Filière MP - ENS de Paris-Saclay, Lyon, Rennes et Paris - Session 2019

Page de garde du rapport de TIPE

NOM : FOURNIER	Prénoms : Victor, Jules, Alexandre
Classe : MP*4	
Lycée : Louis-le-Grand	Numéro de candidat : 3 2 1
Ville : Paris	

Concours auxquels vous êtes admissible, dans la banque MP inter-ENS (les indiquer par une croix) :

ENS Cachan	MP - Option MP		MP - Option MPI	X
	Informatique			
ENS Lyon	MP - Option MP		MP - Option MPI	X
	Informatique - Option M		Informatique - Option P	
ENS Rennes	MP - Option MP		MP - Option MPI	X
	Informatique			
ENS Paris	MP - Option MP		MP - Option MPI	
	Informatique			

Matière dominante du TIPE (la sélectionner d'une croix inscrite dans la case correspondante) :

Informatique		Mathématiques	X	Physique	

Titre du TIPE : Modélisations des chaînes de lettres

Nombre de pages (à indiquer dans les cases ci-dessous) :

Texte	9,5 (11 000 caractères)	Illustration	16 (3 400 caractères)	Bibliographie	0,5 (600 caractères)

Résumé ou descriptif succinct du TIPE (6 lignes, maximum) :

Depuis leurs débuts dans les années 1920, on a pu observer de nombreuses variantes de chaînes de lettres d'argent et de systèmes de vente pyramidale. Ce TIPE détaille plusieurs modèles : Une simulation informatique qui reprend les règles de la chaîne de lettre du « Cercle d'or » qui a circulé aux Etats-Unis en 1978, une étude générique des processus de branchement, de la vente pyramidale, et des chaînes de lettres, ainsi qu'un modèle continu des systèmes de Ponzi. Même si le comportement général est le même pour chacune des situations, on calcule des constantes spécifiques pour mettre en avant les points communs et les différences.

À Paris

Le 08/06/19

Signature du (de la) candidat(e)

Signature du professeur responsable de la classe préparatoire dans la discipline

Cachet de l'établissement

Lycée Louis le Grand
075 0655 E
123, rue St Jacques
75231 PARIS CEDEX 05
Tél. 01 44 32 82 00

La signature du professeur responsable et le tampon de l'établissement ne sont pas indispensables pour les candidats libres (hors CPGE).

Modélisations des chaînes de lettres

Victor Fournier N°321

10/06/2019

Abstract :

Money chain letters and pyramid schemes have taken various forms since their beginnings in the 1920s. This work reviews several models : a computer simulation corresponding to the "Circle of Gold" letter that circulated in the US in 1978, a general study of Galton Watson processes, pyramid schemes and chain letters, and a continuous model of Ponzi schemes. Although the general behavior follows a common pattern, constants are calculated in order to draw out similarities and differences.

1 Introduction

1.1 Un exemple de chaîne de lettres

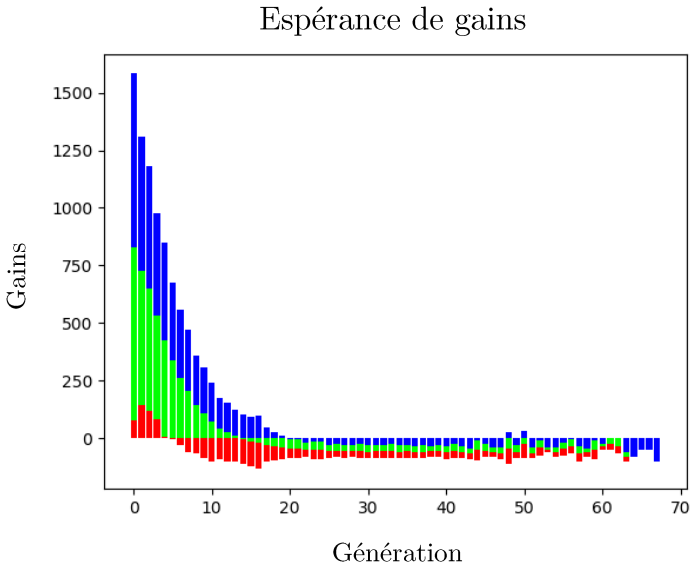
En Californie, en 1976, débute le « Cercle d'or », une chaîne de lettres qui a traversé les Etats-Unis jusqu'à New York et qui est basée sur le principe de la vente pyramidale : la chaîne est telle que les lettres sont des listes de 12 noms. Un nouveau participant donne 50 dollars au détenteur d'une lettre pour la lui acheter. Ensuite, il envoie 50 dollars à la personne dont le nom est au sommet de la liste et raye ce nom. Il inscrit alors son nom en bas de la liste. A ce stade, il a déboursé 100 dollars. Il peut faire une copie de la lettre et tenter de vendre les deux exemplaires à son tour. S'il y parvient, il récupère les 100 dollars déboursés, et si chacune des lettres se vend à son tour en deux exemplaires à chaque fois, au bout de 12 générations de ventes écoulées, son nom apparaîtra au sommet des 4 096 lettres issues de celle qu'il a acheté, ce qui lui fait un gain potentiel de 204 800.

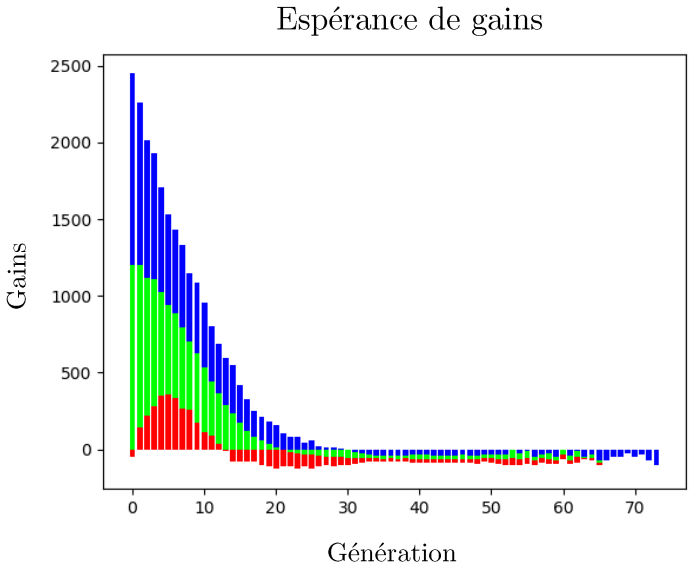
1.2 Simulation informatique

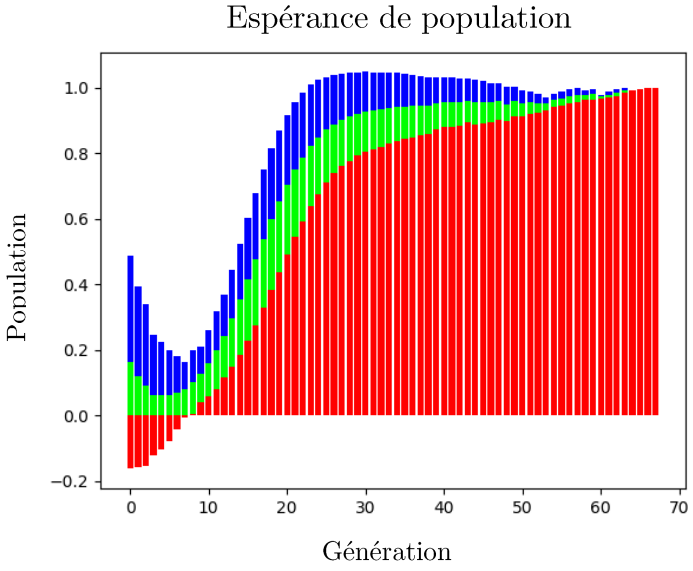
J'ai codé une simulation informatique correspondant à la chaîne de lettres du Cercle d'or. L'objectif du TIPE est de retrouver, en étudiant des situations similaires, les comportements observés dans la simulation. Par exemple, on affiche avec la librairie Python Matplotlib, une moyenne sur 100 et 500 simulations de l'espérance des gains et de la population en fonction de la génération :

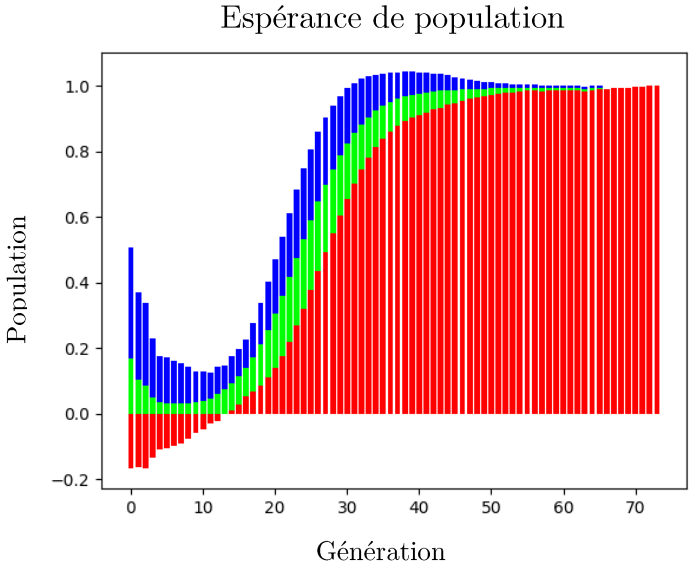
100 simulations

500 simulations









Graphiquement, on trouve la génération où l'espérance des gains devient négative. Ensuite, pour cette même génération, on regarde quelle fraction des participants a déjà rejoint la chaîne de lettres. Dans le deux cas, on obtient un seuil de l'ordre de **0.36**, donc il faut faire partie des **36%** premiers pour avoir une espérance positive.

2 Chaînes de lettres avec plusieurs participations

2.1 Description du modèle

On reprend le principe de la chaîne du Cercle d'or mais on laisse la possibilité aux participants ayant vendu leurs deux lettres d'en racheter. L'espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$ envisagé correspond à l'ensemble des scénarios possibles de vente de lettres au sein d'une population P_{tot} fixée. $\mathbb{P}$ est défini de sorte que, à tout instant, chaque personne ayant une lettre ait la même probabilité de la vendre. On définit les variables aléatoires suivantes : (U_k, V_k, W_k) qui correspondent au nombre de participants ayant vendu respectivement 0, 1 et 2 lettres au moment où, en tout, k lettres ont déjà été vendues. Initialement on pose $(U_0, V_0, W_0) = (0,0,1)$.

2.2 Evolution de la chaîne

On écrit la vente d'une lettre

$$U_{k+1} + V_{k+1} + W_{k+1} = U_k + V_k + W_k + 1$$
$$\Rightarrow\ U_n + V_n + W_n = U_0 + V_0 + W_0 + n$$
$$\Rightarrow\ V_n = n + 1 - U_n - W_n$$

Si un participant ayant vendu une lettre vend sa deuxième, on obtient un nouveau participant de type 0 et de type 2

$$U_{k+1} - U_k = W_{k+1} - W_k = 1$$

Sinon, un participant vend sa première lettre donc le nombre de participants ayant vendu 0 et 2 lettres reste inchangé

$$U_{k+1} - U_k = W_{k+1} - W_k = 0$$

Dans les deux cas on a

$$U_{k+1} - U_k = W_{k+1} - W_k$$

On utilise un télescopage

$$\sum_{k=0}^{n-1} (U_{k+1} - U_k) = \sum_{k=0}^{n-1} (W_{k+1} - W_k)$$
$$\Rightarrow\ W_n = U_n - U_0 + W_0 = U_n - 1$$

On peut donc exprimer le triplet (U_k, V_k, W_k) uniquement en fonction de U_k

$$(U_k, V_k, W_k) = (U_k, k + 2 - 2U_k, U_k - 1)$$

Le nombre de participants n'ayant pas vendu toutes leurs lettres est majoré

$$U_k + V_k \leq P_{tot} \Rightarrow k + 2 - U_k \leq P_{tot} \Rightarrow U_k \geq 2 + k - P_{tot}$$

$$V_k \geq 0 \Rightarrow k + 2 - 2U_k \geq 0 \Rightarrow U_k \leq \frac{k}{2} + 1$$

On obtient un encadrement de U_k

$$k + 2 - P_{tot} \leq U_k \leq \frac{k}{2} + 1$$

En particulier on doit avoir

$$k + 2 - P_{tot} \leq \frac{k}{2} + 1 \Rightarrow k \leq 2P_{tot} - 2$$

Ce qui donne une majoration du nombre total de participants k et qui démontre que la chaîne de lettre s'effondre. On appelle temps d'arrêt la variable aléatoire

$$K = \min\{k, \quad U_k + V_k = N\}$$

U_k augmente ⇔ une personne de V_k a vendu sa lettre. Puisque chaque participant ayant une lettre à vendre, soit $U_k + V_k$ a la même chance de vendre sa lettre, on a

$$\mathbb{P}(U_{k+1} = u | U_k = u) = \frac{U_k}{U_k + V_k} = \frac{u}{k + 2 - u}$$

2.3 Équilibre asymptotique

On considère asymptotiquement un équilibre

$$\mathbb{P}(U_{k+1} = U_k + 1) \xrightarrow[k \to +\infty]{} \alpha$$

La proportion des k tels que $U_{k+1} = U_k + 1$ est $\frac{U_k}{k+1}$ donc d'après la loi faible des grand nombres, on a

$$\frac{U_k}{k+1} \xrightarrow[k \to +\infty]{} \alpha$$

$$\mathbb{P}(U_{k+1} = U_k + 1) = \frac{V_k}{U_k + V_k} = \frac{k + 2 - 2U_k}{k + 2 - U_k} \underset{+\infty}{\sim} \frac{1 - 2\frac{U_k}{k}}{1 - \frac{U_k}{k}} \Rightarrow \alpha = \frac{(1 - 2\alpha)}{1 - \alpha}$$

$$\alpha = \frac{\left(3 - \sqrt{5}\right)}{2} \cong 0.382$$

Si on écrit que α est la proportion de U_k et que k est le nombre des participants mais aussi le nombre de lettres vendues, on obtient le système

$$\begin{cases} U_k = (k+1)\alpha \underset{+\infty}{\sim} k\alpha \\ U_k + V_k + W_k = k+1 \underset{+\infty}{\sim} k \\ 2W_k + V_k = k \end{cases}$$

On en déduit les proportions à l'arrivée

$$\left(\frac{U_k}{k}, \frac{V_k}{k}, \frac{W_k}{k}\right) \xrightarrow[k\to+\infty]{} (0.382, 0.236, 0.382)$$

Par définition du temps d'arrêt K, on a

$$U_K + V_K = P_{tot} \Rightarrow K = \frac{P_{tot}}{(0.382 + 0.236)} = 1.62P_{tot}$$

On vérifie à la fin que les participants supplémentaires sont bien à ceux qui ont vendu 2 lettres $W_K = 0.62P_{tot} = 0.62(U_K + V_K)$ satisfait les proportions asymptotiques obtenues précédemment $0.382 = 0.62 * (0.382 + 0.236)$

3 Vente pyramidale

3.1 Description du modèle

On considère qu'il faut payer un coût c pour entrer dans un club, et que c'est le membre qui recrute le nouvel arrivant qui empoche c. L'espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$ envisagé correspond à l'ensemble des scénarios possibles de recrutement au sein d'une population P_{tot} fixée. $\mathbb{P}$ est défini de sorte qu'à tout instant, chaque membre ait la même probabilité de recruter un nouveau participant. On définit la variable aléatoire X_i du nombre de personnes recrutées à l'étape i, et R la variable aléatoire du nombre totale de personnes recrutées. On note k le rang d'entrée dans la vente pyramidale, la probabilité de recruter le prochain participant est $\frac{1}{k}$, puis $\frac{1}{k+1}$, etc.

$$R = \sum_{i=k}^{N-1} X_i\,, \qquad X_i \sim \mathcal{B}\left(\frac{1}{i}\right)$$

3.2 Calcul du rang d'entrée critique

L'espérance des gains est

$$\mathbb{E} = c(\mathbb{E}(R) - 1)$$

Par linéarité de l'espérance, on a

$$\mathbb{E}(R) \underset{P_{tot}\to+\infty}{\sim} \ln\left(\frac{P_{tot}}{k}\right) \underbrace{\asymp}_{k(P_{tot})} \ln(P_{tot})$$

Le résultat dépend uniquement du quotient $\frac{P_{tot}}{k}$

Le seuil pour que l'espérance des gains soit positive avec P_{tot} grand devient

$$\frac{k}{P_{tot}} < e^{-1} \cong 0.368$$

On connait aussi l'espérance à l'origine :

$$\mathbb{E} \underset{P_{tot}\to+\infty}{\sim} c\ln(P_{tot})$$

3.3 Proportions asymptotiques

La probabilité que le k-ième participants ne regagne plus rien est

$$\mathbb{P}_k(0) = \prod_{i=k}^{P_{tot}-1} (1 - 1/i) = \frac{(k-1)}{(P_{tot}-1)}$$

L'espérance du nombre de participants qui perdent tout est donc

$$\sum_{k=1}^{P_{tot}} \mathbb{P}_k(0) = \frac{1}{P_{tot}-1}(1+\cdots+P_{tot}-1) = \frac{P_{tot}}{2}$$

On peut compléter le résultat précédent

Lemme 1 :

La proportion de participants qui recrutent exactement r personnes tend vers $2^{-(r+1)}$ quand $P_{tot} \to +\infty$

Démonstration 1 :

En annexe

4 Systèmes de Ponzi

4.1 Description du modèle

Les systèmes de Ponzi sont des montages financiers frauduleux où l'on promet un taux r_p plus élevé que le taux d'intérêt r_n et on utilise les capitaux des nouveaux investisseurs pour payer la différence. Pour notre étude, on sort du cadre probabiliste discret et on étudie un modèle continu d'évolution de l'ensemble du capital $S(t)$. Il y a un dépôt initial S_0 et on suppose que le flux entrant - les capitaux des nouveaux investisseurs - est exponentiel $s(t) = \frac{S_0}{1}(e^{r_i t} - 1)$. On rajoute un taux de retrait r_w constant qui s'applique au capital promis $S_f(t)$.

4.1 Taux nominaux et taux réels

Des taux d'intérêt R_n et de retrait R_w réels qui s'appliquent sur une période T vérifient

$$S(t+T) = S(t)(1+R_n)(1-R_w)$$

On obtient une formule générale de la forme

$$S(t) = S_0(1+R_n)^{\frac{t}{T}}(1-R_w)^{\frac{t}{T}}$$

Des taux d'intérêt r_n et de retrait r_w nominaux sur une période T correspondent à la limite quand $dt \to 0$ de taux réels $r_n dt$ et $r_w dt$ qui s'appliquent tous les dt. On note qu'un taux nominal a la dimension de l'inverse d'un temps.

$$S(t+dt) = S(t)(1+r_n dt)(1-r_w dt)$$

$$\Rightarrow S(t+T) = S(t)(1+r_n dt)^{\frac{T}{dt}}(1-r_w dt)^{\frac{T}{dt}} \underset{dt\to 0}{\longrightarrow} S(t)e^{T(r_n - r_w)}$$

On obtient une formule générale de la forme

$$S(t) = S_0 e^{t(r_n - r_w)}$$

On peut définir de façon équivalente les taux nominaux par l'équation différentielle

$$\frac{dS}{dt}(t) = (r_n - r_w)S(t)$$

4.2 Expression des capitaux

L'argent que les clients croient avoir sur leur compte vérifie

$$\frac{dS_f}{dt}(t) = \big(r_p - r_w\big)S_f(t) + s(t)$$

Il s'agit d'une équation différentielle que l'on résout par variation de la constante

$$S_f(t) = e^{(r_p - r_w)t}\left(S_0 + \int_0^t e^{-(r_p - r_w)u} s(u)du\right)$$

En remplaçant $s(u)$ dans l'expression on obtient

$$S_f(t) = S_0\left(-\frac{1}{r_w - r_p}\right) + S_0\left(\frac{1}{r_i + r_w - r_p}\right)e^{r_i t} + S_0\left(1 + \frac{1}{r_w - r_p} - \frac{1}{r_i + r_w - r_p}\right)e^{(r_p - r_w)t}$$

Le capital total vérifie

$$\frac{dS}{dt}(t) = r_n S(t) + s(t) - r_w S_f(t)$$

Il s'agit encore une équation différentielle que l'on résout par variation de la constante

$$S(t) = e^{r_n t}\left(S_0 + \int_0^t e^{-r_n u}\big(s(u) - r_w S_f(u)\big)du\right)$$

En remplaçant $s(u)$ et $S_f(u)$ dans l'expression on obtient le capital comme somme d'exponentielles

$$S(t) = a + be^{r_n t} + ce^{r_i t} + de^{(r_p - r_w)t}$$

Expression des constantes en annexe

En notant $W(t)$ les gains globaux on obtient l'équation puis l'expression suivante

$$W(t) = -S_0 + \int_0^t \big(r_w S_f(u) - s(u)\big)du = \alpha + \beta e^{r_i t} + \gamma e^{(r_p - r_w)}$$

Expression des constantes en annexe

4.3 Quelques valeurs numériques

Le premier zéro de $S(t)$ correspond à la date de l'effondrement du système. On utilise les taux et les montants de l'affaire Madoff (en années et milliards de dollars) pour évaluer les formules obtenues

$$r_p = 10\%, \qquad r_n = 2.5\%, \qquad S_0 = 1$$

$$r_w = ?, \qquad r_i = ?,$$

Les taux r_w et r_i sont des contraintes extérieures qui, en réalité, varient en fonction du temps (tout comme r_n) et du client. Voici les résultats obtenus pour quelques valeurs de r_i et r_w

Valeurs du premier zéro de $S(t)$ (en années)

r_w/r_i	0%	2%	4%	6%	8%
1%	31,3	55,6	63,6	73,9	95,8
3%	20,3	40,6	47,4	54,6	67,6
5%	16,2	34,6	41,1	47,3	57,8
7%	13,8	31,0	37,4	43,1	52,6
9%	12,2	28,6	35,0	40,4	49,2

Valeurs de $W\left(\frac{3}{4}t_f\right)$ où t_f est le premier zéro de $S(t)$ calculé précédemment (en milliards de dollars)

r_w/r_i	0%	2%	4%	6%	8%
1%	−34,0	−48,9	−17,7	142,0	2213,7
3%	−28,7	−54,7	−55,4	−38,0	90,5
5%	−32,2	−67,5	−76,0	−76,5	−49,5
7%	−45,8	−102,3	−120,9	−133,3	−139,4
9%	−121,8	−185,3	−347,3	−396,9	−469,7

L'arnaque de Bernard Madoff a duré 40 ans et a fait perdre environ 50 milliards de dollars. Elle correspond à la case en bleu donc aux taux $r_i = 2\%$ et $r_w = 1\%$. Les cases en rouge sont celles où les participants ont globalement un gain positif au moment de l'effondrement. Naturellement, elles correspondent à un investissement élevé et un retrait faible, ce qui permet au système de ne pas s'effondrer avant longtemps ($t_f > 65$ ans).

5 Processus de branchement

5.1 Description du modèle

Les processus de branchement ont été introduits pour répondre à la question de la survie d'un nom de famille au fil des générations. Il repose sur le fait que les probabilités de reproduction sont constantes. L'espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$ envisagé correspond à l'ensemble des descendances possibles d'un individu. On reste dans le cadre du Cercle d'or avec 2 lettres à vendre en majorant le nombre de fils directs par 2. $\mathbb{P}$ est défini de sorte que quelle que soit la génération, la probabilité d'avoir au moins 1 fils est $p1$ et celle d'avoir deux fils est $p2$. On définit Z_g la variable aléatoire qui donne le nombre de descendants au bout de la génération g, on a en particulier $Z_0 = 1$.

5.2 Fonction génératrice des probabilités

On introduit la fonction génératrice des probabilités

$$F_X(t) = \mathbb{E}(t^X) = \sum_{k=0}^{+\infty} \mathbb{P}(X = k)t^k$$

Elle permet de connaitre facilement $\mathbb{P}(X = 0) = F_X(0)$ ainsi que le k-ième moment factoriel

$$\mu_k = \mathbb{E}\big(X(X-1)\dots(X-k+1)\big) = F_X^{(k)}(1)$$

Si les $(X_k)_{k\leq n}$ sont mutuellement indépendantes, on note $S_n = \sum_{k=1}^{n} X_k$ on a

$$F_{S_n}(t) = \prod_{k=1}^{n} F_{X_i}(t)$$

Si les $(X_k)_{k\leq n}$ sont identiquement distribuées, on a

$$F_{S_n}(t) = \Big(F_{X_1}(t)\Big)^n$$

Lemme 2 :

Pour un processus de branchement, on a pour $n \in \mathbb{N}$

$$F_{Z_n}(t) = F_{Z_1}^{\circ n}(t)$$

Démonstration 2 :

En annexe

J'ai codé un programme Python pour calculer formellement les coefficients de l'itéré de $F_{Z_1}(t) = p_2 t^2 + p_1 t + (1 - p_1 - p_2)$ avec la librairie Sympy. Ceci permet d'avoir $\mathbb{P}(Z_n = k)$, qui est le coefficient de X^k dans le polynôme $F_{Z_n}(X)$ pour des faibles valeurs de n (sinon le résultat est trop gros pour être affiché). On peut calculer explicitement les valeurs de $\mathbb{E}(Z_n)$ et $\mathbb{V}(Z_n)$. On note $\mathbb{E}(Z_1) = m$ et $\mathbb{V}(Z_1) = \sigma^2$

$$\mathbb{E}(Z_n) = F'_{Z_n}(1) = \left(F_{Z_1}^{\circ n}\right)'(1) = \prod_{k=0}^{n-1}\left(F'_{Z_1} \circ F_{Z_1}^{ok}\right)(1) \underbrace{=}_{F_{Z_1}(1)=1} \left(F'_{Z_1}(1)\right)^n = m^n$$

$$\mathbb{V}(Z_n) = F''_{Z_n}(1) + \mathbb{E}(Z_n) - \mathbb{E}(Z_n)^2$$

$$F''_{Z_n}(1) = \sum_{i=0}^{n-1}\left(F'_{Z_1} \circ F_{Z_1}^{oi}\right)'(1) \prod_{\substack{k=0 \\ k\neq i}}^{n-1} F'_{Z_1} \circ F_{Z_1}^{ok}(1)$$

$$= m^{n-1} \sum_{i=0}^{n-1}\left(\prod_{l=0}^{i-1}\left(F'_{Z_1} \circ F_{Z_1}^{\circ l}\right)(1)\right)\left(F''_{Z_1} \circ F_{Z_1}^{\circ i}(1)\right) = m^{n-1}(\sigma^2 + m - m^2)\sum_{i=0}^{n-1} m^i$$

$$\underbrace{=}_{\substack{m\neq 1 \\ m=1}} \begin{cases} m^{n-1}(\sigma^2 + m - m^2)\left(\dfrac{1-m^n}{1-m}\right) = \begin{cases} \sigma^2 m^{n-1}\left(\dfrac{1-m^n}{1-m}\right) + m^n - m^{2n} \\ \sigma^2 n \end{cases} \\ m^{n-1}(\sigma^2 + m - m^2)n \end{cases}$$

$$\mathbb{V}(Z_n) \underbrace{=}_{\substack{m\neq 1 \\ m=1}} \begin{cases} \sigma^2 m^{n-1}\left(\dfrac{1-m^n}{1-m}\right) \\ \sigma^2 n \end{cases}$$

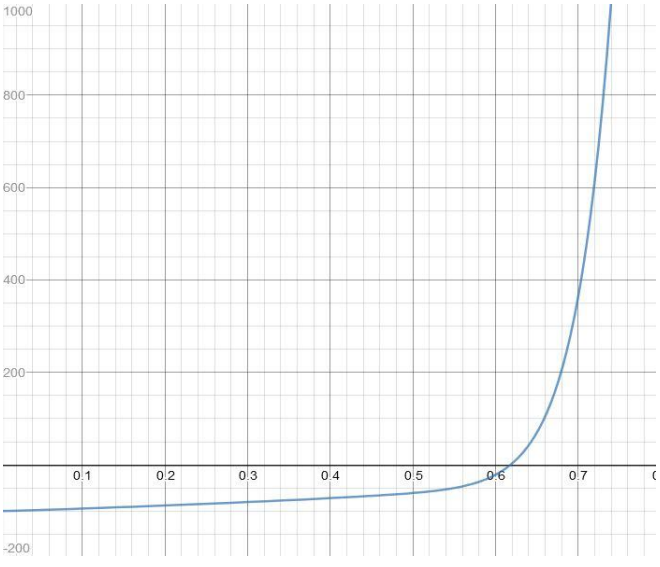
Si on considère que la vente de la première et de la deuxième lettre sont indépendantes, on a $p_2 = p_1^2$ et on peut trouver une probabilité critique à l'origine

$$\mathbb{E} = 50\big(\mathbb{E}(Z_1) + \mathbb{E}(Z_{12})\big) - 100 = 50(m + m^{12} - 2)$$

$$m = p_1 + p_2 = p_1(1 + p_1)$$

Il faut donc $m > 1$ soit $p_1^2 + p_1 - 1 > 0 \Rightarrow p_1 > \frac{\sqrt{5}-1}{2} \sim 0.61$

On peut tracer $\mathbb{E} = f(p_1)$ et voir la croissance rapide au delà de la probabilité critique

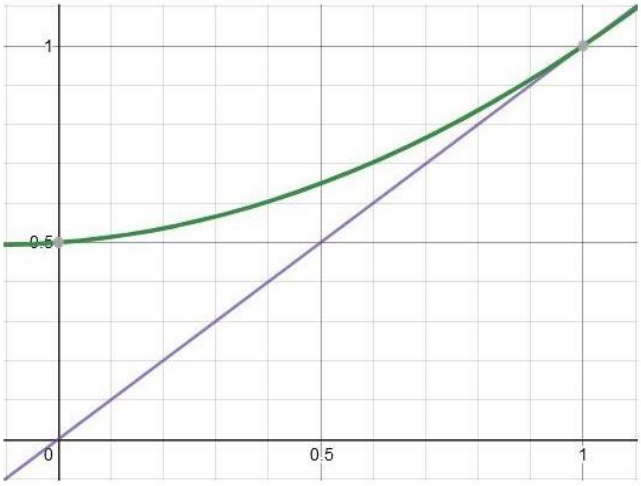


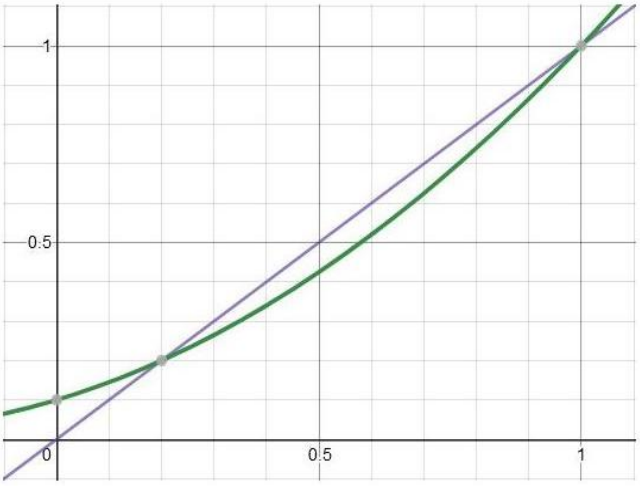
5.3 Probabilité d'extinction

La probabilité d'extinction du processus à l'étape n est $F_{Z_n}(0) = \mathbb{P}(Z_n = 0)$. Si le processus est éteint à l'étape n, alors il le sera à l'étape $n+1$ donc $\left(F_{Z_n}(0)\right)_n$ est une suite croissante majorée donc convergente vers l. l est un point fixe de F_{Z_1}, croissante convexe en tant que limite simple de fonctions croissantes convexes (sommes de monômes à coefficient positifs). Si l' est le plus petit point fixe de F_{Z_1} dans $[0,1]$ alors $0 \leq l' \Rightarrow F_{Z_n}(0) \leq l'$ et $l = l'$. On distingue deux cas :

Si $m \leq 1$ puisque F_{Z_1} convexe est au dessus de sa tangente en 1 elle est au dessus de $y = x$ et n'admet pas de point fixe dans $[0,1[$.

Si $m > 1, F_{Z_1}$ est inférieure à l'identité en 1^-, or $F_{Z_1}(0) = p_0 > 0$ donc d'après le théorème des valeurs intermédiaires, F_{Z_1} admet un point fixe $l' < 1$





Lemme 3 :

1) Si $m > 1$ alors $\mathbb{P}(Z_n > 0) \underset{+\infty}{\longrightarrow} 1 - \min_{[0,1]}\{x,\ F_{Z_1}(x) = x\}$

2) Si $m < 1$ alors $\mathbb{P}(Z_n > 0) \sim Cm^n$

3) Si $m = 1$ alors $\mathbb{P}(Z_n > 0) \sim \frac{2}{\sigma^2 n}$

Dans le cas sous-critique on peut calculer l'espérance du temps d'arrêt

$$\mathbb{E}(K) = \sum_{k=1}^{+\infty} \mathbb{P}(Z_k > 0) = \frac{C}{1 - \mathbb{E}(Z_1)}$$

Démonstration 3 :

En annexe

6 Tableau récapitulatif des constantes

Modèle	Seuil	Temps d'arrêt	Proportions à l'arrivée	Espérance à l'origine
Simulation informatique	0.36	$f(p_1, p_2)\sqrt{P_{tot}}$	72% < 0 20% 0 8% > 0	$P_{tot} - 100$ $P_{tot} \leq 1\,000$
Chaine de lettres avec plusieurs participations	$\frac{(3-\sqrt{5})}{2}$ $\cong 0.382$	$1.62N$	38.2% 0 23.6% 1 38.2% 2	
Vente pyramidale	$1/e \cong 0.368$	P_{tot}	50% 0 25% 1 25% 2 +	$c \ln P_{tot}$
Ponzi	$r_i > 6\%$ $r_w < 3\%$	~ 40 ans		$-50S_0$
Branchement	$p_1 > 0.61$	$\frac{C}{1 - \mathbb{E}(Z_1)}$		$50(m + m^{12} - 2)$

7 Annexes

7.1 Démonstrations

Démonstration 1 :

On introduit les Y_i, de Poisson, stochastiquement plus grandes que les X_i c'est à dire

$$\forall x, \qquad \mathbb{P}(Y_i \geq x) \geq \mathbb{P}(X_i \geq x)$$

On veut avoir $\mathbb{P}(Y_i = 0) = \mathbb{P}(X_i = 0) = 1 - \frac{1}{i}$ pour cela on fixe

$$e^{-\lambda_i} = 1 - \frac{1}{i} \Rightarrow \lambda_i = -\ln\left(1 - \frac{1}{i}\right)$$

On a $\mathbb{P}(Y_i = 1) = -\left(1 - \frac{1}{i}\right)\ln\left(1 - \frac{1}{i}\right) \underset{i \to +\infty}{=} \frac{1}{i} - \frac{1}{2i^2} + O\left(\frac{1}{i^3}\right)$

On définit aussi

$$S_k = \sum_{i=k}^{P_{tot}-1} X_i \preccurlyeq T_k = \sum_{i=k}^{P_{tot}-1} Y_i$$

T_k est de Poisson, de paramètre

$$\gamma_k = \sum_{i=k}^{P_{tot}-1} \lambda_i = \sum_{i=k}^{P_{tot}-1} (\ln(i) - \ln(i-1)) = \ln\left(\frac{P_{tot} - 1}{k - 1}\right)$$

Montrons que

$$\frac{1}{P_{tot} - 2} \sum_{k=2}^{P_{tot}-1} \mathbb{P}(T_k = r) \xrightarrow[P_{tot} \to +\infty]{} 2^{-(r+1)}$$

En effet,

$$\frac{1}{P_{tot} - 2} \sum_{k=2}^{P_{tot}-1} \mathbb{P}(T_k = r) = \frac{1}{P_{tot} - 2} \sum_{k=2}^{P_{tot}-1} e^{-\gamma_k} \frac{\gamma_k^r}{r!}$$

$$= \frac{1}{r!\,(P_{tot} - 2)} \sum_{k=2}^{P_{tot}-1} \frac{k-1}{P_{tot} - 1} \left(\ln\left(\frac{P_{tot} - 1}{k - 1}\right)\right)^r$$

Il s'agit d'une somme de Riemann, on pose $v = \frac{k-1}{P_{tot}-1}$ et la somme converge vers

$$\frac{1}{r!} \int_0^1 v \left(\ln\left(\frac{1}{v}\right)\right)^r dv \underset{z = \ln\left(\frac{1}{v}\right)}{=} \frac{1}{r!} \int_0^{+\infty} e^{-2z} z^r dz = 2^{-(r+1)}$$

Par IPP successives.

On admet que $\mathbb{P}(T_k = r) - \mathbb{P}(S_k = r) \xrightarrow[k\to+\infty]{} 0$ ce qui donne avec le théorème de Cesaro

$$\frac{1}{P_{tot}-2}\sum_{k=2}^{P_{tot}-1} \mathbb{P}(T_k = r) - \mathbb{P}(S_k = r) \xrightarrow[P_{tot}\to+\infty]{} 0$$

■

Démonstration 2 :

On utilise la formule des probabilités totales

$$F_{Z_n}(t) = \sum_{k=0}^{+\infty} \mathbb{P}(Z_n = k)t^k = \sum_{k=0}^{+\infty} t^k \sum_{l=0}^{k} \mathbb{P}(Z_n = k | Z_{n-1} = l)\mathbb{P}(Z_{n-1} = l)$$

La famille est sommable, donc on peut réarranger les termes

$$= \sum_{l=0}^{+\infty} \mathbb{P}(Z_{n-1} = l) \sum_{k=l}^{+\infty} \mathbb{P}(Z_n = k | Z_{n-1} = l)t^k = \sum_{l=0}^{+\infty} \mathbb{P}(Z_{n-1} = l)F_{Z_n|Z_{n-1}=l}(t)$$

Or, la loi de Z_n sachant $Z_{n-1} = l$ est celle d'une somme de l variables aléatoires mutuellement indépendantes identiquement distribuées suivant la même loi que Z_1. D'où $F_{Z_n|Z_{n-1}=l}(t) = \left(F_{Z_1}(t)\right)^l$ et

$$F_{Z_n}(t) = \sum_{l=0}^{+\infty} \mathbb{P}(Z_{n-1} = l)\left(F_{Z_1}(t)\right)^l = F_{Z_{n-1}}\left(F_{Z_1}(t)\right)$$

Il suffit de vérifier que $F_{Z_0} = Id$

■

Démonstration 3 :

1) Déjà démontré

$2)a)$

$$\mathbb{P}(Z_n > 0) \leq \sum_{k=0}^{+\infty} \mathbb{P}(Z_n > 0) = \mathbb{E}(Z_n) = m^n$$

$b)$

$$\mathbb{P}(Z_{n+1} > 0) = \frac{1 - F_{Z_1}\left(F_{Z_n}(0)\right)}{1 - F_{Z_n}(0)}\mathbb{P}(Z_n > 0)$$

Avec le théorème des accroissements finis sur]0,1[il existe c tel que

$$\frac{1-F_{Z_1}\left(F_{Z_n}(0)\right)}{1-F_{Z_n}(0)} = F'_{Z_1}(c) \underset{convexité}{\leq} F'_{Z_1}(1) = m$$

$\left(\frac{\mathbb{P}(Z_n>0)}{m^n}\right)_n$ est décroissante minorée donc converge vers une limite $C \geq 0$, on a $C > 0$ en appliquant $C - S$ à $Z_n \mathbf{1}_{Z_n>0}$.

3) $F''_{Z_1}(1) = \sigma^2$ car $m = 1$

$$F_{Z_1}(1-x) \underset{0}{=} F_{Z_1}(1) - F'_{Z_1}(1)x + F''_{Z_1}(1)\frac{x^2}{2} + o(x^2) \underset{0}{=} 1 - x + \sigma^2\frac{x^2}{2} + o(x^2)$$

$$u_n = \mathbb{P}(Z_n > 0), \qquad u_{n+1} = 1 - F_{Z_1}(1-u_n)$$

$$\frac{1}{u_{n+1}} - \frac{1}{u_n} = \frac{1}{u_n}\left(\frac{1}{1-\frac{\sigma^2}{2}u_n + o(u_n)} - 1\right) = \frac{1}{u_n}\left(\frac{\sigma^2}{2}u_n + o(u_n)\right) \xrightarrow[n\to+\infty]{} \frac{\sigma^2}{2}$$

$$\frac{1}{u_n} = \frac{1}{u_0} + \sum_{k=0}^{n-1}\left(\frac{1}{u_{k+1}} - \frac{1}{u_k}\right) \sim \frac{\sigma^2 n}{2}$$

$$u_n \sim \frac{2}{\sigma^2 n}$$

■

7.2 Expression des constantes

$$S(t) = a + be^{r_n t} + ce^{r_i t} + de^{(r_p - r_w)t}$$

$$a = S_0\left(\frac{1}{r_n} - \frac{r_w}{r_n(r_w - r_p)}\right)$$

$$b = S_0\left(1 - \frac{1}{r_i - r_n} - \frac{1}{r_n} + \frac{r_w}{r_n(r_w - r_p)} + \frac{r_w}{(r_i + r_w - r_p)(r_i - r_n)} + \frac{r_w}{r_p - r_w - r_n}\left(1 + \frac{1}{r_w - r_p} - \frac{1}{r_i + r_w - r_p}\right)\right)$$

$$c = S_0\left(\frac{1}{r_i - r_n} - \frac{r_w}{(r_i - r_n)(r_i + r_w - r_p)}\right)$$

$$d = S_0\left(-\frac{r_w}{r_p - r_w - r_n}\right)\left(1 + \frac{1}{r_w - r_p} - \frac{1}{r_i + r_w - r_p}\right)$$

$$W(t) = \alpha + \beta e^{r_i t} + \gamma e^{(r_p - r_w)}$$

$$\alpha = S_0\left(-1 + \frac{1}{r_i} - \frac{r_w}{r_i(r_i + r_w - r_p)} - \frac{r_w}{r_p - r_w}\left(1 + \frac{1}{r_w - r_p} - \frac{1}{r_i + r_w - r_p}\right)\right)$$

$$\beta = S_0\left(-\frac{1}{r_i} + \frac{r_w}{r_i(r_i + r_w - r_p)}\right)$$

$$\gamma = S_0\left(\frac{r_w}{r_p - r_w}\right)\left(1 + \frac{1}{r_w - r_p} - \frac{1}{r_i + r_w - r_p}\right)$$

7.3 Bibliographie

[1] GRINSTEAD, Charles Miller et SNELL, James Laurie : Introduction to probability : *American Mathematical Soc., 2012*
[2] VANARSDALE, Daniel W. : Chain letter evolution : *Persion: 1998-12-03, 1998*
[3] GASTWIRTH, Joseph L. et BHATTACHARYA, P. K. : Two probability models of pyramid or chain letter schemes demonstrating that their promotional claims are unreliable : *Operations Research, 1984, vol. 32, no 3, p. 527-536*
[4] GASTWIRTH, Joseph L : A probability model of a pyramid scheme : *The American Statistician, 1977, vol. 31, no 2, p. 79-82*
[5] ARTZROUNI, Marc : The mathematics of Ponzi schemes : *Mathematical Social Sciences, 2009, vol. 58, no 2, p. 190-201*

7.4 Code

Le code commence page suivante, il est en 3 parties, tout d'abord la mise en place de la simulation, j'ai laissé les premières versions où les lettres n'avaient pas de noms ainsi que les algorithme pour vérifier la pertinence des résultats. Ensuite, les algorithmes d'affichage, d'abord sans, puis avec Matplotlib. Enfin, une partie pour calculer les itérés d'un polynômes d'abord sans, puis avec Sympy.

```
# C:\Users\Victor\Desktop\TIPE Code.py
001| import random
002|
003| def sold(p1, p2):
004|     p0 = 1-p1-p2
005|     return(random.choices((0,1,2),(p0,p1,p2))[0])
006|
007| def pop_gen(g_max, p1, p2):
008|     g = 0
009|     pop = [[g, -50]]
010|     indice = 0
011|     while g < g_max and len(pop)-indice > 0:
012|         g += 1
013|         l = len(pop)
014|         for i in range(indice, l):
015|             s = sold(p1, p2)
016|             pop[i][1] += 50*s
017|             pop += [[g, -50]]*s
018|         indice = l
019|     return(pop)
020|
021| def pop_gen_repaired(g_max, p1, p2):
022|     g = 0
023|     pop = [[g, -50]]
024|     indice = 0
025|     while g < g_max and len(pop)-indice > 0:
026|         g += 1
027|         l = len(pop)
028|         for i in range(indice, l):
029|             s = sold(p1, p2)
030|             pop[i][1] += 50*s
031|             for i in range(s):
032|                 pop += [[g, -50]]
033|         indice = l
034|     return(pop)
035|
036| def total_sum(t):
037|     sum = 0
038|     for elem in t:
039|         sum += elem[1]
040|     return(sum)
041|
042| def update_letter(pop, i):
043|     letter = pop[i][2]
044|     first = letter[0]
045|     if first is not None:
046|         pop[first][1] += 50
047|     return(letter[1::]+[i])
048|
049| def pop_gen2(p1, p2, population):
050|     g = 0
051|     pop = [[g, -100, [None]*12]]
052|     indice = 0
053|     while len(pop)-indice > 0:
054|         g += 1
```

```
055|         l = len(pop)
056|         pop_dispo = population-l
057|         p1_g = p1*(pop_dispo)/population
058|         p2_g = p2*(pop_dispo)/population
059|         for i in range(indice,l):
060|             s = sold(p1_g, p2_g)
061|             pop[i][1] += 50*s
062|             for j in range(s):
063|                 pop.extend([[g, -100, update_letter(pop, i)]])
064|         indice = l
065|     return(pop)
066|
067| def data(t):
068|     return([elem[:2] for elem in t])
069|
070| def data_display(t):
071|     for elem in t:
072|         print(elem[0], elem[1])
073|
074| def avg(t):
075|     if t == []:
076|         return([])
077|     gen_max = t[-1][0]
078|     stats = []
079|     i = 0
080|     for gen in range(gen_max+1):
081|         sum = 0
082|         n = 0
083|         while i < len(t) and t[i][0] == gen :
084|             n += 1
085|             sum += t[i][1]
086|             i += 1
087|         stats += [[gen, n, sum//n]]
088|     return(stats)
089|
090| def avg_sumpop(t):
091|     if t == []:
092|         return([])
093|     gen_max = t[-1][0]
094|     stats = []
095|     i = 0
096|     for gen in range(gen_max+1):
097|         sum = 0
098|         n = 0
099|         while i < len(t) and t[i][0] == gen :
100|             n += 1
101|             sum += t[i][1]
102|             i += 1
103|         stats += [[gen, i, sum//n]]
104|     return(stats)
105|
106| def avg_display(t):
107|     if t == []:
108|         return([])
109|     gen_max = t[-1][0]
```

```
110|     i = 0
111|     for gen in range(gen_max+1):
112|         sum = 0
113|         n = 0
114|         while i < len(t) and t[i][0] == gen:
115|             n +=1
116|             sum += t[i][1]
117|             i += 1
118|         print(gen, n, sum//n)
119|
120| def avg_sumpop_display(t):
121|     if t == []:
122|         return([])
123|     gen_max = t[-1][0]
124|     i = 0
125|     for gen in range(gen_max+1):
126|         sum = 0
127|         n = 0
128|         while i < len(t) and t[i][0] == gen:
129|             n +=1
130|             sum += t[i][1]
131|             i += 1
132|         print(gen, i, sum//n)
133|
134| import matplotlib.pyplot as plt
135|
136| def coord_gen(n, g_max, p1, p2):
137|     coord = []
138|     for i in range(n):
139|         t = pop_gen(g_max, p1, p2)
140|         x = [elem[0] for elem in avg(t)]
141|         y1 = [elem[1] for elem in avg(t)]
142|         y2 = [elem[2] for elem in avg(t)]
143|         coord += [[x,y1,y2]]
144|     return(coord)
145|
146| def coord_gen2(n, p1, p2, population):
147|     coord = []
148|     for i in range(n):
149|         t = pop_gen2(p1, p2, population)
150|         x = [elem[0] for elem in avg(t)]
151|         y1 = [elem[1] for elem in avg(t)]
152|         y2 = [elem[2] for elem in avg(t)]
153|         coord += [[x,y1,y2]]
154|     return(coord)
155|
156| def coord_plot(coord, type):
157|     for elem in coord:
158|         x = elem[0]
159|         y1 = elem[1]
160|         y2 = elem[2]
161|         if type == 1:
162|             plt.plot(x,y1)
163|         else :
164|             plt.plot(x,y2)
```

```
165|
166|  def plot_display(type):
167|      plt.xlabel("Génération")
168|      if type == 1:
169|          plt.ylabel("Nb_acheteurs")
170|          plt.title("Graphique des participants par génération")
171|      else:
172|          plt.ylabel("Gain moyen")
173|          plt.title("Graphique des gains par génération")
174|      plt.show()
175|
176|  def histo_data(p1, p2, population):
177|      t = pop_gen2(p1, p2, population)
178|      g_max = t[-1][0]
179|      h = [[elem[1] for elem in data(t) if elem[0] == g] for g in
range(g_max+1)]
180|      return(h)
181|
182|  def histo_bar(h, g):
183|      if len(h) <= g:
184|          dat = [-2]
185|      else:
186|          dat = h[g]
187|      categories = [50*i for i in range(-2, max(dat)//50)]
188|      plt.hist(dat, categories, rwidth = 0.9)
189|
190|  def histo_display():
191|      plt.xlabel("Gain")
192|      plt.ylabel("Nombre")
193|      plt.title("Probabilité de gains")
194|      plt.show()
195|
196|  def prob_bar(n, p1, p2, population, g):
197|      dat = []
198|      for i in range(n):
199|          h = histo_data(p1, p2, population)
200|          if len(h) > g:
201|              dat += h[g]
202|      categories = [50*i for i in range (-2, max(dat)//50)]
203|      plt.hist(dat, categories, rwidth = 0.9)
204|
205|  def prob_gain(n, cap, p1, p2, population):
206|      dat = [[]]
207|      for i in range(n):
208|          h = histo_data(p1, p2, population)
209|          diff = len(h)-len(dat)
210|          if diff > 0:
211|              for j in range(diff):
212|                  dat += [[]]
213|          for g in range(len(h)):
214|              p = len([elem for elem in h[g] if elem >
cap])/len(h[g])
215|              dat[g] += [p]
216|      for i in range(len(dat)):
217|          dat[i] = sum(dat[i])/len(dat[i])
```

```
218|     return(dat)
219|
220| def bar_display(dat):
221|     plt.bar([i for i in range(len(dat))], dat)
222|     plt.xlabel("Gain")
223|     plt.ylabel("Proba")
224|     plt.title("Probabilité de gains")
225|     plt.show()
226|
227| def expect_val(n, p1, p2, population):
228|     dat = [[]]
229|     for i in range(n):
230|         h = histo_data(p1, p2, population)
231|         diff = len(h)-len(dat)
232|         if diff > 0:
233|             for j in range(diff):
234|                 dat += [[]]
235|         for g in range(len(h)):
236|             e = sum(h[g])/len(h[g])
237|             dat[g] += [e]
238|     for i in range(len(dat)):
239|         dat[i] = sum(dat[i])/len(dat[i])
240|     return(dat)
241|
242| def expect_val2(n, p1, p2, population):
243|     dat_money = [[]]
244|     dat_pop = [[]]
245|     for i in range(n):
246|         h = histo_data(p1, p2, population)
247|         diff = len(h)-len(dat_money)
248|         tot = 0
249|         if diff > 0:
250|             for j in range(diff):
251|                 dat_money += [[]]
252|                 dat_pop += [[]]
253|         for g in range(len(h)):
254|             l = len(h[g])
255|             e = sum(h[g])/l
256|             dat_money[g] += [e]
257|             tot += l
258|             dat_pop[g] += [tot]
259|         for j in range(len(h)):
260|             dat_pop[j][-1] = dat_pop[j][-1]/tot
261|     for i in range(len(dat_money)):
262|         money_moy = sum(dat_money[i])/len(dat_money[i])
263|         money_var = sum([(e-money_moy)**2 for e in
dat_money[i]])/len(dat_money[i])
264|         dat_money[i] = (money_moy, money_var**(1/2))
265|         pop_moy = sum(dat_pop[i])/len(dat_pop[i])
266|         pop_var = sum([(e-pop_moy)**2 for e in
dat_pop[i]])/len(dat_pop[i])
267|         dat_pop[i] = (pop_moy, pop_var**(1/2))
268|     return(dat_money,dat_pop)
269|
270| def money_display(dat):
```

```
271|     plt.bar([i for i in range(len(dat))], dat)
272|     plt.xlabel("Génération")
273|     plt.ylabel("Gain")
274|     plt.title("Espérance de gain")
275|     plt.show()
276|
277| def pop_display(dat):
278|     plt.bar([i for i in range(len(dat))], dat)
279|     plt.xlabel("Génération")
280|     plt.ylabel("Population")
281|     plt.title("Espérance de population")
282|     plt.show()
283|
284| def double_display(dat):
285|     plt.bar([i for i in range(len(dat[0]))], dat[0])
286|     plt.bar([i for i in range(len(dat[1]))], dat[1])
287|     plt.xlabel("Génération")
288|     plt.ylabel("Gain et population")
289|     plt.title("Espérance de gain et population")
290|     plt.show()
291|
292| def money_var_display(dat):
293|     plt.xlabel("Génération")
294|     plt.ylabel("Gain")
295|     plt.title("Espérance de gain")
296|     dat1 = [e[0]+e[1] for e in dat]
297|     dat2 = [e[0] for e in dat]
298|     dat3 = [e[0]-e[1] for e in dat]
299|     plt.bar([i for i in range(len(dat))], dat1, color=[0,0,1])
300|     plt.bar([i for i in range(len(dat))], dat2, color=[0,1,0])
301|     plt.bar([i for i in range(len(dat))], dat3, color=[1,0,0])
302|     plt.show()
303|
304| def money_var_display2(dat):
305|     plt.xlabel("Génération")
306|     plt.ylabel("Gain")
307|     plt.title("Espérance de gain")
308|     dat1 = [e[0]-e[1] for e in dat]
309|     dat2 = [e[0] for e in dat]
310|     dat3 = [e[0]+e[1] for e in dat]
311|     plt.bar([i for i in range(len(dat))], dat1, color=[1,0,0])
312|     plt.bar([i for i in range(len(dat))], dat2, color=[0,1,0])
313|     plt.bar([i for i in range(len(dat))], dat3, color=[0,0,1])
314|     plt.show()
315|
316| def pop_var_display(dat):
317|     plt.xlabel("Génération")
318|     plt.ylabel("Population")
319|     plt.title("Espérance de population")
320|     dat1 = [e[0]+e[1] for e in dat]
321|     dat2 = [e[0] for e in dat]
322|     dat3 = [e[0]-e[1] for e in dat]
323|     plt.bar([i for i in range(len(dat))], dat1, color=[0,0,1])
324|     plt.bar([i for i in range(len(dat))], dat2, color=[0,1,0])
325|     plt.bar([i for i in range(len(dat))], dat3, color=[1,0,0])
```

```
326|     plt.show()
327|
328| def Fn(p1,p2,n,x):
329|     p0 = 1-p1-p2
330|     a = x
331|     for i in range(n):
332|         a = p0+p1*a+p2*a**2
333|     return(a)
334|
335| from sympy import symbols, simplify
336|
337| p1 = symbols('p1', float = True)
338| p2 = symbols('p2', float = True)
339|
340| F = [1-p1-p2,p1,p2]
341|
342| def Somme(P,Q):
343|     S = []
344|     m = min(len(P), len(Q))
345|     for  i in range(m):
346|         S.append(P[i]+Q[i])
347|     S.extend(P[m:])
348|     S.extend(Q[m:])
349|     return(S)
350|
351| def Produit(P,Q):
352|     S = [0]*(len(P)+len(Q)-1)
353|     for i, a in enumerate(P) :
354|         for j, b in enumerate(Q):
355|             S[i+j] = S[i+j] + a*b
356|     return(S)
357|
358| def Puissances(P,n):
359|     Pn = [[1]]
360|     for i in range(n):
361|         Pn.append(Produit(Pn[-1],P))
362|     return(Pn)
363|
364| def Compose(P,Q):
365|     PQ = []
366|     for a, Qi in zip(P, Puissances(Q, len(P)-1)):
367|         PQ = Somme(PQ, Produit([a], Qi))
368|     return(PQ)
369|
370| def Iteres(P,n):
371|     L = [[0,1]]
372|     for i in range(n):
373|        L.append(Compose(L[-1],P))
374|     return(L)
375|
376| a = simplify(Iteres(F,3)[-1][0])
```