

DM: Le fil qui chante

Ce sujet a pour objectif de choisir où placer des poteaux télégraphiques pour relier le point le plus à gauche d'un paysage bidimensionnel au point le plus à droite en fonction de critères de coût. On fera les simplifications suivantes : entre deux points d'abscisses entières, le paysage est affine. Les fils sont sans poids et tendus ; ils relient donc en ligne droite les sommets de deux poteaux consécutifs. Les normes de sécurité imposent que les fils soient en tout point à une distance d'au moins Δ (mesurée verticalement) au-dessus du sol. Les poteaux sont tous de longueur identique $\ell \geq \Delta$. Voici par exemple une proposition valide de placement de poteaux pour le paysage ci-dessous avec $\Delta = 0.5$ et $\ell = 2.0$.

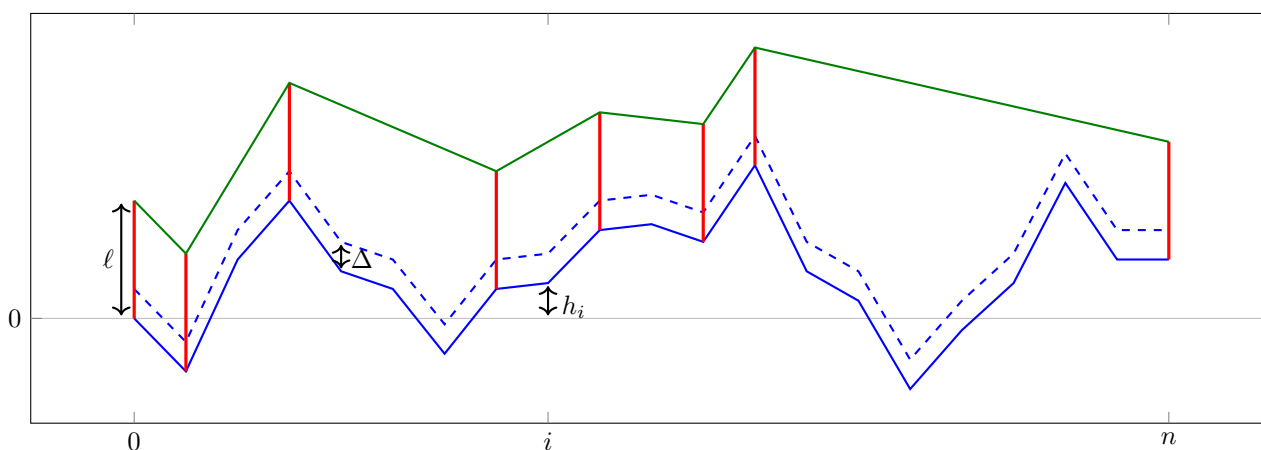
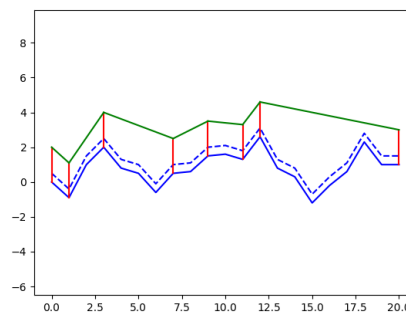


FIGURE 1: Un exemple de poteaux où le fil est en tout point au moins Δ au-dessus du sol. La ligne brisée bleue représente le sol, celle au dessus en pointillés représente la limite inférieure pour le fil (distance Δ du sol), les poteaux sont en rouges et le fil est en vert. Il y a beaucoup trop de poteaux dans cet exemple!

Test en Python. Pour ceux qui voudraient travailler sur ordinateur (recommandé), vous pouvez télécharger sur mon site web un script contenant la liste `deniveles` ($n = 20$) utilisée dans les exemples de ce sujet, ainsi qu'une fonction permettant de tracer le paysage et les poteaux de manière très similaire aux images du sujet (voir ci-contre). Cette fonction pourra être utilisée dès que la liste `hauteurs` de la question 1 aura été calculée.



1 Planter le paysage

On définit le paysage à partir d'une suite de relevés de dénivelés stockés dans une liste `deniveles` de nombres de taille $n + 1$. L'entier n et la liste `deniveles` sont supposés être des variables globales. Le paysage est alors décrit par une ligne brisée de $n + 1$ points, dont le i -ème point P_i est de coordonnées (i, h_i) où :

$$h_0 = 0 \quad \text{et} \quad h_i = h_{i-1} + \text{deniveles}[i] \quad \text{pour } i \in \{1, \dots, n\}.$$

Question 1. Écrire une fonction `calculHauteurs()` qui renvoie les hauteurs des points sous la forme d'une liste de taille $n + 1$. *Remarque :* le premier élément de `deniveles` n'est pas utilisé. Avec l'annexe, vous devez trouver une liste de 21 points, dont les premiers termes sont $0, -1, 1, 2$, aux approximations flottantes près.

Dans la suite, on suppose que l'appel `hauteurs = calculHauteurs()` a été réalisé, de sorte que `hauteurs` est une variable globale, avec `hauteurs[i]` la hauteur du point P_i pour tout $0 \leq i \leq n$.

Question 2. Écrire une fonction `calculFenetre()` renvoyant un quadruplet de valeurs (`hMin`, `hMax`, `iMin`, `iMax`). `hMin` et `hMax` sont respectivement les hauteurs minimale et maximale du paysage, tandis que `iMin` et `iMax` sont les indices des points les plus à gauche de hauteur minimale et maximale respectivement.

On appelle *distance au sol* de i à j , la longueur de la ligne brisée allant du point P_i au point P_j .

Question 3. Écrire une fonction `distanceAuSol(i, j)` qui calcule et renvoie la distance au sol de P_i à P_j .

On appelle un *pic* un point P_i d'indice $1 \leq i < n$ tel que $h_i > \max(h_{i-1}, h_{i+1})$. On appelle *point remarquable*, les pics et les points des bords gauche (point P_0) et droit (point P_n). On appelle *bassin* toute partie du paysage allant d'un point remarquable au suivant.

Question 4. Écrire une fonction `longueurDuPlusLongBassin()` qui calcule et renvoie la distance au sol maximale d'un bassin dans le paysage.

2 Planter les poteaux

On souhaite relier le point le plus à gauche au point le plus à droite par un fil télégraphique. Pour cela, on doit choisir en quels points parmi les $(P_i)_{0 \leq i \leq n}$ planter les poteaux télégraphiques intermédiaires. Rappelons que le fil est supposé sans poids et tendu et qu'il relie donc les sommets de chacun des poteaux en ligne droite. Les poteaux sont plantés verticalement et ont tous une longueur identique $\ell \geq \Delta$.

ℓ et Δ seront respectivement représentées par les variables globales `l` et `delta`. La législation impose que le fil doit rester à une distance supérieure ou égale à Δ (mesurée verticalement) au-dessus du sol. Pour tester si un fil tiré entre un poteau placé au point P_i et un poteau placé au point P_j (avec $j > i$ ou $j < i$) respecte la législation, notons

$$\alpha_{i,k} = \frac{(h_k + \Delta) - (h_i + \ell)}{k - i} \text{ pour } k \neq i \quad \text{et} \quad \beta_{i,j} = \frac{(h_j + \ell) - (h_i + \ell)}{j - i}$$

les pentes des fils tirés depuis le poteau en P_i jusqu'à une hauteur Δ au-dessus du point P_k d'une part et jusqu'à une hauteur ℓ au-dessus du point P_j d'autre part (voir la Figure 2).

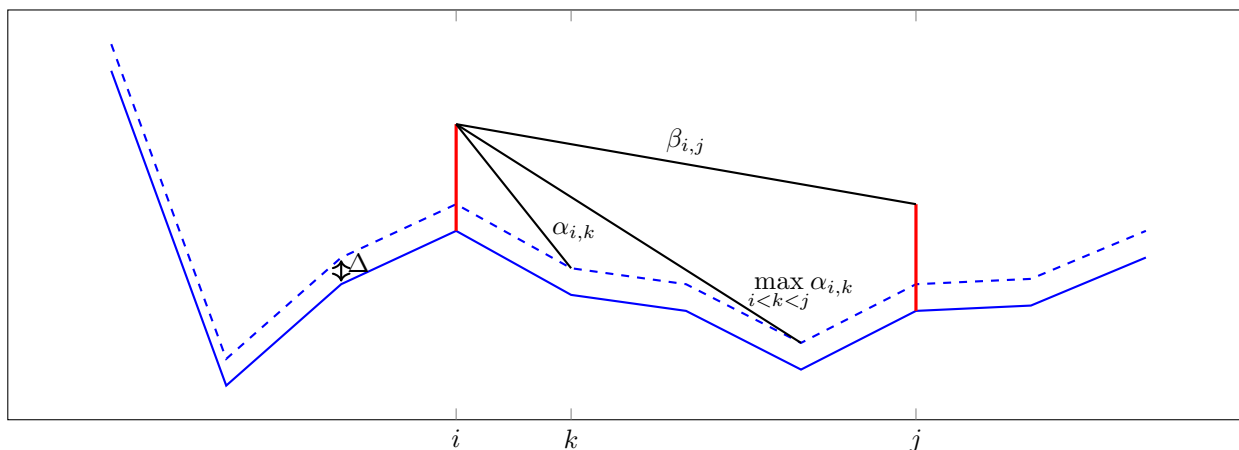


FIGURE 2: Condition sur les pentes pour pouvoir tirer un fil.

On admet que le fil tiré d'un poteau en P_i à un poteau en P_j respecte la législation si et seulement si :

$$\beta_{i,j} \geq \max_{i < k < j} \alpha_{i,k} \quad \text{lorsque } j > i \quad \text{et} \quad \beta_{i,j} \leq \min_{j < k < i} \alpha_{i,k} \quad \text{lorsque } j < i.$$

Question 5. En utilisant cette méthode, écrire une fonction `estDeltaAuDessusDuSol(i, j)` qui renvoie `True` si un fil tiré entre les sommets d'un poteau placé au point P_i et d'un poteau placé au point P_j respecte la législation, et renvoie `False` dans le cas contraire.

Considérons une première stratégie, dite *algorithme glouton en avant*. Le premier poteau est planté en P_0 . Pour calculer l'emplacement du prochain poteau, on part du dernier poteau planté et on avance (à droite) avec le fil tendu tant que la législation est respectée (et que P_n n'est pas atteint). Un nouveau poteau est alors planté, et on recommence jusqu'à ce que P_n soit atteint. La figure 3 illustre la solution produite par cet algorithme.

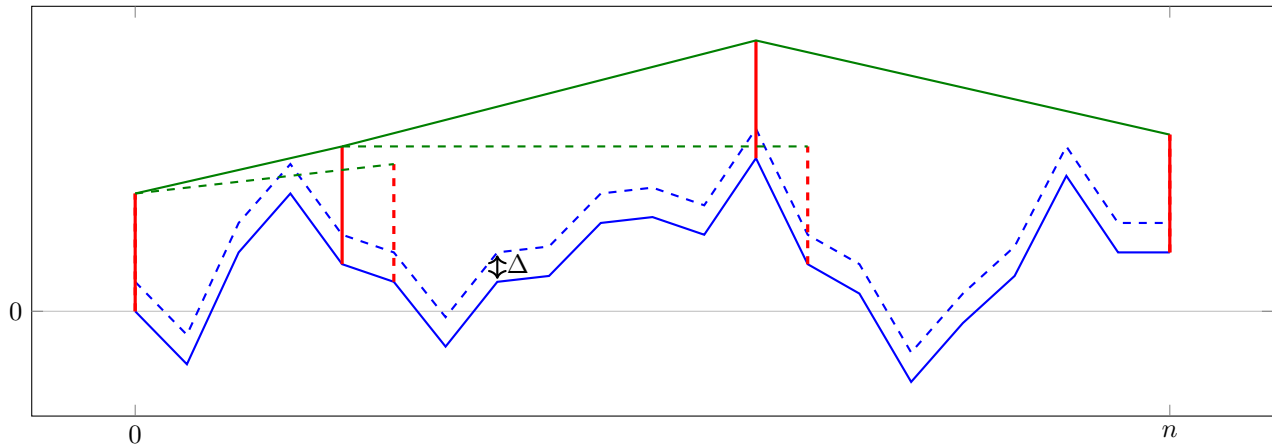


FIGURE 3: Solution produite par l'algorithme glouton en avant. Les poteaux et les fils en pointillés sont ceux violant la législation.

Question 6. Écrire une fonction `placementGloutonEnAvant()` qui calcule la disposition des poteaux selon l'algorithme ci-dessus. La solution retournée sera donnée sous la forme d'une liste `poteaux` dans laquelle la case `poteaux[t]`, pour $t \geq 0$, contiendra l'indice i indiquant que le t -ème poteau (les poteaux sont numérotés à partir de 0), s'il existe, est planté en P_i .

Question 7. Donner une majoration de la complexité du temps de calcul de votre algorithme en fonction de n . Justifier qu'on peut se contenter du calcul de $O(n)$ pentes pour implémenter l'algorithme glouton en avant, et modifier votre fonction (si nécessaire) pour obtenir une complexité en $O(n)$.

L'algorithme glouton en avant a tendance à placer beaucoup trop de poteaux, en particulier dans les vallées alors qu'il suffirait de relier les deux extrémités par un unique fil. Nous considérons donc une alternative, dite glouton au plus loin, qui consiste à planter le prochain poteau le plus à droite possible de la position courante. Le premier poteau est toujours planté en P_0 . La figure 4 illustre la solution produite par cet algorithme sur le même paysage que celui de la figure 3.

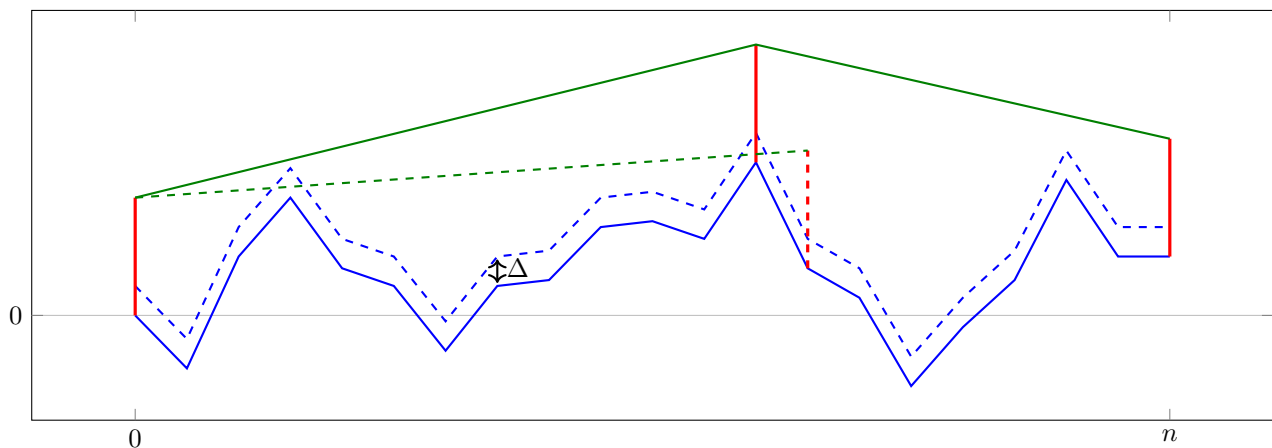


FIGURE 4: Solution produite par l'algorithme glouton au plus loin. Le poteau et le fil en pointillés sont ceux violant la législation.

Question 8. Écrire une fonction `placementGloutonAuPlusLoin()` qui calcule la disposition des poteaux selon l'algorithme ci-dessus. La solution sera donnée sous la forme d'une liste `poteaux` dans laquelle la case `poteaux[t]`, pour $t \geq 0$, contiendra l'indice i indiquant que le t -ème poteau (les poteaux sont numérotés à partir de 0), s'il existe, est planté en P_i .

3 Minimiser la longueur du fil

L'objectif est maintenant de calculer un placement optimal des poteaux en terme de longueur totale du fil (le nombre de poteaux pouvant être maintenant arbitraire). Une liste `optL` peut être calculée de proche en proche tel que `optL[i]` désigne la longueur minimale de fil à utiliser si l'on souhaitait relier le point P_0 au point P_i .

Question 9. Soit $L_{i,j}$ la longueur du segment d'extrémités P_i et P_j . Montrer que la liste `optL` vérifie : `optL[0] = 0`, et pour $1 \leq i \leq n$,

$$\text{optL}[i] = \min\{L_{i,j} + \text{optL}[j], \text{ où } 0 \leq j < i \text{ et il est possible de tirer un fil de } P_j \text{ à } P_i\}$$

La case `optL[n]` contient alors la longueur minimale de fil pour relier le bord gauche au bord droit.

Question 10. Écrire une fonction `longueurMinimale()` qui renvoie la longueur minimale de fil pour relier le bord gauche au bord droit, en respectant la législation.

Question 11. Modifier votre fonction `longueurMinimale` pour qu'elle retourne une liste `precOptL` de taille $n + 1$, où `precOptL[i]` contient l'indice du poteau précédant le poteau placé en P_i dans une solution de longueur minimale reliant P_0 à P_i . (Par convention, `precOptL[0] = -1`.)

Question 12. Écrire une fonction qui retourne un placement des poteaux minimisant la longueur du fil de P_0 à P_n à partir de la liste `precOptL`. La solution sera retournée sous la forme d'une liste `poteaux` dans laquelle la case `poteaux[t]`, pour $t \geq 0$, contiendra l'indice i indiquant que le t -ème poteau (les poteaux sont numérotés à partir de 0), s'il existe, est planté en P_i . Quelle est la complexité de votre fonction en fonction de n ?

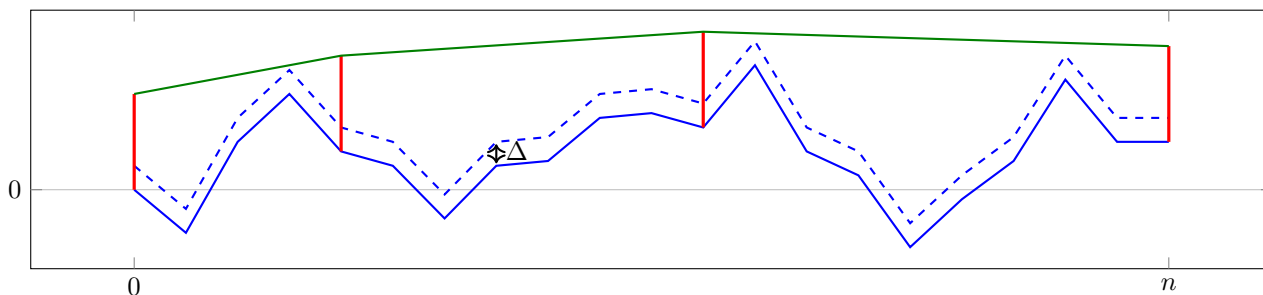


FIGURE 5: Un planté de poteaux qui minimise la longueur du fil.