COMPTE-RENDU DE TP

Optimisation – Mini-projet TP – Thomas ROBERT

1. Introduction

1.1. Le problème

On dispose de points $p_i = (x_i, y_i)^\top \in \mathbb{R}^2$ et de temps $t_i \in [0,1]$ pour chaque point ($i \in [\![1, N]\!]$). On veut approximer ces points par une fonction polynomiale $m(t) = \big(x(t), y(t)\big)$.

Ce problème demande une optimisation en deux étapes similaire aux algorithmes de type *Expectation-Maximisation* (EM) (partie 4).

Une première étape (traitée en partie 2) d'estimation des t_i pour $m(t)$ et p_i connus afin de trouver les t_i qui minimisent les écarts aux carrés (critère des moindres carrés) entre les points p_i et $m(t_i)$:

$$\min_{t_i} J = \sum_{i=1}^{N} J_i(t_i) = \sum_{i=1}^{N} d_i^2(t_i) = \sum_{i=1}^{N} \|m(t_i) - p_i\|^2$$

Et une seconde étape (traitée en partie 3) d'estimation des paramètres du modèle $m(t)$ pour t_i et p_i connus.

1.2. Notations

On a pour modèle $m(t) = at^3 + bt^2 + ct + d$ avec $a, b, c, d \in \mathbb{R}^2$.

Pour simplifier les calculs, on passe en notation matricielle. On pose ainsi M la matrice des paramètres et θ_i le vecteur des puissances de $t_{i,n}$, tel que :

$$M = \begin{bmatrix} a^\top \\ b^\top \\ c^\top \\ d^\top \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \\ c_1 & c_2 \\ d_1 & d_2 \end{bmatrix} \qquad \theta_i = \begin{pmatrix} {t_{i,n}}^3 \\ {t_{i,n}}^2 \\ t_{i,n} \\ 1 \end{pmatrix} \qquad m(t_{i,n}) = M^\top \theta_i$$

On notera par « $'$ » les dérivés des éléments.

Enfin les questions du sujet de TP seront identifiées dans le texte tel que la question 1.1 sera notée « **(Q1.1)** » afin d'identifier où sont les réponses aux diverses questions.

1.3. Code Matlab

A chaque partie de ce compte-rendu correspond un fichier Matlab nommé *PartieX.m*. Certains fichiers sont composé de sections de code Matlab[1] correspondant aux diverses étapes de la partie.

[1] Voir http://fr.mathworks.com/help/matlab/matlab_prog/run-sections-of-programs.html

2. Estimation des t_i

On commence par estimer les t_i par descente de gradient et descente de Gauss-Newton, en supposant connus le modèle M et les points p_i.

On cherche à résoudre le problème :

$$\min_{t_i, i\in[\![1,N]\!]} J(t_i) = \sum_{i=1}^{N} J_i(t_i) = \sum_{i=1}^{N} d_i^2(t_i) = \sum_{i=1}^{N} \|m(t_i) - p_i\|^2$$

On résout ce problème indépendamment pour chaque t_i, puisqu'ils sont indépendants. On résout donc N fois le problème :

$$\min_{t_i} J_i(t_i) = d_i^2(t_i) = \|m(t_i) - p_i\|^2$$

2.1. Descente de gradient

(Q1.1) On utilise la méthode du gradient, on va donc modifier t_i afin de descendre d'un pas α_n dans le sens opposé au gradient du critère J_i. On a donc :

$$\boxed{t_{i,n+1} = t_{i,n} - \alpha_n \nabla_{t_i} J_i(t_{i,n})}$$

Avec :

$$\begin{aligned}
\nabla_{t_i} J_i(t_{i,n}) &= \nabla_{t_i} d_i^2(t_{i,n}) \\
&= \nabla_{t_i} \|m(t_{i,n}) - p_i\|^2 \\
&= \nabla_{t_i} \left[(m(t_{i,n}) - p_i)^\top (m(t_{i,n}) - p_i) \right] \\
&= \nabla_{t_i} \left[m(t_{i,n})^\top m(t_{i,n}) + p_i^\top p_i - 2 m(t_{i,n})^\top p_i \right] \\
\boxed{\nabla_{t_i} J_i(t_{i,n}) &= 2 \left[m(t_{i,n})^\top m'(t_{i,n}) - m'(t_{i,n})^\top p_i \right]}
\end{aligned}$$

(Q1.2) On peut par ailleurs calculer la dérivée de $m(t)$ qui est :

$$\boxed{m'(t) = M^\top \theta_i'} = 3at^2 + 2bt + c \quad \text{avec} \quad \boxed{\theta_i' = \begin{pmatrix} 3{t_{i,n}}^2 \\ 2t_{i,n} \\ 1 \\ 0 \end{pmatrix}}$$

En réinjectant m et m' dans l'expression de $\nabla_{t_i} J_i(t_{i,n})$ et on obtient :

$$\boxed{\nabla_{t_i} J_i(t_{i,n}) = 2\left[\theta_i^\top M M^\top \theta_i' - {\theta_i'}^\top M p_i\right]}$$

Et on obtient donc :

$$\boxed{t_{i,n+1} = t_{i,n} - 2\alpha_n\left[\theta_i^\top M M^\top \theta_i' - {\theta_i'}^\top M p_i\right]}$$

Notons que l'on utilisera une méthode d'adaptation du pas afin d'améliorer la vitesse de descente et de s'assurer une diminution du coût.

2.2. Descente de Gauss-Newton

On implémente désormais la méthode de Gauss-Newton. Pour cela, on utilise une nouvelle direction de descente basée sur la hessienne du critère.

(Q1.4) La modification de t_i s'écrit cette fois comme :

$$\boxed{t_{i,n+1} = t_{i,n} - H_{t_i}(t_{i,n})^{-1} \nabla_{t_i} J_i(t_{i,n})}$$

Avec $\nabla J_i(t_{i,n})$ déjà calculé précédemment, et :

$$\begin{aligned} H_{t_i}(t_{i,n}) &= \nabla_{t_i} \nabla_{t_i} J_i(t_{i,n}) \\ &= 2\nabla_{t_i} \left[m(t_{i,n})^\top m'(t_{i,n}) - m'(t_{i,n})^\top p_i \right] \end{aligned}$$

$$\boxed{H_{t_i}(t_{i,n}) = 2 \left[m(t_{i,n})^\top m''(t_{i,n}) + m'(t_{i,n})^\top m'(t_{i,n}) - m''(t_{i,n})^\top p_i \right]}$$

(Q1.5) On peut calculer la valeur de m'' dans notre cas particulier. On a :

$$\boxed{m''(t_{i,n}) = M^\top \theta_i''} \quad \text{avec} \quad \boxed{\theta_i'' = \begin{pmatrix} 6t_{i,n} \\ 2 \\ 0 \\ 0 \end{pmatrix}}$$

On a donc :

$$\boxed{H_{t_i}(t_{i,n}) = 2\left[\theta_i^\top M M^\top \theta_i'' + \theta_i'^\top M M^\top \theta_i' - \theta_i''^\top M p_i \right]}$$

En réinjectant ce résultat dans $t_{i,n+1}$, on a :

$$\boxed{t_{i,n+1} = t_{i,n} - \left[\theta_i^\top M M^\top \theta_i'' + \theta_i'^\top M M^\top \theta_i' - \theta_i''^\top M p_i \right]^{-1} \left[\theta_i^\top M M^\top \theta_i' - \theta_i'^\top M p_i \right]}$$

2.3. Résultats

(Q1.3) (Q1.6) On implémente ces deux méthodes en Matlab. **(Q1.7)** On obtient, pour 4 itérations, des résultats conformes à l'énoncé du TP.

(Q1.8) On peut observer la façon dont les deux méthodes convergent selon la position initiale.

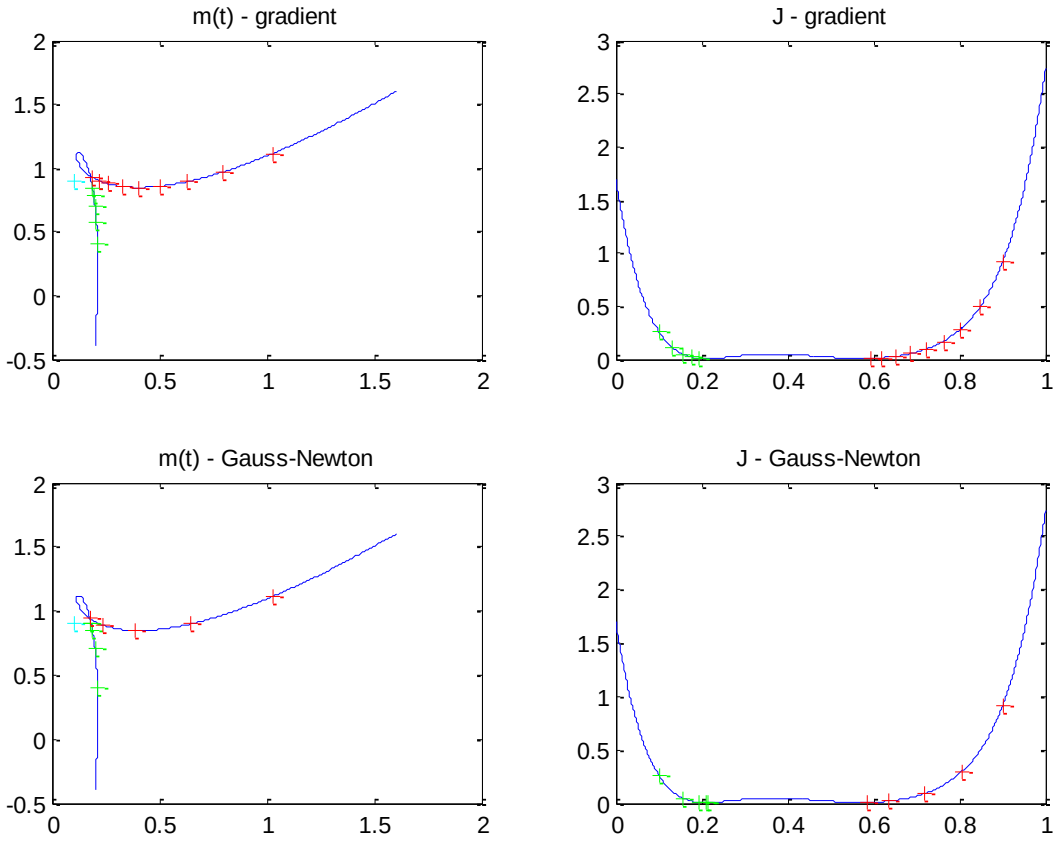

Figure 1 : Essai des méthodes de descente de gradient et Gauss-Newton

On essaye ici des initialisations à 0,1 et 0,9. On constate que la fonction de coût n'est pas convexe et qu'il n'existe pas de minimum global. En effet, les deux méthodes convergent vers deux positions différentes. On note aussi que, comme prévu, la méthode de Gauss-Newton converge en moins d'itérations que la méthode du gradient.

3. Estimation du modèle $m(t)$

Cette fois, on considère connu les points p_i et leurs positions t_i, on cherche à estimer le modèle polynomial $m(t)$ présenté au début de ce rapport, et plus précisément les paramètres M de ce modèle.

(Q2.1) Commençons par poser le critère J de ce problème, à minimiser. On veut minimiser la distance entre le modèle et les points pour chaque couple (p_i, t_i). Soit :

$$J = \sum_{i=1}^{N} \|m(t_i) - p_i\|_2^2$$

$$\boxed{J = \sum_{i=1}^{N} \|M^\top \theta_i - p_i\|^2}$$

$$= \sum_{i=1}^{N} (M^\top \theta_i - p_i)^\top (M^\top \theta_i - p_i)$$

$$J = \sum_{i=1}^{N} \theta_i^\top M M^\top \theta_i + p_i^\top p_i - 2 p_i^\top M^\top \theta_i$$

(Q2.2) On peut décomposer M en deux vecteur colonne tel que $M = [M_x \quad M_y]$. On peut alors écrire le coût comme :

$$J = \sum_{i=1}^{N} \theta_i^\top [M_x \quad M_y][M_x \quad M_y]^\top \theta_i + p_i^\top p_i - 2 p_i^\top [M_x \quad M_y]^\top \theta_i$$

$$= \sum_{i=1}^{N} \theta_i^\top M_x M_x^\top \theta_i + \theta_i^\top M_y M_y^\top \theta_i + p_i^\top p_i - 2\left(p_{i_x} M_x^\top \theta_i + p_{i_y} M_y^\top \theta_i\right)$$

$$J = \underbrace{\sum_{i=1}^{N} \theta_i^\top M_x M_x^\top \theta_i - 2 p_{i_x} M_x^\top \theta_i}_{J_x} + \underbrace{\sum_{i=1}^{N} \theta_i^\top M_y M_y^\top \theta_i - 2 p_{i_y} M_y^\top \theta_i}_{J_y} + p_i^\top p_i$$

On peut donc décomposer le critère en deux critères puisque chaque moitié du modèle n'intervient qu'avec une seule des deux composantes de chaque point. Les deux coordonnées sont donc indépendantes l'un de l'autre dans ce problème de minimisation.

(Q2.3) On a donc affaire à un problème de moindres carrés que l'on implémente en Matlab. Pour cela, on propose de réécrire le problème de façon matricielle sans la somme sur i.

Pour cela, on pose :

$$\boxed{\Theta = \begin{bmatrix} \theta_1^\top \\ \vdots \\ \theta_N^\top \end{bmatrix} \qquad P = \begin{bmatrix} p_{1_x} & p_{1_y} \\ \vdots & \vdots \\ p_{N_x} & p_{N_y} \end{bmatrix}}$$

On peut alors réécrire le coût comme :

$$\boxed{J = \|\Theta M - P\|_F^2}$$

La solution du problème est donc :

$$\boxed{\widehat{M} = (\Theta^\top \Theta)^{-1} (\Theta^\top M)}$$

(Q2.4) On teste cette méthode avec les données p_i fournies pour le TP et t_i répartis uniformément entre 0 et 1.

On obtient la solution suivante :

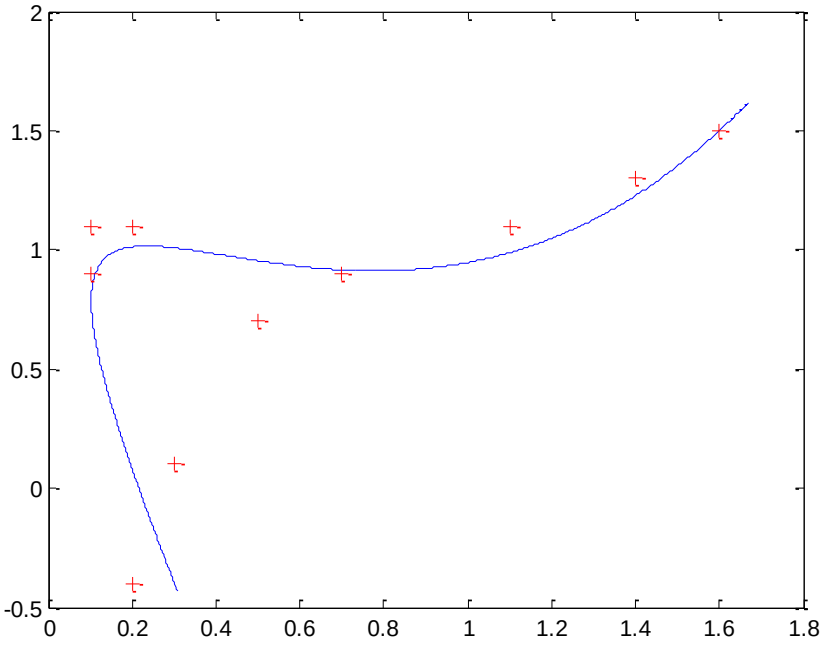

Figure 2 : Résultats de l'optimisation du modèle $m(t)$ par la méthode des moindres carrés

4. Estimation E-M des t_i et du modèle $m(t)$

L'objectif de cette partie est donc d'utiliser alternativement les deux étapes précédentes à l'image des algorithmes E-M (*Expectation-Maximisation*), et ce afin de déterminer conjointement les t_i et le modèle $m(t)$ à partir des points p_i. Ce problème peut également être vu comme un problème d'optimisation bi-niveau :

$$\begin{aligned} \min_{M} \quad & \|\Theta M - P\|_F^2 \\ \text{s.c.} \quad & t_i = \underset{t_i}{\operatorname{argmin}} \|\theta_i M - p_i\|^2 \ \ \forall\, i \in [\![1; N]\!] \end{aligned}$$

(Q3.1) On met donc en place un premier algorithme qui alterne ces deux niveaux d'optimisation, et on observe sa convergence, tout d'abord à partir d'une initialisation des t_i uniforme dans [0,1].

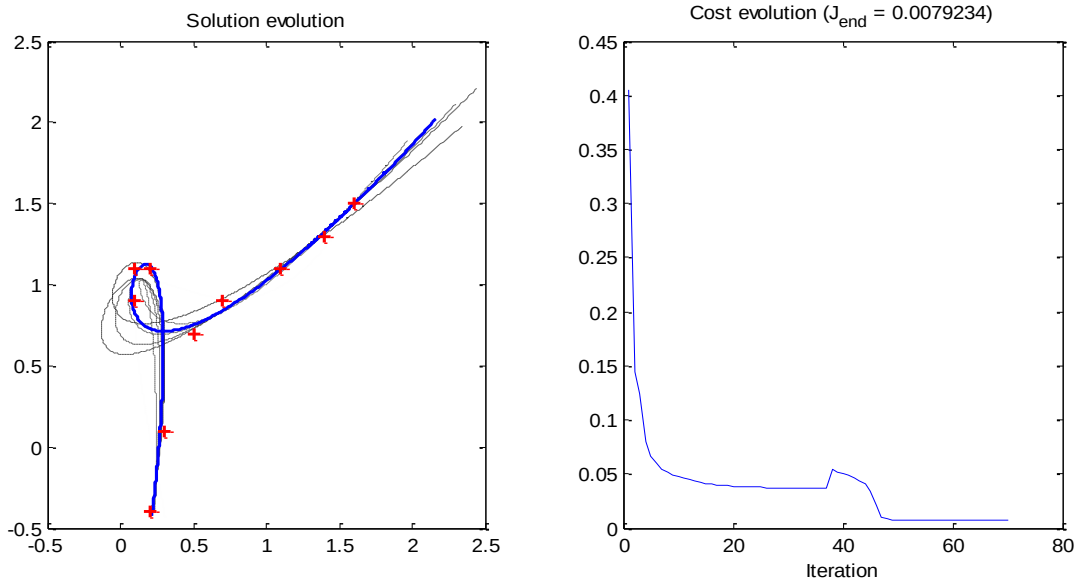

Figure 3 : Résultats avec une initialisation linéaire des t_i dans [0,1]

On visualise sur la figure ci-dessus diverses itérations de la méthode ainsi que la solution finalement trouvée. Le résultat est plutôt satisfaisant.

(Q3.2) Si les t_i sont initialisé aléatoirement entre [0,1], on obtient des solutions différentes, montrant qu'il existe des minimas locaux à notre problème d'optimisation bi-niveau.

On peut par exemple obtenir le résultat suivant, dont le coût final est supérieur à celui de la solution précédente :

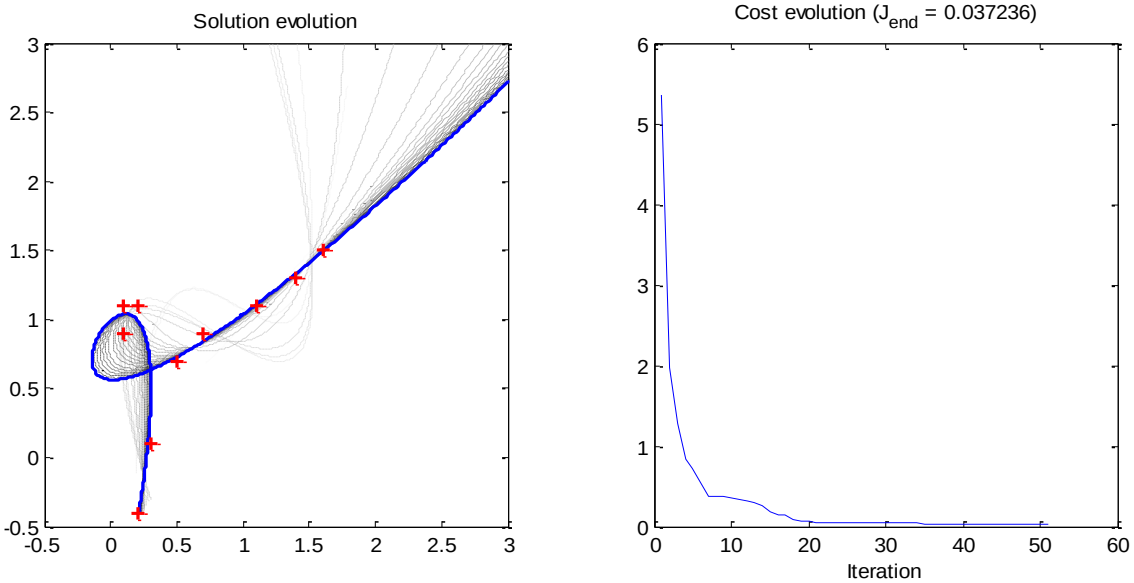

Figure 4 : Résultats avec une initialisation aléatoire des t_i

(Q3.3) Injectons maintenant de l'aléatoire dans les valeurs de t_i si le critère varie de moins de 10^{-3} d'une itération à l'autre. On peut ainsi espérer sortir d'un minimum local et espérer converger finalement vers un autre minimum local, et peut-être, avec de la chance, le minimum global.

(Q3.4) Le problème de cette injection d'aléatoire, c'est que l'on ne sait plus quand arrêter l'algorithme, puisque dès que l'on arrive à une solution, on en part. Un critère d'arrêt possible avec cette solution pourrait être de fixer un nombre d'itération à réaliser. Par exemple, pour 300 itérations on obtient la solution suivante :

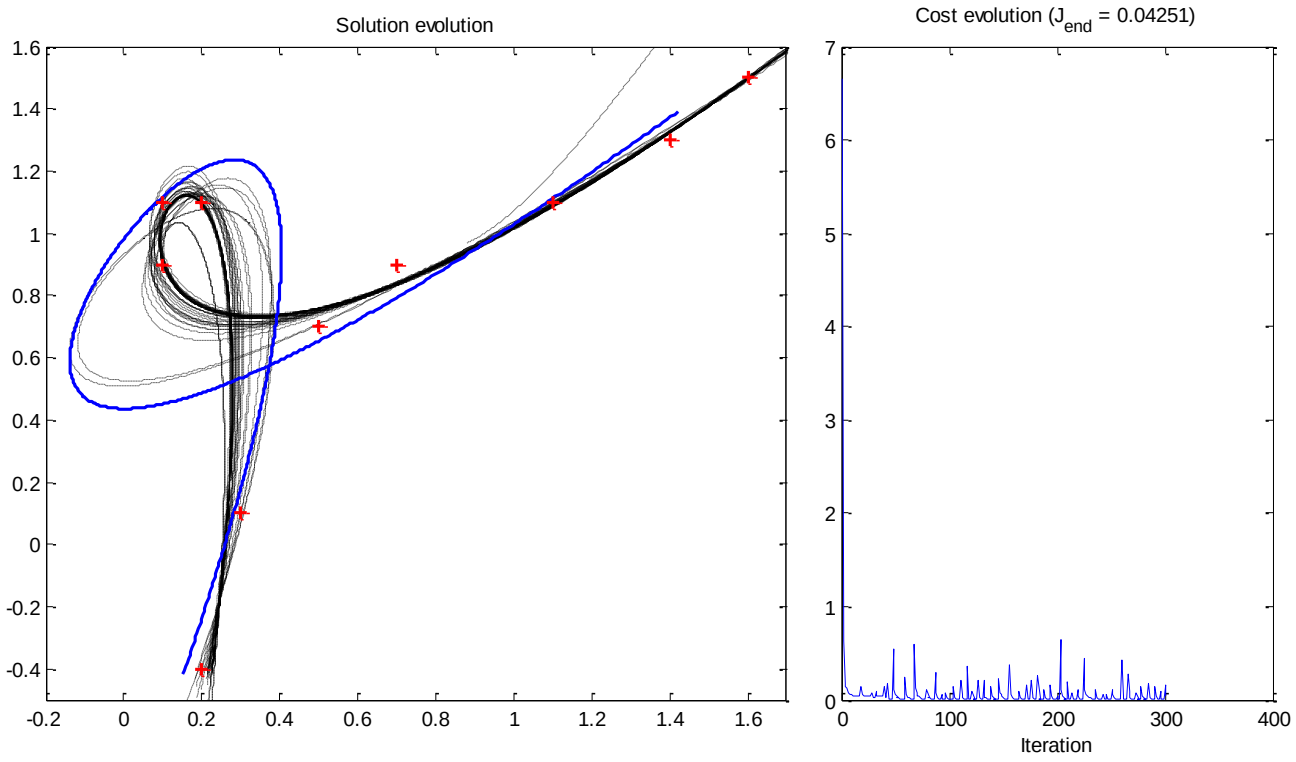

Figure 5 : Résultats obtenus par injection d'aléatoire après chaque minimum local

On constate que l'on a effectivement convergé plusieurs fois. Notons d'ailleurs que les lignes pointillées noires sont les minimas locaux à partir desquels de l'aléatoire a été injecté.

(Q3.5) Cependant, en utilisant un nombre d'itération prédéfini, on ne s'assure absolument pas de s'arrêter dans un minimum, et surtout, il n'y a aucune raison que le dernier minimum trouvé soit le meilleur (au sens du critère). La solution finale présentée sur la figure précédente n'est très probablement pas un minimum local.

Il semble donc nécessaire d'enregistrer la meilleure solution trouvée au cours des itérations, afin de finalement conserver celle-ci.

On relance l'algorithme en conservant la meilleure solution, que l'on affiche en bleu, avec les minimas locaux en noir. Voir la Figure 6 ci-après. On trouve bien la solution donnée par l'énoncé du TP.

Le meilleur coût trouvé est de $8{,}06 \times 10^{-3}$. Au début de cette partie, nous avions trouvé une solution avec un coût de $7{,}92 \times 10^{-3}$, c'est en fait dû au fait que l'on arrête ici l'optimisation dès que le coût ne varie que de 10^{-3} alors qu'au début, le critère d'arrêt était une variation de 10^{-9}. On retrouve effectivement cette valeur de $7{,}92 \times 10^{-3}$ dans cette partie avec aléatoire si on attend une variation de 10^{-9} avant d'injecter de l'aléatoire.

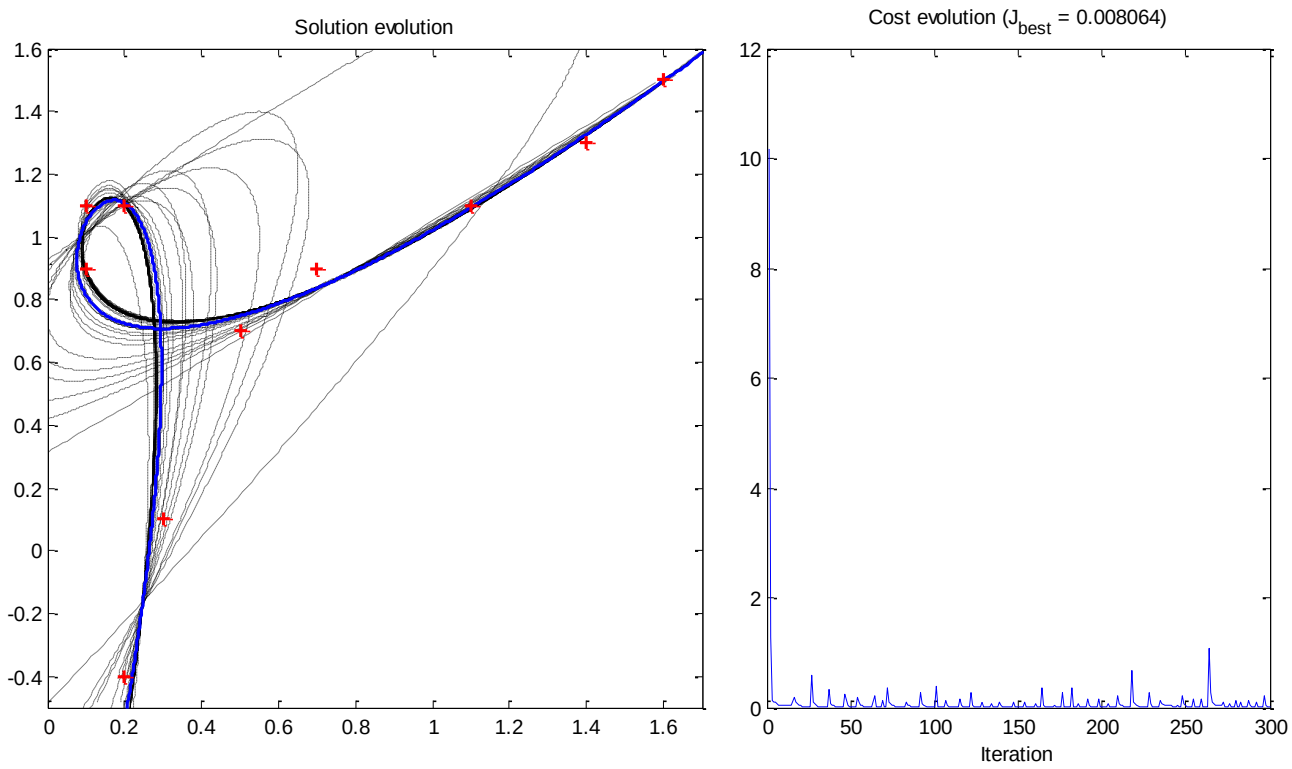

Figure 6 : Résultats obtenus par injection d'aléatoire après chaque minimum local et conservation de la meilleure solution

5. Optimisation E-M et algorithme génétique

Pour terminer ce TP, on remplace l'optimisation Gauss-Newton permettant l'estimation des t_i par une méthode d'algorithme génétique. En effet, comme on a pu le voir, le coût que l'on cherche à optimiser pour déterminer t_i n'est pas convexe, il existe plusieurs minimas. Une méthode de descente de gradient ne nous assure donc pas de trouver le minimum global, mais seulement un minimum local.

Pour pallier à ce problème, on se propose donc d'utiliser une méthode d'optimisation globale, et en particulier un algorithme génétique.

5.1. Opérateurs de l'algorithme génétique

Un algorithme génétique est constitué d'un certain nombre d'opérateurs qui peuvent être choisis plus ou moins complexe et donneront de plus ou moins bons résultats. Voyons les choix qui ont été faits.

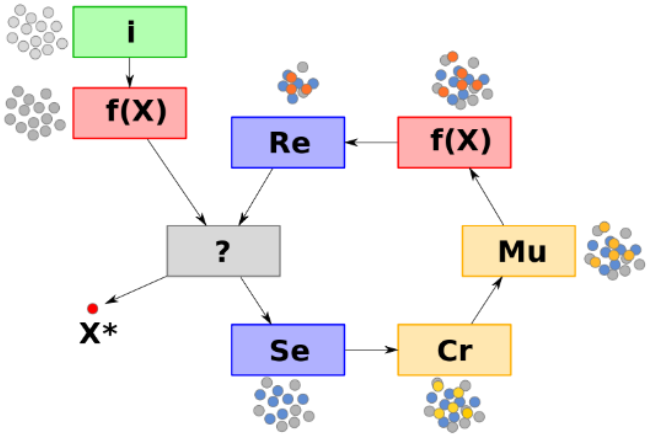

Figure 7 : Schéma d'un algorithme génétique (Wikipédia)

Etape d'initialisation *I* : On initialise $n = 100$ candidats linéairement entre 0 et 1.

Etape d'évaluation *f(X)* : Chaque candidat est évalué par la valeur de critère J présenté précédemment qu'il produit.

Etape de sélection *Se* : On conserve tous les points existants pour le croisement, la sélection est en quelque sorte effectuée par l'opérateur de réduction.

Etape de croisement *Cr* : On croise les candidats 2 par 2 avec la méthode du BlendingXover (BLX) qui consiste à produire un enfant à partir de deux parents tel que :

$$x_{enfant} = (1-\gamma)x_i + \gamma x_j \quad \text{avec} \quad \gamma = (1+2\alpha)U_{[0,1]} - \alpha \quad \text{et} \quad \alpha = 0{,}2$$

Etape de mutation *Mu* : Aucun opérateur de mutation n'a été implémenté, partant du principe que la méthode de croisement incluait déjà un effet de mutation car elle permet de s'éloigner aléatoirement des parents plutôt que de rester dans l'intervalle défini par les deux parents.

Etape de réduction *Re* : L'opérateur de réduction est un tournoi qui conserve les n meilleurs candidats vis-à-vis de leur évaluation.

Etape de terminaison *?* : L'algorithme s'arrête si le critère du meilleur candidat ne varie pas de plus de 10^{-6} pendant 4 itérations. Dans ce cas, on retourne le meilleur candidat. Un algorithme génétique ayant une convergence très peu maitrisée et grandement due à l'aléatoire des croisements et des mutations, il m'a semblé judicieux d'ajouter cette nécessité que la variation du coût persiste sur plusieurs itérations, permettant d'éviter qu'une mauvaise itération de croisement ne produisant pas de nouveau meilleur candidat ne stoppe l'algorithme avant convergence.

L'opérateur de réduction est sans doute celui qui mériterait le plus d'être amélioré dans cet algorithme. En effet, il est complètement « déterministe » et ne permet pas de s'assurer que l'on préserve la diversité des candidats. Un opérateur tel qu'une roue de loterie avec sharing serait très probablement plus performant et permettrait de conserver de la diversité. Cependant, compte-tenu de la taille de l'espace des paramètres à parcourir ([0,1]), du nombre de points (100), et du faible nombre de portions convexes du coût (au maximum quelques-unes), cet opérateur est suffisant pour obtenir de bons résultats.

5.2. Résultats obtenus

Voici un résultat obtenu par l'algorithme d'optimisation bi-niveau vu précédemment, mais utilisant cette fois une optimisation par algorithme génétique plutôt que par Gauss-Newton afin de déterminer les paramètres.

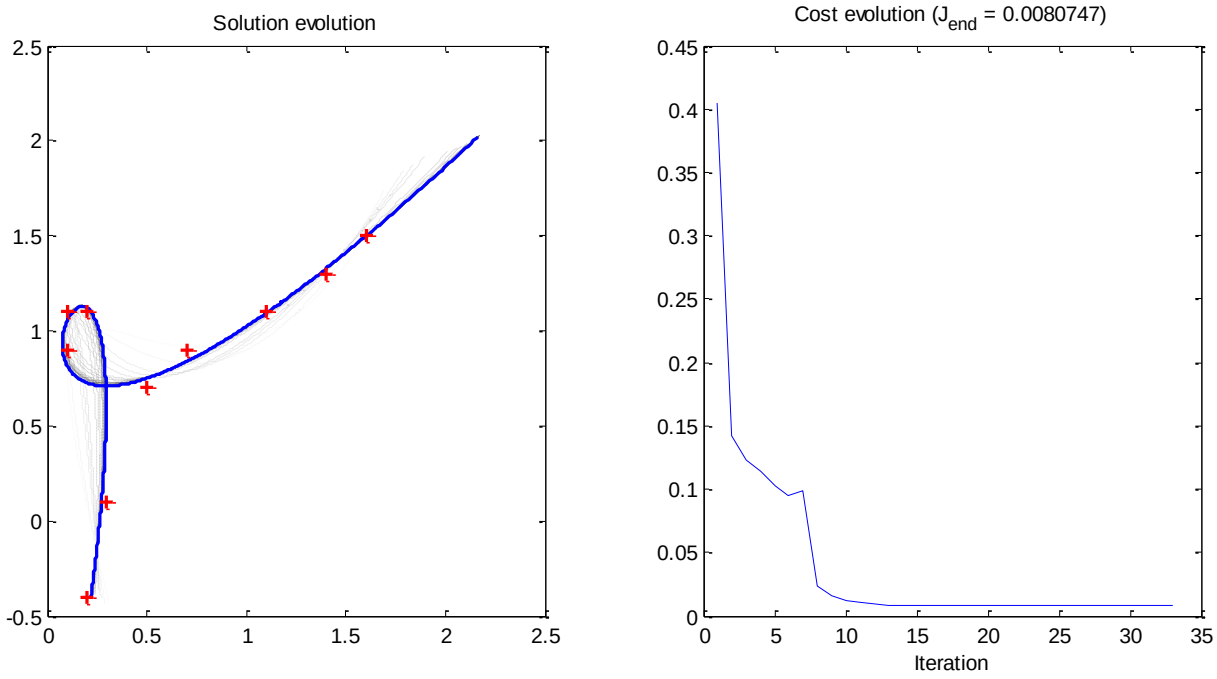

Figure 8 : Résultats obtenus avec utilisation d'un algorithme génétique

On constate que l'algorithme met une dizaine d'itérations à converger vers l'optimum global, alors que la solution utilisant une descente de Gauss-Newton mettait au moins une vingtaine d'itérations à converger vers un optimum local, et bien plus si on voulait essayer d'explorer plusieurs optimums en introduisant de l'aléatoire.

Cette solution semble donc très intéressante dans ce cas, puisqu'elle permet d'obtenir des résultats bien plus stables.

Conclusion

Ce TP aura été l'occasion d'essayer diverses méthodes d'optimisation : les moindres carrés, les descentes de gradient et de Gauss-Newton, et les algorithmes génétiques.

Ces méthodes ont été combinées et alternées au travers d'un algorithme itératif de type Expectation-Maximisation permettant une optimisation bi-niveau du problème global.

Les méthodes itératives ont par ailleurs été appliquées à un problème non-convexe et pouvant disposer de plusieurs optimums.

L'introduction d'aléatoire et l'utilisation d'une méthode d'optimisation globale (algorithme génétique) ont ainsi permit de trouver l'optimum global du problème.