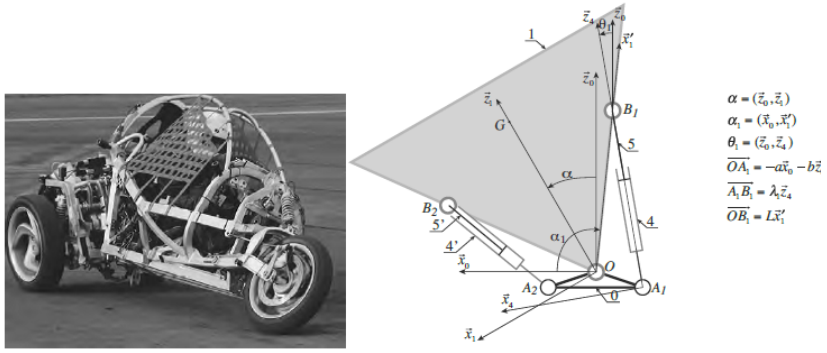
Équations numériques

1 Mise en situation

On s'intéresse au mouvement de la cabine du véhicule à trois roues Clever dont la cabine s'incline à l'image d'une moto pour prendre un virage.



Pour piloter le mécanisme, il est nécessaire de connaître l'angle de la cabine en fonction de l'élongation des vérins. L'étude géométrique permet d'obtenir facilement l'élongation en fonction de l'angle :

$$\lambda_1(\alpha) = \sqrt{(L\cos(\alpha - 130°) + a)^2 + (L\sin(\alpha - 130°) - b)^2}$$

avec $\alpha \in [-50°, 50°]$, a = 0.14 m, b = 0.046 m et L = 0.49 m.
L'objectif du TP est de déterminer l'angle α pour une valeur d'élongation λ_1 donnée.

Question 1. Définir la fonction `lambda1(alpha)`

2 Approche graphique

Question 2. Tracer l'angle α en fonction de λ_1 sur le domaine d'étude considéré.

Question 3. Déterminer graphiquement, à l'aide de zoom sur la figure, l'angle α pour un allongement de $\lambda_1 = 0.4$ m.

3 Utilisation de numpy et scipy

Un certain nombre d'algorithmes existent pour résoudre une équation, la plupart d'entre eux sont déjà implantées dans le module `scipy`.
Nous allons utiliser les méthodes de la dichotomie et de Newton pour déterminer une solution de référence. Ces méthodes permettent de résoudre une équation de la forme $f(x) = 0$.

On importe préalablement les modules `numpy` et `scipy.optimize`

La méthode de dichotomie est accessible à l'aide de :

```
optimize.bisect(fonction, borne inférieure, borne supérieure, précision)
```

La méthode de Newton est accessible à l'aide de :

```
optimize.newton(fonction, valeur initiale, fonction dérivée, précision)
```

Question 4. Essayer ces deux méthodes pour déterminer la solution de cette équation pour un allongement de $\lambda_1 = 0.4$ m avec une précision de 10^{-6} près. On fera varier les bornes et les valeurs initiales et on commentera ce qui se passe.

4 Méthode de la dichotomie

La méthode par dichotomie est d'approcher la solution par réduction successive de l'intervalle de recherche. Cette méthode est adaptée pour les fonctions (localement) monotones dont on recherche le zéro.

Méthode :
- choix de l'intervalle de départ [a, b].
- tant que $|b - a| > \varepsilon$
- $c = \dfrac{a+b}{2}$
  - si f(c)*f(a)>0 alors a=c
  - sinon b=c

Question 5. Implémenter cet algorithme dans une fonction `dichotomie(f, a, b, epsilon)` et vérifier que le résultat renvoyé correspond à celui attendu.

Question 6. Modifier votre fonction pour qu'elle renvoie également le nombre d'itérations ainsi que la solution quand la convergence est atteinte.

Question 7. Créer une nouvelle fonction `dichotoliste` d'arguments à déterminer qui implémente cet algorithme en renvoyant la liste contenant les solutions successives obtenues à chaque itération. On supposera que la solution à chaque itération est $\dfrac{a+b}{2}$

Question 8. On pose :

$$\text{ordre}(i) = \frac{\log(|x_i - x_{i-1}|)}{\log(|x_{i-1} - x_{i-2}|)}$$

Écrire une fonction `ordre(liste_x)` qui renvoie la liste contenant l'ordre de convergence en fonction des itérations. En déduire l'ordre de convergence de la méthode par dichotomie.

5 Méthode de Newton

Nous rappelons la relation de récurrence vue en cours pour la méthode de Newton :

$$x_{k+1} = xk - \frac{f(x_k)}{f'(x_k)}$$

Cette méthode a un ordre de convergence théorique de 2 (voir le cours) quand on est capable de déterminer précisément la dérivée.

Question 9. Calculer la dérivée de λ_1 et implémenter la fonction dérivée comme fonction `fprime`

Question 10. Écrire une fonction `newton(f, xini, eps)` qui affiche la solution, le nombre d'itérations et qui renvoie la liste des approximations x_k successives en prenant la dérivée exacte avec comme critère d'arrêt $|f(x_k)| < \varepsilon$

Question 11. Déterminer l'évolution de l'ordre de convergence en fonction des itérations et en déduire l'ordre de convergence.

6 Méthode de la sécante

Cette méthode est une variante de la méthode de Newton dans laquelle la dérivée est approximée par la pente de la droite passant par les deux points d'abscisses x_{k-1} et x_{k-2} calculés précédemment.

$$x_{k+2} = x_{k+1} - \frac{x_{k+1}}{x_{k+1} - xk}(x_{k+1} - x_k)$$

Question 12. Faire un dessin pour expliquer cette méthode

Question 13. Écrire une fonction `secante(f, xini1, xini2,eps)` qui affiche la solution, le nombre d'itérations et qui renvoie la liste des approximations x_k successives avec comme critère d'arrêt $|f(x_k)| < \varepsilon$.

Question 14. Déterminer l'évolution de l'ordre de convergence en fonction des itérations et en déduire l'ordre de convergence. Comparer à la valeur théorique qui est le nombre d'or : $\dfrac{1+\sqrt{5}}{2}$

7 Méthode du point fixe

La méthode consiste cette fois à résoudre le problème sous la forme $g(x) = x$.
L'algorithme itératif consiste à déterminer successivement $x_{k+1} = g(x_k)$.
Toute la subtilité de cette méthode consiste à trouver une fonction g suffisamment intelligente pour obtenir un ordre de convergence correct.

Question 15. Déterminer une fonction g qui permettrait d'appliquer la méthode du point fixe.

Question 16. Écrire une fonction `point_fixe(g, xini,eps)` qui affiche le résultat, le nombre d'itérations et qui renvoie la liste des approximations x_k successives avec comme critère d'arrêt $|f(x_k)| < \varepsilon$.

Question 17. Déterminer l'évolution de l'ordre de convergence en fonction des itérations et en déduire l'ordre de convergence.

8 Comparaisons temporelles

Il est possible de mesurer le temps mis par un programme (ou une fonction) en se servant de la bibliothèque `time` : pour cela on importe la bibliothèque à l'aide par exemple de

```
import time as tm
```

puis, on évalue la durée d'exécution d'une suite d'instructions à l'aide de la séquence suivante :

```
debut=tm.perf_counter()
procédure
fin=tm.perf_counter()
temps=fin-debut
```

Notez que `tm.clock()` et `tm.time()` existent aussi mais sont actuellement dépréciés.

Question 18. Testez et comparer les différents temps d'exécution pour les méthodes programmées et celles de scipy. Il pourra être pertinent de répéter les différents tests plusieurs fois et de faire une moyenne.