TELECOM Nancy *(ESIAL)*
Maths Numériques

TP 3

Exercice 1 *Moindres carrés*

1. Compléter votre module d'algèbre linéaire avec une fonction qui, à partir de la forme standard d'un problème de moindres carrés (minimiser la fonction d'erreur $E(x) = \|Ax - b\|^2$), c'est à dire à partir de la matrice A (de taille $m \times n$) et du vecteur b (de taille $m \times 1$) calcule la solution x^* en résolvant les équations normales (le système linéaire associé étant résolu par les deux fonctions déjà présente dans ce module). Fonction numpy utile : `dot(A1,A2)` (produit matriciel), méthode numpy utile : `A.transpose()` (ou plus court `A.T` sans parenthèses) pour transposer une matrice (dans les deux cas l'argument n'est pas modifié et la matrice transposée est retournée).
2. Un chercheur bavarois, Arnd Leike, a montré que lorsqu'on sert un verre de bière, juste après la formation du nuage de mousse (cet instant étant repéré par $t = 0$), la hauteur de la mousse diminue de manière exponentielle au cours du temps, c'est à dire qu'en notant $h(t)$ la hauteur à l'instant t on a le modèle suivant :
$$h(t) = Ce^{-\frac{t}{\tau}}$$
où C est la hauteur à l'instant $t = 0$ et τ la constante de temps associée à la décroissance exponentielle. On pourrait même reconnaître différents types de bière à cette constante... Arnd Leike a reçu un prix Nobel "Ig" en 2002 pour cette "découverte". Le modèle n'est pas linéaire par rapport au paramètre τ mais nous avons vu en TD (cf exercice 3) qu'en passant en log on obtient un modèle linéaire en les paramètres $x_1 = \log(C)$ et $x_2 = \frac{1}{\tau}$.
   (a) Compléter votre module d'algèbre linéaire (ou celui proposé sur Arche si vous n'êtes pas sûr de sa justesse) par une fonction d'entête :
   ```
   def fit_expo_decay(tm,hm):
   ```
   qui à partir des données $(t_i, h_i)_{1\leq i\leq m}$ permet d'obtenir la forme standard du problème de moindres carrés de paramètres x_1 et x_2, appelle la fonction précédente pour obtenir la solution optimale x^* et finalement retourne les deux paramètres C^* et τ^*. Pour former la matrice A de la fonction d'erreur, vous avez besoin de concaténer deux vecteurs pour obtenir une matrice $m \times 2$. Avec python-numpy c'est un peu moins pratique qu'avec matlab. Une solution naturelle serait d'utiliser la fonction `concatenate` mais cela est finalement compliqué... Une solution plus compacte est d'utiliser la fonction `array` qui peut admettre comme argument un tuple formé de tableaux [1]. Par exemple avec deux tableaux 1d `t1` et `t2` de même taille m `array((t1,t2))` retourne un tableau 2d de taille $2 \times m$ que vous pouvez ensuite transposer avec l'attribut `.T`.
   (b) Compléter le script `beer.py` [2] qui contient les données sur les variations de hauteur en fonction du temps pour la bière préférée d'Arnd Leike, à savoir, la "Erdinger Weissbier". Il suffit :
      i. d'appeler la fonction précédente pour déterminer les paramètres optimaux C^* et τ^* ;
      ii. et d'afficher la courbe obtenue avec le modèle avec les points expérimentaux, par exemple avec [3] :

1. Jusqu'à présent on avait utilisé `array` sur une liste.
2. Disponible sur Arche.
3. Autre astuce non documentée dans le tutoriel : la fonction `legend` peut admettre un tuple formé par les différentes chaînes des légendes et pas seulement un tableau 1d, au lieu d'"écrire `legend(array(['experience','modele']))` vous pouvez utiliser `legend(('experience','modele'))` ce qui est plus court !

```
# C et tau obtenus par moindres carres
tt = linspace(0, max(tm), 100);
hh = C*exp(-tt/tau);
plot(tm,hm,'ro',tt,hh,'b')
legend(('experience','modele'))
```

(c) Dans l'expérience de Leike, les erreurs sur les mesures de hauteur de mousse δh_i sont estimées : elles sont nulles en moyenne ($E[\delta h_i] = 0$) et l'écart type est évalué à $\sigma(\delta h_i) = \sigma_i$. Pour cela l'auteur a effectué plusieurs fois l'expérience ; notez que ces erreurs σ_i sont données dans `beer.py` (vecteur `dhm`). Sachant qu'elles sont différentes selon les mesures $\sigma_i \neq Cte, \forall i$ il est préférable d'utiliser les moindres carrés pondérés. Cependant comme on passe en log pour obtenir le modèle linéaire, les erreurs ne sont plus les mêmes :

$$\begin{aligned}
\log(h_i + \delta h_i) &= \log\left(h_i\left(1 + \frac{\delta h_i}{h_i}\right)\right) \\
&= \log(h_i) + \log\left(1 + \frac{\delta h_i}{h_i}\right) \\
&\simeq \log(h_i) + \frac{\delta h_i}{h_i} \quad \text{si} \quad \left|\frac{\delta h_i}{h_i}\right| << 1
\end{aligned}$$

on a donc intérêt à utiliser des moindres carrés pondérés (cf cours et l'exercice 2 du TD 6) correspondant à des données bruitées dont l'écart type est de σ_i/h_i et donc avec des poids $w_i = h_i/\sigma_i$ (on remarque d'ailleurs que même si $\sigma_i = Cte, \forall i$ il faudrait pondérer par $w_i = h_i/Cte$ du fait du passage en log).

Compléter votre module d'algèbre linéaire par une fonction d'entête :

```
def fit_expo_decay_bis(tm,hm,dhm):
```

qui intègre ces modifications et compléter le script précédent avec cette méthode de moindres carrés pondérés. De même, visualiser graphiquement la concordance du modèle avec les données : on s'apercevra que le modèle prend mieux en compte la première mesure ($t = 0$) qui correspond à une erreur très faible.