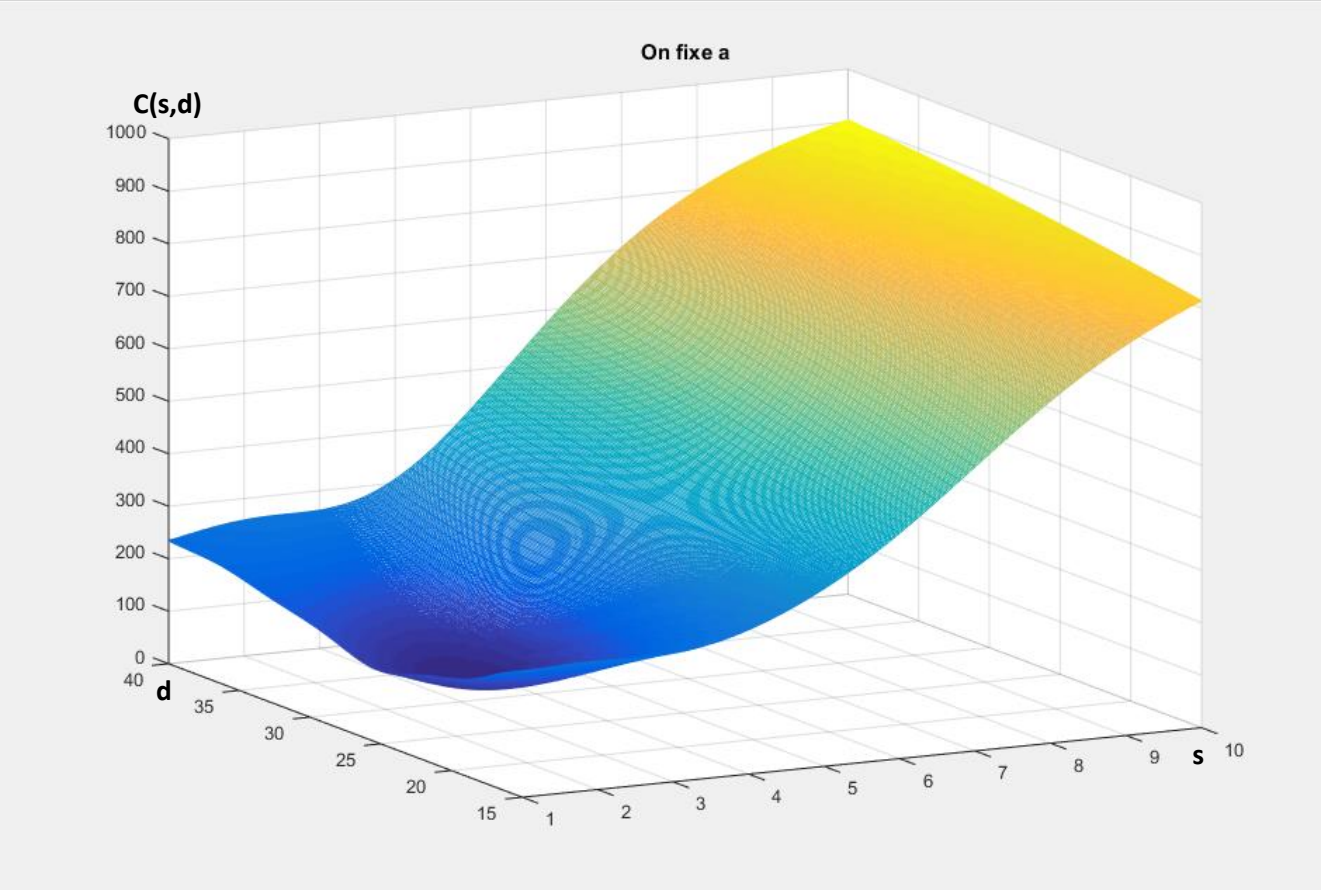
Azag Djessim
Gybels Hugo

Projet Optimisation Continue Partie 2 :

Question 10 : Représentation de la fonction de Coût en fonction du vecteur d'inconnues $\theta = (d, a, s)^t$.

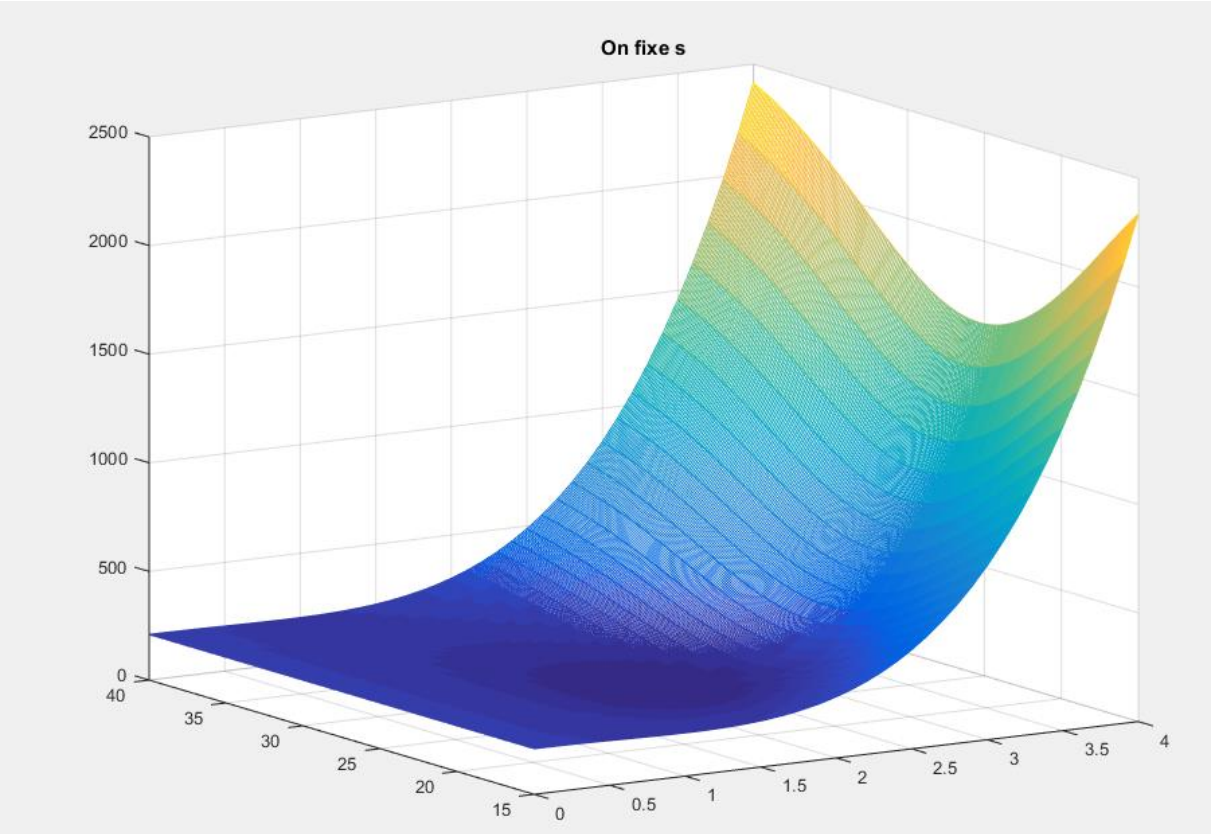
On fixe a = 1.8, et on trace C(d,s) :



On retrouve bien les valeurs minimales de C pour $d \approx 25$ et $s \approx 2$.

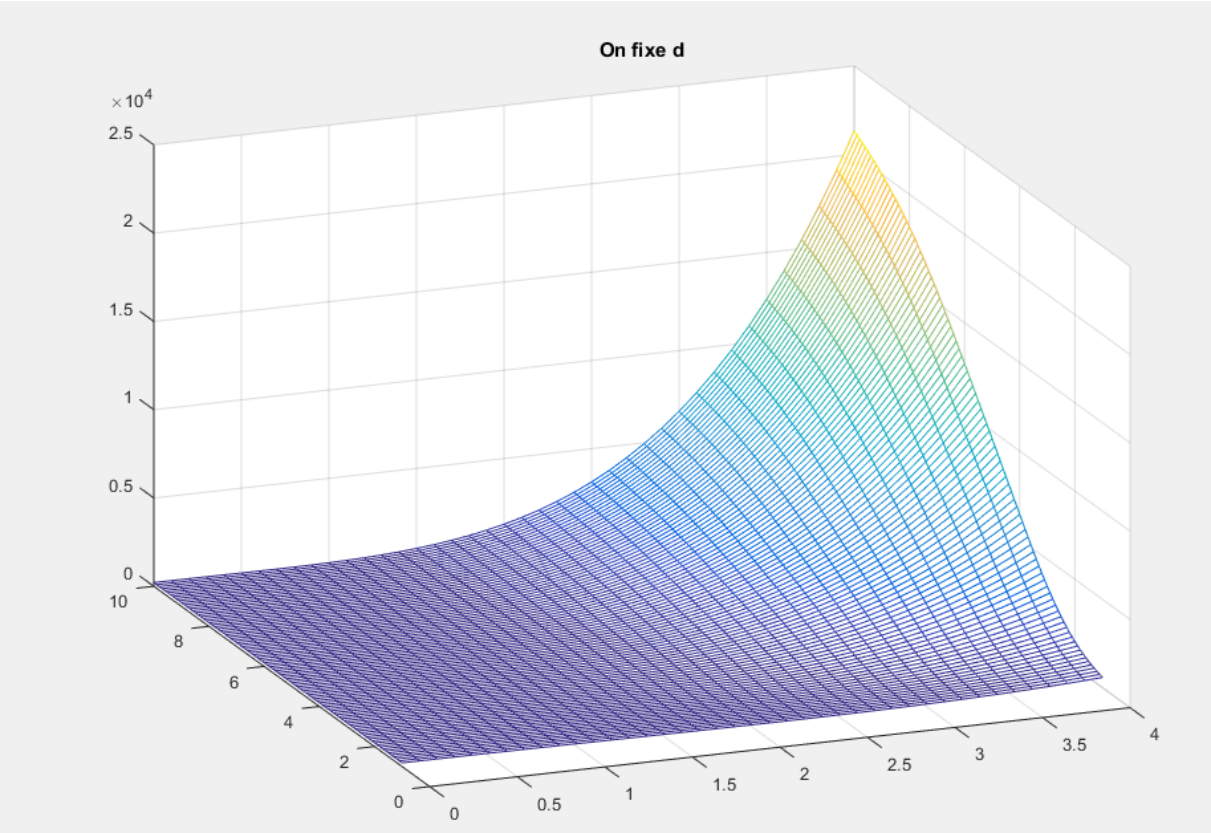
Azag Djessim
Gybels Hugo

On fixe s = 2.1, et on trace C(d,a) :



On retrouve bien les valeurs minimales de C pour $d \approx 25$ et $a \approx 2$.

On fixe a = 1.8, et on trace C(d,s) :



Azag Djessim
Gybels Hugo

Question 11 : Estimation des paramètres

On cherche à estimer les trois paramètres de la fonction de coût en observant les points donnés dans la description du problème. Pour cela, on choisit un intervalle dans lequel on cherchera chacune des variables.

- On commence par définir un intervalle dans lequel chercher d. **d** représente **le point autour duquel la gaussienne est centrée**. On choisit donc d entre **15 et 40** de manière à être certain de trouver une approximation du minimal.
  Il faudra pour finir déterminer un pas pour d. Plus ce pas sera petit, plus le résultat sera précis ; et plus il sera grand plus le résultat sera rapide mais moins précis.
- On cherche également à définir une valeur de **s** (l'écart-type). Pour cela on estime la largeur de la gaussienne lorsque **l'amplitude est égale à la moitié de l'amplitude maximale**. On estime donc s autour de **15**.
- Finalement on estime l'**amplitude maximale** de la gaussienne (**a**) entre **2 et 4**.

Question 12 : Division de l'erreur absolue

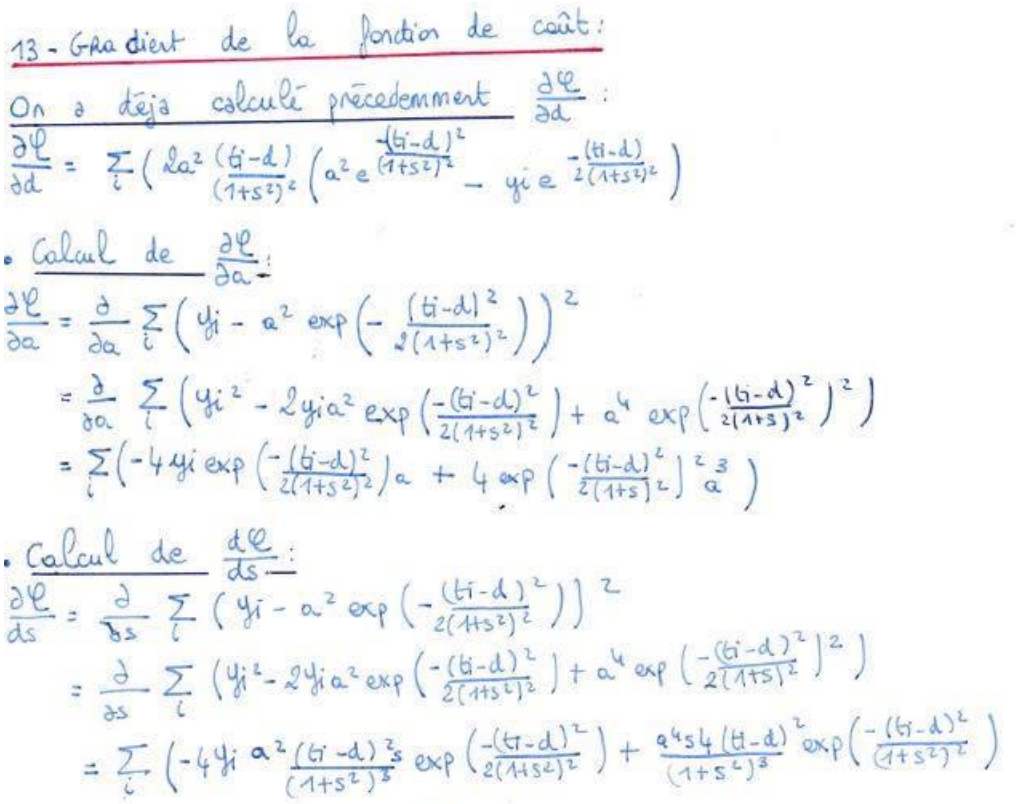
Nous avions choisi comme erreur 0.1. On souhaite diviser cette erreur par 2 pour **chacune des trois variables**. Lorsqu'on divise par deux l'erreur sur une variable, on divise le pas de la variable ; ce qui revient à multiplier le nombre d'itérations par 2.

De plus, on multiplie le nombre d'itérations pour chacune des trois variables ; ce qui revient à multiplier le nombre total d'itérations par 2^3soit **par 8.**

Pour n variables on multiplie donc globalement de temps de recherche par $\mathbf{2^n}$**.**

Azag Djessim
Gybels Hugo

Question 13 : Gradient



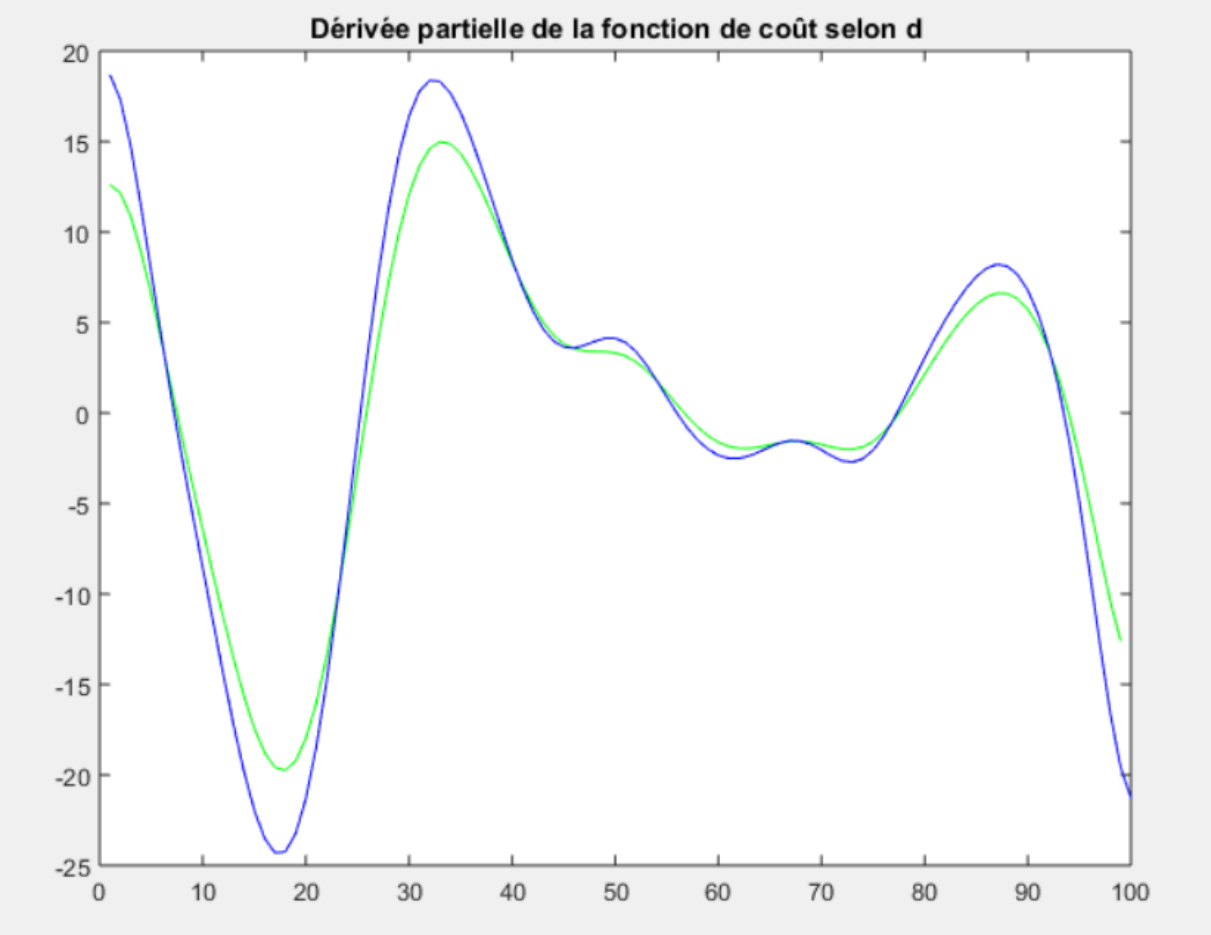
On a donc le gradient de la forme suivante :

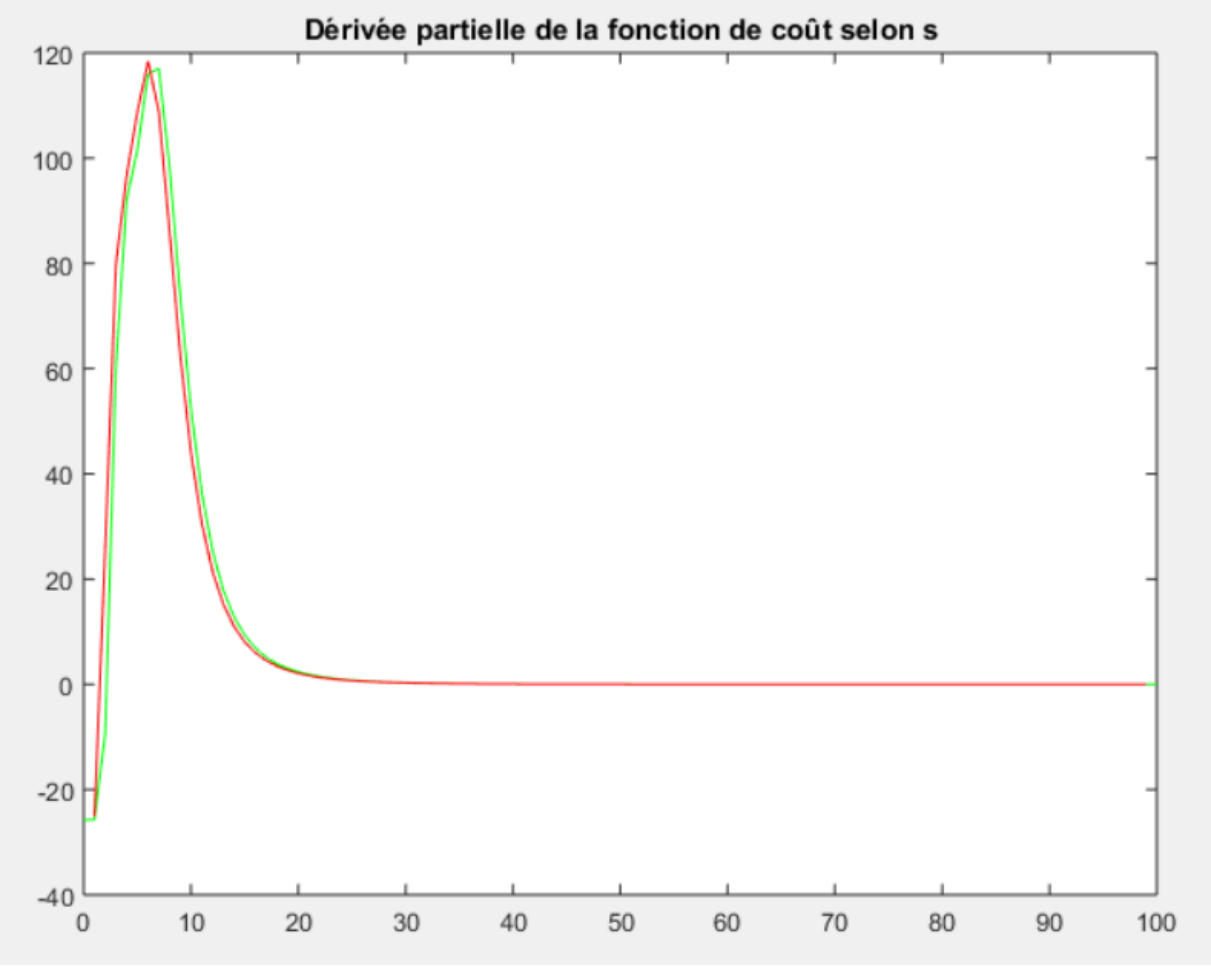
$$\nabla_\theta f = \begin{matrix} \frac{\partial f}{\partial d} \\ \frac{\partial f}{\partial s} \\ \frac{\partial f}{\partial a} \end{matrix}$$

Question 14 : Dérivées avec Diff()

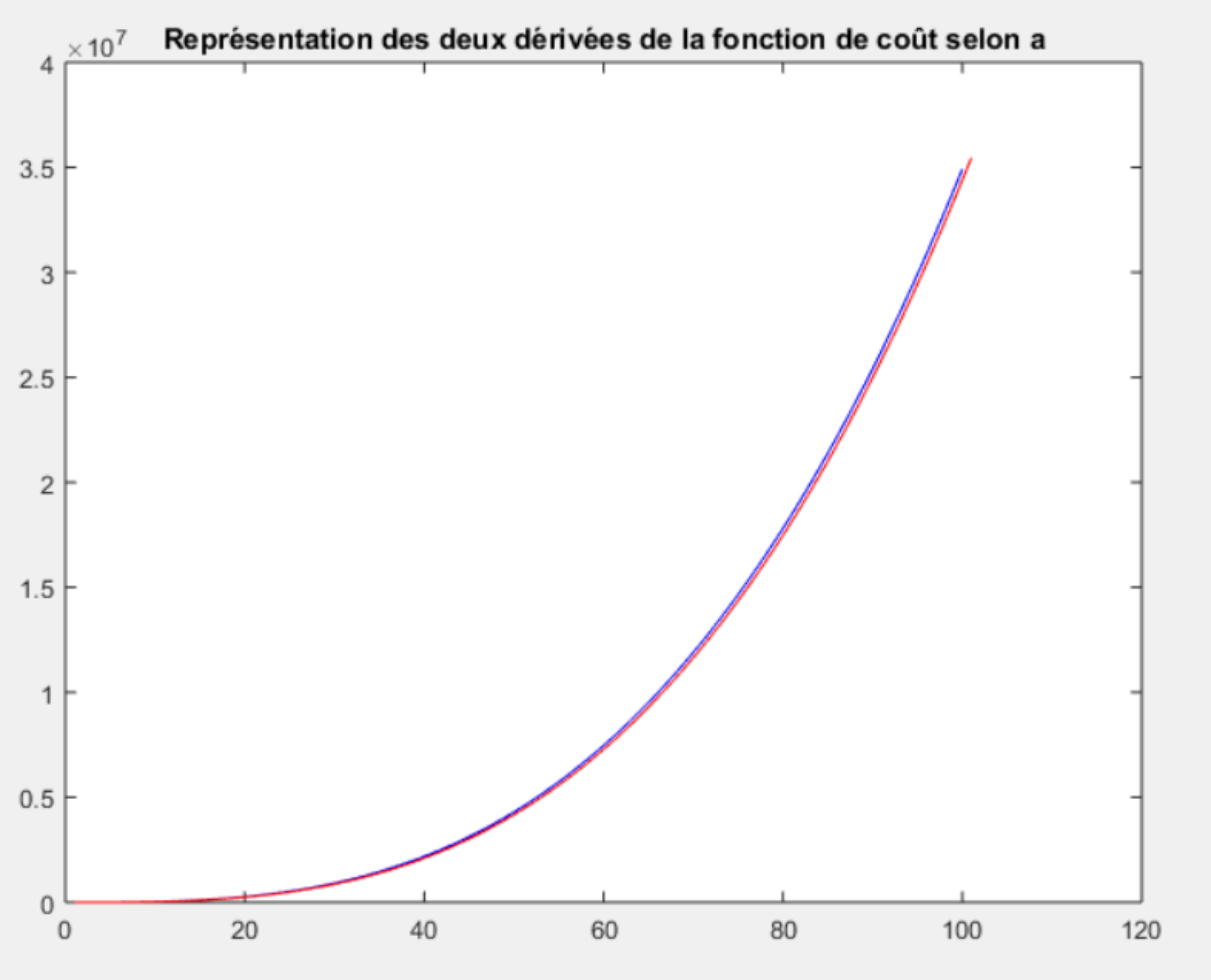
On souhaite désormais vérifier le calcul des dérivées obtenues ci-dessus. Pour cela on utilise la fonction de dérivée intégrée à matlab Diff(). Pour chacune des trois variables on en fixe 2 et on calcule la dérivée de la fonction de coût pour les variations de la troisième.

Azag Djessim
Gybels Hugo





Azag Djessim
Gybels Hugo



Nous avons ensuite calculé l'erreur relative maximale commise pour chacune des trois courbes. On trouve les valeurs suivantes :

- ErreurRelativeMaximaleD ≈ 20%
- ErreurRelativeMaximaleS ≈ 23%
- ErreurRelativeMaximaleA ≈ 1%

L'erreur relative maximal est importante sur S bien que les deux courbes semblent proche à cause de l'importante pente de la courbe, ainsi un légers retard entraine une importante erreur à un point donné.

Question 15 : Méthode des plus fortes pentes

Pour trouver le minimum de la fonction coût dépendant de 3 variables, il nous fait implémenter la méthode des plus fortes pentes. Pour cela on calcule d'abord la direction dk qui correspond à la direction opposé du gradient car c'est la direction pour laquelle la descente est la plus importante.

On utilise ensuite l'algorithme de Fletcher-Lemarchal qui va nous permettre de trouver un pas adéquat grâce aux conditions de Wolfe. Ce pas va assurer une diminution de la fonction coût jusqu'à trouver le minimum.

Une fois qu'on a la direction et le pas on peut calculer le nouveau point se rapprochant du minimum.

On répète ces étapes jusqu'à obtenir le minimum avec une erreur satisfaisante.

Pour notre part, on est parti du point X = [18 6 3] avec un pas initial de 0.01

On obtient le point suivant Xmin = [25.3505 2.0851 1.8517] au bout de 339 itérations. Ce calcul a demandé 2.6 secondes ce qui est beaucoup plus court que les méthodes utilisées précédemment pour une précision aussi faible, on remarque également que ce point est en adéquation avec ce que nous avions trouvé précédemment.

On remarque que l'on s'approche relativement rapidement du résultat lorsque l'on est très éloigné : le pas s'adapte et est plus grand lorsqu'on est loin du résultat.

Question 16 : Impacte de la variation des paramètres

Nous proposons maintenant d'étudier l'impact des paramètres d'entrée de la fonction de recherche linéaire.

Premièrement, nous pouvons remarque que peu importe le point de départ, l'algorithme converge toujours vers le même point correspondant au minimum.

Le choix d'une direction différente de l'opposé du gradient ralentirait considérablement la recherche car par définition l'opposé du gradient constitue la direction optimale.

Si on prend une valeur plus grande pour α, le nombre d'itération de l'algorithme total n'augmente pas, au contraire il est légèrement plus faible. Cependant, le temps de recherche du pas est plus de 10 fois plus grande pour α=2. C'est la méthode de Fletcher Lemarechal qui met beaucoup plus de temps à converger.

La variation de lambda a très peu d'influence sur le temps de calcul total et le nombre d'itérations. Le résultat peut varier de quelques secondes mais son choix est moins décisif que pour α.

Si on prend une valeur trop élevée de β1 ou trop faible pour β2 l'algorithme de Fletcher Lemarechal ne converge plus, il ne retourne jamais une valeur de α. Si on les prend suffisamment espacés pour que l'algorithme converge on trouve rapidement un résultat, cependant ce dernier est beaucoup moins précis.

La valeur de ε permet simplement de choisir la précision du résultat, plus ce dernier est petit plus le calcul est long et inversement.

Question 17 : Méthode de Quasi-Newton

Nous voulons maintenant trouver ce résultat à l'aide d'une autre méthode. Nous avons donc implémenté la méthode de Quasi-Newton et plus précisément la méthode BFGS. Cet algorithme reprend l'algorithme de Newton linéaire, seulement il utilise une approximation qui est tout le temps définie ce qui rend cet algorithme fonctionnel et rapide.

Azag Djessim
Gybels Hugo

Pour l'implémentation nous réutilisons la méthode de Fletcher-Lemarchal utilisée précédemment pour déterminer le pas. Chaque itération de cet algorithme se déroule en 4 étapes :

- Le calcul de la direction dk grâce au gradient et à l'approximation de l'inverse de la matrice Hesssienne. On initialise cette matrice à la matrice I pour qu'elle soit inversible à la première itération.
- On détermine ensuite alpha de la même façon que pour l'algorithme de la plus forte pente.
- Une fois alpha et la direction calculés, on détermine le nouveau point.
- Puis pour terminer on met à jour la valeur de la matrice Hessienne grâce à l'approximation BFGS.

On approche ici très rapidement le résultat (seulement 16 itérations) et le temps d'exécution est de seulement quelques secondes.

On a donc une méthode beaucoup plus rapide, qui converge dans la plupart des situations (même si le point est très éloigné). Cependant le résultat est un peu moins précis que pour la méthode des plus fortes pentes.

Question 18 : Fonction fminunc()

On utilise maintenant la fonction fournie par Matlab permettant de déterminer un minimum pour une fonction à plusieurs variables de façon à pouvoir évaluer nos résultats. Cette fonction s'implémente de la façon suivante :

```
%Question 18 :
i = 1:100;
c = @(x) sum((A.sig_noisy(i) - x(2).^2*exp(-(i-x(1)).^2/(2*(1+x(3)^2).^2))).^2);
x0 = [20,3,4];
x = fminunc(c,x0)
```

Et on obtient :

```
x =

   25.3523    1.8517    2.0852
```

Ce résultat est donc cohérent avec tous les résultats que nous avions obtenus précédemment. Toutefois il faut simplement savoir que cette fonction retourne le minimum le plus proche du point initial donc si on prend une valeur trop éloignée du minimum global on ne tombera pas obligatoirement sur le bon résultat.