Une chasse au trésor

http://www.mp933.fr/ - stephane@gonnord.org

L'objectif est d'obtenir un couple de réels (flottants) avec une assez bonne précision (précision absolue inférieure à 0.001). Pendant les diverses étapes, il convient donc de garder en tête cet objectif : plus on avance, plus les erreurs s'accumulent !

Voici le plan de la chasse au trésor :
- récupérer depuis un fichier les cent coefficients de $A \in \mathcal{M}_{10}(\mathbb{R})$ et les dix de $Y \in \mathcal{M}_{10,1}(\mathbb{R})$;
- résoudre $AX = Y$;
- résoudre une équation polynomiale (dont les coefficients sont issus de X) pour trouver $t_0 \in [1, 3]$;
- calculer une intégrale dépendant de t_0 ; en déduire un nouveau paramètre a ;
- résoudre un système différentiel paramétré par a ; en déduire la position du trésor !

Après la récupération des données, les quatre étapes « numériques » pourront être traitées soit avec des fonctions de bibliothèques, soit avec des fonctions écrites par vos soins. Une attaque raisonnable peut être de comparer les deux...

1 Récupération des données

1. Calculer $n = 31 \times m + j$, avec $1 \leqslant m \leqslant 12$ désignant le numéro de son mois de naissance et j le jour dans le mois. Par exemple, le premier janvier correspond à $n = 32$ et le 31 décembre à $n = 403$.
2. Récupérer le fichier de données correspondant dans l'archive elle-même récupérée sur le web [1]. Le 9 novembre correspondra à $n = 350$ donc au fichier `donnees350.txt`
3. Ouvrir le fichier avec un éditeur de texte pour découvrir le format : les 10 premières lignes sont constituées chacune de 10 entiers, suivies d'un flottant. Les 10 entiers correspondent à une ligne i de la matrice A, et le flottant au second membre de l'équation, donc y_i.
   Les lignes suivantes, généreusement fournies, donnent les solutions intermédiaires de la chasse.
4. Dans un script python, ouvrir le fichier (via `open`), lire les 10 premières lignes et ranger (après `split` et compagnie) les données dans un tableau bidimensionnel A et un tableau Y.
   On pourra préférer les `array` de `numpy` : débrouillez-vous !

2 Résolution du système

Résoudre l'équation $AX = Y$. Vous devez donc récupérer un tableau de 10 valeurs $x_0, ..., x_9$.

Au choix : votre propre méthode de résolution, ou bien la fonction `solve` de la bibliothèque d'algèbre linéaire `numpy.linalg`.

3 Résolution d'une équation polynomiale

Il s'agit de résoudre l'équation $P(t) = 840$, avec $P = \sum_{i=0}^{9} x_i t^i$. On est certain que cette équation possède une unique solution positive : trouvez-la ! Elle sera notée t_0 dans la suite.

Au choix : par dichotomie, méthode de Newton, ou bien la fonction `bisect` de la bibliothèque dédiée `scipy.optimize`... Je conseille de garantir une précision absolue de l'ordre de 10^{-10} : c'est facile, rapide et prudent pour la suite.

1. sauvegarder le fichier .zip ou .tar.gz sur son disque, l'ouvrir, etc...

4 Une intégrale absurde

Calculer une évaluation de $a = 20\displaystyle\int_{3/2}^{t_0} \frac{|\cos t|}{t} dt$. *On pourra utiliser une des méthodes programmées en TD. On peut également utiliser la fonction* `quad` *de la bibliothèque* `scipy.integrate`. *Je conseille de garantir une précision absolue de l'ordre de* 10^{-5}, 10^{-6}.

5 Proies et prédateurs

Allez, un dernier pour la route ! On considère maintenant le système différentiel

$$\begin{cases} x' &= (a-y)x \\ y' &= (x-1)y \end{cases}$$

On prend comme conditions initiales $x(0) = 2$ et $y(0) = 1$. Le trésor est au point $(x(10), y(10))$! Allez voir si vous avez gagné, en tentant de browser l'adresse correspondant au format suivant :

`http://blog.mp933.fr/public/info-pour-tous/dm2013_2014/tresors/1.755.5.121.txt`

Le nom de fichier est de la forme `x.xxx.y.yyy.txt` : l'abscisse et l'ordonnée sont écrites avec trois décimales après la virgules, arrondies inférieurement. Ici, la position du trésor était $(1.755721..., 5.121406...)$. Vous pouvez alors m'envoyer un mail avec votre programme, et un identifiant trouvé dans le fichier !

On pourra utiliser la méthode d'Euler programmée en TD... au moins dans le corrigé ! On peut également utiliser la fonction `odeint` *de la bibliothèque* `scipy.integrate`.