Travaux pratiques d'optimisation de fonctions continues non linéaires

L'objectif de ces travaux pratiques est d'expérimenter les algorithmes vus en cours sur quelques problèmes non linéaires à l'aide de la toolbox d'optimisation de C.T. Kelley complément de son ouvrage *Iterative Methods for Optimization*.

Partie A
Prise en main

Avant de pouvoir commencer le TP, il est nécessaire de configurer Matlab :

1. copier les fichiers du TP dans votre répertoire personnel,
2. ouvrir Matlab,
3. modifier le répertoire courant de Matlab (`current directory`) par celui où vous avez copié les fichiers,
4. ouvrir le fichier `testRosenbrock.m`,
5. exécuter ce script avec la commande `Debug->Run` (ou `F5`), une fenêtre doit s'ouvrir montrant une courbe.

Pour minimiser des fonctions multivariables, la toolbox de Kelley propose plusieurs commandes, notamment :

– la commande `steep` permet d'appliquer l'algorithme de la plus forte pente,
– la commande `bfgs` permet d'appliquer l'algorithme BFGS (quasi-newton),
– la commande `nelder` permet d'appliquer l'algorithme du simplexe de Nelder et Mead.

Vous pouvez obtenir de l'aide sur ces fonctions en tapant `help steep`, `help bfgs` ou `help nelder` dans la fenêtre de commande (`command window`).

Le script `testRosenbrock.m` fournit un exemple d'utilisation de ces trois commandes pour minimiser la fonction `rosenbrock`.

A.1. Étudier ce script et faire varier les différents paramètres pour en comprendre leurs rôles respectifs.

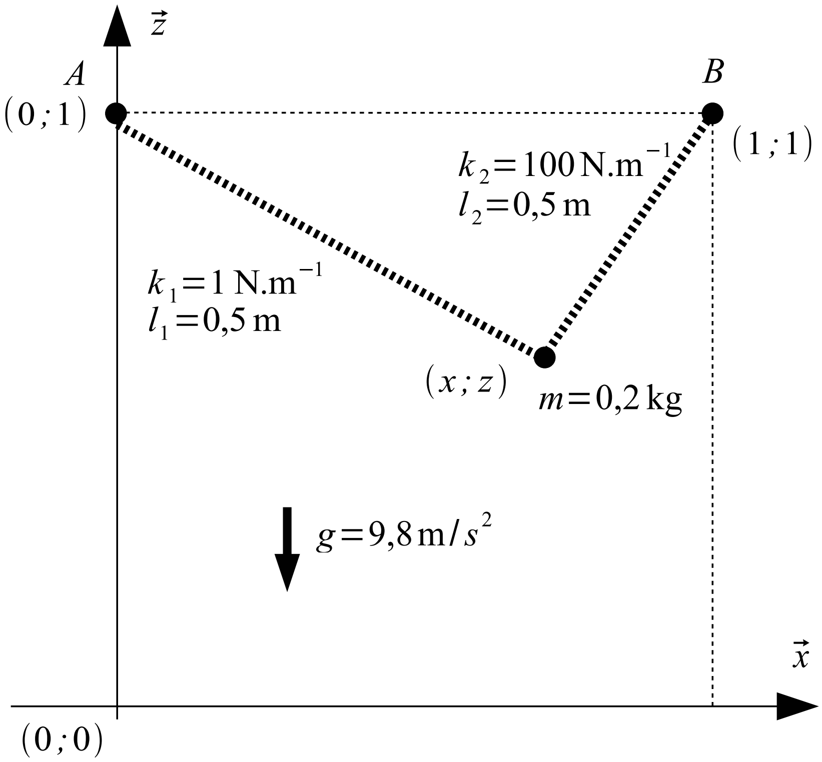
A.2. Compléter le tableau ci-dessous avec le nombre d'itérations minimal pour que la norme du gradient soit inférieure à 0.01 à partir du point initial $X_0 = \begin{bmatrix}-2 & 2\end{bmatrix}$.

Méthode	Nombre d'itérations	Nombre d'évaluations de la fonction	Nombre d'évaluations du gradient
Plus forte pente			
Nelder et Mead			
BFGS			
BFGS avec estimation du gradient			

Partie B

Système mécanique à deux ressorts

Soit le système mécanique suivant constitué d'une masse m suspendue par deux ressorts de longueur à vide l_1 et l_2 et de raideur k_1 et k_2.



Au repos, ce système minimise son énergie potentielle mécanique définie comme la somme de son énergie potentielle élastique et de son énergie potentielle de pesanteur.

B.1. En vous inspirant du canevas ci-dessous, créer une nouvelle fonction `ressorts` *calculant l'énergie potentielle de ce système en fonction du vecteur* $\begin{bmatrix} x & z \end{bmatrix}$. *On rappelle que l'énergie potentielle d'un ressort est égale à* $\frac{1}{2}k\delta^2$ *pour un allongement* δ. *On négligera le poids des ressorts.*

```
function E=ressorts(V)

x=V(1);
z=V(2);

E=    ...    ;

end
```

B.2. Écrire un nouveau script en vous inspirant de `testRosenbrock.m` *pour trouver la position de la masse au repos. Une fonction* `plotRessorts` *est fournie et permet de représenter graphiquement une solution.*

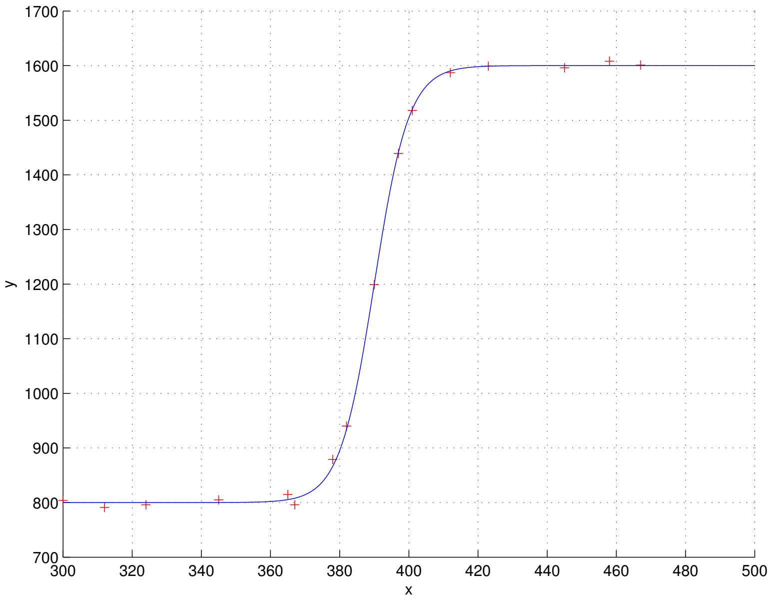
Partie C
Ajustement d'un modèle non linéaire

On souhaite ajuster les paramètres d'un modèle non linéaire à des données expérimentales entachées d'erreurs de mesure. Le modèle est de la forme :

$$g(x) = a + b * \tanh\left(\frac{x-c}{d}\right)$$

où a, b, c et d sont des coefficients réels.

Les données expérimentales et le modèle sont représentés par la figure suivante.



L'objectif est de minimiser l'écart entre le modèle et les données expérimentales (x_i, y_i) au sens des moindres carrés, c'est-à-dire minimiser :

$$f(a,b,c,d) = \frac{1}{2}\sum_{i=1}^{n}(g(x_i) - y_i)^2$$

Une fonction `ajustement` est fournie et permet de calculer de calculer f en fonction du vecteur $\begin{bmatrix} a & b & c & d \end{bmatrix}$.

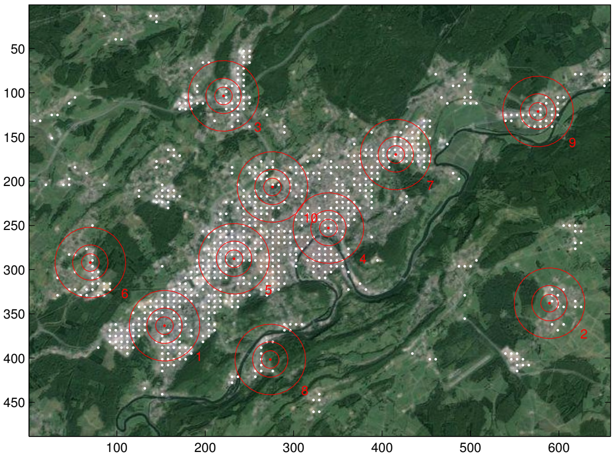
Une fonction `plotAjustement` est également fournie pour représenter graphiquement une solution.

C.1. Écrire un nouveau script pour trouver les paramètres a, b, c et d qui minimisent le critère des moindres carrés.

C.2. Observer la valeur de la fonction objectif au cours des itérations. Qu'observe-t'on ?

Partie D
Placement d'antennes relais

La société *Fruit Telecom* souhaite déployer son réseau 4G sur Besançon. Il est prévu pour cela d'installer dix antennes relais. L'objectif est de trouver l'emplacement optimal de ces dix antennes.



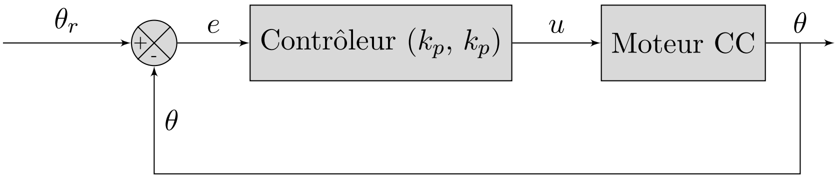
La fonction `couverture` donne la distance moyenne des utilisateurs à l'antenne la plus proche en fonction du vecteur $\begin{bmatrix} x_1 & y_1 & \cdots & x_{10} & y_{10} \end{bmatrix}$ donnant les coordonnées (en unités arbitraires) des antennes.

Une fonction `plotCouverture` est également fournie et permet de représenter graphiquement une solution.

D.1. Trouver un vecteur des coordonnées $\begin{bmatrix} x_1 & y_1 & \cdots & x_{10} & y_{10} \end{bmatrix}$ qui permet d'atteindre une distance moyenne inférieure à 43.

Partie E
Commande d'un servomoteur

Un moteur à courant continu est asservi en position à l'aide d'un correcteur proportionnel-dérivé.



On souhaite trouver les coefficients k_d et k_p du correcteur qui minimisent un critère défini à la fois sur l'erreur de position angulaire $e(t)$ et la tension $u(t)$ appliquée au moteur.

$$J(\alpha) = (1-\alpha)\int_0^{10} e(t)^2 dt + \frac{\alpha}{1000}\int_0^{10} u(t)^2 dt = (1-\alpha)J_1 + \alpha J_2$$

Le coefficient $\alpha \in [0;1]$ permet de pondérer l'influence de chacun des deux sous-critères J_1 et J_2. Minimiser J_1 revient à minimiser l'intégrale de l'erreur de position et donc d'obtenir un asservissement plus rapide. Minimiser J_2 revient à minimiser l'intégrale de la tension appliquée et donc d'obtenir un asservissement qui sollicite moins l'actionneur au détriment de la rapidité.

Une fonction `servomoteur` est fournie. Cette fonction permet que calculer J en fonction du vecteur $[k_d \;\; k_p]$ et de α. Pour préciser que l'on veut optimiser la fonction par rapport au premier de ses paramètres et avec $\alpha = 0.5$, il faut écrire : `@(X)servomoteur(X,0.5)`.

Une fonction `plotServomoteur` est également fournie et permet de représenter graphiquement la position angulaire en fonction du temps. Elle affiche également les valeurs de J_1 et de J_2.

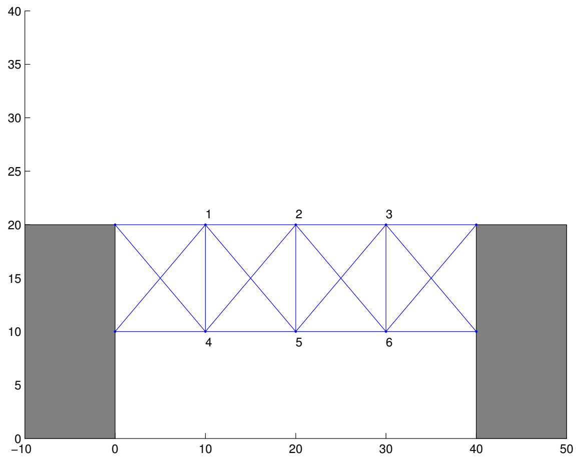
E.1. Trouver les coefficients k_d et k_p qui minimise $J(0.5)$.

En faisant varier α, on trouve différentes solutions Pareto-optimales (solutions non dominées) pour J_1 et J_2.

E.2. Tracer quelques points du front de Pareto. On pourra utiliser la commande `plot` de MATLAB.

Partie F
Déformation d'une structure mécanique

Un pont est construit à l'aide de poutrelles métalliques selon la structure suivante :



La fonction `structure` donne l'énergie potentielle de la structure (en utilisant un calcul similaire au problème des deux ressorts) en fonction du vecteur $V = \begin{bmatrix} x_1 & z_1 & \cdots & x_6 & z_6 \end{bmatrix}$ donnant les coordonnées des points 1 à 6.

La fonction `structure` admet un deuxième paramètre correspondant aux coordonnées initiales $V_0 = \begin{bmatrix} x_{0,1} & z_{0,1} & \cdots & x_{0,6} & z_{0,6} \end{bmatrix}$ des points avant déformation. Pour préciser que l'on veut optimiser la fonction par rapport au premier de ses paramètres, il faut écrire : `@(X)structure(X,V0)`

Une fonction `plotStructure` est également fournie et permet de représenter graphiquement une solution.

F.1. Trouver le vecteur des coordonnées x_1, z_1, $\ldots x_6$, z_6 des points qui minimise l'énergie potentielle du pont.

F.2. Trouver des valeurs pour les critères d'arrêt qui donnent un résultat satisfaisant en un minimum d'itérations.

On souhaite maintenant trouver la géométrie la plus rigide, c'est-à-dire celle qui minimise la déformation du pont sous son poids. L'idée est lancer plusieurs fois l'optimisation précédente à partir de points initiaux différents et de chercher par une seconde optimisation les positions initiales minimisant la déformation du pont.

F.3. Écrire une fonction `deformation` *qui calcule la valeur absolue de la variation d'altitude du point 2 en fonction de la position initiale* $(x_{0,4}, z_{0,4})$ *du point 4 et de la hauteur* $z_{0,5}$ *du point 5. Les positions initiales des points 1, 2 et 3 sont définies selon la figure précédente. Le point 6 est le symétrique du point 4.*

```
function d=deformation(P)

x4=P(1);
z4=P(2);
z5=P(3);

d=   ...   ;

end
```

F.4. Trouver les coordonnées $x_{0,4}$*,* $z_{0,4}$*,* $z_{0,5}$ *qui minimisent la déformation au centre du pont.*

Une fonction `plotDeformation` est fournie et permet de représenter graphiquement une solution en fonction du vecteur $\begin{bmatrix} x_{0,4} & z_{0,4} & z_{0,5} \end{bmatrix}$.

Partie G
Bonus

Le solveur d'Excel permet également de minimiser une fonction. On propose d'illustrer son fonctionnement avec la fonction de Rosenbrock.

1. Ouvrir Excel,
2. Entrer -2 dans la cellule `A1` et entrer 2 dans la cellule `A2`,
3. Entrer la fonction de Rosenbrock dans la cellule `C1` en utilisant `A1` et `A2` comme variables (dans Excel une formule commence par =),
4. Ouvrir le solveur qui se trouve dans le groupe `Analyse` de l'onglet `Données`,

Il suffit ensuite de renseigner l'objectif par la cellule `C1`, les cellules variables par la zone `A1:A2` (qui définit le point initial), de choisir `Min` et la résolution `GRG non linéaire` et enfin de cliquer sur `Résoudre`. Le résultat s'affiche dans la zone `A1:A2`.

On peut modifier certains paramètres de l'algorithme dans les options du solveur.