Agro 2022

- Parties 1 à 4 : études mathématiques de différents modèles de dynamiques de populations.
- Partie 5 : Étude informatique

Cette partie consiste à mettre en place des programmes informatiques pour simuler les différents modèles considérés dans le sujet. Bien que cette partie soit largement indépendante des parties précédentes, il est vivement conseillé de les avoir lues. Les programmes sont à rédiger en langage Python. L'annexe comporte des rappels sur les commandes utiles. Avant chaque algorithme, on écrira brièvement le raisonnement suivi et la formule qu'il est censé calculer.

1. On considère le code :

```
1  r = 2
2  def sol(t) :
3      return m.exp(r*t)
4  def liste(T, h) :
5      t = 0
6      L = [0]
7      while t+h <= T :
8          t = t+h
9          L.append(t)
10     return L
11 def mystere() :
12     Lt = liste(20, 10**-2)
13     n = len(Lt)
14     Ls = []
15     for k in range(0, n) :
16         Ls.append(sol(Lt[k]))
17     plt.plot(Lt, Ls)
18     plt.show()
```

(a) La fonction `exp` est présente dans le module `math`.
Indiquer comment charger ce module pour que l'utilisation de `exp` soit correcte dans `sol`.

(b) Que renvoient `liste(1, 0.2)` et `liste(1, 0.3)` ? Indiquer, de façon générale, ce que renvoie `liste(T, h)`.

(c) Que contient `Ls` dans la fonction `mystere` ? Que fait cette fonction ?

2. (a) On considère `L = [[3,1], [7], [1,9,8,0]]`. Que vaut `L[1]` ? Que vaut `L[0][1]` ?
Que vaut `len(L)` ? Que vaut `L` après avoir exécuté `L.append(9.75)` ?

(b) Écrire une fonction d'entête `lapin(x, y)` qui renvoie $x - xy$ et une fonction d'entête `lynx(x, y)` qui renvoie $-ax + xy$. La constante a est supposée être préalablement définie dans une variable globale.

(c) On ajoute au code précédent la fonction suivante, où les arguments `x0, y0, T` et `h` définissent respectivement x_0, y_0, l'intervalle d'étude $[0, T]$ et le pas $h > 0$ de la méthode d'Euler $(*)$ définie à la **partie 4** :

```
1  def resol_1(x0, y0, T, h) :
2      x = x0; y = y0
3      t = 0
4      Lx = [x]
5      Ly = [y]
6      Lt = [t]
7      while t+h <= T :
8      ----- ligne(s) à compléter -----
9      return [Lt, Lx, Ly]
```

$(*)$ Les réels x_k et y_k représentent respectivement les populations de lapins et de lynx à l'instant t_k. L'équation (1) ci-dessous est une discrétisation de :

$$(\mathcal{S}) \quad \begin{cases} x' = x - xy \\ y' = -ay + xy \end{cases}$$

où $x(t), y(t)$ sont les populations de lapins et de lynx au temps t.

Compléter les lignes manquantes dans la boucle, afin que les listes `Lx` et `Ly` contiennent respectivement les x_k et y_k définis par l'équation (1) et que la liste `Lt` contienne les $t_k = hk$:

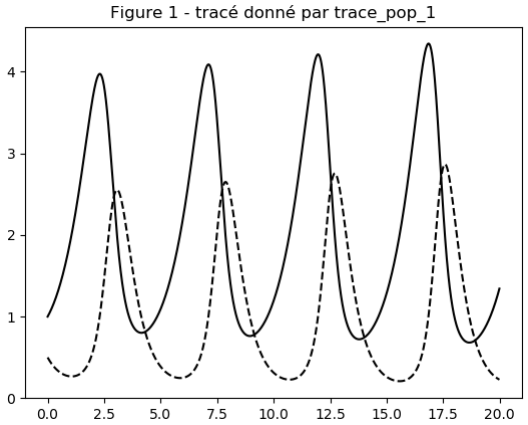
$$\begin{cases} x_{k+1} = x_k + h(x_k - x_k y_k) \\ y_{k+1} = y_k + h(-ay_k + x_k y_k) \end{cases} \qquad (1)$$

(d) On ajoute au code précédent :

```
1  x0 = 1; y0 = 0.5; T = 20; h = 10**-2
2  def trace_pop_1() :
3      L = resol_1(x0, y0, T, h)
4      plt.plot(L[0], L[1])
5      plt.plot(L[0], L[2])
6      plt.show()
```

Expliquer à quoi correspond chaque ligne de la fonction `trace_pop_1`, en fonction des x_k et y_k de la méthode d'Euler.

(e) L'exécution de `trace_pop_1` donne les courbes représentées figure 1. Identifier (avec explications), en lien avec la modélisation effectuée, chacune des deux courbes.



(f) On ajoute au code précédent :

```
1  def fonctionV(x, y) :
2      return x-a*m.log(x)+y-m.log(y)
```

Écrire une fonction d'entête `trace_V_1()` qui effectue la représentation graphique des valeurs v_k en ordonnées, en fonction des valeurs t_k en abscisses.

Une étude mathématique en **Partie 2** a montré que si $x(t), y(t)$ sont solutions du système $(\mathcal{S})$ de la question **2c)**, alors $V\big(x(t), y(t)\big)$ est une fonction constante.

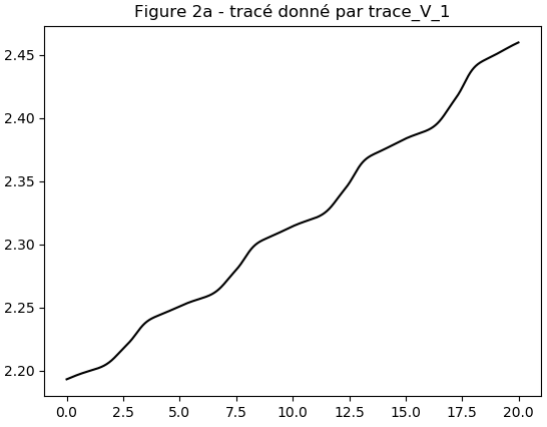
3. On cherche dorénavant à effectuer une résolution numérique avec une autre méthode, appelée méthode de Heun, où la relation de récurrence définissant les x_k et y_k est :

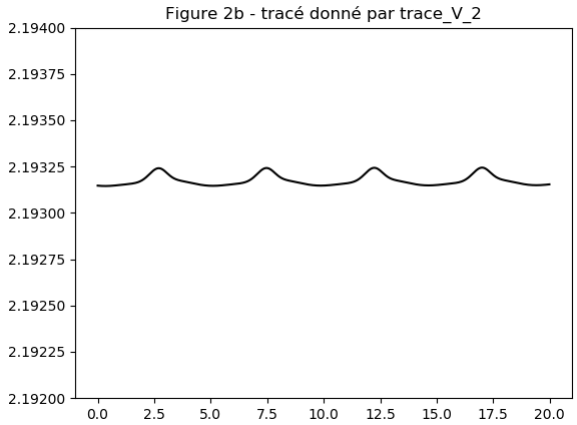
$$\begin{cases} u_k = x_k + h(x_k - x_k y_k) \\ w_k = y_k + h(-a y_k - x_k y_k) \\ x_{k+1} = x_k + \frac{h}{2}(x_k - x_k y_k + u_k - u_k w_k) \\ y_{k+1} = y_k + \frac{h}{2}(-a y_k + x_k y_k - a w_k + u_k w_k) \end{cases} \tag{2}$$

   (a) Écrire une fonction d'entête `resol_2(x0, y0, T, h)`, analogue de `resol_1` utilisant la méthode de Heun.

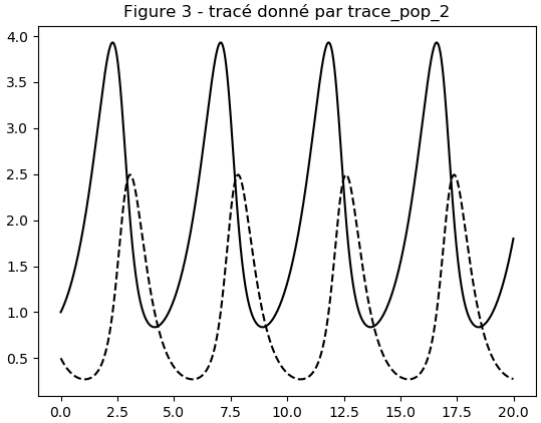
   (b) Que suffit-il de modifier dans la fonction `trace_V_1` pour lui faire utiliser la méthode de Heun ?
   On appelle `trace_V_2` la fonction ainsi modifiée.

   (c) L'exécution de `trace_V_1` et `trace_V_2` donne les courbes présentées en figure 2 :
   justifier quelle méthode semble la plus satisfaisante.





4. On définit, de manière analogue à la fonction `trace_pop_1`, la fonction `trace_pop_2` associée à la méthode de Heun. L'exécution de `trace_pop_2` donne les courbes présentées figure 3. Comparer les figures 1 et 3. Quel problème numérique est soulevé par la méthode d'Euler ? Que permet d'améliorer la méthode de Heun ?



5. Interpréter l'évolution des dynamiques de population des lièvres et des lynx obtenues.

Annexe : Rappels de syntaxe Python pour la partie 5

On suppose que le module `matplotlib.pyplot`, qui permet de tracer des graphiques, est importé avec l'alias `plt` ; que le module `math`, qui permet d'utiliser des fonctions mathématiques, est importé avec l'alias `m`. Les variables `Lx` et `Ly` sont deux listes de réels, de même longueur ; la variable `x` est un réel.

- `plt.plot(Lx,Ly)` place les points dont les abscisses sont contenues dans `Lx` et les ordonnées dans `Ly`, et les relie entre eux par des segments. Si cette fonction n'est pas suivie de `plt.show()`, le graphique n'est pas affiché.
- `plt.show()` affiche le(s) tracé(s) précédemment créé(s) par `plt.plot`.
- `m.exp(x)` renvoie $\exp(x)$.
- `m.log(x)` renvoie $\ln(x)$.

* * *

Agro 2021

- Parties 1,2 et 4 : étude de l'effet de la vaccination contre la variole sur l'espérance de vie.
- Partie 3 : étude informatique.

Seul le résultat final de la question 1 est utile pour répondre aux questions suivantes : si besoin, on pourra donc admettre dès le début de la question 2 que la liste `Lnorm` est créée. Une annexe en dernière page du sujet rappelle quelques fonctions Python.

1. On dispose d'un fichier texte `data.txt`, composé de deux lignes, où figurent des données sous la forme suivante :
   - la première ligne du fichier contient le nombre de morts par d'autres maladies, pour chaque année ;
   - la seconde ligne contient le nombre de survivants au début de chaque année :
   - sur chaque ligne, les données sont séparées par un espace (caractère `" "`).

   Par exemple, si l'on considère une version simplifiée avec seulement sept années, le fichier `data.txt` est constitué du texte suivant :

   ```
   283 133 47 30 21 16 13
   1000 855 798 760 732 710 692
   ```

   L'objectif de cette question est d'extraire ces informations pour créer la liste `Lnorm` qui contient, pour chaque année, le nombre de morts par d'autres maladies divisé par le nombre de survivants :
   `Lnorm = [283/1000, 133/855, 47/798, ...]`.

   a. Écrire une fonction d'entête `extraction_ch()` qui renvoie une liste composée de deux chaînes de caractères : une correspondant à la première ligne de `data.txt` et l'autre à la seconde. Cette fonction devra, au moins, ouvrir le fichier `data.txt`, le lire puis le fermer. Pour l'ouverture, la lecture et la fermeture d'un fichier, on pourra utiliser les syntaxes décrites dans l'annexe (en dernière page du sujet). Par exemple, sur le contenu simplifié ci-dessus, `extraction_ch()` renvoie :

   ```
   ['283 133 47 30 21 16 13\n', '1000 855 798 760 732 710 692']
   ```

   b. Compléter la fonction suivante d'entête `ch_vers_list(ch)` prenant en entrée une chaîne de caractères (de même forme que celles renvoyées par `extraction_ch`) et qui renvoie la liste de nombres correspondant ; ces nombres seront de type flottant.

   Par exemple, `ch_vers_list('1000 855 798 760 732 710 692')` renvoie :

   ```
   [1000.0, 855.0, 798.0, 760.0, 732.0, 710.0, 692.0]
   ```

   Les éléments complétés sont à justifier, en particulier le rôle de la variable `sh`.

   ```
   1  def ch_vers_liste(ch) :
   2      L = _________
   3      n = len(ch)
   4      sh = ""
   5      for k in range(0, n) :
   6          if ch[k] != " " :
   7              sh = sh+ch[k]
   8          else :
   9              _________
   10             _________
   11     _________
   12     return L
   ```

   c. Écrire une fonction d'entête `division(L1, L2)` prenant en entrée deux listes de nombres flottants. Cette fonction renvoie le booléen `False` si les deux listes ne sont pas de la même longueur ; sinon, elle renvoie la liste contenant les divisions terme à terme des éléments de `L1` par ceux de `L2` (on suppose qu'il n'y a pas de terme nul dans `L2`).

   Par exemple, `division([2, 10.5, 6, 4], [2, 2, 3, 1])` renvoie :

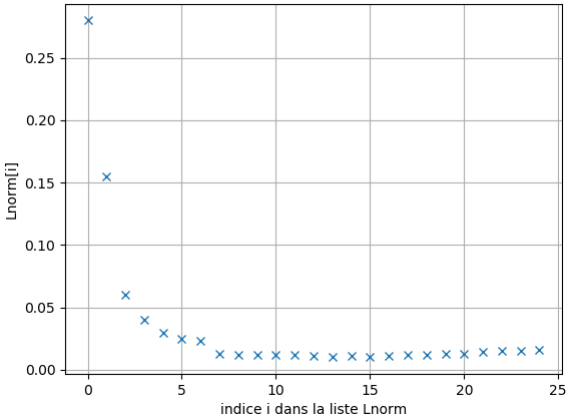
   ```
   [1.0, 5.25, 2.0, 4.0]
   ```

d. Proposer une suite d'instructions qui crée la liste `Lnorm` précédemment définie.
Si besoin, on rappelle que `float("6.5\n")` renvoie `6.5`.

2. On se propose d'estimer la fonction a représentant le taux de morts hors variole au cours du temps.
On note $(\ell_t)_{t\in\{0,\dots,83\}}$ les valeurs relevées sur la cohorte du nombre de morts hors variole pour les 84 années de l'étude, stockées dans la liste `Lnorm`. Le graphique des valeurs numériques de ℓ_t pour les 24 premières années est donné en figure 1. On admettra que le comportement en temps long reste stable.
On définit la fonction $M : \mathbf{R}^3 \longrightarrow \mathbf{R}$ par :

$$\forall(\lambda,\mu,\gamma)\in\mathbf{R}^3,\ M(\lambda,\mu,\gamma)=\sum_{t=0}^{83}\left(\ell_t-\lambda e^{-\mu t}-\gamma\right)^2$$

Figure 1 - tracé des valeurs ℓ_t



a. Un premier choix, non retenu pour la suite, aurait été de considérer, à la place de M, la fonction $\tilde{M}$ suivante :

$$\forall(\alpha,\beta)\in\mathbf{R}^2,\ \tilde{M}(\alpha,\beta)=\sum_{t=0}^{83}(\ell_t-\alpha t-\beta)^2.$$

(i) Expliquer ce que l'on aurait cherché à faire en minimisant $\tilde{M}$ par rapport à α et β.
(ii) Expliquer ce que l'on cherche à faire en minimisant M par rapport à λ, μ et γ ; on pourra faire un dessin. Pourquoi choisir $t\mapsto\lambda e^{-\mu t}+\gamma$ plutôt qu'une fonction polynomiale ?

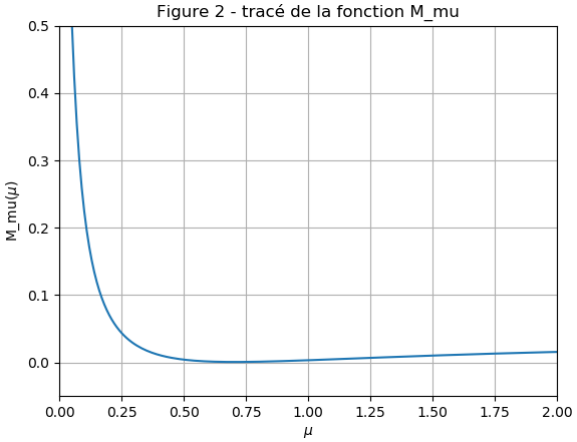
b. À l'aide des données représentées sur la figure 1, indiquer avec justification quelle valeur numérique approximative peut être choisie pour γ. Faire de même pour λ. On note ces choix γ_0 et λ_0, stockés dans les variables globales `gamma0` et `lambda0`. Dans tout le reste de cette question 2, on considère que $\gamma=\gamma_0$ et $\lambda=\lambda_0$: seule μ reste à déterminer.

c. À quoi correspond la fonction suivante ? (*La fonction Python exp est la fonction exponentielle.*)

```
1  def M_mu(mu) :
2      lamb = lambda0
3      gam = gamma0
4      s = 0
5      for k in range(0, len(Lnorm)) :
6          s = s+(Lnorm[k]-lamb*exp(-mu*k)-gam)**2
7      return s
```

d. Le tracé de la fonction `M_mu` est donné en figure 2. Expliquer si ce tracé est utile pour déterminer la valeur de μ cherchée. Si oui, proposer une valeur, notée μ_0 ; si non, que faudrait-il tracer ?

Figure 2 - tracé de la fonction M_mu



3. Dans cette question, on dispose d'une fonction $f : x \mapsto f(x)$ définie sur un intervalle $[A, B]$ de $\mathbf{R}$ qui vérifie l'hypothèse suivante : il existe $x^\star \in [A, B]$ tel que f est strictement décroissante sur $[A, x^\star]$ et strictement croissante sur $[x^\star, B]$; ainsi, f admet un minimum en $x^\star$. L'objectif de la question est de déterminer $x^\star$ de façon approchée.

   L'algorithme utilisé consiste à partir de $g_0 = A$ et $d_0 = B$ puis à construire une suite d'intervalles $[g_k, d_k]$ qui sont de plus en plus petits et qui contiennent $x^\star$ (ainsi, lorsqu'on aura atteint un intervalle « suffisamment petit », on aura localisé $x^\star$ avec une « précision suffisante »). Pour passer de $[g_k, d_k]$ à $[g_{k+1}, d_{k+1}]$, on utilise le raisonnement suivant :
   – on choisit deux nombres x^g et x^d qui vérifient $g_k < x^g < x^d < d_k$;
   – si $f(x^g) < f(x^d)$ alors $x^\star$ se situe à gauche de x^d donc on pose $g_{k+1} = g_k$ et $d_{k+1} = x^d$;
   – sinon si $f(x^g) > f(x^d)$ alors $x^\star$ se situe à droite de x^g donc on pose $g_{k+1} = x^g$ et $d_{k+1} = d_k$;
   – sinon, c'est que $f(x^g) = f(x^d)$ donc $x^\star$ se situe dans $[x^g, x^d]$ et on pose $g_{k+1} = x^g$ et $d_{k+1} = x^d$.

   On calcule cette suite d'intervalles et on s'arrête dès que la taille de l'intervalle $[g_k, d_k]$ est inférieure ou égale à ε (où $\varepsilon > 0$ est une valeur fixée) ; on renvoie alors, parmi les trois points g_k, d_k et m_k (où m_k est le milieu entre g_k et d_k), celui en lequel f est la plus petite.

   a. On dispose de la fonction suivante, où `f` est une fonction et `x,y` deux réels.

```
def argmin2(f, x, y) :
    if f(x) <= f(y) :
         return x
    else :
         return y
```

   Écrire une fonction `argmin3(f, x, y, z)` qui renvoie parmi les trois valeurs `x,y` et `z` celle en laquelle `f` est minimale.

   b. Écrire une fonction d'entête `minimum(f, A, B, eps)` qui programme l'algorithme déterminant $x^\star$. Elle suivra la structure suivante (en utilisant autant de lignes que nécessaire dans le corps de boucle) ; par ailleurs, le choix de x^g et de x^d sera tel qu'ils découpent $[g_k, d_k]$ en trois parties égales.
   On prendra soin d'expliquer les choix effectués pour compléter la fonction.

```
def minimum(f, A, B, eps) :
    g = A
    d = B
    while __________
         # calcul de x^g  et x^d
         xg = __________
         xd = __________
         # calcul de g et d
         .
         .
    return __________
```

4. On revient dans cette question à la problématique de la question 2 et on considère toujours γ et λ fixés aux valeurs γ_0 et λ_0 précédemment trouvées. Cependant, μ_0 est à déterminer à l'aide de la fonction `minimum` écrite en question 3, plutôt que par lecture graphique.

   a. Écrire une fonction d'entête `test_croissant(L)` prenant en entrée une liste de nombres et qui renvoie un booléen indiquant si les valeurs de `L` sont triées par ordre croissant.
   Cette fonction devra parcourir `L` une seule fois ; en particulier, elle ne devra pas réaliser un quelconque tri.

   b. À l'aide de `test_croissant`, proposer un raisonnement (sans écrire de code) pour tester si la fonction `M_mu` est bien croissante sur l'intervalle $[0,8\,;\,3]$.

   c. Proposer une instruction pour trouver la valeur μ_0 cherchée.

   d. On trouve $\mu_0 \approx 0,7061$: commenter et conclure sur l'estimation de la fonction a. Que penser de l'hypothèse « a constant » du premier modèle ? (qui suppose que le taux de mortalité est constant au cours des différents âges de la vie)

Annexe - Rappels de syntaxe Python pour la partie 3

– `f = open("monfichier.txt", "r")` ouvre le document intitulé `monfichier.txt` (en mode lecture) ; le fichier est alors désigné par la variable `f`.
– `f.read()` lit l'intégralité du contenu du fichier `f` et le renvoie sous la forme d'une chaîne de caractères.
– `f.readline()` lit la première ligne du fichier `f` et la renvoie sous forme d'une chaîne de caractères.
Ré-exécuter `f.readline()` lit alors la deuxième ligne ; et ainsi de suite.
– `f.close()` ferme le fichier `f`.
– Le caractère `"\n"` (de longueur 1) désigne le passage à la ligne.