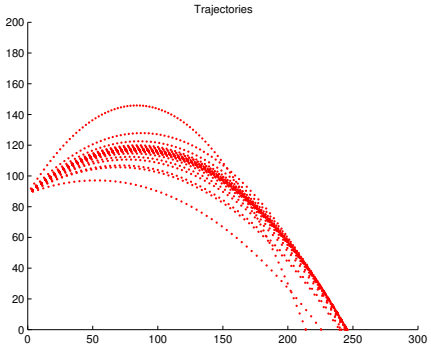
Matlab 14-15 for dummies : exercice 6 : Déduire portée maximale d'un obusier !

L'objectif de ce problème est de déterminer l'angle d'élévation θ du tube d'une pièce d'artillerie afin que la portée soit maximale. En d'autres mots, il faut que la distance horizontale parcourue par l'obus soit la plus longue possible.



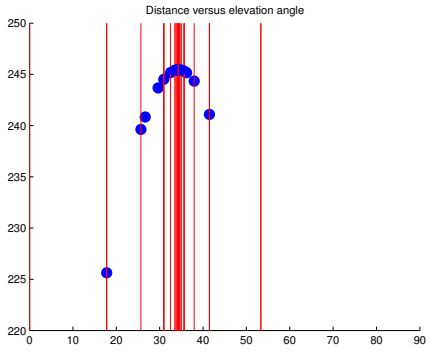
La trajectoire de l'obus sera obtenue en intégrant les équations du mouvement qui pourraient s'écrire sous la forme :

$$\begin{cases} x''(t) &= -\gamma\ x'(t) \\ y''(t) &= -\gamma\ y'(t) - g \end{cases}$$

où $g = 9.81$ est l'accélération de la gravité et γ est un coefficient de friction. Toutefois, nos artilleurs ne sont pas encore totalement sûrs de la validité de ce modèle mathématique et il est donc possible qu'il doive être modifié...

On vous demande d'établir une stratégie d'ajustement pour n'importe quel système d'équations différentielles ordinaires qui pourraient modéliser de manière réaliste la trajectoire de l'obus. En particulier, lla trainée de l'air sur l'obus pourrait plutôt être écrite sous la forme d'un terme quadratique. On pourrait aussi souhaiter tenir compte de l'influence du vent. C'est pourquoi, on vous demande d'imaginer un algorithme d'optimisation pour une fonction unimodale. Une fonction unimodale est une fonction qui présente un seul maximum (ou minimum) en étant d'abord uniquement croissante et puis décroissante.... La trajectoire presque parabolique est évidemment un exemple quasi parfait de trajectoire unimodale.

Technique d'encadrement pour l'optimisation unimodale



Si $f(x)$ atteint son maximum dans l'intervalle $[a, d]$. Si nous divisons cet intervalle en trois sous-intervalles $[a, b]$, $[b, c]$ et $[c, d]$ de taille égale avec $b = a + (d - a)/3$ et $c = a + 2(d - a)/3$, on peut alors calculer $f(b)$ et $f(c)$, et déduire que :

- Le maximum ne peut pas se trouver sur l'intervalle $[c, d]$, lorsque $f(c) < f(b)$.
- Le maximum ne peut pas se trouver sur l'intervalle $[a, b]$, lorsque $f(b) < f(c)$.

Cela permet de réduire l'intervalle d'encadrement $[a, d]$ en le remplaçant par $[a, c]$ ou $[b, d]$. A chaque itération, la longueur de l'intervalle d'encadrement se réduit d'un facteur $2/3$. Evidemment, il est essentiel d'être certain que la fonction f est bien unimodale pour que la stratégie d'encadrement converge.

On vous demande d'implémenter une stratégie d'encadrement pour un modèle mathématique de trajectoire qui n'est pas connu a priori mais qui est une fonction unimodale.

1. Il s'agit d'écrire deux fonctions :

   ```
   function [theta] = adjustFire(y0,v0,epsilon,h,f,bonus)
   function [distance] = HeunIntegrate(theta,y0,v0,h,f)
   ```

   La fonction `adjustFire` calcule l'angle de tir optimal pour une position verticale initiale $(0, y_0)$ et un vitesse initiale v_0. Les trois derniers arguments sont `epsilon` la précision requise pour l'angle, le pas requis pour l'intégration numérique et le pointeur vers la fonction contenant les équations décrivant la trajectoire de l'obus.

   La fonction `HeunIntegrate` intègre les équations différentielles ordinaires avec la méthode de Heun. Les conditions initiales sont spécifiées par les arguments `y0`, `v0` et `theta`. Le pas est donné par `h`. Cette fonction est évidemment nécessaire pour la précédente. L'intégration temporelle se fait jusqu'au moment où la hauteur de l'obus est nulle. La longueur du dernier pas sera adapté afin d'obtenir une valeur nulle pour la dernière hauteur. La fonction retourne la distance horizontale parcourue.

2. Il est possible d'améliorer la vitesse de convergence de l'algorithme en réduisant le nombre de tirs dans la stratégie d'encadrement, lorsque l'argument `bonus=1`, vous pouvez proposer un algorithme plus rapide... Oui, un meilleur algorithme existe ! Par contre, lorsque `bonus=0`, la stratégie devra être exactement celle décrite plus haut. Par exemple, on pourrait tirer profit de l'information présente dans les points qui ont été calculés précédemment...

3. Un petit programme `test_matlab6` vous est fourni pour tester votre fonction et obtenir les graphiques.

```
function test_matlab6
    global shot
    figure();
    subplot(2,1,1); axis([0 300 0 200]); title('Trajectories'); axis manual; hold on;
    subplot(2,1,2); axis([0 90 220 250]); title('Distance versus angle'); axis manual; hold on;
    shot = 0;
    adjustFire(88.48,42,0.01,0.05,@f);
    fprintf('=== Number of shots : %d \n', shot);
end

function dudt = f(u)
    dudt = u;
    dudt(1) = -0.01 * u(1);
    dudt(2) = u(1);
    dudt(3) = -9.81 - 0.01 * u(3);
    dudt(4) = u(3);
end
```

4. Votre programme doit être exact et surtout utiliser un minimum de tirs... N'hésitez pas à modifier les équations différentielles a résoudre et les paramètres afin de vérifier la robustesse et l'exactitude de votre programme. Pour vous aide, nous vous avons donné un canevas de départ contenant une ébauche des deux fonctions : il suffit de compléter les quelques bribes de programmes manquantes :-)

5. Votre fonction (avec les éventuelles sous-fonctions que vous auriez créées) sera incluse dans un unique fichier `adjustFire.m`, sans y adjoindre le programme de test fourni ! Cette fonction devra être soumise via le web avant le 5 décembre à 23h59 : ce travail est individuel et sera évalué. Pour permettre une correction plus aisée, ne pas inclure les commandes `clc` et `close all` dans votre fonction `adjustFire`.