Méthodes numériques – TD4

Moindres carrés avec contraintes

Dans ce TD, nous allons dans un premier temps linéariser une fonction d'objectif Gaussienne. Dans un deuxième temps, nous nous intéresserons à améliorer cette approximation dont la linéarisation naïve conduit à une solution éloignée de la solution du problème initial. A la fin du TD, nous utiliserons le script développé pour calculer la température d'un nuage d'atome froid.

1 Linéarisation d'une fonction Gaussienne 1D

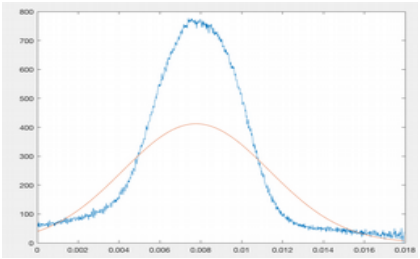
Dans la continuité des TD précédents, notre objectif est d'approcher au mieux un ensemble de N mesures (x_i,y_i) par une fonction Gaussienne d'amplitude $A>0$, d'espérance μ (moyenne), et d'écart type σ :

$$g_{A,\mu,\sigma}(x)=A\,e^{\frac{-(x-\mu)^2}{2\sigma^2}}.$$

Pour tester le script, vous utiliserez les données « data_temperature1.mat ».

1. Définissez la fonction d'objectif aux moindres carrés et montrez que le système à résoudre n'est pas linéaire.
2. Quelle transformation faut-il appliquer à la fonction Gaussienne (et donc aux mesures y_i) pour obtenir une fonction d'objectif quadratique ? (et ainsi retomber sur la résolution d'un simple système d'équations linéaires)
3. Formez le système linéaire à résoudre. Quelle condition devrions nous rajouter pour garantir l'obtention d'une forme Gaussienne ? Pour l'instant nous allons nous contenter de vérifier que la solution obtenue satisfait cette condition.
4. Testez cette procédure sur vos données, visualisez l'approximation obtenue superposée aux données. Vous devez obtenir un résultat similaire à la figure ci-contre.
5. Calculez le résidu de la fonction d'objectif initiale :

$$\sum_i (g(x_i)-y_i)^2$$



2 Améliorations de la solution

La solution obtenue précédemment est très éloignée de la solution idéale dont le résidu est de l'ordre de **1.3e+06**. Le principal problème provient du fait que la transformation que nous avons appliquée n'est pas du tout linéaire et donne beaucoup plus d'importance aux petites valeurs. Afin de tenter de nous rapprocher d'un résultat exploitable, nous allons dans la suite tester différentes stratégies.

1. Pour notre 1er essai nous allons imposer à notre solution de passer **exactement** par le point (x_i,y_i) de valeur y maximale (voir l'aide de la fonction **max** pour trouver y_i et i). Nous rajouterons cette contrainte linéaire à notre système précédent via la méthode des multiplicateurs de Lagrange. Testez, visualisez, et calculez le nouveau résidu.
   → Nous avons légèrement réduit le résidu, mais l'écart-type est encore largement surestimé.

2. Pour notre seconde tentative nous nous proposons d'imposer à la solution de rester au plus proche des mesures en la contraignant à rester dans des bornes d'erreur d'amplitude h fixée, c'est à dire : $y_i-h\leqslant g(x_i)\leqslant y_i+h$. Remarquez que ces inégalités peuvent être linéarisées de la même façon que précédemment. Le problème consiste donc à trouver la fonction g qui respecte ces contraintes et dont l'énergie quadratique établie en 1.2 est minimale. Pour résoudre ce type de problème avec Matlab, référez vous à la fonction **lsqlin**. Au passage, vous pouvez en profiter pour inclure la condition établie en 1.3. Une difficulté de cette

approche consiste à trouver le *h* le plus faible qui admet une solution. Vous pouvez par exemple partir avec *h=100* et converger vers le *h* optimal via une recherche dichotomique. Testez, visualisez, et calculez le nouveau résidu.
→ Le résultat est enfin exploitable. Remarquez que cette approche revient à minimiser la norme L^{∞} du résidu non linéaire.

3. Pour notre 3ème et dernière tentative, l'idée est de pondérer chacune de nos contraintes *douces* par un poids w_i (comme pour les « moindres carrées pondérée » vues au TD précédent) de manière à donner plus d'importance aux mesures les plus fortes et ainsi de compenser l'effet introduit par l'application du logarithme. Testez avec $w_i = y_i, y_i^2, y_i^3, y_i^4, y_i^5$ et choisissez la pondération donnant le meilleure résidu.
→ Vous devez obtenir un résidu très proche de celui de la solution obtenue par un processus d'optimisation non-linéaire.
→ Pour une explication plus rigoureuse sur le bien-fondé de cette approche, voir les explications détaillées en annexe (à lire à la maison).

3 Application : température d'un nuage d'atome froid.

Un nuage d'atomes froids piégés dans le vide a une distribution de densité de particules Gaussienne : $g_{A,\mu,\sigma}(x) = A\, e^{\frac{-(x-\mu)^2}{2\sigma^2}}$. Lorsqu'il est relâché, le nuage s'étend sous l'effet de sa température (distribution de vitesse). Lors de cette expansion, sa distribution de densité conserve une forme Gaussienne dont la variance σ est relié au temps d'expansion TOF (time of flight), à la variance de la distribution initiale σ_0 (TOF=0), et à la température T :

$$\sigma^2(\mathrm{TOF}) = \sigma_0^2 + \frac{k_b T}{m}\mathrm{TOF}^2$$

Les données « data_temperature_tot.mat » sont un temps de vol d'un nuage d'atomes froids de Rubidium pour TOF=[6 8 10 12 14 16 18] ms.

Quelle est la température du nuage ?

k_b=1.38 x 10^{-23} J/K: constante de Boltzman
m=1.44 x 10^{-25} kg : masse d'un atome de Rubidium

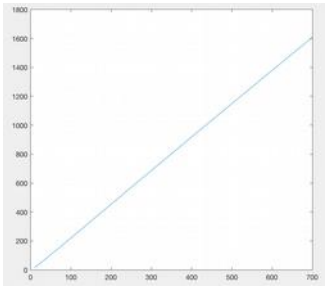
4 Annexe

En 2.3 nous avons réussi à nous rapprocher grandement du minimum de la fonction d'objective initiale : $f_{nl}(x)=\sum_i (g(x_i)-y_i)^2$ simplement en ajoutant une pondération non uniforme aux échantillons. Cela n'a rien de magique. Rappelons que nous avons linéarisé cette énergie non-linéaire en une énergie quadratique en appliquant une transformation T à chacun des termes :

$$f_l(x)=\sum_i (T(g(x_i))-T(y_i))^2 \quad .$$

En section 1, nous avons simplement pris $T(y)=\log(y)$, et comme cette transformation T n'est pas du tout linéaire le minimum de f_l n'est pas du tout égal au minimum de f_{nl}. Remarquez en revanche que si T est linéaire, c'est à dire que si T est de la forme $T(y)=a\,y+b$, alors les deux énergies admettent le même minimum. L'idée est donc trouver une nouvelle transformation T qui soit quasi linéaire sur le domaine considéré [1,700].

Par exemple prenons T de la forme $T(y)=\log(y)\,y^d$ et cherchons la valeur de l'exposant d pour T soit le plus rectiligne possible, par exemple en cherchant d tel que : $(T(1)+T(y_{max}))/2=T(1+y_{max})$. Puisque $\log(1)=0$, nous pouvons facilement résoudre cette équation, et pour $y_{max}=700$ nous trouvons *d=0.84* qui effectivement donne visuellement une transformation très linéaire (voir figure ci-contre).



Bien sur, cette transformation appliquée à notre Gaussienne g ne donne plus une simple forme quadratique, mais nous pouvons l'approcher par : $T(g(x_i))=\log(g(x_i))\,g(x_i)^{0.84}\approx\log(g(x_i))\,y_i^{0.84}$ ce qui est raisonnable puisque $g(x_i)$ est sensé être proche de y_i . Nous obtenons donc une pondération par $y_i^{0.84}$ similaire à ce que nous avons fait intuitivement en 2.3. Cependant, vous avez probablement trouvé une pondération optimale bien différente de celle-ci. La raison est que les données sur lesquels nous travaillons, notamment à cause des zones de faible valeur, ne peuvent pas être bien approchées par une Gaussienne ce qui viole notre hypothèse que $g(x_i)\approx y_i$. A ce stade, une idée simple consiste donc a réaliser plusieurs itérations en raffinant notre estimation de $g(x_i)$ a chaque itération :

1. $z_i=y_i$ (initial estimate)
2. répéter jusqu'à convergence :
3. $g=\underset{g}{argmin}\sum_i \left(\log(g(x_i))\,z_i^{0.84}-\log(y_i)\,y_i^{0.84}\right)^2$ (linear solve - minimization)
4. $z_i \leftarrow \alpha\, z_i+(1-\alpha)\,g(x_i)$ (update estimate - expectation)

où le facteur α permet de mettre à jour progressivement l'estimation et à l'algorithme de converger.

Remarque finale : quelque soit la stratégie choisie, malgré la linéarisation du problème, afin d'obtenir une solution satisfaisante nous avons du passer un processus itératif afin de trouver une borne d'erreur minimale en 2.2, une pondération optimale en 2.3, ou encore ici pour raffiner itérativement notre expectation.