M2 STIM
Projet d'Optimisation 2015-2016
Moindres Carrés, Descente de gradient
Gauss-Newton, Métaheuristiques

Préambule

On considère dans ce projet le problème de l'approximation d'une séquence de points par une courbe paramétrique. On considère pour cela une suite de points $p_i = (x_i, y_i)^T$ avec $1 \leq i \leq n$. Ces points peuvent par exemple avoir été acquis par l'intermédiaire d'une table à digitaliser.

i	1	2	3	4	5	6	7	8	9	10
x_i	0.2	0.3	0.2	0.1	0.1	0.5	0.7	1.1	1.4	1.6
y_i	−0.4	0.1	1.1	1.1	0.9	0.7	0.9	1.1	1.3	1.5

TABLE 1 – Observations

On a alors une trajectoire dépendant du temps dans $\mathbf{R}^2$. La figure 1 illustre graphiquement cette trajectoire.

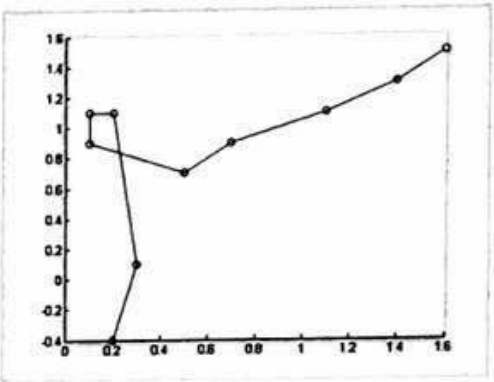

FIGURE 1 – Trajectoire à modéliser par une courbe paramétrique $m(t) = [x(t), y(t)]^T$

L'objectif de ce projet est d'approximer au mieux la séquence de points p_i du tableau 1 par une courbe paramétrique $m(t) = [x(t), y(t)]^T$ où $x(t)$ et $y(t)$ sont deux polynômes dépendant de la variable t appartenant à $\mathbf{R}^2$. On s'appuie pour ce faire sur les deux définitions suivantes :

— Soit $t_0 \in [0, 1]$, on définit $d_i(t_0)$ comme la distance entre le point $m(t_0)$ et le point p_i.

$$d_i(t_0) = \sqrt{(x(t_0) - x_i)^2 + (y(t_0) - y_i)^2} \tag{1}$$

— Soit $d_i(t)$ la distance entre un point $m(t)$ et un point p_i, on définit D_i comme la distance d'un point p_i à une courbe définie par $m(t)$

$$D_i = D(p_i, m) = \min_t d_i(t) \tag{2}$$

La distance entre un point p_i et une courbe est ainsi la distance minimale entre le point p_i et tous les points de la courbe $m(t)$.

PARTIE 1 : Descente de gradient et Gauss-Newton

Pour un modèle $m(t)$ fixé, et pour un point p_i fixé, on veut dans un premier temps estimer la valeur de t_i qui minimise le critère $J_i(t) = d_i^2(t)$. On cherche ainsi quel est le point de la courbe paramétrique le plus proche d'un point donné de l'espace. On va pour ce faire utiliser et comparer les méthodes de la descente de gradient et de Gauss-Newton, toutes deux initialisées avec le même $t_{i,0}$

1. Dans le cas de la méthode du gradient, donner l'expression générale de $t_{i,n+1}$ en fonction de $t_{i,n}$, p_i, $m(t)$, $m'(t)$ et λ
2. Transformer l'écriture précédente dans le cas où $m(t)$ s'écrit sous forme polynomiale telle que :

$$m(t) = at^3 + bt^2 + ct + d \qquad (3)$$

Avec

$$a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \quad d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \qquad (4)$$

3. Écrire une fonction Matlab/Octave `UnPasGradient` permettant d'effectuer une itération de la méthode du Gradient sur ce problème. La fonction aura le prototype suivant :

```
tinew=UnPasGradient(param,tiold,px,py,lambda)
```

Vous utiliserez ici une règle simple d'adaptation de λ telle que $\lambda_{i+1} = 0.5 \times \lambda_i$ ou $\lambda_{i+1} = 1.5 \times \lambda_i$.

4. Dans le cas de la méthode de Gauss-Newton, donner l'expression générale de $t_{i,n+1}$ en fonction de $t_{i,n}$, p_i, $m(t)$, $m'(t)$, $m''(t)$
5. Transformer l'expression précédente dans le cas où $m(t)$ s'écrit :

$$m(t) = at^3 + bt^2 + ct + d \qquad (5)$$

Avec

$$a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \quad d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \qquad (6)$$

6. Écrire une fonction `UnPasGaussNewton` permettant d'effectuer une itération de la méthode de Gauss-Newton sur ce problème. La fonction aura le prototype suivant :

```
tinew=UnPasGaussNewton(param,tiold,px,py)
```

7. Réaliser 4 itérations des deux méthodes pour les données suivantes, en initialisant t_i à 1 et λ à 0.02 dans le cas de la méthode du gradient. Vous vérifierez que les critères après 4 itérations valent respectivement 0.030421 et 0.017323.

$$a = \begin{pmatrix} 4.2 \\ 12.3 \end{pmatrix} \quad b = \begin{pmatrix} -3.2 \\ -20.2 \end{pmatrix} \quad c = \begin{pmatrix} 0.4 \\ 9.9 \end{pmatrix} \quad d = \begin{pmatrix} 0.2 \\ -0.4 \end{pmatrix} \quad p_i = \begin{pmatrix} 0.1 \\ 0.9 \end{pmatrix} \qquad (7)$$

8. Observez le comportement des deux méthodes pour différentes initialisations de t_i, de λ, et du point à approcher. Vous tracerez pour ce faire sur une seule figure (cf. fig. 2) l'évolution de la position du point sur la courbe $m(t)$ et l'évolution de la valeur du critère, pour les deux méthodes. Commenter à partir de vos connaissances de cours.

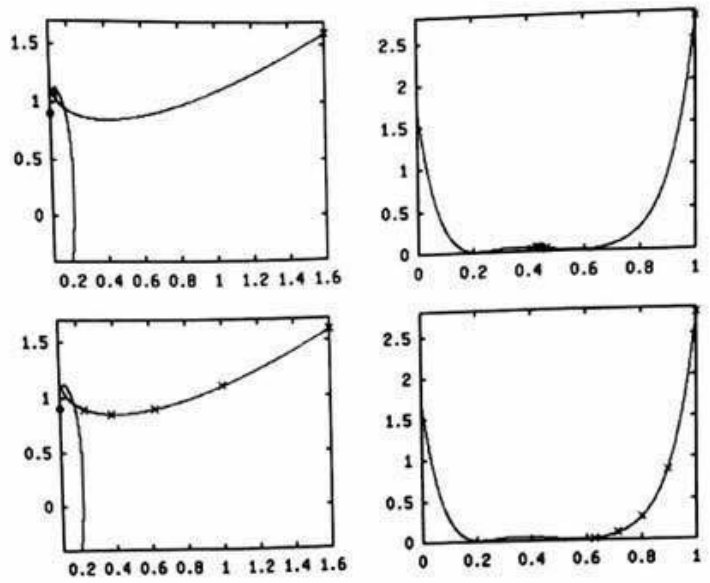

FIGURE 2 – Comparaison des deux méthodes

PARTIE 2 : Moindres carrés

On se place maintenant dans un contexte dans lequel les valeurs de t_i sont fixées, tout comme les points d'approximation. On cherche alors à déterminer les valeurs optimales des paramètres de la fonction $m(t)$ pour que la courbe la représentant passe au plus près des points d'approximation fixés, au sens de la distance Euclidienne.

On reprend pour cela l'expression précédente de $m(t)$ dont on cherche les valeurs optimales de a, b, c et d :

$$m(t) = at^3 + dt^2 + ct + d \tag{8}$$

Avec

$$a = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \quad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \quad d = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \tag{9}$$

1. Formuler l'expression du critère $J(a, b, c, d)$ à minimiser
2. On peut ici décomposer ce critère en deux critères $J_x(a1, b1, c1, d1)$ et $J_y(a2, b2, c2, d2)$. Pourquoi ?
3. Écrire une fonction `Estime_a_b_c_d_MC` permettant de calculer les valeurs de a,b,c et d. La fonction aura le prototype suivant :

```
[param,crit] = Estime_a_b_c_d_MC(ti,Pxy);
```

4. Tester votre méthode avec les points donnés en introduction de ce TP et sur un vecteur de t_i uniformément répartis entre 0 et 1 et vérifiez bien que vous obtenez les résultats de la figure 4.

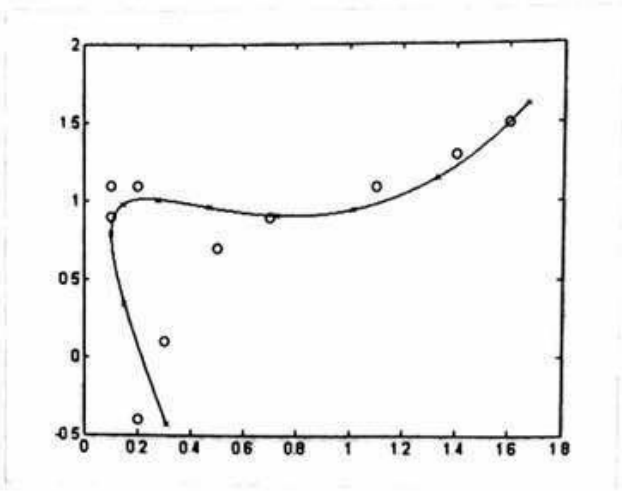

FIGURE 3 – Resultats de l'optimisation par moindres carrés $J = 0.4043$

PARTIE 3 : Moindres carrés et Gauss-Newton

Dans cette partie, on vous demande de combiner les outils développés dans la partie 1 et la partie 2 pour approximer au mieux l'ensemble de points par une courbe paramétrique polynomiale de degré 3. On va dans ce cadre alterner les étapes d'optimisation des t_i et des paramètres de $m(t)$.

1. Programmer l'algorithme suivant.

```
Début
   Initialiser Pxy par les valeurs données dans l'énoncé
   Initialiser les t_i.
   Faire
      Optimiser les paramètres a,b,c,d à t_i fixés
      Optimiser les ti pour les paramètres a,b,c,d obtenus
      Corriger les ti pour qu'ils respectent les contraintes
   Tant que le critère d'arrêt n'est pas atteint (à vous de choisir)
   Tracer la solution finale
   Tracer l'évolution de la valeur du critère
Fin
```

2. Observer le comportement de cet algorithme ainsi que l'évolution de la valeur du critère. Vous testerez d'abord sur les données des exercices précédents, puis sur des données générées aléatoirement (uniformmément dans [0,1] pour les t_i comme pour les paramètres). Vous prendrez garde à bien respecter les contraintes sur les t_i en les restreignant à l'intervalle [0,1] et en les triant.
3. Le principal défaut de l'algorithme précédent réside dans le fait qu'il ne prend pas en compte d'éventuels optima locaux. Pour remédier à ce problème, on propose, lorsque le critère n'évolue plus significativement lors d'une itération des deux optimisations, d'injecter une part d'aléatoire dans les t_i. Plus précisément, on vous demande de modifier votre algorithme pour que lorsque le critère ne varie pas d'une valeur supérieure à 10^{-3}, les t_i subissent une modification telle que $new = (0.8 + 0.4 * U_{[0,1]}) * old$.
4. Quelle est la modification apportée par une telle équation ?
5. Tester ce nouvel algorithme, et comparer ses performances à celles du premier. De nouveau, vous observerez l'évolution du critère au cours des itérations. Vous regarderez plus précisément si la valeur finale obtenue pour ce critère est toujours la meilleure et en déduirez une dernière modification de votre programme. Vous devriez obtenir un résultat final tel que le suivant

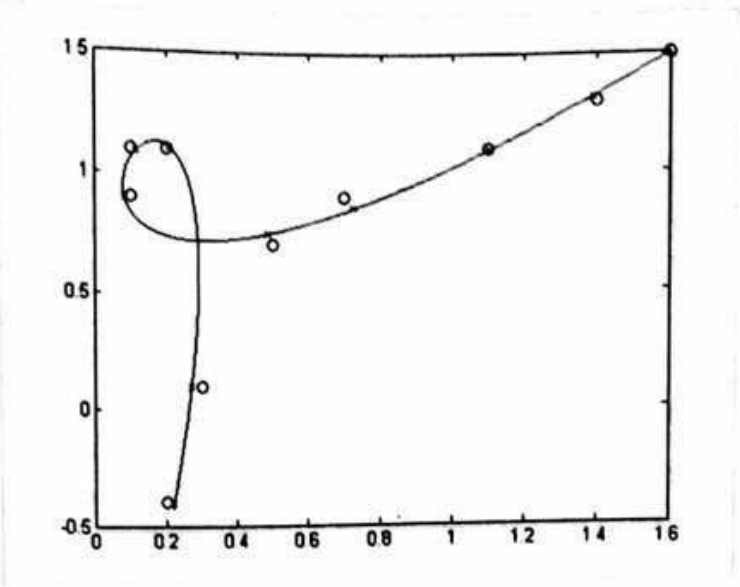

FIGURE 4 – Resultats attendus de l'optimisation

PARTIE 4 : Métaheuristiques

Dans cette dernière partie, on vous demande de faire parler votre imagination, en programmant une méthode méta-heuristique (Recuit simulé, AG, OEP, SE, ANT), couplée ou non avec les moindres carrés pour résoudre le problème posé.