Agro 2024

Problème A : On définit la fonction f par son expression : $\forall x \in \mathbf{R},\ f(x) = x + \frac{1}{2}\, x \left(\frac{S-x}{S}\right)$
où S est une constante strictement positive fixée.
On fixe $v_0 \in \mathbf{R}_+^\star$ et on s'intéresse à la suite $(v_n)_n$ définie par : $\forall n \in \mathbf{N},\ v_{n+1} = f(v_n)$.

1. Compléter le code suivant pour qu'il trace les 20 premiers termes de la suite.

```
S = 30
v0 = 5
L = [v0]
v = v0
for k in ## LIGNE A COMPLETER ##
   ## LIGNE A COMPLETER ##
   ## LIGNE A COMPLETER ##

plt.plot(L)
plt.xlabel("n")
plt.ylabel("v_n")
plt.show()
```

2. Pour le tracé, on a utilisé le module `matplotlib.pyplot` sous l'alias `plt`. Comment importer le module ?

Problème B : Dans cette partie, on s'intéresse à l'implémentation informatique du processus de Galton-Watson (*NB : voir DS3*). Notre objectif est de faire des conjectures sur le comportement de la population dans le cas où la loi de reproduction est plus complexe : on considèrera dans cette partie que les variables aléatoires $X_{n,i}$ suivent la loi de Poisson de paramètre $\lambda > 0$. On rappelle que la commande `rd.poisson(x)` simule une variable aléatoire qui suit une loi de Poisson de paramètre `x`. La fonction suivante simule l'évolution d'une population et stocke le nombre d'individus à chaque génération dans une liste.

```
1  def galton_watson(lambda_,n) :
2    population = np.zeros(n+1)
3    population[0] = 1
4    Z = 1
5    for i in range(1,n+1) :
6      descendants = 0
7      for j in range(Z) :
8        descendants += rd.poisson(lambda_)
9      population[i] = descendants
10     Z = descendants
11     if descendants == 0 :
12       return population
13   return population
```

1. À quoi servent les deux arguments de la fonction `galton_watson` ?
2. À quoi servent les lignes suivantes ?

```
11      if descendants == 0 :
12        return population
```

```
2    population = np.zeros(n+1)
```

```
6      descendants = 0
7      for j in range(Z) :
8        descendants += rd.poisson(lambda_)
```

3. Pourquoi peut-on remplacer les lignes 6 à 8 par :

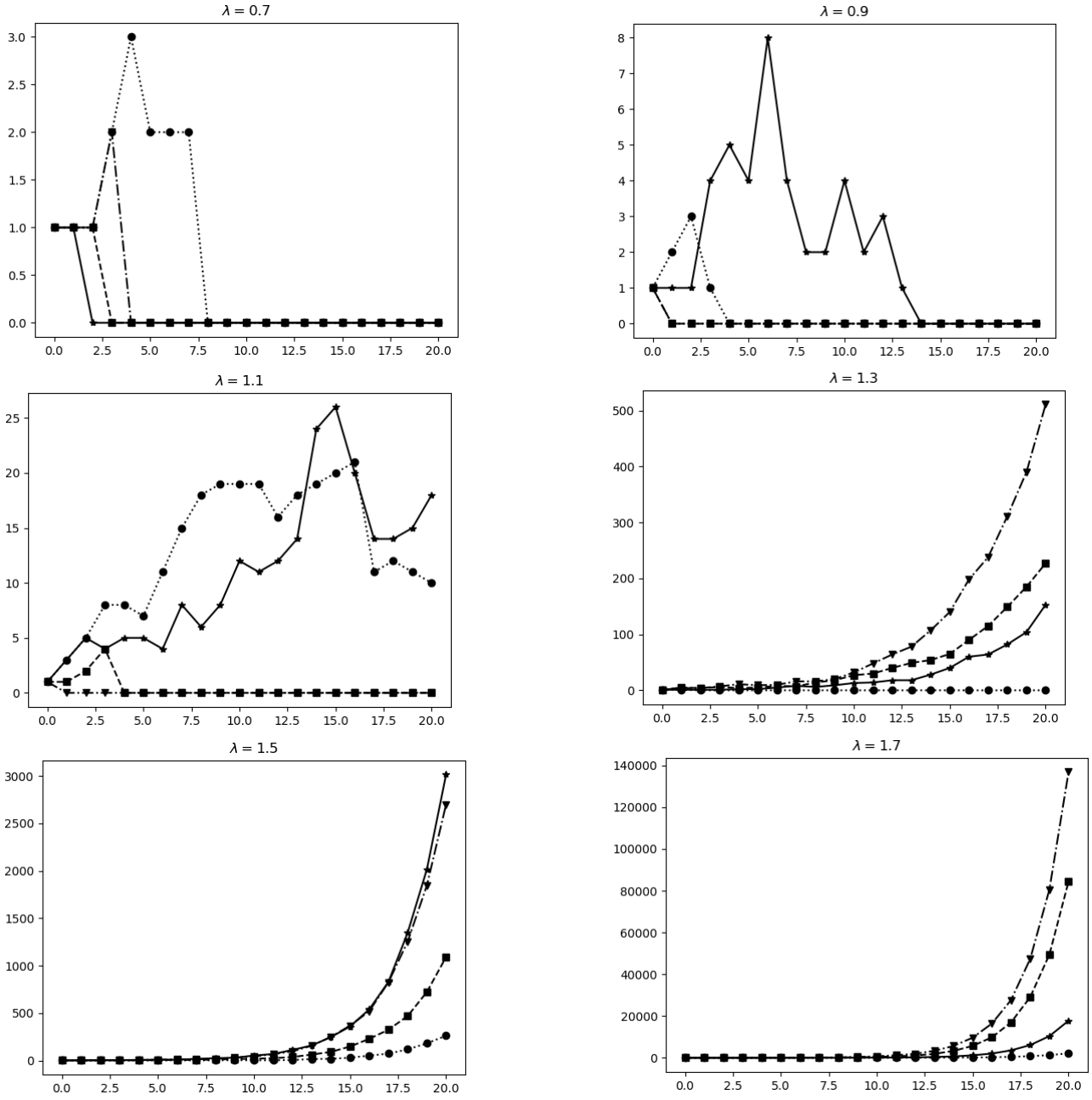
```
6      descendants = rd.poisson(Z*lambda_)
```

4. On souhaite réaliser :
   - des simulations pour différentes valeurs de λ,
   - pour chacun des choix, plusieurs simulations.

   a. Compléter le programme suivant pour qu'il réalise 10 simulations pour $\lambda = 0,7$ et 20 générations et les trace sur un même graphique.

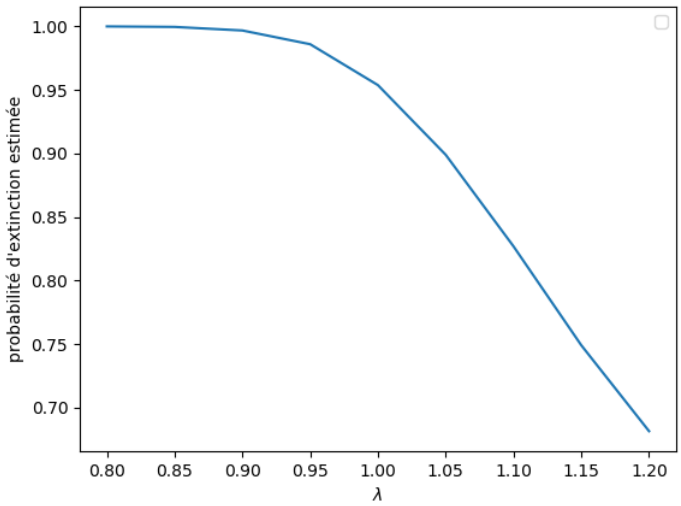
```
1  lambda_ = 0.7
2  for k in ## LIGNE A COMPLETER ##
3      plt.plot( ## LIGNE A COMPLETER ##)
4  plt.show()
```

b. Les simulations donnent les résultats suivants pour λ variant entre $0,7$ et $1,7$ avec un pas de $0,2$. Quelles conjectures peut-on faire quant à la probabilité d'extinction de l'espèce ?



Les valeurs de la population sont déterminées pour des nombres *entiers* de générations. Des lignes ont cependant été tracées *entre* les valeurs de la population pour les générations successives, afin de faciliter la visualisation de l'évolution d'une population au cours des générations.

5. Dans cette question, on s'intéresse à la probabilité d'extinction.
   a. Comment modifier la fonction `galton_watson` pour qu'elle renvoie 1 si la lignée est éteinte et 0 si la lignée n'est pas éteinte ? Pour la suite on appelle la fonction ainsi modifiée : `galton_watson_2`.
   b. Écrire une fonction `extinction`, qui prend en rentrée un paramètre `lambda_` et qui, à partir de 5000 simulations de Galton-Watson, renvoie une approximation de la probabilité d'extinction. (On s'arrêtera à 60 générations).
6. La simulation pour différentes valeurs de λ donne le graphique suivant. (On a pris λ entre $0,8$ et $1,2$ avec un pas de $0,05$.) On note p_λ la probabilité d'extinction. Quelles conjectures peut-on faire sur p_λ ?



Annexe Python

Dans le module matplotlib.pyplot importé sous l'alias plt :
`plt.plot(X,Y)` prend en entrée deux vecteurs ou deux listes de même taille, et réalise le tracé des points d'abscisses prises dans X et d'ordonnées prises dans Y. Si on donne un seul argument à `plt.plot`, cela trace juste la suite des termes de X.
On utilise `plt.show()` pour afficher le tracé.

Dans le module numpy importé sous l'alias np :
`np.zeros(n)` crée une matrice unidimensionnelle de n coefficients tous nuls.

Dans le module numpy.random importé sous l'alias rd :
`rd.poisson(x)` simule une variable aléatoire suivant une loi de Poisson de paramètre x.

* * *

Agro 2023 **Partie 2**

1. Soit la fonction Python suivante :

```
1  def C(n,k) :
2    Num = 1
3    for i in range(2,n+1) :
4       Num = Num*i
5    Den = 1
6    for i in range(2,k+1) :
7       Den = Den*i
8    for i in range(2,n-k+1) :
9       Den = Den*i
10   return (Num/Den)
```

   Justifier que cette fonction renvoie le coefficient binomial $\binom{n}{k}$. Évaluer le nombre d'opérations effectuées par la fonction `C` en fonction de n. Quel autre problème cette fonction pose-t-elle en terme de calcul numérique ?

2. Compléter en le recopiant le code suivant afin de renvoyer le coefficient binomial $\binom{n}{k}$ de façon optimisée en nombre d'opérations. Expliquer la démarche suivie :

```
1  def Copt(n,k) :
2    Num = ...
3    for i in range(... , ...) :
4       Num = Num*i
5    Den = ...
6    for i in range(... , ...) :
7       Den = Den*i
8    return (Num/Den)
```

3. On appelle c_n le *prix d'un call sur le prix du blé* de prix d'exercice s^* et de terme n.

   On démontre que : $\forall n \geqslant 1,\ c_n = \sum_{k=a}^{n} \left(u^{2k-n}s_0 - s^\star\right) \times \binom{n}{k} p^k (1-p)^{n-k}$ (1)

   où a est le plus petit entier tel que : $\forall k \geqslant a,\ u^{2k-n}s_0 - s^\star \geqslant 0$.

   u, s_0 et $s^\star$ désignent des constantes strictement positives ($u > 1$) dont l'interprétation n'a pas d'intérêt ici.

   p désigne un réel compris strictement entre 0 et 1.

   Que renvoie la fonction suivante ? Justifier :

```
1  def Feuilles(n,u,s0,setoile) :
2    F = []
3    for k in range(n+1) :
4       F.append(max(u**(k*2-n)*s0-setoile,0))
5    return F
```

4. Écrire une fonction `call` qui utilise la fonction `Feuilles` et (1) pour calculer le prix du call de prix initial s_0, de prix d'exercice $s^\star$, de terme n et de paramètres u et p. Justifier.

Partie 3

Pour trouver un zéro d'une fonction $f(u)$ dérivable (résoudre $f(u) = 0$) on peut utiliser la **méthode de Newton** appelée aussi **méthode de la tangente**. Elle consiste à définir une suite (u_n) par le relation de récurrence :

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}$$

Cette suite converge vers un zéro de f sous de bonnes conditions qu'on ne demande pas de remontrer ici. Numériquement on s'arrête de calculer les termes successifs de la suite quand $|u_{n+1} - u_n| < \varepsilon$ ou lorsque le nombre d'itérations dépasse un seuil n_{max}. On supposera que les deux fonctions $f(u)$ et $f'(u)$ sont programmées sous Python dans les codes `F` et `Fprime` (on ne demande pas ici la programmation de ces fonctions).

1. Écrire une fonction `Newton(u0, epsilon, F, Fprime)` renvoyant un zéro de f approché par la méthode de Newton. Cette fonction prendra en entrée u_0, ε, et les codes `F` et `Fprime` de f et de sa dérivée f'. On supposera que n_{max} est une variable globale.
2. Choisir en justifiant une valeur de ε cohérente.
3. Que renvoie cette fonction si $f'(u_n) = 0$? Modifier le code pour renvoyer un message d'erreur dans ce cas.
4. On admet que l'entier a de la **question 3, Partie 2** est compris entre $\left\lfloor 1 + \frac{n}{2} \right\rfloor$ et n, où $\lfloor . \rfloor$ désigne la partie entière. On cherche une valeur de u qui annule la fonction $F(u) =$`call`$(u) - c^\star$, où la fonction `call` est celle de la **question 4, Partie 2**, et où $c^\star$ est un réel positif fixé.

   Supposant les fonctions `F` et `Fprime` programmées sous Python, que renvoie l'algortihme suivant ? Justifier.

```
1  def mystere(F,Fprime,s0,setoile,epsilon) :
2    a = n
3    u = (setoile/s0)**(1/(2*a-n))
4    while F(u)<0 and a >= m.floor(1+n/2) :
5       a -= 1
6       u = (setoile/s0)**(1/(2*a-n))
7    a += 1
8    u = (setoile/s0)**(1/(2*a-n))
9    u = Newton(u,epsilon,F,Fprime)
10   return u
```

5. On appelle **volatilité** le réel strictement positif σ tel que, lorsque $p = \frac{1}{2}$, on a : $u = \exp\left(\sqrt{\sigma^2}\right)$.

   Écrire une fonction `Volatilite` estimant la volatilité σ.

Annexe Python

On suppose que le module `math`, qui permet d'utiliser des fonctions mathématiques, est importé avec l'alias `m`. Ce module contient les fonctions :
- `floor(x)` qui renvoie la partie entière inférieure de x.
- `exp(x)` qui renvoie la valeur de la fonction exponentielle en x.
- `log(x)` qui renvoie la valeur de la fonction logarithme en x.

Par ailleurs on rappelle que la valeur absolue de x est donnée par la fonction `abs(x)`.