau_{0,a}$ défini tel que :

$$\tau_{0,a} = \min\{n \geqslant 0 \mid X_n = a \text{ ou } X_n = 0\}$$

On supposera ici que $\mathbb{P}(\tau_{0,a} < \infty) = 1$, et on définit pour tout $x \in \{0, ..., a\}$:

$$f(x) = \mathbb{P}(X_{\tau_{0,a}} = a | X_0 = x)$$

On sait que $f(0) = 0$ et que $f(a) = 1$, et que pour tout entier i tel que $0 < i < a$, on a :

$$\begin{aligned}
f(i) &= \mathbb{P}(X_{\tau_{0,a}} = a \,|\, X_0 = i) \\
&= \sum_{j=0}^{a} \mathbb{P}(X_{\tau_{0,a}} = a \,|\, X_0 = i, X_1 = j)\, \mathbb{P}(X_1 = j \,|\, X_0 = i) && \text{(loi des probabilités totales)} \\
&= \sum_{j=0}^{a} f(j)\, p(i,j) && \text{(propriété de Markov)} \\
&= \sum_{j=1}^{a-1} f(j)\, p(i,j) \; + p(i,a) && \text{(car } f(0) = 0 \text{ et } f(a) = 1\text{)}
\end{aligned}$$

D'où le système suivant :

$$\boxed{\forall i \in \mathbb{N},\, 0 < i < a, \quad f(i) = p(i,a) + \sum_{j=1}^{a-1} f(j)\, p(i,j)}$$

On peut alors écrire ce système sous forme matricielle, en notant $b = (p(i,a))_{0<i<a}$, $x = (f(i))_{0<i<a}$ et $M = (p(i,j))_{0<i,j<a}$:

$$\begin{aligned}
&\forall i \in \mathbb{N},\, 0 < i < a, \quad f(i) = p(i,a) + \sum_{j=1}^{a-1} f(j)\, p(i,j) \\
\iff \quad & x = b + Mx \\
\iff \quad & \boxed{[I_{a-1} - M]\, x = b}
\end{aligned}$$

Or, $P = (p(i,j))_{0 \le i,j \le a}$ est une matrice stochastique à coefficients non nuls (déjà vu dans la première partie du projet). Donc, on peut en déduire que la matrice $A = [I_{a-1} - M]$ est **une matrice à diagonale strictement dominante**.

En effet, on a :

$$\forall i \in [1, a-1], \quad a_{i,i} = 1 - p(i,i) \qquad \text{et} \qquad \sum_{\substack{j=1 \\ j \neq i}}^{a-1} a_{i,j} = \sum_{\substack{j=1 \\ j \neq i}}^{a-1} p(i,j)$$

Et par définition de P, comme $\displaystyle\sum_{j=0}^{a} p(i,j) = 1$ avec $p(i,0), p(i,a) > 0$, on en déduit que :

$$\begin{aligned}
&\forall i \in [1, a-1], \quad 1 - p(i,i) = \sum_{\substack{j=0 \\ j \neq i}}^{a} p(i,j) > \sum_{\substack{j=1 \\ j \neq i}}^{a-1} p(i,j) \\
\implies \quad &\forall i \in [1, a-1], \quad |a_{i,i}| > \sum_{\substack{j=1 \\ j \neq i}}^{a-1} a_{i,j}
\end{aligned}$$

Donc, d'après le **lemme d'Hadamard**, on en déduit que A **est inversible**. Donc, **notre système admet une unique solution**.\

Le système matricielle peut être facilement résolue grâce à Python :

[4]:
```
def resolv(a,p) :
  # Matrice M
  M = np.zeros([a-1,a-1])
  for i in range (a-2) :
      M[i,i+1]=p
      M[i+1,i]=1-p
  A = np.eye(a-1) - M
  # Vecteur b
  b = [0 for i in range(a-1)]
  b[a-2] = p
  b =  np.array(b)
  # Résolution
  return np.linalg.solve(A, b)
x = resolv(a,prob)
print(x)
```

```
[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]
```

Dans le cadre fixé par le sujet, c'est-à-dire $a = 10$, $p = 0.5$, on a :

$$\forall x \in [1, a-1], \quad \mathbb{P}(X_{\tau_{0,a}} = a \,|\, X_0 = x) = \frac{x}{a}$$

Avec évidemment $\mathbb{P}(X_{\tau_{0,a}} = a \,|\, X_0 = 0) = 0$ et $\mathbb{P}(X_{\tau_{0,a}} = a \,|\, X_0 = a) = 1$.

Nous venons donc de déterminer la *vraie* probabilité associée à $\mathbb{P}(X_{\tau_{0,a}} = a \,|\, X_0 = 5)$, que l'on note ici p^*.

Et, empiriquement, nous avons déterminé une estimation $\widetilde{p}$ de $\mathbb{P}(X_{\tau_{0,a}} = a \,|\, X_0 = 5)$. Pour cela, on a simulé M trajectoires que l'on note : $\forall i \in [1, M]$, $X_\tau^{(i)}$. Et on s'intéresse alors à :

$$\widetilde{p} = \frac{\#\left(\text{trajectoires } i \text{ pour lesquelles } \tau \leq N \text{ et } X_\tau^{(i)} = a\right)}{M}$$

On peut alors réécrire $\widetilde{p}$ tel que :

$$\widetilde{p} = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}_{\{X_\tau^{(i)} = a\}}$$

On note pour simplifier : $\forall i \in [1, M]$, $p_i \triangleq \mathbb{1}_{\{X_\tau^{(i)} = a\}}$. Or, par définition de l'indicatrice, les $(p_i)_{1 \leq i \leq M}$ suivent alors chacun une loi de Bernoulli de paramètre p^* puisque :

$$\forall i \in [1, M], \quad \mathbb{E}[p_i] = \mathbb{E}\left[\mathbb{1}_{\{X_\tau^{(i)} = a\}}\right] = \mathbb{P}\left(X_\tau^{(i)} = a\right) = p^*.$$

Et donc,

$$\forall i \in [1, M], \quad p_i \sim \mathcal{B}(1, p^*) \Longrightarrow \begin{cases} \mathbb{E}[p_i] = p^* < \infty \\ \text{Var}(p_i) = p^*(1-p^*) < \infty \end{cases}$$

Finalement, $\$ \widetilde{p}\$estlamoyenneempiriquedeM$ variables i.i.d de Bernoulli (les p_i). Donc, d'après le **Théorème Central Limite**, on a :

$$\boxed{\sqrt{M}\,\frac{\widetilde{p} - p^*}{\sqrt{p^*(1-p^*)}} \xrightarrow{\mathcal{L}} \mathcal{N}(0,1)}$$

Seulement, nous voulons déterminer un intervalle de confiance à 95% de notre estimation, pour ensuite vérifier si p^* se trouve dans cet intervalle.

Si on ne change rien, notre intervalle de confiance va dépendre de p^*, ce qui n'est pas du tout souhaitable. On va donc utiliser **le théorème de Slutsky** pour se ramener à un intervalle de confiance indépendant de p^*.

Premièrement, comme $p_1, p_2, \ldots, p_M$ sont des variables aléatoires i.i.d, **la loi forte des grands nombres** nous assure que :

$$\widetilde{p} = \frac{1}{M}\sum_{i=1}^{M} p_i \xrightarrow{p.s.} \mathbb{E}[p_1] = p^*$$

Et donc, on a nécessairement le résultat plus faible suivant :

$$\widetilde{p} \xrightarrow{\mathbb{P}} p^*$$

Ainsi, en appliquant le théorème de continuité avec la fonction $x \longmapsto \frac{1}{\sqrt{x(1-x)}}$ sur $]0,1[$, on a :

$$\boxed{\frac{1}{\sqrt{\widetilde{p}\,(1-\widetilde{p})}} \xrightarrow{\mathbb{P}} \frac{1}{\sqrt{p^*(1-p^*)}}}$$

Par conséquent, d'après le **théorème de Slutsky**, on en déduit que :

$$\boxed{\sqrt{M}\,\frac{\widetilde{p} - p^*}{\sqrt{\widetilde{p}\,(1-\widetilde{p})}} \xrightarrow{\mathcal{L}} \mathcal{N}(0,1)}$$

On en déduit donc, en notant u_α les quantiles d'ordre α pour la loi $\mathcal{N}(0,1)$, que :

$$\mathbb{P}\left(u_{\frac{\alpha}{2}} \leq \sqrt{M}\,\frac{\widetilde{p} - p^*}{\sqrt{\widetilde{p}\,(1-\widetilde{p})}} \leq u_{1-\frac{\alpha}{2}}\right) = 1-\alpha$$

$$\Longrightarrow \quad \mathbb{P}\left(\sqrt{\frac{\widetilde{p}\,(1-\widetilde{p})}{M}}u_{\frac{\alpha}{2}} \leq \widetilde{p} - p^* \leq \sqrt{\frac{\widetilde{p}\,(1-\widetilde{p})}{M}}u_{1-\frac{\alpha}{2}}\right) = 1-\alpha$$

Or, on cherche un intervalle de confiance à 95%. Et, on sait que pour $\mathcal{N}(0,1)$, on a : $u_{0,975} \approx 1,96$. On en déduit donc l'intervalle de confiance de notre estimation $\widetilde{p}$:

$$\boxed{\text{IC} = \left[\widetilde{p} - 1,96\sqrt{\frac{\widetilde{p}\,(1-\widetilde{p})}{M}}\,;\,\widetilde{p} + 1,96\sqrt{\frac{\widetilde{p}\,(1-\widetilde{p})}{M}}\right]}$$

[5]:
```
print("Proba empirique =",p)
```

```
Proba empirique = 0.469
```

La valeur trouvée est très proche de la probabilité théorique qui est égale à 0.5. On va pouvoir vérifier cela à l'aide d'un intervalle de confiance à 95% de notre estimation.

[6]:
```
IC = [p -1.96 * np.sqrt(p*(1 - p)/M), p + 1.96 * np.sqrt(p*(1 - p)/M)]
print("Intervalle de confiance =",IC)
```

```
Intervalle de confiance = [0.43806929967815145, 0.4999307003218485]
```

Or, on a avait trouvé que $p^* = P(X_{\tau_{0,a}}=a \,|\, X_0=5) = 0.5$.*Ainsi, on remarque effectivement que* : $\boxed{p^* \in \text{IC}}$

1.b

[7]:
```
moy = np.mean([L[i][0] for i in range (M)])
print("moy =",moy)
```

```
moy = 24.181
```

On rappelle que le temps d'arrêt $\tau_{0,a}$ est défini tel que:

$$\tau_{0,a} = \min\{n \geqslant 0 \mid X_n = a \text{ ou } X_n = 0\}$$

Le théorème énonce alors qu'en posant $\forall i \in [0,a],\ g(i) = \mathbb{E}\left[\tau_{0,a} \,|\, X_0 = i\right]$, $(g(i))_{0\leq i\leq a}$ est la plus petite solution positive du système suivant :

$$\boxed{\begin{cases} g(i) = 0 & \text{si } i \in \{0,a\} \\ g(i) = 1 + \displaystyle\sum_{j=1}^{a-1} p(i,j)\, g(j) & \text{sinon} \end{cases}}$$

On peut alors écrire ce système sous forme matricielle, en notant $b = (1)_{0<i<a}$, $x = (g(i))_{0<i<a}$ et $M = (p(i,j))_{0<i,j<a}$:

$$\begin{aligned} \forall i \in \mathbb{N},\ 0 < i < a, \quad & g(i) = 1 + \sum_{j=1}^{a-1} p(i,j)\, g(j) \\ \Longrightarrow \quad & x = b + Mx \\ \Longrightarrow \quad & \boxed{[I_{a-1} - M]\, x = b} \end{aligned}$$

On remarque, de la même manière d'après le **lemme d'Hadamard**, la matrice $A = I_{a-1} - M$ est inversible, et on peut alors déterminer les différents valeurs des $g(i)$ dans le cadre de notre exemple qui est le jeu "ruine du joueur".

[8]:
```
def resolv_esperance(a,p) :
  # Matrice M
  M = np.zeros([a-1,a-1])
  for i in range (a-2) :
      M[i,i+1]=p
      M[i+1,i]=1-p
  A = np.eye(a-1) - M
  # Vecteur b
  b = np.array([1 for i in range(a-1)])
  # Résolution
  return np.linalg.solve(A, b)
x = resolv_esperance(a,prob)
print(x)
```

```
[ 9. 16. 21. 24. 25. 24. 21. 16.  9.]
```

On trouve alors : $\$\mathrm{E}[\tau_{0,a} \,|\, X_0 = 5] = 25\$$.

De la même manière qu'à la question 1.a, nous venons de déterminer la *vraie* probabilité associée à $\mathbb{E}\left[\tau_{0,a} \,|\, X_0 = 5\right]$, que l'on note ici θ^*.

Notre estimateur est ici une moyenne empirique :

$$\hat{\theta} = \overline{\tau_{0,a}} = \frac{1}{M} \sum_{i=1}^{M} \tau_{0,a}^{(i)}$$

Donc, on a bien : $\mathbb{E}[\hat{\theta} \,|\, X_0 = 5] \underset{\text{not.}}{=} \mathbb{E}_5[\hat{\theta}] = \mathbb{E}_5[\tau_{0,a}^{(1)}] = 25$ en théorie (par identique distribution et linéarité de l'espérance).

Il est nécessaire de trouver $\mathrm{Var}_5[\hat{\theta}]$ pour pouvoir par la suite utiliser le Théorème Central Limite.

Pour cela, on pose :

$$\widehat{S^2} = \frac{1}{M-1} \sum_{i=1}^{M} \left(\tau_{0,a}^{(i)} - \overline{\tau_{0,a}} \right)^2$$

Et il se trouve que $\widehat{S^2}$ est un estimateur sans biais convergent vers $\mathrm{Var}_5[\hat{\theta}]$.

Ainsi, on peut alors appliquer le Théorème Central Limite :

$$\boxed{\sqrt{M} \frac{\hat{\theta} - \theta^*}{\sqrt{\widehat{S^2}}} \xrightarrow{\mathcal{L}} \mathcal{N}(0,1)}$$

On en déduit donc, en notant u_α les quantiles d'ordre α pour la loi $\mathcal{N}(0,1)$, que :

$$\mathbb{P}\left(u_{\frac{\alpha}{2}} \leq \sqrt{M}\,\frac{\hat{\theta}-\theta^*}{\sqrt{\widehat{S^2}}} \leq u_{1-\frac{\alpha}{2}}\right) = 1-\alpha$$

$$\Longrightarrow \quad \mathbb{P}\left(\sqrt{\frac{\widehat{S^2}}{M}}u_{\frac{\alpha}{2}} \leq \hat{\theta}-\theta^* \leq \sqrt{\frac{\widehat{S^2}}{M}}u_{1-\frac{\alpha}{2}}\right) = 1-\alpha$$

Or, on cherche un intervalle de confiance à 95%. Et, on sait que pour $\mathcal{N}(0,1)$, on a : $u_{0,975} \approx 1,96$. On en déduit donc l'intervalle de confiance de notre estimation $\hat{\theta}$:

$$\boxed{\text{IC} = \left[\hat{\theta} - 1,96\sqrt{\frac{\widehat{S^2}}{M}};\, \hat{\theta} + 1,96\sqrt{\frac{\widehat{S^2}}{M}}\right]}$$

Et on peut donc le calculer grâce à Python. En premier lieu, on détermine la valeur de $\widehat{S^2}$.

[9]:
```
S2 = (1/(M-1)) * np.sum([(L[i][0] - moy)**2 for i in range(M)])
print("Variance empirique =",S2)
```

```
Variance empirique = 343.60984884884886
```

On peut alors déterminer notre intervalle de confiance IC :

[10]:
```
print("Espérance empirique =",moy)

IC = [moy - 1.96*np.sqrt(S2/M), moy + 1.96*np.sqrt(S2/M)]
print("Intervalle de confiance à 95% =",IC)
```

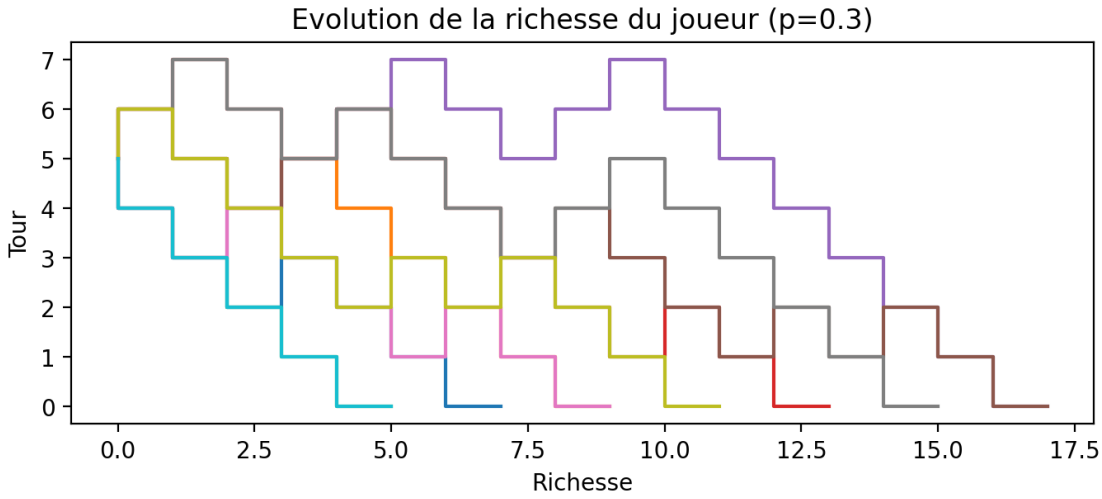
```
Espérance empirique = 24.181
Intervalle de confiance à 95% = [23.032082424480443, 25.32991757551956]
```

Or, on a avait trouvé que : $$\vartheta^* = \mathrm{E}[\tau_{0,a} \,|\, X_0 = 5] = 25. Ainsi, on remarque effectivement que : $$ $\boxed{\theta^* \in \text{IC}}$ $$

1.c On choisit arbitrairement $p = 0.3\,(\neq 0.5)$.

[11]:
```
prob = 0.3

N = 100
X0 = 5
a = 10
M = 1000

fig = plt.figure(figsize=(8,3),dpi=200)

L = []
sum = 0
for i in range (M) :
```

```
  L.append(ruine_du_joueur(N, X0, a, prob, i<10))
  sum+=(L[i][1]==a)
plt.xlabel("Richesse")
plt.ylabel("Tour")
plt.title("Evolution de la richesse du joueur (p="+str(prob)+")")
plt.show()
p = sum / M
print("Probabilité empirique =",p)

moy = np.mean([L[i][0] for i in range (M)])
print("Moyenne empirique =",moy)
```




```
Probabilité empirique = 0.011
Moyenne empirique = 12.436
```

De la même manière qu'en 1.a, on peut obtenir les valeurs exactes des $\mathbb{P}(X_{\tau_{0,a}} = a | X_0 = x)$ en résolvant un système linéaire.

[12]:
```
x = resolv(a,prob)
print("f( 0 ) =",0)
for i in range (1,a) :
  print("f(",i,") =",x[i-1])
print("f( a ) =",1)
```

```
f( 0 ) = 0
f( 1 ) = 0.0002787800416548339
f( 2 ) = 0.000929266805516113
f( 3 ) = 0.0024470692545257642
f( 4 ) = 0.00598860830221495
f( 5 ) = 0.01425219941348972
f( 6 ) = 0.03353391200646419
```

```
f( 7 ) = 0.07852457472340463
f( 8 ) = 0.183502787729599
f( 9 ) = 0.4284519514107193
f( a ) = 1
```

[13]:
```
print("Probabilité empirique =",p)
print("La vraie probabilité vaut :",np.round(x[4],5))
```

```
Probabilité empirique = 0.011
La vraie probabilité vaut : 0.01425
```

On peut aussi déterminer la vraie valeur de la moyenne des temps $\mathbb{E}[\tau_{0,a} \,|\, X_0 = x]$ pour tout $x \in \{1, \ldots, 10\}$

[14]:
```
e = resolv_esperance(a,prob)
print(e)
```

```
[ 2.4930305   4.97676833  7.43882327  9.85028479 12.14369501 14.1616522
 15.53688563 15.41243031 11.78870121]
```

[15]:
```
print("Moyenne empirique =",moy)
print("La vraie moyenne de temps d'atteinte vaut :",e[4])
```

```
Moyenne empirique = 12.436
La vraie moyenne de temps d'atteinte vaut : 12.143695014662748
```

1.1.2 Question 2

Dans un premier temps, on va construire aléatoirement la matrice stochastique P qui régira les déplacements des élèves et du monstre dans les 10 zones.

[16]:
```
def matrice_stochastique(n, m):
  # On crée une matrice aléatoire de taille n x m
  P = np.random.rand(n, m)
  # On divise chaque coefficient de P par la somme des coefficients de sa ligne
  P /= np.sum(P, axis=1, keepdims=True)
  return P

P = matrice_stochastique(10,10)
print(P)
```

```
[[1.54668955e-01 7.92329018e-02 1.66716121e-01 1.58243660e-01
  2.81642569e-02 1.79511347e-01 5.76302724e-02 6.87650725e-02
  1.59150098e-02 9.11524035e-02]
 [2.25483948e-02 1.51517578e-01 1.09836042e-01 2.51396503e-02
  1.40179156e-01 1.17257671e-01 6.54884464e-02 8.15796756e-02
  1.28632965e-01 1.57820422e-01]
 [9.69318840e-02 1.93570761e-01 3.59482136e-02 1.90501821e-01
  1.27232995e-01 1.49696913e-01 7.88612125e-02 4.50236261e-02
  1.09046466e-03 8.11421094e-02]
 [1.76754093e-01 3.42063737e-02 4.74052781e-05 1.36526738e-01
```

```
 1.49275915e-01 4.23367969e-02 2.58086354e-02 6.02000244e-02
 1.81036520e-01 1.93807499e-01]
[1.79363633e-01 8.05090662e-02 7.41490500e-02 8.45604000e-02
 1.44024003e-01 1.42748882e-01 8.98917897e-02 9.95005874e-02
 2.08734023e-03 1.03165249e-01]
[4.46093205e-02 1.10837823e-01 1.67028355e-01 4.66108060e-02
 2.01085205e-01 2.10521662e-01 4.31476672e-02 1.52636418e-02
 6.95780326e-04 1.60199740e-01]
[2.07382481e-02 1.19873742e-01 2.29006093e-01 1.85591151e-01
 6.38969771e-02 7.95708796e-02 5.66973544e-04 4.83612040e-02
 1.42472167e-01 1.09922566e-01]
[1.54178214e-01 1.46187888e-01 1.42555851e-01 1.67714744e-01
 1.98576463e-02 4.64599023e-02 3.09488270e-02 1.66682885e-01
 3.39414510e-02 9.14725904e-02]
[7.31245338e-02 8.22907526e-02 6.46874996e-02 2.18955304e-01
 5.94625607e-02 2.77746643e-02 1.73103964e-01 1.54860829e-01
 5.41275211e-02 9.16123720e-02]
[9.16153452e-02 1.18978969e-01 5.55659058e-02 1.23222434e-01
 1.21573768e-01 1.04384621e-01 1.10965964e-01 6.51746816e-02
 1.05020987e-01 1.03497325e-01]]
```

2.a

[17]:
```
N = 10000

def simulation_etudiant_compteur_zone(P, X0, N):
  # Initialisation
  Xn = X0 #zone actuelle
  L_zone = [0 for i in range (10)] #compteur de passages dans une zone
  L_zone[X0-1]+=1
  # Tours
  for i in range(N):
    Xn = sim_dis(P[Xn-1], range(1,11), 1)[0] #déplacement de l'élève
    L_zone[Xn-1]+=1
  return L_zone

L_zone = simulation_etudiant_compteur_zone(P, 1, N)
p_zone = [L_zone[i]/N for i in range (10)]
print("Fraction de temps passée par l'étudiant dans chaque zone :",p_zone)

ecart = np.dot(p_zone,P) - p_zone
erreur = 0
for i in range (10) :
  erreur += abs(ecart[i])
print("Valeur de l'erreur =",erreur)
```

```
Fraction de temps passée par l'étudiant dans chaque zone : [0.1052, 0.1034,
0.0997, 0.1308, 0.117, 0.1117, 0.0645, 0.0755, 0.0695, 0.1228]
Valeur de l'erreur = 0.02681169639091696
```

Soit $\pi = (\pi_i)_{1\leq i\leq 10}$ la solution de l'équation $\pi^\top P = \pi^\top$ avec $\pi_i \in [0,1]$ et $\sum_{i=1}^n \pi_i = 1$. Un tel vecteur s'appelle une probabilité invariante pour P.

Or,

$$\begin{aligned} \pi^\top P = \pi^\top \quad &\iff \quad (\pi^\top P)^\top = (\pi^\top)^\top \\ &\iff \quad P^\top \pi = \pi \\ &\iff \quad \pi \text{ est un vecteur propre associé à la valeur propre 1 de } P^\top \end{aligned}$$

Et le fait que $\pi_i \in [0,1]$ et $\sum_{i=1}^n \pi_i = 1$ nous permet de conclure que π est **le vecteur propre stochastique** de la matrice $P^\top$ associé à la valeur propre 1.

\

À partir de maintenant, on note $A = P^\top$, avec P notre matrice stochastique à coefficients strictement positifs. Montrons pour commencer ce premier résultat :

$$\boxed{1 \text{ est valeur propre de } P \text{ et toute valeur propre complexe } \lambda \text{ de } P \text{ vérifie } |\lambda| \leq 1.}$$

\

En effet, soit $U = (1)_{1\leq i\leq 10}$. Alors, comme P est une matrice stochastique, on sait que : $\sum_{j=1}^{10} p_{i,j} = 1$, ce qui équivaut à dire que : $PU = U$.

Cela prouve que 1 est bien valeur propre de P, et U est un vecteur propre associé. À noter que 1 est donc aussi valeur propre de $A = P^\top$. \

Soit $\lambda \in \mathrm{Sp}(P)$, et soit $X = (x_i)_{1\leq i\leq 10}$ un vecteur propre associé. Soit $i \in [1,10]$ tel que $|x_i| = \max_{1\leq k\leq 10} |x_k|$. Comme par définition $PX = \lambda X$, en regardant la i-ième coordonnée, on obtient :

$$p_{i,1}\, x_1 + \cdots + p_{i,10}\, x_{10} = \lambda x_i$$

En passant au module, on obtient donc que :

$$|\lambda x_i| = |\lambda||x_i| = |p_{i,1}\, x_1 + \cdots + p_{i,10}\, x_{10}| \leq (p_{i,1} + \cdots + p_{i,10})|x_i| = |x_i|$$

On en conclut donc que : $|\lambda| \leq 1$.

\

Il est alors temps d'utiliser **le théorème de Perron-Frobenius**. Bien qu'ici, le théorème de Perron nous suffira.

À noter que l'écriture la plus connue de ce théorème concerne les matrices réelles primitives, mais on s'intéressera ici uniquement aux matrices réelles strictement positives (pour simplifier un peu).

> Soit A une matrice réelle strictement positive. Son rayon spectral $\rho(A)$ est une valeur propre simple et dominante (i.e. de module strictement supérieur à celui des autres valeurs propres). Elle admet un vecteur propre strictement positif. [source]

Or, dans notre cas, on vient de montrer que : $\forall \lambda \in \mathrm{Sp}(P),\ |\lambda| \leq 1$, et que 1 est bien valeur propre de P. Donc,

$$\rho(P) = 1 \quad \Longrightarrow \quad \rho(A) = 1$$

Donc, d'après le théorème de Perron appliqué à la matrice A, 1 est une valeur propre **simple** et dominante de A. Mais on en déduit aussi que A **admet un vecteur propre** v **strictement positif**.

Et, comme l'espace propre associé à la valeur propre 1 est de dimension 1 (d'après le théorème de Perron), il est engendré par le vecteur $v > 0$. Donc, en le normalisant, c'est-à-dire en posant π tel que :

$$\pi = \frac{v}{\|v\|}$$

On vient donc de prouver que :

$$\boxed{P \text{ admet une unique probabilité invariante } \pi \text{ (avec } \pi > 0)}$$

\

Il est également possible de déterminer explicitement une approximation de ce vecteur π à epsilon près à l'aide de **la méthode de la puissance itérée**.

En effet, comme la valeur propre 1 est dominante, c'est-à-dire que :

$$\forall \lambda \in \mathrm{Sp}(A) \backslash \{1\}, \quad |\lambda| < 1$$

on peut appliquer le théorème correspondant à la méthode de la puissance itérée qui s'écrit comme suit :

> On définit la suite $(v_k)_{k \in \mathbb{N}}$ telle que :
>
> $$\begin{cases} v_0 \text{ choisi arbitrairement dans } \mathbb{R}^{10} \\ \forall k \in \mathbb{N}, \quad v_{k+1} = \dfrac{A\, v_k}{\|A\, v_k\|} \end{cases}$$
>
> La valeur propre dominante est 1, donc le théorème peut alors s'écrire de cette manière : - **Si** v_0 n'appartient pas au sous-espace engendré par les vecteurs propres associés aux autres valeurs propres, avec $\|v_0\| = 1$, - **Alors** $v_k \underset{k \to +\infty}{\longrightarrow} \pi$

[source]

Dans la vraie version du théorème, on aurait :

$$v_k \underset{k \to +\infty}{\longrightarrow} v \quad \text{où } v \text{ est un vecteur unitaire de } A \text{ associé à la valeur propre 1}$$

Mais cela se ramène bien ici à $v_k \underset{k \to +\infty}{\longrightarrow} \pi$ puisque, par **unicité** de π, si v est un vecteur propre associé à la valeur propre 1 tel que $v > 0$ et $\|v\| = 1$, alors $\boxed{v = \pi}$. \

\

Pour ce qui est de la preuve de ce théorème, voici quelques liens qui permettent d'y voir plus clair :

- Une preuve sympathique de la méthode de la puissance itérée : https://moodle.utc.fr/file.php/665/MT09-ch8.pdf
- Les corollaires 2.8 et 2.10 permettent de mieux comprendre le pourquoi de l'utilisation de cette méthode dans le cadre des matrices stochastiques aux coefficients strictements positifs : https://www.imo.universite-paris-saclay.fr/~daniel.perrin/CAPES/algebre/Markov1.pdf

\

Voici alors l'implémentation de cette méthode algorithmique sur notre matrice $A = P^\top$:

[18]:
```
def methode_puissance_iteree(A, epsilon=1e-8, max_iterations=1000):
    n = A.shape[0]
    v = np.ones(n) / n  # Choix arbitraire pour v_0 de norme 1

    for i in range(max_iterations):
        v_new = np.dot(A.T, v)  # On s'intéresse à la transposée de la matrice␣
↪dans notre cas
        v_new = v_new / np.linalg.norm(v_new, ord=1)  # On normalise le nouveau␣
↪vecteur v

        if np.linalg.norm(v_new - v, ord=1) < epsilon: # On répète l'opération␣
↪jusqu'à se situer assez proche de vecteur v limite
            break

        v = v_new

    # Juste un test final pour vérifier la convergence de la suite v_k vers le␣
↪vecteur propre v pour la matrice A^T
    eig_val = np.dot(A.T, v) / v
    if np.allclose(eig_val, 1):
        return v
    else:
        raise ValueError("La suite v_k ne converge pas vers un vecteur propre v")

pi = methode_puissance_iteree(P)
print(pi)
```

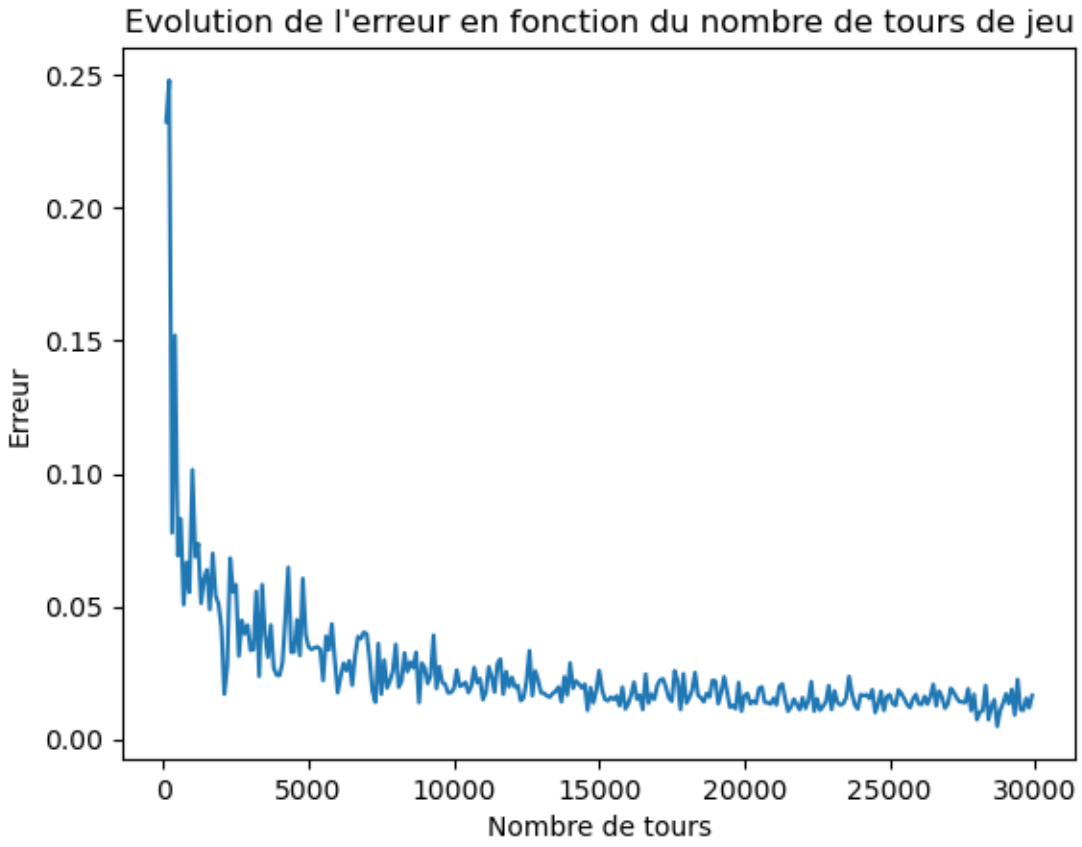
```
[0.10513222 0.10938556 0.09815112 0.12516746 0.11384361 0.11583024
 0.06706938 0.07545473 0.06746245 0.12250322]
```

Notre vecteur π déterminé empiriquement (p_{zone}) est alors très proche de la valeur théorique attendue déterminée par la méthode de la puissance itérée. Et cela se vérifie d'autant plus que N est grand.

\

Nous nous sommes alors intéressés à étudier la différence (l'erreur) entre $p_{zone} \times P$ et notre vecteur empirique p_{zone}. Et effectivement, on peut remarquer que cette erreur tend vers 0 lorque N (le nombre de tours de jeu) tend vers l'infini :

[19]:
```python
def calcul_erreur_1(P, X0, i_max) :
  L_erreur = []
  L_N = [i*100 for i in range (1,i_max)]
  for N in L_N :
    L_zone = simulation_etudiant_compteur_zone(P, 1, N)
    p_zone = [L_zone[i]/N for i in range (10)]

    ecart = np.dot(p_zone,P) - p_zone
    erreur = 0
    for i in range (10) :
      erreur += abs(ecart[i])
    L_erreur.append(erreur)
  plt.plot(L_N,L_erreur)
  plt.xlabel("Nombre de tours")
  plt.ylabel("Erreur")
  plt.title("Evolution de l'erreur en fonction du nombre de tours de jeu")
  plt.show()

calcul_erreur_1(P, 1, 300)
```



On peut donc constater que le vecteur π, probabilité empirique de P, peut être approchée par la limite des $p_{zone,N}$ lorsque N tend vers l'infini.

2.b On pose maintenant $X_0 = 5$.

[20]:
```
L_zone = simulation_etudiant_compteur_zone(P, 5, N)
P_zone = [L_zone[i]/N for i in range (10)]
print(P_zone)
```

```
[0.1107, 0.1028, 0.0997, 0.1274, 0.1152, 0.1153, 0.0643, 0.0822, 0.0661, 0.1164]
```

La nouvelle valeur de p_{zone} est très similaire à celle trouvée lorsque X_0 valait 1. En effet, lorsque l'on avance dans les tours de jeu, l'influence de la case de départ deviens de plus en plus faible. On peut vérifier cela en comparant l'écart entre des deux vecteurs en fonction de N.

[ ]:
```
def calcul_erreur_2(P, i_max) :
  L_erreur = []
  L_N = [i*100 for i in range (1,i_max)]
  for N in L_N :
    L_zone_1 = simulation_etudiant_compteur_zone(P, 1, N)
    p_zone_1 = [L_zone_1[i]/N for i in range (10)]
    L_zone_2 = simulation_etudiant_compteur_zone(P, 5, N)
    p_zone_2 = [L_zone_2[i]/N for i in range (10)]

    ecart = [p_zone_1[i] - p_zone_2[i] for i in range (10)]
    erreur = 0
    for i in range (10) :
      erreur += abs(ecart[i])
    L_erreur.append(erreur)
  plt.plot(L_N,L_erreur)
  plt.xlabel("Nombre de tours")
  plt.ylabel("Erreur")
  plt.title("Evolution de l'erreur en fonction du nombre de tours de jeu")
  plt.show()

calcul_erreur_2(P, 100)
```

Du fait de la case de départ qui importe peu lorsque le nombre de tours de jeu N augmente, on tire les mêmes conclusions que pour la 2.a., c'est-à-dire que :

On peut donc constater que le vecteur π, probabilité empirique de P, peut être approchée par la limite des $p_{zone,N}$ lorsque N tend vers l'infini.

1.1.3 Question 3

[ ]:
```
M = 100000
N = 1000
X0_eleve = 1
X0_monstre = 10
```

```python
def simulation_etudiant_survie_statique(P, X0_eleve, X0_monstre, N):
  # Si l'élève et le monstre sont déjà sur la même case (rip)
  if X0_eleve==X0_monstre :
    return 0
  # Initialisation
  Xn = X0_eleve #zone actuelle de l'étudiant
  # Tours
  for i in range(N):
    Xn = sim_dis(P[Xn-1], range(1,11), 1)[0] #déplacement de l'élève
    if Xn==X0_monstre : #si l'élève et le monstre sont sur la même case (rip)
      return (i+1)
  return N #l'élève a survécu toute la partie (pendant N tours)

L_tps_survie = [simulation_etudiant_survie_statique(P, X0_eleve, X0_monstre, N)␣
 ↪for _ in range (M)]
moy = np.mean(L_tps_survie)
print("Temps de survie moyen =",moy)
```

Soit $(X_n)_{n\in\mathbb{N}}$ la chaîne de Markov associé à l'étudiant. Le monstre reste ici immobile dans une zone précise noté a.

On note ici :

$$\tau_{0,a} = \min\{n \geqslant 0 \mid X_n = a\}$$

L'objectif ici est de calculer le temps de survie moyen exact de l'étudiant. C'est-à-dire le nombre de tour où l'étudiant n'atteint pas la zone a. Par conséquent, à l'image de la question 1.b, on s'intéresse ici au temps d'atteinte moyen de la zone a pour la chaîne de Markov (X_n).

Ainsi, on cherche ici à calculer :

$$\forall i \in [1,a], \quad \mathbb{E}_i[\tau_a] = \mathbb{E}[\tau_a \mid X_0 = i]$$

D'après le théorème sur le temps moyen d'atteinte, on peut alors énoncer que $(\mathbb{E}_i[\tau_a])_{1\leq i\leq a}$ est la plus petite solution positive du système suivant :

$$\boxed{\begin{cases} y_i = 0 & \text{si } i = a \\ y_i = 1 + \displaystyle\sum_{j=1}^{a-1} p(i,j)\, y_j & \text{sinon} \end{cases}}$$

On peut alors écrire ce système sous forme matricielle, en notant $b = (1)_{0<i<a}$, $x = (\mathbb{E}_i[\tau_a])_{0<i<a}$ et $P' = (p(i,j))_{0<i,j<a}$ matrice extraite de P :

$$\begin{aligned} \forall i \in \mathbb{N},\, 0 < i < a, \quad & \mathbb{E}_i[\tau_a] = 1 + \sum_{j=1}^{a-1} p(i,j)\, \mathbb{E}_j[\tau_a] \\ \Longrightarrow \quad & x = b + P'x \\ \Longrightarrow \quad & \boxed{[I_{a-1} - P']\, x = b} \end{aligned}$$

On remarque, de la même manière d'après le **lemme d'Hadamard**, que la matrice $A = I_{a-1} - P'$ est inversible, et on peut alors déterminer les différents valeurs des $\mathbb{E}_i[\tau_a]$.

[ ]:
```
def resolv_Q3(a,P) :
  # Matrice A
  A = np.eye(a-1) - np.array(P[:a-1,:a-1])
  # Vecteur b
  b = np.array([1 for i in range(a-1)])
  # Résolution
  return np.linalg.solve(A, b)

X0_monstre = 10
x = resolv_Q3(X0_monstre,P)
x = np.append(x,0) # Partir de la zone 10 revient à partir dans la zone du␣
 ↪monstre, donc le temps de survie moyen exact de l'étudiant vaut 0.
print("Toutes les solutions pour les zones de départ variant de 1 à 10 :",x)
print("")
print("Temps de survie moyen exact de l'étudiant partant de la zone 1 =",np.
 ↪round(x[0],3))
print("Temps de survie moyen empirique =", moy)
```

La valeur qu'on trouve pour $\mathbb{E}[\tau_a \mid X_0 = 1]$ est vraiment similaire à celle trouvée empiriquement par une méthode de Monte Carlo (et ce d'autant plus lorsque M est grand).

Et cela peut par exemple se vérifier à l'aide d'un intervalle de confiance.

En effet, en suivant la méthode de la question 1.b, on peut alors déterminer un intervalle de confiance à 95% notre estimation de Monte-Carlo, pour pouvoir comparer notre estimateur à la vraie valeur de $\mathbb{E}_1[\tau_a]$.

En notant $\hat{\theta}$ notre estimateur, et $\widehat{S^2}$ l'estimateur sans biais de la variance, on en déduit que l'intervalle de confiance correspondant est alors :

$$\boxed{\text{IC} = \left[\hat{\theta} - 1,96\sqrt{\frac{\widehat{S^2}}{M}}; \hat{\theta} + 1,96\sqrt{\frac{\widehat{S^2}}{M}}\right]}$$

[ ]:
```
print("Espérance empirique =",moy)
print("Vraie espérance =",np.round(x[0],3))

S2 = (1/(M-1)) * np.sum([(L_tps_survie[i] - moy)**2 for i in range(M)])
IC = [moy - 1.96*np.sqrt(S2/M), moy + 1.96*np.sqrt(S2/M)]
print("Intervalle de confiance à 95% =",IC)
```

Ainsi, en notant $\theta^* = \mathbb{E}[\tau_a \mid X_0 = 1]$, on remarque effectivement que :

$$\boxed{\theta^* \in \text{IC}}$$

1.1.4 Question 4

[ ]:
```
M = 100000
N = 10000
X0_eleve = 1
X0_monstre = 10

def simulation_etudiant_survie_mobile(P, X0_eleve, X0_monstre, N):
  # Si l'élève et le monstre sont déjà sur la même case (rip)
  if X0_eleve==X0_monstre :
    return 0
  # Initialisation
  Xn_eleve = X0_eleve #zone actuelle de l'étudiant
  Xn_monstre = X0_monstre #zone actuelle du monstre
  # Tours
  for i in range(N):
    Xn_eleve = sim_dis(P[Xn_eleve-1], range(1,11), 1)[0] #déplacement de l'élève
    Xn_monstre = sim_dis(P[Xn_monstre-1], range(1,11), 1)[0] #déplacement du␣
 ↪monstre
    if Xn_eleve==Xn_monstre : #si l'élève et le monstre sont sur la même case␣
 ↪(rip)
      return i+1
  return N+1 #l'élève a survécu toute la partie (pendant N tours)

L_tps_survie = [simulation_etudiant_survie_mobile(P, X0_eleve, X0_monstre, N)␣
 ↪for i in range (M)]
moy = np.mean(L_tps_survie)
print("Temps de survie moyen =",moy)
```

Soient les chaînes de Markov $(X_{k,\text{monstre}})$ et $(X_{k,\text{élève}})$ traduisant le déplacement de l'élève et du monstre sur les 10 zones en fonction du temps. Ces déplacements sont régis par la matrice stochastique P. La suite des $X_k = (X_{k,\text{monstre}}, X_{k,\text{élève}})$ forme elle-même une chaîne de Markov. \

Soit le temps d'arrêt τ défini par :

$$\tau = \min\{k \geqslant 0 \mid X_{k,\text{monstre}} = X_{k,\text{élève}}\}$$

On cherche ici à calculer :

$$\forall i, j \in [|1, 10|], \quad \mathbb{E}_{(i,j)}[\tau] = \mathbb{E}[\tau \mid X_0 = (i, j)]$$

D'après le théorème sur le temps moyen d'atteinte (cf. annexe), on peut alors énoncer que $(\mathbb{E}_{(i,j)}[\tau])_{1\leq i,j\leq 10}$ est la plus petite solution positive du système suivant :

$$\begin{cases} y_{(i,j)} = 0 & \text{si } i = j \\ y_{(i,j)} = 1 + \displaystyle\sum_{k\neq l} p((i,j),(k,l))\, y_{(k,l)} & \text{sinon} \end{cases}$$

On peut alors écrire ce système sous forme matricielle, en notant $b = (1)_{1\leq n\leq 90}$, $x = (\mathbb{E}_{(i,j)}[\tau])_{i\neq j}$ et $P' = (p((i,j),(k,l)))_{i\neq j, k\neq l}$ matrice extraite de P :

$$\begin{aligned} \forall (i,j)\in\mathbb{N}^2,\ 1\leq i,j\leq 10, i\neq j, \quad & \mathbb{E}_{(i,j)}[\tau] = 1 + \sum_{k\neq l} p((i,j),(k,l))\,\mathbb{E}_{(k,l)}[\tau] \\ \Longrightarrow \quad & x = b + P'x \\ \Longrightarrow \quad & \boxed{\left[I_{90} - P'\right] x = b} \end{aligned}$$

On remarque, de la même manière d'après le **lemme d'Hadamard**, que la matrice $A = I_{90} - P'$ est inversible, et on peut alors déterminer les différents valeurs des $\mathbb{E}_{(i,j)}[\tau]$.

[ ]:
```
def matrice_stochastique_couple(P):
  taille = len(P[0][:])
  PP = np.zeros([taille**2,taille**2])
  for i in range (taille): #Dizaines : zone du monstre avant le tour
    for j in range (taille): #Unités : zone de l'élève avant le tour
      for k in range (taille): #Dizaines : zone du monstre après le tour
        for l in range (taille): #Unités : zone de l'élève après le tour
          PP[10*k+l,10*i+j] = P[k,i] * P[l,j]
  return PP

def remove_line_column(P,l,k):
  PP1 = np.array(P[:l,:k])
  PP2 = np.array(P[l+1:,:k])
  PP3 = np.array(P[:l,k+1:])
  PP4 = np.array(P[l+1:,k+1:])
  PPP1 = np.vstack((PP1, PP2))
  PPP2 = np.vstack((PP3, PP4))
  PPP = np.hstack((PPP1, PPP2))
  return PPP

def resolv_Q4(P):
  # Matrice A
  PP = matrice_stochastique_couple(P)
  for i in range (10):
    PP = remove_line_column(PP,10*i,10*i)
  A = np.eye(90) - PP
  # Vecteur b
  b = np.array([1 for i in range(90)])
  # Résolution
  return np.linalg.solve(A, b)

x = resolv_Q4(P)
xx = []
i=0
for E in x:
  if (i%10)==0:
```

```
    xx.append(0)
  xx.append(E)
  i+=1
if (i%10)==0:
  xx.append(0)
Solution = np.zeros([10,10])
for i in range (10):
  for j in range (10):
    Solution[i,j] = xx[i+10*j]

print("Toutes les solutions pour les zones de départ variant de 1 à 10 :",x)
print("")
print("Temps de survie moyen exact de l'étudiant partant de la zone 1 avec le␣
 ↪monstre partant de la zone 10 =", np.round(Solution[9,0],3))
print("Temps de survie moyen empirique =", moy)
```

Encore une fois, la valeur qu'on trouve pour $\mathbb{E}\left[\tau_a \,|\, X_{0,lve} = 1, X_{0,monstre} = 10\right]$ est vraiment similaire à celle trouvée empiriquement par une méthode de Monte Carlo (et ce d'autant plus lorsque M est grand).

Et cela peut notamment se vérifier à l'aide d'un intervalle de confiance par exemple.

En effet, en suivant la méthode de la question 1.b, on peut alors déterminer un intervalle de confiance à 95% notre estimation de Monte-Carlo, pour pouvoir comparer notre estimateur à la vraie valeur de $\mathbb{E}_{(10,1)}[\tau]$.

En notant $\hat{\theta}$ notre estimateur, et $\widehat{S^2}$ l'estimateur sans biais de la variance, on en déduit que l'intervalle de confiance correspondant est alors :

$$\boxed{\text{IC} = \left[\hat{\theta} - 1,96\sqrt{\frac{\widehat{S^2}}{M}}; \hat{\theta} + 1,96\sqrt{\frac{\widehat{S^2}}{M}}\right]}$$

[ ]:
```
print("Espérance empirique =",moy)
print("Vraie espérance =",np.round(Solution[0,9],3))

S2 = (1/(M-1)) * np.sum([(L_tps_survie[i] - moy)**2 for i in range(M)])
IC = [moy - 1.96*np.sqrt(S2/M), moy + 1.96*np.sqrt(S2/M)]
print("Intervalle de confiance à 95% =",IC)
```

Ainsi, en notant $\theta^* = \mathbb{E}\left[\tau_a \,|\, X_{0,lve} = 1, X_{0,monstre} = 10\right]$, on remarque effectivement que :

$$\boxed{\theta^* \in \text{IC}}$$

1.1.5 Question 5

Dans cette question, on réalise la même expérience que précedemment, mais en consiférent plusieurs étudiants en même temps. Tous commencent sur la zone 1, le monstre commence sur la zone 10. Tous les individus sont soumis à la même matrice de probabilité P. On arrête l'expérience lorsque le

premier étudiant est mangé par le monstre mobile, et on étudiera le temps moyen avant le premier mort en fonction du nombre d'étudiants au départ.

[ ]:
```
def simulation_etudiants_survie_mobile_min(P, nb_eleves, X0_eleve, X0_monstre,␣
↪N):
  # Si les élèves et le monstre sont déjà sur la même case (rip)
  if X0_eleve==X0_monstre :
    return 0
  # Initialisation
  Xn_eleve = [X0_eleve for i in range (nb_eleves)] #zone actuelle des étudiants
  Xn_monstre = X0_monstre #zone actuelle du monstre
  # Tours
  for t in range(N):
    Xn_eleve = [sim_dis(P[Xn_eleve[i]-1], range(1,11), 1)[0] for i in range␣
↪(nb_eleves)] #déplacement de l'élève
    Xn_monstre = sim_dis(P[Xn_monstre-1], range(1,11), 1)[0] #déplacement du␣
↪monstre
    for i in range(nb_eleves):
      if Xn_eleve[i]==Xn_monstre : #si l'élève et le monstre sont sur la même␣
↪case (rip)
        return t
  return N #l'élève a survécu toute la partie (pendant N tours)

M = 1000
N = 100
X0_eleve = 1
X0_monstre = 10
nb_eleve = 20

L_tps_survie_min_moy = []
for i in range (1,nb_eleve+1):
  L_tps_survie_min = [simulation_etudiants_survie_mobile_min(P, i, X0_eleve,␣
↪X0_monstre, N) for _ in range (M)]
  moy = np.mean(L_tps_survie_min)
  L_tps_survie_min_moy.append(moy)
plt.plot([i for i in range (1,nb_eleve+1)], L_tps_survie_min_moy)
plt.xlabel("Nombre d'élèves au début de la partie")
plt.ylabel("Date de la première mort")
plt.title("Evolution du temps moyen de la première mort d'un des étudiants")
plt.show()
```

On observe que l'évolution du temps avec la première mort est décroissante lorsque le nombre d'étudiants augmente.

Soit n le nombre d'élèves.
On généralise le procédé étudié en Q.4 avec n élèves et un monstre qui se déplacent.
Soient les chaînes de Markov $(X_{k,monstre})$, $(X_{k,lve\ 1})$, ..., $(X_{k,lve\ n})$ traduisant le déplacement des élèves et du monstre sur les 10 zones en fonction du temps. Ces déplacements sont régis par la

matrice stochastique P. La suite des $X_k = (X_{k,monstre}, X_{k,lve\ 1}, ..., X_{k,lve\ n})$ forme elle-même une chaîne de Markov.
Soit le temps d'arrêt τ défini par :

$$\tau = \min\{k \geqslant 0 \mid \exists m \in [|1,n|], X_{k,lve\ m} = X_{k,monstre}\}$$

On cherche ici à calculer :

$$\forall i, j_1, ..., j_n \in [|1,10|], \quad \mathbb{E}_{(i,j_1,...,j_n)}[\tau] = \mathbb{E}[\tau \mid X_0 = (i, j_1, ..., j_n)]$$

D'après le théorème sur le temps moyen d'atteinte, on peut énoncer que $(\mathbb{E}_{(i,j_1,...,j_n)}[\tau])_{1\leq i,j_1,...,j_n\leq 10}$ est la plus petite solution positive du système suivant :

$$\boxed{\begin{cases} y_{(i,j_1,...,j_n)} = 0 & \text{si } \exists m \in [|1,n|] : i = j_m \\ y_{(i,j_1,...,j_n)} = 1 + \displaystyle\sum_{k \neq l_1} & \\ \quad \dots & \\ k \neq l_n p((i,j_1,...,j_n),(k,l_1,...,l_n))\, y_{(k,l_1,...,l_n)} & \text{sinon} \end{cases}}$$

On pose M le nombre de $(n+1)$-ulpets $(i, j_1, ..., j_n)$ tels que $\forall m \in [|1,n|]$, $i \neq j_m$.
On peut alors écrire ce système sous forme matricielle, en notant $b = (1)_{1\leq n\leq M}$, $x = (\mathbb{E}_{(i,j_1,...,j_n)}[\tau])_{i\neq j_1,...,i\neq j_n}$ et $P' = (p((i,j_1,...,j_n),(k,l_1,...,l_n)))_{i\neq j_1,...,i\neq j_n,k\neq l_1,...,k\neq l_n}$ où P' est une matrice extraite de P :

$$\begin{aligned} \forall (i,j_1,\dots,j_n) \in \mathbb{N}^2,\ 1 \leq i,j \leq 10, i \neq j_1, ..., i \neq j_n, \quad \mathbb{E}_{(i,j_1,...,j_n)}[\tau] &= 1 + \sum_{k\neq l} p((i,j_1,...,j_n),(k,l_1,...,l_n))\, \mathbb{E}_{(k,l_1,...,l_n)} \\ \implies \quad x &= b + P'x \\ \implies \quad \boxed{\left[I_M - P'\right] x = b} & \end{aligned}$$

On remarque, d'après le **lemme d'Hadamard**, que la matrice $A = I_M - P'$ est inversible, et on peut alors déterminer les différents valeurs des $\mathbb{E}_{(i,j)}[\tau]$.

Si, en théorie, le calcul est plutôt similaire, la compléxité de la définition du temps d'arrêt complexifie très grandement la construction de la matrice P' définie plus haut : il est compliqué de trouver efficacement les coefficients de la matrice stochastique à garder pour construire P'.
De plus, les coefficients de la matrice P (et donc de la matrice extraite P') sont de plus en plus petits lorsque n grandi, donc à moins d'avoir la précision nécessaire sur notre machine, la solution par pivot de Gauss est grandement soumise aux erreurs de précision.

Ainsi, cette question est plus difficile à aborder, tant sur le plan théorique que sur le plan pratique.

1.2 Annexes

1.2.1 Théorème du temps moyen d'atteinte

Dans le cadre général Soit $(X_n)_{n\in\mathbb{N}}$ une chaîne de Markov homogène avec comme matrice de transition $P = (p_{i,j})_{(i,j)\in E^2}$ sur un espace d'état fini ou dénombrable E.

Soit $A \subseteq E$, on définit : - Le temps d'atteinte de A par :

$$T_A = \inf\{n \in \mathbb{N} \,|\, X_n \in A\}$$

- Le temps moyen d'atteinte de A partant de i

$$v_i^A = \mathbb{E}_i[T_A].$$

On rappelle également que la notation $\mathbb{P}_i$ (idem pour $\mathbb{E}_i$) :

$$\forall i \in E, \text{Pour tout évènement } B, \quad \mathbb{P}_i(B) = \mathbb{P}(B \,|\, X_0 = i).$$

\

Le théorème peut alors s'écrire de cette manière :

Théorème du temps moyen d'atteinte

Le vecteur des temps moyen d'atteinte $(v_i)_{i\in E}$ est la plus petite solution positive du système

$$\begin{cases} y_i = 0 & \text{si } i \in A \\ y_i = 1 + \sum_{j \notin A} p_{i,j}\, y_j & \text{sinon} \end{cases}$$

\

Démonstration :

On supposera dans cette démonstration pour simplifier (et car c'est dans le cadre du projet de maths) que $\mathbb{P}(T_A < \infty) = 1$. \

Montrons tout d'abord que $(v_i)_{i\in E}$ est solution du système. - Si $X_0 = i \in A$ alors on a bien sûr $T_A = 0$ et donc $v_i = \mathbb{E}_i[T_A] = 0$ - Si $X_0 = i \notin A$ alors on a :

$$v_i = \mathbb{E}_i[T_A] = \sum_{j\in E} \mathbb{E}_i\left[T_A \mathbb{1}_{\{X_1=j\}}\right]$$

Et pour tout $j \in E$, on a:

$$\begin{aligned}
\mathbb{E}_i\left[T_A \mathbb{1}_{\{X_1=j\}}\right] &= \left(\sum_{k\in\mathbb{N}^*} k\, \mathbb{P}_i(T_A = k, X_1 = j)\right) \\
&= \left(\sum_{k\in\mathbb{N}^*} k\, \mathbb{P}_i(T_A = k \,|\, X_1 = j)\, \mathbb{P}_i(X_1 = j)\right) \quad \text{(formule des probabilités totales)} \\
&= \left(\sum_{k\in\mathbb{N}^*} k\, \mathbb{P}_j(T_A = k-1)\, p_{i,j}\right) \quad \text{(par propriété de Markov)} \\
&= p_{i,j}\left(\sum_{k\in\mathbb{N}^*} k\, \mathbb{P}_j(T_A = k-1)\right) \\
&= p_{i,j}\left(\left(\sum_{k\in\mathbb{N}} k\, \mathbb{P}_j(T_A = k)\right) + \left(\sum_{k\in\mathbb{N}} \mathbb{P}_j(T_A = k)\right)\right) \quad \text{(chgt. de var. } k = \widetilde{k}+1) \\
&= p_{i,j}\left(\mathbb{E}_j[T_A] + 1\right)
\end{aligned}$$

D'où

$$\mathbb{E}_i[T_A] = \sum_{j\in E} p_{i,j}\,(\mathbb{E}_j[T_A]+1) = \sum_{j\in E} p_{i,j} + \sum_{j\in E} p_{i,j}\,\mathbb{E}_j[T_A]$$

Et donc :

$$v_i = 1 + \sum_{j\in E} p_{i,j}\,v_j = 1 + \sum_{j\notin A} p_{ij}\,v_j \qquad \text{(car } v_j = 0 \text{ lorsque } j\in A\text{)}$$

On en conclut donc que $(v_i)_{i\in E}$ est solution du système. \

Montrons maintenant la **minimalité** de $(v_i)_{i\in E}$. Soit $(y_i)_{i\in E}$ solution du système. Alors si $i\in A$, on a immédiatement $y_i = 0 = u_i$. Et si $i\notin A$ on a

$$\begin{aligned}
y_i &= 1 + \sum_{j\notin A} p_{i,j}\,y_j \\
&= 1 + \sum_{j\notin A} p_{i,j}\left(1 + \sum_{k\notin A} p_{j,k}\,y_k\right) \\
&= 1 + \sum_{j\notin A} p_{i,j} + \sum_{j\notin A}\sum_{k\notin A} p_{i,j}\,p_{j,k}\,y_k \qquad \text{(car } y_j \text{ est, lui aussi, solution du système)} \\
&= \mathbb{P}_i(T_A\geq 1) + \mathbb{P}_i(T_A\geq 2) + \sum_{j\notin A}\sum_{k\notin A} p_{i,j}\,p_{j,k}\,y_k.
\end{aligned}$$

En guise de petite explication de cette dernière ligne, on a $i\notin A$ donc $\mathbb{P}_i(T_A\geq 1) = 1$, et par définition de l'ensemble A, et sachant que $i\notin A$, on a bien : $\mathbb{P}_i(T_A\geq 2) = \sum_{j\notin A} p_{i,j}$. \

Et par récurrence sur n (qu'il faudrait normalement bien rédiger, mais que nous admettrons ici), on obtient que, pour tout $n\geq 1$

$$y_i = \sum_{k=1}^{n} \mathbb{P}_i(T_A\geq k) + \sum_{j_1\notin A}\sum_{j_2\notin A}\cdots\sum_{j_n\notin A} p_{i,j_1}\,p_{j_1,j_2}\cdots p_{j_{n-1},j_n}\,y_{j_n}.$$

On a de plus supposé que $y_i\geq 0$ pour tout $i\in E$ (car on a posée $(y_i)_{i\in E}$ comme solution positive du système), d'où :

$$y_k \geq \sum_{k=1}^{n} \mathbb{P}_i(T_A\geq k)$$

Ceci étant vrai pour tout $n\geq 1$, on a :

$$y_i \geq \sum_{k=1}^{+\infty} \mathbb{P}_i(T_A\geq k)$$

et le passage à la limite est autorisé car la suite $\left(\sum_{k=1}^{n} \mathbb{P}_i(T_A\geq k)\right)_{n\geq 1}$ est croissante.

De plus, par propriété de l'espérance dans le cadre des variables aléatoires discrètes :

$$\sum_{k=1}^{+\infty} \mathbb{P}(T_A\geq k) = \sum_{k=1}^{\infty} k\,\mathbb{P}(T_A = k) = \mathbb{E}_i(T_A)$$

Donc $\boxed{\forall i \in E, \quad y_i \geq \mathbb{E}_i(T_A) = v_i}$.

Et ainsi, le vecteur des temps moyen d'atteinte $(v_i)_{i\in E}$ est **la plus petite solution positive du système**.

Dans le cadre d'un couple de chaînes de Markov On note ici $E = [|1, 10|]^2$, et $A = \{(i, j) \in E \,|\, i = j\}$.

On s'intéresse ici à la suite des $X_k = (X_{k,\text{monstre}}, X_{k,\text{élève}})$, qui forme elle-même une chaîne de Markov.

Le théorème du temps moyen d'atteinte peut alors s'écrire de cette manière :

Théorème du temps moyen d'atteinte :

> Le vecteur des temps moyen d'atteinte $(v_{(i,j)})_{(i,j)\in E}$ est la plus petite solution positive du système
> $$\begin{cases} y_{(i,j)} = 0 & \text{si } i \in A \\ y_{(i,j)} = 1 + \displaystyle\sum_{(k,l)\notin A} p_{(i,j)\to(k,l)}\, y_{(k,l)} & \text{sinon} \end{cases}$$

\

Démonstration :

On supposera dans cette démonstration pour simplifier (et car c'est dans le cadre du projet de maths) que $\mathbb{P}(T_A < \infty) = 1$. \

Montrons tout d'abord que $(v_{(i,j)})_{(i,j)\in E}$ est solution du système. - Si $X_0 = (i, j) \in A$ alors on a bien sûr $T_A = 0$ et donc $v_{(i,j)} = \mathbb{E}_{(i,j)}[T_A] = 0$ - Si $X_0 = (i, j) \notin A$ alors on a :

$$v_{(i,j)} = \mathbb{E}_{(i,j)}[T_A] = \sum_{(k,l)\in E} \mathbb{E}_{(i,j)}\left[T_A \mathbb{1}_{\{X_1=(k,l)\}}\right]$$

Et pour tout $(k,l) \in E$, on a:

$$\begin{aligned}
\mathbb{E}_{(i,j)}\left[T_A \mathbb{1}_{\{X_1=(k,l)\}}\right] &= \left(\sum_{n\in\mathbb{N}^*} n\,\mathbb{P}_{(i,j)}(T_A = k, X_1 = (k,l))\right) \\
&= \left(\sum_{n\in\mathbb{N}^*} n\,\mathbb{P}_{(i,j)}(T_A = k \,|\, X_1 = (k,l))\,\mathbb{P}_{(i,j)}(X_1 = (k,l))\right) \\
&\qquad\qquad \text{(probabilités totales)} \\
&= \left(\sum_{n\in\mathbb{N}^*} n\,\mathbb{P}_{(k,l)}(T_A = n-1)\,p_{(i,j)\to(k,l)}\right) \qquad \text{(par propriété de Markov)} \\
&= p_{(i,j)\to(k,l)}\left(\sum_{n\in\mathbb{N}^*} n\,\mathbb{P}_{(k,l)}(T_A = n-1)\right) \\
&= p_{(i,j)\to(k,l)}\left(\left(\sum_{n\in\mathbb{N}} n\,\mathbb{P}_{(k,l)}(T_A = n)\right) + \left(\sum_{n\in\mathbb{N}} \mathbb{P}_{(k,l)}(T_A = n)\right)\right) \\
&\qquad\qquad \text{(chgt. de var. } n = \widetilde{n}+1) \\
&= p_{(i,j)\to(k,l)}\left(\mathbb{E}_{(k,l)}[T_A] + 1\right)
\end{aligned}$$

D'où

$$\mathbb{E}_{(i,j)}[T_A] = \sum_{j\in E} p_{(i,j)\to(k,l)}\left(\mathbb{E}_{(k,l)}[T_A] + 1\right) = \sum_{(k,l)\in E} p_{(i,j)\to(k,l)} + \sum_{(k,l)\in E} p_{(i,j)\to(k,l)}\,\mathbb{E}_{(k,l)}[T_A]$$

Et donc :

$$v_{(i,j)} = 1 + \sum_{(k,l)\in E} p_{(i,j)\to(k,l)}\,v_{(k,l)} = 1 + \sum_{(k,l)\notin A} p_{(i,j)\to(k,l)}\,v_{(k,l)} \quad (\text{car } v_{(k,l)} = 0 \text{ lorsque } (k,l)\in A)$$

On en conclut donc que $(v_{(i,j)})_{(i,j)\in E}$ est solution du système. \

La minimalité de $(v_{(i,j)})_{(i,j)\in E}$ se montre de la même manière que dans le cas plus classique. (On se passera donc de la réécriture en adaptant seulement les notations.) \

Ainsi, le vecteur des temps moyen d'atteinte $(v_{(i,j)})_{(i,j)\in E}$ est **la plus petite solution positive du système**.