UNIVERSITÉ
CÔTE D'AZUR

2023-2024

Master 1 Ingénierie mathématique

Rapport du projet de processus stochastique

Résumé :

Ce document examine de près, un modèle et une construction d'un processus aléatoire $(Y_t)_{t\geq 0}$ représentant le capital d'un casino en fonction du temps. Cette étude vise à analyser les différents ajustements et paramétrages de faillite ou de gain de la maison sur le long terme. Dans le cadre de cette investigation, la modélisation et la simulation probabiliste du capital à différente dates nous permettra de mettre en évidence comment les probabilités sont ajustées pour créer un avantage de la maison, garantissant une rentabilité statistique sur une période étendue.

Rapporteurs :

YANG Quiyan

KPATCHA Essowaza Celestin

MAN Castillo David

Introduction :

La gestion du capital dans l'industrie du jeu a toujours été un défi complexe. Le contexte aléatoire des jeux de hasard exige une approche stratégique pour assurer la pérennité financière des établissements tels que les casinos. Notre projet se penche sur cette problématique en développant un modèle de processus aléatoire qui représente de manière réaliste l'évolution du capital d'un casino. Dans cette démarche, nous explorerons la construction du modèle, en accordant une attention particulière au choix des paramètres afin de garantir la stabilité financière tout en permettant une croissance raisonnable.

Notre objectif final est d'analyser la probabilité de ruine du casino, offrant ainsi des perspectives cruciales pour la gestion des risques financiers dans le secteur du jeu. De manière complémentaire, notre étude intègrera une composante de simulation numérique du processus, permettant une validation pratique du modèle. Cette approche nous fournira des données tangibles et des résultats concrets pour évaluer la performance du modèle dans des scénarios réalistes.

I. Aspects mathématiques :

1. Modèle Mathématique de travail :

Considérons un casino où les rentrées d'argent entre les instants 0 et t sont déterministes et gouvernées par un paramètre $\alpha > 0$, représentant le gain constant αt . Les gains des joueurs suivent un processus de Poisson de paramètre 1, avec des variables aléatoires indépendantes identiquement distribuées (Xi) strictement positives et admettant un moment d'ordre 2, notées $\mu = E(X1)$ et $\sigma^2 = Var(X1)$.

Le capital du casino à l'instant t, noté, Y_t avec un capital initial Y_0 , est décrit par :

$$Y_t = Y_0 + \alpha t - \sum_{i=1}^{N_t} X_i$$

Où N_t est le nombre de joueurs ayant gagné jusqu'à l'instant t.

Il est important de souligner que l'objectif est de minimiser la probabilité de ruine $r(y)$ du casino sachant un investissement initial y : $r(y) = \mathbb{P}(\exists\, t \in \mathbb{R}_+\, t.q.\, Y_t < 0 \mid Y_0 = y)$.

2. Analyse des Propositions :

Proposition 1: $\forall y \geq 0,\ r(y) > 0$.

Démostration : $\exists\, k \in \mathbb{N}^*,\ \mathbb{P}(N_t = k) > 0$**, soient** $t > 0,\ y \geq 0$ **telle que** $Y_0 + \alpha t < \sum_{i=1}^{N_t} X_i\ ,\ p.s \Longrightarrow$ $\mathbb{P}\left(Y_0 + \alpha t < \sum_{i=1}^{N_t} X_i\right) > 0 \Leftrightarrow \mathbb{P}(Y_t < O \mid Y_0 = y)$

Analyse : Cette démonstration met en évidence que, quel que soit le montant initial d'argent investi par le casino, il existe toujours un risque non négligeable que le capital atteigne un niveau négatif à un moment donné. Cela souligne la difficulté pour le casino de minimiser totalement le risque de faillite, même en ajustant les paramètres du modèle.

Proposition 2: Si $\alpha < \mu$ alors $\forall y \geq 0,\ r(y) = 1$

Démostration : $N_t = \left(N_t - N_{\lfloor t \rfloor}\right) + \sum_{i=1}^{\lfloor t \rfloor}(N_i - N_{i-1})$ posons $n_i = N_i - N_{i-1}, \Longrightarrow N_t = \sum_{i=1}^{t} n_i\ avec\ n_i\ i.i.d$

$$\mathbb{E}\left[e^{it\left(\sum_{i=1}^{t} n_i\right)}\right] = \mathbb{E}\left[e^{itn_1}\right] \times \ldots\ldots\ldots \times \mathbb{E}\left[e^{itn_t}\right]$$

La moyenne empirique de n_i est $\frac{N_t}{t}$

D'après la loi forte des grands nombres, $\frac{N_t}{t} = 1 = \mathbb{E}[n_i]\ p.s\ \forall i$

$$\Longrightarrow \lim_{t\to\infty} Y_t = Y_0 + \alpha t - \sum_{i=1}^{t} X_i \Longrightarrow \lim_{t\to\infty} \frac{Y_t}{t} = \frac{1}{t}\left(Y_0 + \alpha t - \sum_{i=1}^{t} X_i\right) = \alpha - \frac{1}{t}\left(\sum_{i=1}^{t} X_i - Y_0\right) \Longrightarrow \lim_{t\to\infty} \frac{Y_t}{t} = \alpha - \mu,\ avec\ \mu$$

$$= \frac{1}{t}\left(\sum_{i=1}^{t} X_i - Y_0\right)$$

Analyse : si $\alpha < \mu,\ alors\ \alpha t < \sum_{i=1}^{t} X_i - Y_0 \Longrightarrow\ \alpha t + Y_0 < \sum_{i=1}^{t} X_i \Longrightarrow\ r(y) = 1$

Proposition 3: Si $\alpha = \mu$ alors $\forall y \geq 0,\ r(y) = 1$

Démostration : $\boldsymbol{S_n = \sum_{i=1}^{n}(X_i - \alpha\xi_i) = \sum_{i=1}^{n} X_i - \alpha \sum_{i=1}^{n} \xi_i = \sum_{i=1}^{n} X_i - \alpha T_i}$

On a $\frac{N_n}{n} \xrightarrow[n\to+\infty]{} 1, donc\ \boldsymbol{S_n = \sum_{i=1}^{N_n} X_i - \alpha T_i}$

Et $Y_{T_n} = Y_0 - \boldsymbol{\sum_{i=1}^{N_n} X_i - \alpha T_i}$ donc $\lim_{n\to+\infty} Y_{T_n} = Y_0 - S_n$

Donc pour montrer que $r(y) = 1$ i.e. $Y_{T_n} < 0$

On va montrer que $Y_{T_n} \xrightarrow[S_n\to+\infty]{} -\infty$ i.e. $Y_{T_n} \xrightarrow[n\to+\infty]{} -\infty$

Il suffit de montrer que $S_n \xrightarrow[n\to+\infty]{} +\infty$, à l'aide du lemme 1 on peut le démontrer.

Lemme 1 : Soit $(Z_n)_{n\geq 1}$ une suite de variables aléatoires indépendantes.

$\sigma(Z_1, Z_2, \ldots\ldots, Z_k) \coprod \sigma(Z_{k+1}, Z_{k+2}, \ldots\ldots), \forall k \in \mathbb{N}^*$ Soit $\mathcal{F} \subset \sigma(Z_{k+1}, Z_{k+2}, \ldots\ldots) \forall k \in \mathbb{N}$, $\mathcal{F} \coprod \sigma(Z_1, Z_2, \ldots\ldots Z_k)\ \forall k \in \mathbb{N}^*$ donc $\mathcal{F} \coprod \sigma(\cup_{k\in\mathbb{N}^*}^{\sigma}(Z_1, Z_2, \ldots\ldots Z_k))$, donc $\forall i \in \mathbb{N},\ Z_i \in \mathcal{A}_i\ on\ a\ \mathcal{A}_i \coprod \mathcal{A}_j\ si\ i \neq j \forall i, j \in \mathbb{N}$, où $\mathcal{A}_i$, est mesurable et $\forall k \in \mathbb{N},\ \cap_{k\in\mathbb{N}} \sigma(Z_{k+1}, Z_{k+2}, \ldots\ldots) \subset \sigma(Z_1, \ldots\ldots Z_k) \Longrightarrow$ $\cap_{n\geq 1} \sigma(Z_n, Z_{n+1}, Z_{n+2}, \ldots\ldots)\ est\ un\ \pi - système$ donc l'ensemble $A = \cap_{n\geq 1} \sigma(Z_n, Z_{n+1}, Z_{n+2}, \ldots\ldots)$ est indépendant à lui-même. Donc $\mathbb{P}(A \cap A) = \mathbb{P}^2(A) = \mathbb{P}(A) \Longrightarrow \mathbb{P}(A) = 0\ ou\ \mathbb{P}(A) = 1$

3. Assertions sur la statistique Hn :

Introduction de la statistique :

$$H_n = \sup_{x\geq 0} \left|\widehat{F_n}(x) - \left(1 - e^{-x/\overline{X_n}}\right)\right|$$

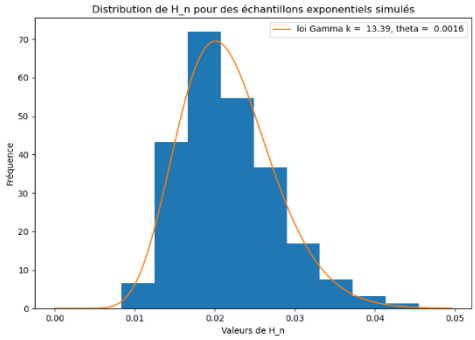

Figure 1. Distribution de Hn pour un échantillon exponentiel de taille 100. Simulations = 5000

D'ici pour une confiance de 95% on peut avoir C = 0,03 pour zone de rejet $\{H_n > C\}$.

Nous avons tracé une distribution gamma en haut, parce que nous travaillons avec des distributions exponentielles et qu'il est possible que Hn ait une distribution gamma. En regardant le graphique, nous pouvons voir qu'il y a une grande similarité entre la distribution simulée et une distribution gamma ajustée aux paramètres correspondants dans le graphique. Nous avons calculé ces paramètres à l'aide de la fonction gamma.fit de scipy.stats. Nous avons également calculé C avec un niveau de signification de 0,05 et avons trouvé C_gamma = 0,032.

Si $\{H_n < C\}$ et sachant que le lemme 2 nous garantit l'indépendance donc on peut utiliser la statistique :

$$\widehat{\gamma_n} = \frac{1}{\overline{X_n}} := \frac{n}{X_1 + \cdots + X_n}$$

comme paramètre de notre modèle.

Lemme 2. Si les $(X_i)_{i\geq 1}$ ont une distribution exponentielle de paramètre γ, la loi de la statistique H_n est indépendante de γ.

Convergence vers 0. Si les $(X_i)_{i\geq 1}$ ont une distribution exponentielle, $H_n \to 0$ presque sûrement quand $n \to \infty$. Simulation :

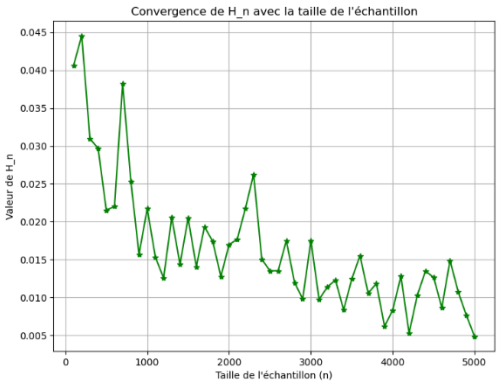

Figure 2. Convergence de Hn vers 0 quand la taille de l'échantillon tend vers l'infini

Le graphique vise à illustrer si Hn tend vers 0 lorsque la taille de l'échantillon n augmente. Une tendance décroissante vers 0 indiquerait que plus l'échantillon est grand, plus la fonction de répartition empirique se rapproche de la fonction de répartition théorique de la distribution exponentielle.

II. Simulation et discussion des résultats :

Au cœur de ce projet, l'objectif n'est pas de noyer le lecteur dans des graphiques, mais de percer le sens profond des résultats. Utilisant le langage Python pour effectuer plusieurs tests avec des données et paramètres délibérément choisi, notre démarche se concentre sur la compréhension fine plutôt que sur une simple présentation de données visuelles. Chaque test a été pour nous une exploration, chaque résultat une facette à comprendre, dans le but d'extraire des enseignements essentiels.

Le code python sur ce projet est annexé au document afin de permettre au lecteur d'effectuer une infinité de test.

1. Cas $\alpha < \mu$

$$\alpha < \mu, \; alors \; \forall y \geq 0, \; r(y) = 1$$

Graphe :

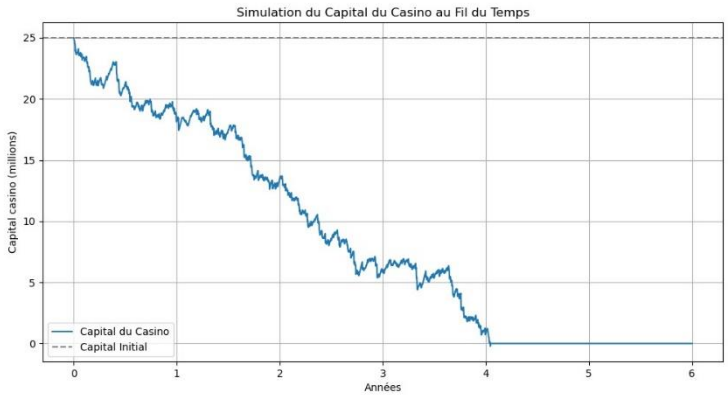

Figure 3. Capital du Casino au Fil du Temps avec α > μ

Analyse : Ce graphe montre que lorsque le paramètre $\alpha < \mu$, l'argent du casino diminue au fil du temps c'est à dire d'année en année a telle point de perdre tout son investissement au bout de quatre ans dans notre exemple. On constate régulièrement de petites fluctuations qui représente sans doute les moments où les joueurs perdent ; mais il est clair que ces pertes ne sont rien comparé au gain. Ce graphe permet donc de confirmer la proposition selon laquelle lorsque le paramètre d'entrées d'argent du casino est plus petit que l'espérance de gain des joueurs, alors on a une faillite presque certaine du casino.

2. Cas $\alpha = \mu$

$$\alpha = \mu, \; alors \; \forall y \geq 0, \; r(y) = 1$$

Graphe 1 :

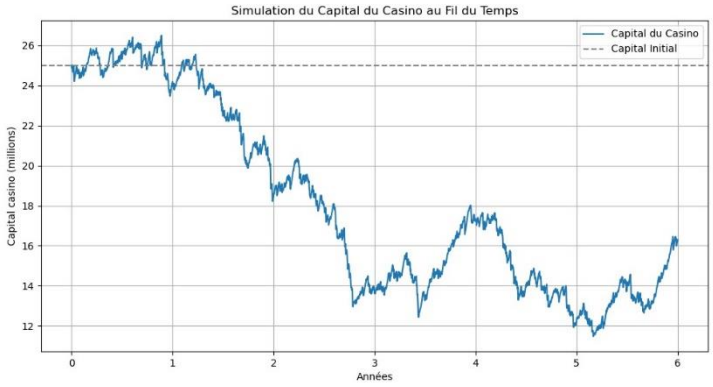

Figure 4. Capital du Casino au Fil du Temps avec α = μ et temps de la simulation de 6 ans

Analyse : Le graphe montre comme dans le premier cas, que quand $\alpha = \mu$, l'argent du casino diminue au fil du temps mais avec une évolution vers la faillite plus lente. On constate même dans notre exemple que la première année a connu des moments de bénéfice pour le casino. De plus même si on remarque une chute du capital on constate aussi des fluctuations plus importantes que le cas ci-dessus. Ce graphe permet donc de confirmer la proposition selon laquelle lorsque le paramètre d'entrées d'argent du casino est égal à l'espérance de gain des joueurs, alors le capital du casino chute.

Graphe 2 :

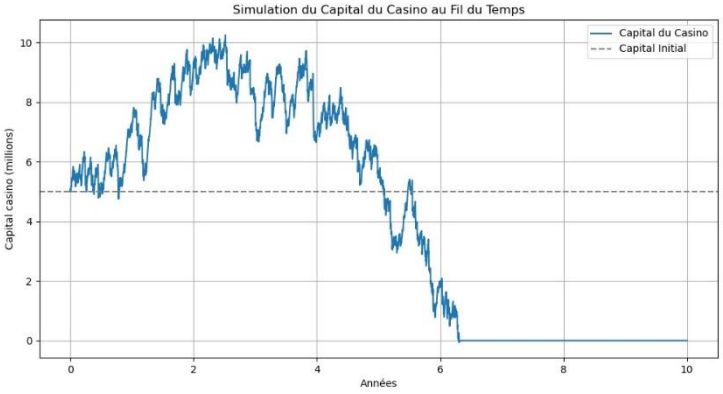

Figure 5 . Capital du Casino au Fil du Temps avec α = μ et temps de la simulation de 10 ans

Analyse : Ce graphe montre juste que même si la chute du capital est moins rapide, la ruine du casino est inévitable dans un temps plus long.

En conclusion, quand le paramètre $\alpha = \mu$, le casino peut avec peu de chance gagné à certaine période, mais il y a beaucoup de chance de noté plusieurs pertes du casino lorsqu'il y a un grand nombre de joueurs et lorsqu'on se plonge dans un temps plus long. On est donc presque sûr que le casino sera ruiné au fil de plusieurs années.

3. Cas $\alpha > \mu$

$$\alpha > \mu,\ alors\ \forall y \geq 0,\ r(y)\ \leq\ e^{-Ay}\ (majoration)$$

Graphe :

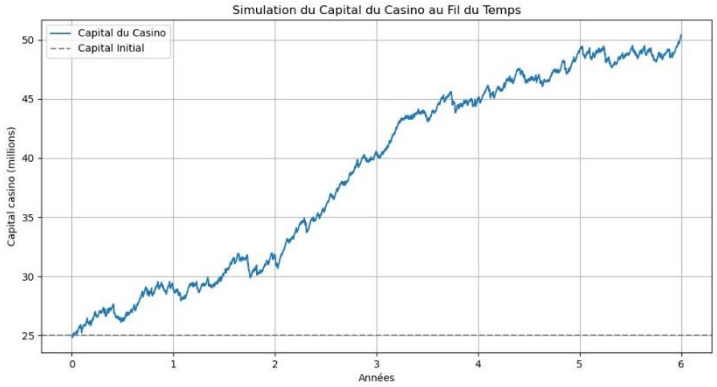

Figure 6. Capital du Casino au Fil du Temps avec α > μ

Analyse : Lorsque $\alpha > \mu$ **,** le graphe illustre clairement l'évolution du capital du casino. Ceci veut donc dire que si le gérant du casino choisi un paramètre plus grand que l'espérance de gain des joueurs, il est possible qu'il connaisse des moments de chute d'intérêt mais il ne sera jamais en faillite. Il faut aussi noter que plus le gérant choisit un paramètre grand plus le capital grandit. Le gérant a donc intérêt à choisir un paramètre plus grand que l'espérance de gain des joueurs, mais ce paramètre ne doit pas non plus être très supérieur ceci afin de garder et d'attirer de nouveau joueurs.

4. Théorème 1

$$r(y) = \frac{e^{-(\gamma - 1/\alpha)y}}{\alpha\gamma}, \quad \forall y \geq 0 \; avec \; \gamma = \frac{1}{\mu} > \frac{1}{\alpha}$$

Graphe :

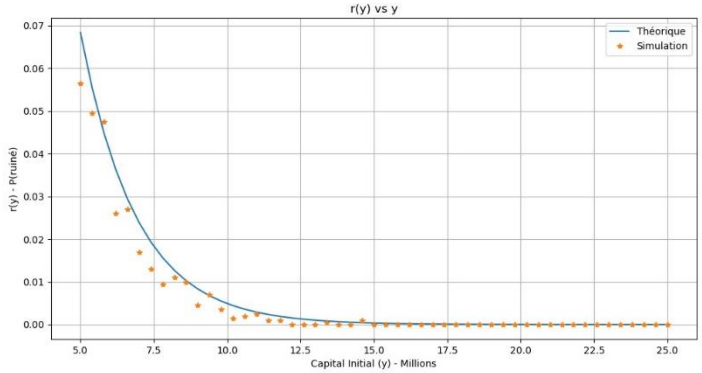

Figure 7. Probabilité d'être ruiné r(y) théorique vs r(y) simulation

Analyse : L'examen du graphique met en évidence une relation décroissante entre le capital initial et la probabilité de ruine. Il est visuellement perceptible que plus le capital initial est important, moins la probabilité de se retrouver ruiné est élevée.

La version simulée de r(y) a été obtenue en générant 2000 scénarios différents, avec la condition que le paramètre α soit supérieur à μ, conformément à notre modèle. L'emploi de la méthode de Monte-Carlo a permis d'estimer $E[1_A]$, où $A = \{ \omega \in \Omega: \exists t < t_{max} \mid Y\_t(\omega) < 0 \}$ est l'événement de la ruine.

La concordance entre la courbe théorique et les résultats de simulation est remarquable, bien que les données simulées tendent à sous-estimer la probabilité de ruine par rapport au modèle théorique. Cette

divergence pourrait être expliquée par les limitations inhérentes aux simulations numériques, telles que les hypothèses simplificatrices ou le nombre fini de scénarios considérés.

III. Modélisation et réflexions :

1. Pertinence du modèle :

Dans le modèle, les gains du casino sont modélisés comme linéaires dans le temps αt. Cette simplification pourrait ne pas refléter la réalité où les gains du casino peuvent être influencés par de nombreux facteurs tels que les variations saisonnières, les événements spéciaux, ou les changements dans les habitudes des joueurs. Une analyse plus réaliste pourrait inclure une modélisation stochastique ou des variations cycliques des gains.

Le modèle suppose que les gains des joueurs se produisent selon un processus de Poisson, impliquant des temps d'inter-arrivées exponentiels. Cela suppose une absence de mémoire, où l'événement suivant ne dépend pas du temps écoulé depuis le dernier. Dans la réalité, les comportements des joueurs peuvent ne pas être entièrement indépendants ou sans mémoire.

2. Pourquoi n'observe-t-on jamais de ruine en pratique ?

Les casinos réels ont rarement une « ruine » car ils sont conçus pour avoir un avantage statistique dans presque tous les jeux ($\alpha > \mu$). De plus, ils gèrent un grand nombre de jeux et de joueurs, ce qui, selon la loi des grands nombres, stabilise leurs gains autour d'une moyenne attendue. Les casinos gèrent également activement leurs risques, limitant les mises et ajustant les règles pour maintenir leur avantage.

CONCLUSION :

En conclusion de cette exploration fascinante du modèle de processus aléatoire appliqué au capital d'un casino, notre étude a révélé des éléments intéressants concernant la gestion financière dans l'industrie du jeu. À travers la construction et la simulation numérique de notre modèle, nous avons observé avec satisfaction une robustesse surprenante du casino face aux aléas du marché.

Les résultats de notre analyse suggèrent que, dans notre contexte spécifique, le casino affiche une capacité impressionnante à résister, indiquant ainsi que des stratégies de gestion des risques bien conçues peuvent véritablement atténuer la probabilité de ruine. Cette conclusion positive (pour le casino) souligne l'importance de la prudence dans la gestion financière.

Cependant, il est essentiel de rappeler que notre modèle repose sur des hypothèses spécifiques et des paramètres choisis délibérément. Les conditions réelles peuvent être plus complexes, et l'application de ces résultats doit être empreinte de réalisme. Notre étude, bien qu'offrant des perspectives encourageantes, nous incite également à rester attentifs aux nuances du monde réel.

En définitive, cette exploration éveille notre curiosité sur les multiples facettes de la gestion financière dans un environnement aléatoire.

ANNEXE :

Code:

```
#Libs

import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats


#%% Justification des paramètres par rapport au problème réel

Y0 = np.linspace(5, 25, 51) * 1e6 #capital entre 5 - 25 millions
alpha = 0.1 * 1e6 # le casino gagne 0.1 million per jour (100k/jour)
mu = alpha * (1 - 0.05)  #alpha (5% >) mu
mu = alpha * (1 + 0.05)  #alpha (5% <) mu
mu = alpha * (1 - 0.00)  #alpha = mu
annees = 6 #temps arbitraire que nous avons choisi pour visualiser la simulation

#%% modèle simulation individuel

### Paramètres du modèle

Y0 = 25 *1e6 # Capital initial du casino
alpha = 0.1 *1e6   # Taux de gain du casino par unité de temps
mu = alpha * (1 - 0.05)    # Moyenne des gains des joueurs (distribution exponentielle)
lambda_gain = 1 / mu  # Paramètre lambda pour la loi exponentielle des gains (variable gamma)
lambda_temps_gains = 1 # Paramètre lambda pour la loi exponentielle des temps de gains
annees = 6 #temps de simulation
#-----------------------------------------------------------------------------------------

### Simuler un processus de Poisson pour les instants de gain

temps_total = annees * 365 # Durée totale de la simulation

#pour s'assurer que la somme cumulée est toujours supérieure au temps de simulation
size_gains = (lambda_temps_gains) * (temps_total + 100)

#somme chi_i -> T_i
temps_gains = np.random.exponential(scale = 1/lambda_temps_gains, size = size_gains).cumsum()
temps_gains = temps_gains[temps_gains <= temps_total]

#-----------------------------------------------------------------------------------------

### Simuler les gains des joueurs
gains_joueurs = np.random.exponential(scale=1/lambda_gain, size=len(temps_gains)) #variable X_i

#-----------------------------------------------------------------------------------------

### Calculer le capital du casino au fil du temps

temps = np.arange(temps_total + 1) # Durée totale de la simulation
capital_casino = np.zeros_like(temps) #vecteur composé de zéros pour stocker l'évolution du
capital
index_gains = 0
somme_gains = 0

for i, t in enumerate(temps):

    while index_gains < len(temps_gains) and temps_gains[index_gains] <= t:
        # cond 1: Éviter un Débordement de l'Index
        # cond 2: Condition N_t definition
        somme_gains += gains_joueurs[index_gains]
        index_gains += 1

    #somme_gains accumulés jusqu'à présent t
    capital_casino[i] = Y0 + alpha * t - somme_gains
```

```
    if capital_casino[i] < 0:
        break

#-------------------------------------------------------------------------------------------

### Visualiser la simulation

annees_temps = temps / 365
capital_casino_millions = capital_casino  #/ 1e6

plt.figure(figsize=(12, 6))
plt.plot(annees_temps, capital_casino_millions, label='Capital du Casino')
plt.axhline(Y0 / 1e6, color='grey', linestyle='--', label='Capital Initial')
plt.title('Simulation du Capital du Casino au Fil du Temps')
plt.xlabel('Années')
plt.ylabel('Capital casino (millions)')
plt.legend()
plt.grid(True)
plt.show()


#%% Fonctions
#-------------------------------------------------------------------------------------------

### Probabilité

# Définition de la probabilité exacte si les gains des joueurs
# sont distribués de manière exponentielle
def r(Y0,mu,alpha):
    gamma = 1/mu
    return np.exp(-Y0 * (gamma - (1/alpha)))/(alpha * gamma)


# essai de une simulation de l'évolution du capital du casino.
def test_simulation(Y0,mu,alpha,annees):

    lambda_gain = 1 / mu  # Paramètre lambda pour la loi exponentielle des gains (variable gamma)
    lambda_temps_gains = 1 # Paramètre lambda pour la loi exponentielle des temps de gains
    #-------------------------------------------------------------------------------------------

    ### Simuler un processus de Poisson pour les instants de gain

    temps_total = annees * 365 # Durée totale de la simulation
    #pour s'assurer que la somme cumulée est toujours supérieure au temps de simulation
    size_gains = (lambda_temps_gains) * (temps_total + 100)
    #somme chi_i -> T_i
    temps_gains = np.random.exponential(scale = 1/lambda_temps_gains, size = size_gains).cumsum()
    temps_gains = temps_gains[temps_gains <= temps_total]

    #-------------------------------------------------------------------------------------------

    ### Simuler les gains des joueurs
    gains_joueurs = np.random.exponential(scale=1/lambda_gain, size=len(temps_gains)) #variable
X_i

    #-------------------------------------------------------------------------------------------

    ### Calculer le capital du casino au fil du temps

    temps = np.arange(temps_total + 1)
    capital_casino = np.zeros_like(temps) #vecteur composé de zéros pour stocker l'évolution du
capital
    index_gains = 0
    somme_gains = 0

    for i, t in enumerate(temps):

        while index_gains < len(temps_gains) and temps_gains[index_gains] <= t:
            # cond 1: Éviter un Débordement de l'Index
```

```
            # cond 2: Condition N_t definition
            somme_gains += gains_joueurs[index_gains]
            index_gains += 1

        #somme_gains accumulés jusqu'à présent t
        capital_casino[i] = Y0 + alpha * t - somme_gains

        if capital_casino[i] < 0:
            return 1

    return 0


# r(y) = P( \exists t \in R+ t.q. Y_t < 0 | Y_0 = y )
def r_simulation(Y0,mu,alpha,annees,nombre_tests):

    tests = np.zeros(nombre_tests) #vecteur composé de zéros pour stocker si Y_t < 0

    for i in range(nombre_tests):

        tests[i] = test_simulation(Y0, mu, alpha, annees) #stoker des tests

    return np.sum(tests)/nombre_tests #return probabilité




#%% probabilité theorique vs simulation


Y0 = np.linspace(5, 25, 51) * 1e6 #capital entre 5 - 25 millions
alpha = 0.1 * 1e6
 # le casino gagne 0.1 million per jour (100k)
mu = alpha * (1 - 0.05) #le casino perds 5% du alpha, ce qui gagnent des joueurs
annees = 5

proba_ruine = r(Y0,mu,alpha)
proba_ruine_simulation = np.zeros_like(Y0)

for i, y in enumerate(Y0):

    proba_ruine_simulation[i] = r_simulation(y, mu, alpha, annees, 2000)
    print(i)

#%% affichage probabilité theorique vs simulation

Y0_millions = Y0/1e6


plt.figure(figsize=(12, 6))
plt.plot(Y0_millions, proba_ruine, label='Théorique')
plt.plot(Y0_millions, proba_ruine_simulation, label='Simulation', marker='*', linestyle = '')
plt.title('r(y) vs y [alpha = 0.1 M et mu = 95%alpha]')
plt.xlabel('Capital Initial (y) - Millions')
plt.ylabel('r(y) - P(ruiné)')
plt.legend()
plt.grid(True)
plt.show()



#%% version 2 simulation pour montrer alpha>mu, alpha = mu et alpha<mu

def casino_simulation(Y0,mu,alpha,annees):

    lambda_gain = 1 / mu  # Paramètre lambda pour la loi exponentielle des gains (variable gamma)
    lambda_temps_gains = 1 # Paramètre lambda pour la loi exponentielle des temps de gains
    #-----------------------------------------------------------------------------------------
```

```
    ### Simuler un processus de Poisson pour les instants de gain

    temps_total = annees * 365 # Durée totale de la simulation
    #pour s'assurer que la somme cumulée est toujours supérieure au temps de simulation
    size_gains = (lambda_temps_gains) * (temps_total + 100)
    #somme chi_i -> T_i
    temps_gains = np.random.exponential(scale = 1/lambda_temps_gains, size = size_gains).cumsum()
    temps_gains = temps_gains[temps_gains <= temps_total]

    #-------------------------------------------------------------------------------------------

    ### Simuler les gains des joueurs
    gains_joueurs = np.random.exponential(scale=1/lambda_gain, size=len(temps_gains)) #variable
X_i

    #-------------------------------------------------------------------------------------------

    ### Calculer le capital du casino au fil du temps

    temps = np.arange(temps_total + 1)
    capital_casino = np.zeros_like(temps) #vecteur composé de zéros pour stocker l'évolution du
capital
    index_gains = 0
    somme_gains = 0

    for i, t in enumerate(temps):

        while index_gains < len(temps_gains) and temps_gains[index_gains] <= t:
            # cond 1: Éviter un Débordement de l'Index
            # cond 2: Condition N_t definition
            somme_gains += gains_joueurs[index_gains]
            index_gains += 1

        #somme_gains accumulés jusqu'à présent t
        capital_casino[i] = Y0 + alpha * t - somme_gains

        if capital_casino[i] < 0:
            break

    return np.array(temps), np.array(capital_casino)

#%%

annees = 10
Y0 = 6 * 1e6
alpha = 0.1 * 1e6
mu = [ alpha * (1 + 0.05), alpha * (1 + 0.0), alpha * (1 - 0.05) ]

temps1, casino1 = casino_simulation(Y0, mu[0], alpha, annees)
temps2, casino2 = casino_simulation(Y0, mu[1], alpha, annees)
temps3, casino3 = casino_simulation(Y0, mu[2], alpha, annees)

x_data = np.array([temps1, temps2, temps3])
y_data = np.array([casino1, casino2, casino3])
x_data = x_data/365
y_data = y_data/1e6
titles = ['Simulation du Capital du Casino - alpha (5% <) mu', 'Simulation du Capital du Casino -
mu = alpha', 'Simulation du Capital du Casino - alpha (5% >) mu']  # Titres correspondants

# Nombre de lignes et de colonnes pour les sous-graphiques
n_rows = 3
n_cols = 1

# Création de la figure et des axes
fig, axs = plt.subplots(n_rows, n_cols, figsize=(8 * n_cols, 4 * n_rows))

if n_rows == 1 or n_cols == 1:
    axs = np.array(axs).reshape(n_rows, n_cols)

# Parcourir et créer chaque sous-graphique
```

```
for i in range(n_rows):
    for j in range(n_cols):
        index = i * n_cols + j
        if index < len(y_data):
            axs[i, j].plot(x_data[index], y_data[index])
            axs[i, j].set_title(titles[index])
            axs[i, j].set_xlabel('temps (années)')
            axs[i, j].set_ylabel('Capital Casino (millions)')

# Ajuster l'espacement
plt.tight_layout()
plt.show()


#%% Determination de C, generation de la distribution de Hn

def F_chapeau_empirique(echantillon, x):
    #Calculer la Fonction Repartion empirique à la valeur x.
    return np.mean(echantillon <= x)

def F_exponentielle(x, gamma):
    #Calculer la Fonction Repartition d'une distribution exponentielle à la valeur x
    return 1 - np.exp(-gamma * x)

def calculer_Hn(echantillon, gamma_chapeau, resolution):
    #Calculer la statistique H_n pour l'échantillon donné et le gamma estimé

    valeurs_y = np.linspace(0, max(echantillon), resolution)

    valeurs_Hn = np.zeros(resolution)

    for i, y in enumerate(valeurs_y):
        valeurs_Hn[i] = abs(F_chapeau_empirique(echantillon, y) - F_exponentielle(y,
gamma_chapeau))

    return max(valeurs_Hn)

# Simulation d'un échantillon exponentiel
alpha = 0.1 * 1e6
mu = alpha * (1 - 0.05)
n = 1000  # taille de l'échantillon
gamma_vrai = 1/mu  # paramètre réel de la distribution exponentielle
echantillon = np.random.exponential(1/gamma_vrai, n)

# Estimation de gamma à partir de l'échantillon
gamma_chapeau = n / np.sum(echantillon)

# Calcul de H_n (pour un seul test)
resolution = 500 #résolution de la grille sur laquelle H_n est calculé
Hn = calculer_Hn(echantillon, gamma_chapeau, resolution)

# Affichage des résultats
print(f'Hn = {Hn}')
print(f'gamma chapeau = {gamma_chapeau}')
print(f'gamma_vrai = {gamma_vrai}')


#%%
#pour plusieurs tests

# Nombre de simulations
nombre_simulations = 5000
n = 1000  # taille de l'échantillon


# Simulation des valeurs de H_n sous l'hypothèse que la distribution est exponentielle
simulations_Hn = np.zeros(nombre_simulations)

for i in range(nombre_simulations):
```

```
    # Simulation d'un échantillon de la distribution exponentielle
    echantillon_simule = np.random.exponential(scale = 1/gamma_vrai, size = n)

    # Calcul de la fonction répartition empirique de l'échantillon simulé
    gamma_chapeau_simule = n / np.sum(echantillon_simule)
    Hn_simule = calculer_Hn(echantillon_simule, gamma_chapeau_simule, resolution)
    simulations_Hn[i] = Hn_simule
    print(i)

#%%

# Définition du niveau de signification
alpha = 0.05

# Détermination du seuil C comme le quantile (1-alpha) des valeurs simulées de H_n
C = np.quantile(simulations_Hn, 1 - alpha)
print(C)

shape, loc, scale = stats.gamma.fit(simulations_Hn, floc=0)
C_gamma = stats.gamma.ppf(1-alpha, shape, loc=0, scale=scale)
print(C_gamma)



#%% Affichage de la distribution de H_n

x_valeurs = np.linspace(0, max(simulations_Hn), 1000)
distribution_gamma = stats.gamma.pdf(x_valeurs, shape, loc=0, scale=scale)


plt.hist(simulations_Hn, density=True)
plt.plot(x_valeurs, distribution_gamma, label = f"loi Gamma k = {shape: .2f}, theta =
{scale: .4f}")
plt.title("Distribution de H_n pour des échantillons exponentiels simulés")
plt.xlabel("Valeurs de H_n")
plt.ylabel("Fréquence")
plt.legend(loc='upper right')
plt.show()

#%% tendance vers zéro de Hn quand n vers le infini


# Paramètres de la simulation
tailles_echantillon = np.arange(100, 5001, 100)  # Tailles d'échantillon croissantes de 100 à
5000
# Simulation pour différentes tailles d'échantillon

valeurs_Hn = np.zeros(len(tailles_echantillon))
for i, n in enumerate(tailles_echantillon):

    # Simulation d'un échantillon de la distribution exponentielle
    echantillon_simule = np.random.exponential(scale = 1/gamma_vrai, size = n)

    # Calcul de la fonction répartition empirique de l'échantillon simulé
    gamma_chapeau_simule = n / np.sum(echantillon_simule)
    Hn_simule = calculer_Hn(echantillon_simule, gamma_chapeau_simule, resolution)
    valeurs_Hn[i] = Hn_simule


# Affichage du graphique
plt.plot(tailles_echantillon, valeurs_Hn, marker='*', linestyle = '-', color = 'g')
plt.title("Convergence de H_n avec la taille de l'échantillon")
plt.xlabel("Taille de l'échantillon (n)")
plt.ylabel("Valeur de H_n")
plt.grid(True)
plt.show()
```