Séq4 Informatique et intelligence artificielle

Dossier travaux pratiques

Résolution de problème numérique

MICHELET

Préambule : la majorité des exemples sont tirés du poly de J-L Biondi.

1.Relation entrée sortie

60 minutes

Exemple - Camion élévateur

L'actionneur d'inclinaison du mât 2 est le vérin hydraulique (4+5). La longueur λ de celui-ci conditionne donc l'inclinaison α du mât.

On cherche une lois entrée-sortie en position : α et β en fonction de l'allongement du vérin λ. L'expression directe de a et b en fonction de lambda n'est pas « immédiate ».

Modèle mécanique et paramétrage

$\overrightarrow{AC} = a\overrightarrow{x_1} - b\overrightarrow{y_1}$

$\overrightarrow{AO} = c\overrightarrow{x_1} - d\overrightarrow{y_1}$

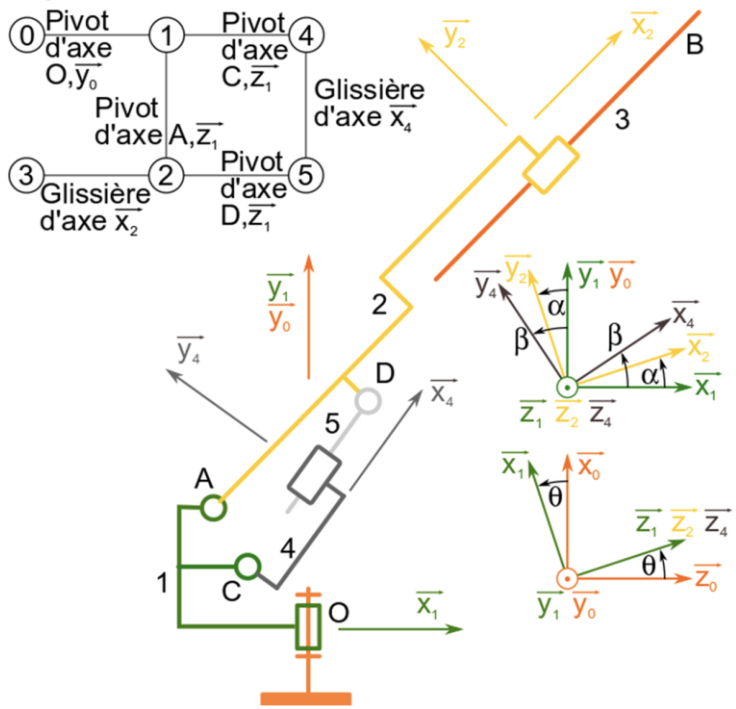
$\overrightarrow{AD} = L\overrightarrow{x_2} - e\overrightarrow{y_2}$

$\overrightarrow{AB} = \mu(t)\overrightarrow{x_2}$; $\overrightarrow{CD} = \lambda(t)\overrightarrow{x_4}$

$a = 0.2$ m ; $b = 0.8$ m

$e = 0.1$ m ; $L = 1.6$ m

Plage d'inclinaison du mât souhaitée : $\alpha \in [0 ; \pi/2]$



Question 1 : Déterminez les expressions de λ et β en fonction de α.

Question 2 : Déterminez la course du vérin ainsi que sa plage de position angulaire (β) compte tenu de la plage d'inclinaison du mât.

Question 3 : Écrivez le programme Python permettant de tracer les lois E/S $\alpha(\lambda)$ et $\beta(\lambda)$.

Exemple – Camion élévateur correction 1/2

$$\overrightarrow{AD} = \overrightarrow{AC} + \overrightarrow{CD} \Leftrightarrow L\overrightarrow{x_2} - e\overrightarrow{y_2} = a\overrightarrow{x_1} - b\overrightarrow{y_1} + \lambda\overrightarrow{x_4} \Leftrightarrow \lambda\overrightarrow{x_4} = L\overrightarrow{x_2} - e\overrightarrow{y_2} - a\overrightarrow{x_1} + b\overrightarrow{y_1}$$

Par élévation au carré :

$$\lambda^2 = L^2 + e^2 + a^2 + b^2 - 2aL\cos\alpha + 2bL\sin\alpha - 2ae\sin\alpha - 2be\cos\alpha$$

$$\boxed{\lambda = \sqrt{L^2 + e^2 + a^2 + b^2 - 2(aL + be)\cos\alpha + 2(bL - ae)\sin\alpha}}$$

En projetant dans le repère 1 :

$$\lambda\cos\beta = L\cos\alpha + e\sin\alpha - a$$
$$\lambda\sin\beta = L\sin\alpha - e\cos\alpha + b$$

En élevant au carré puis en sommant les deux équations, on retrouve l'expression de λ.

En divisant pour faire disparaître λ :

$$\boxed{\beta = \text{Arc tan}\left(\frac{L\sin\alpha - e\cos\alpha + b}{L\cos\alpha + e\sin\alpha - a}\right)}$$

Avec $\alpha \in [0° ; 90°] \rightarrow$ $\lambda \in [1.565\text{ m} ; 2.402\text{ m}]$ soit une course de 0.837 m

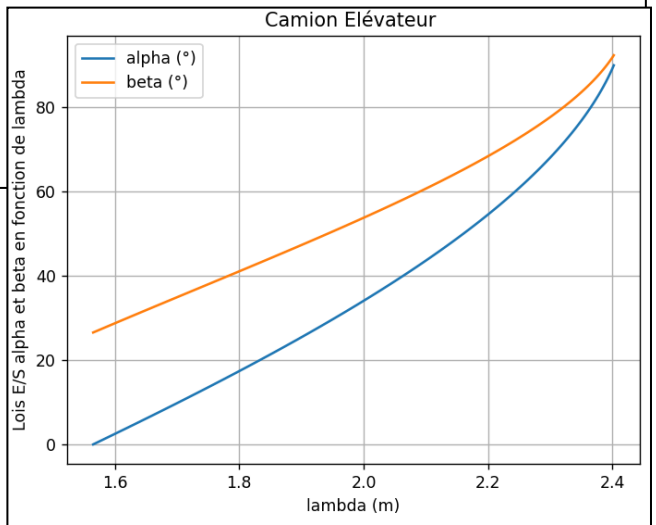
$\beta \in [26.6° ; 92.4°]$ (la fonction tangente est π périodique)

Pour éviter tout problème avec arc-tangente lorsque β « passe » par 90°.

$$\boxed{\beta = \text{Arc cos}\left(\frac{L\cos\alpha + e\sin\alpha - a}{\sqrt{L^2 + e^2 + a^2 + b^2 - 2(aL + be)\cos\alpha + 2(bL - ae)\sin\alpha}}\right)}$$

Exemple - Camion élévateur correction 2/2

```
import numpy as np ; import matplotlib.pyplot as plt # importation des modules

# parametres en mètres
a = 0.2;b=0.8;e=0.1;L=1.6

# plage alpha pour aller jusqu'à pi/2 inclus par pas de 1°
alpha=np.arange(0,91/180*np.pi,np.pi/2/90)

# calcul de la longueur lambda et de l'angle beta
lamb=np.sqrt(L**2+e**2+a**2+b**2-2*np.cos(alpha)*(a*L+b*e)+2*np.sin(alpha)*(b*L-a*e))
beta=180/np.pi*np.arccos((L*np.cos(alpha)+e*np.sin(alpha)-a)/lamb)
alphadeg=180/np.pi*alpha

# Tracé des courbes
plt.plot(lamb, alphadeg, label="alpha (°)") ; plt.plot(lamb, beta, label="beta (°)") # trace y(x)
plt.ylabel('Lois E/S alpha et beta en fonction de lambda') ; plt.xlabel("lambda (m)") # Titre des axes
plt.title("Camion Elévateur ") # Titre du graphique
plt.legend() # Imprime les légendes
plt.grid(True) # Affichage de la grille
plt.savefig("Camion_Elevateur.png",dpi=125) # sauvegarde la figure
plt.show() # Affichage de la fenêtre
```



Exercice 1.a : Bielle - manivelle

La manivelle (1) tourne à vitesse constante.

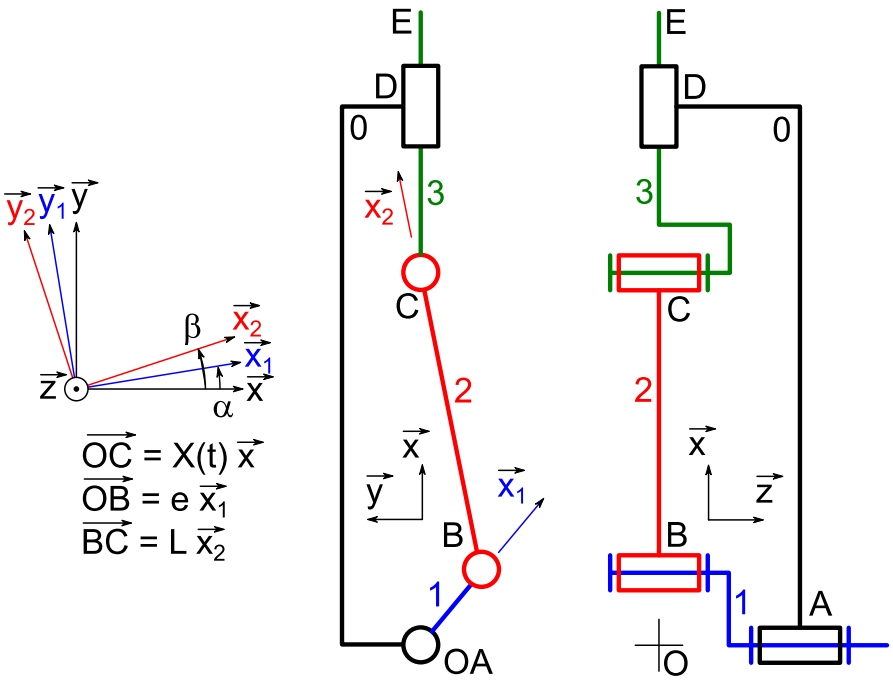
On souhaite visualiser la position, la vitesse et l'accélération du piston (3) en fonction de la position angulaire et la vitesse angulaire de la manivelle (1).

On souhaite aussi connaître la position angulaire, la vitesse angulaire et l'accélération angulaire de la bielle (2) en fonction de ces mêmes paramètres d'entrée et géométriques.

On devra pouvoir choisir la vitesse de la manivelle, la longueur L de la bielle et la valeur e de l'excentricité.

Enfin, on veut visualiser la différence entre les valeurs exactes et les valeurs approchées si on considère que le rapport L/e est petit (c'est-à-dire $L \approx 10 * e$).

Dans la mesure où l'on fait une résolution numérique, il ne faut surtout pas chercher à tout exprimer en fonction de la position d'entrée α.



Question 1 : Donnez l'expression de X(t) en fonction de α(t) et β(t) , puis celle de β(t) en fonction de α(t).

Question 2 : Donnez les expressions des dérivées première et seconde de X(t) en fonction de α(t), β(t) et leurs dérivées.

Question 3 : Donnez des expressions simplifiées des grandeurs précédentes (en considérant un développement limité à l'ordre 1 pour β(t) tel que sin $\beta \approx \beta$ et cos $\beta \approx 1$).

Question 4 : Proposez un programme python qui permet de :

- saisir la vitesse de rotation N en tr/mn, l'excentricité e et la longueur de bielle L en mm (fonction Input : `N=float(input("vitesse de rotation en tr/mn : "));`),
- tracer sur 4 graphiques X(mm), b(°), X'(m/s) et X''(m/s²), en valeur exacte et approchée, en fonction du temps et faire figurer le rapport L/e dans un titre.

Suite slide suivante

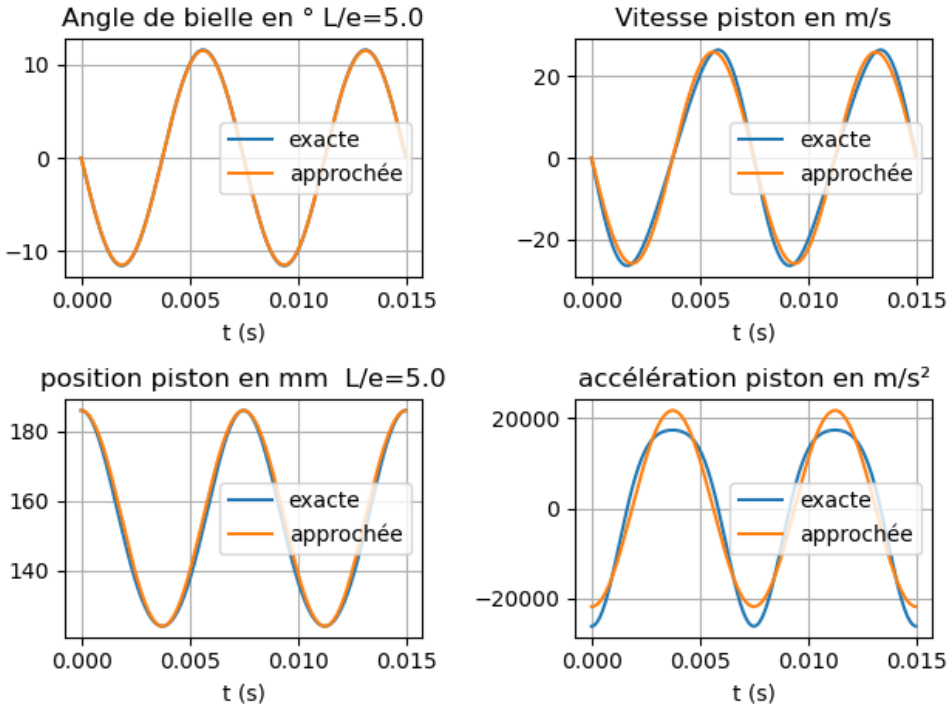
Exercice 1.a : Bielle – manivelle (suite)

Question 5 : Réalisez 4 simulations pour N = 8000 tr/mn : e = 31 mm, L = [155, 130, 100, 35] mm. Les rapports e/L des couples (31,130) et (31,100) correspondent aux dimensions de deux moteurs de 500 cm^3. Le premier est dit à bielle longue, l'autre à bielle courte

Question 6 : Commentez les résultats. Vous pouvez aussi faire des tests pour voir jusqu'où l'approximation est acceptable pour X et b et voir ce qui se passe avec L ≈ e.

Vous devriez obtenir les courbes suivantes (pour le cas L=5e) :

La correction est donnée sur cahier de prépa.

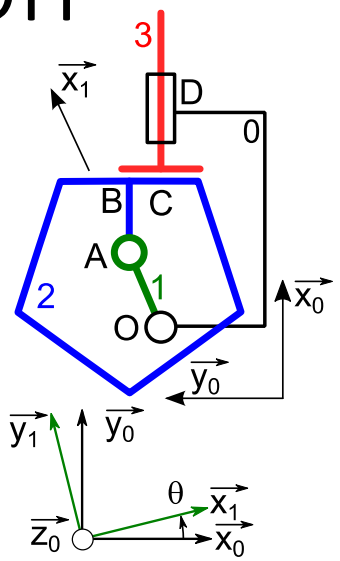


Exercice 1.b : pompe à piston

Ce modèle est celui associé à un moteur hydraulique radial à cylindrée fixe.

Modèle mécanique et caractéristiques constructeur :

$\overrightarrow{OA} = e\overrightarrow{x_1}$ $\overrightarrow{AB} = a\overrightarrow{x_0}$ $\overrightarrow{OD} = H\overrightarrow{x_0}$ $\overrightarrow{OC} = X(t)\overrightarrow{x_0}$ $\overrightarrow{CB} = Y(t)\overrightarrow{y_0}$ $\theta = (\overrightarrow{x_0}, \overrightarrow{x_1})$



- Excentricité e = 10 mm ; Rayon piston R = 20 mm
- Cylindrée : 125 cm3
- Vitesse nominale : 600 tr/mn (ou « RPM »)
- Pression continue de fonctionnement : 250 bars – Maxi 400 bars
- Couple spécifique théorique 1.8 N.m/bar
- Puissance de sortie maxi mesurée : 24 kW

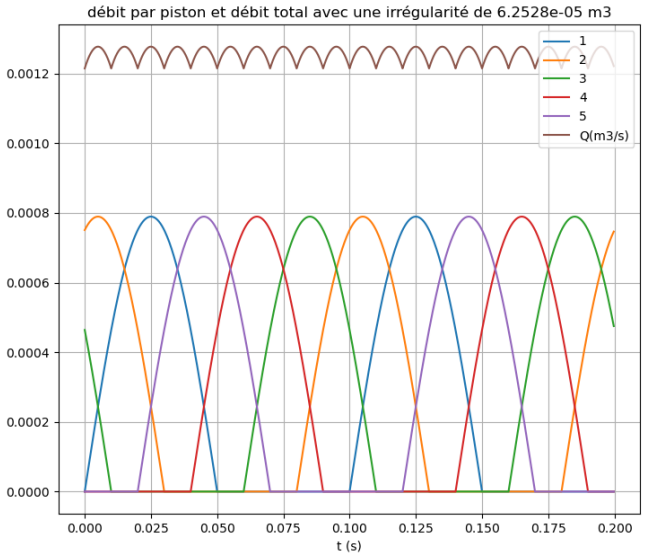
Question 1 : Par fermeture cinématique OU géométrique (en passant par les points OABC, et en utilisant le fait que $\overrightarrow{V_{C\in 3/0}}.\overrightarrow{x_0} = V_{30}$ et $\overrightarrow{V_{C\in 3/2}}.\overrightarrow{y_0} = V_{32}$) déterminez les expressions de V_{30} et V_{32} en fonction de e, θ et la sortie attendue w_{10}.

On définit le débit Q comme étant la quantité « vitesse du piston * surface du piston ».

Question 2 : Écrivez le programme python permettant de tracer la courbe de débit pour chaque piston, le débit instantané pour l'ensemble des pistons alimentant le moteur ainsi que l'irrégularité cyclique (amplitude de variation du débit total).

Question 3 : Vous préciserez, au vu de la courbe, combien de pistons sont à l'alimentation en même temps selon la phase du mouvement.

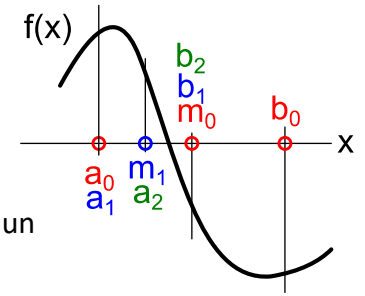
Vous devriez obtenir la courbe suivante :
La correction est donnée sur cahier de prépa.



2.Recherche de 0

60 minutes

Cours : dichotomie

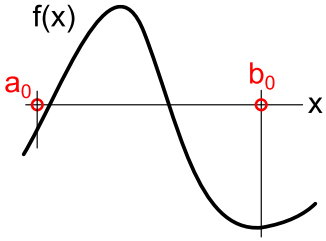


Soit la fonction f(x) continue. On recherche x* solution de l'équation f(x) = 0 dans l'intervalle [a,b].

Si $x^* \in [a,b] \rightarrow f(a)$ et f(b) sont de signes opposés $\rightarrow f(a) \cdot f(b) < 0$. Si ce n'est pas le cas, il faut définir un autre intervalle.

On part du principe que l'on dispose du tracé de la fonction et que l'on connait l'intervalle où chercher la solution x*.

Pour trouver la solution, on divise l'intervalle en deux parties égales avec comme milieu $m_0=(a+b)/2$.



Si $f(a) \cdot f(m) > 0$, $x^* \in [m,b]$ sinon $x^* \in [a,m]$. On réitère alors la recherche dans le nouvel encadrement jusqu'à ce que $(a-b) < \varepsilon$ où ε est la précision voulue.

Exercice 2 : dichotomie

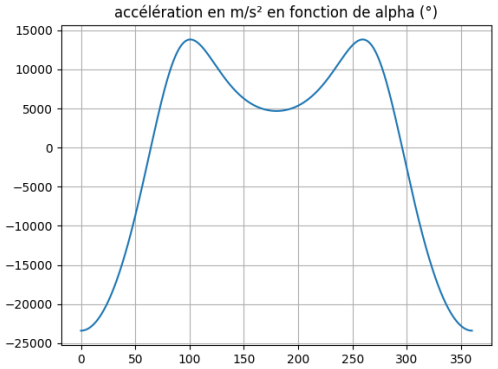
On réutilise ici le système bielle manivelle et les équations de mécanique déjà définies. On souhaite connaître la valeur de α (°), angle de rotation de la manivelle, pour laquelle la vitesse est maximum et donc pour laquelle l'accélération du piston est nulle.

Question 1 : Définissez une fonction "acceleration(alpha)" qui retourne la valeur de l'accélération du piston pour une valeur de α. La correction est donnée slide suivante.

Question 2 : Le corps du programme devra permettre le tracé de l'accélération du piston (m/s^2) en fonction de $\alpha \in [0°,360°]$ pour une valeur de e = 0.02 m, L = 0.03 m et N = 8000 tr/mn. Il devra permettre de choisir un encadrement de α, calculer α pour lequel l'accélération est nulle avec une précision de 0.01° et afficher la solution.

Le tracé à obtenir pour l'accélération est le suivant :

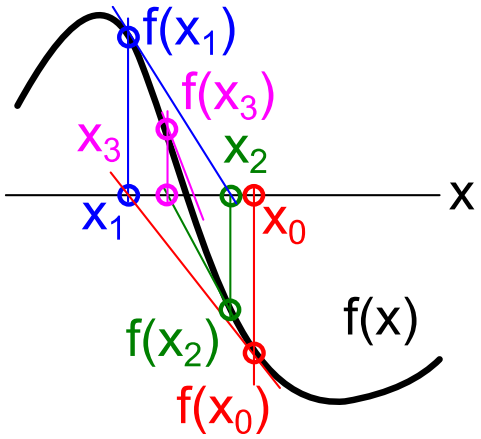
La correction est donnée sur cahier de prépa.



Dichotomie – fonction accélération

```
import numpy as np ; import matplotlib.pyplot as plt
N=8000 ; e = 0.02 ; L = 0.03 ; w=N*np.pi/30 # paramètres
def acceleration(alpha): # définition de l'accélération
            alpha=alpha*np.pi/180;beta=np.arcsin(-e/L*np.sin(alpha)); dep=e*np.cos(alpha)+L*np.cos(beta)
            betaprim=-e*w/L*(np.cos(alpha)/np.cos(beta))
            betasec=(betaprim**2*np.sin(beta)+e/L*w**2*np.sin(alpha))/np.cos(beta)
            depprim=-e*w*np.sin(alpha)-L*betaprim*np.sin(beta)
            depsec=-e*w**2*np.cos(alpha)-L*betasec*np.sin(beta)-L*betaprim**2*np.cos(beta)
            return(depsec) # retourne l'accélération du piston pour une valeur de alpha

# calcul de l'accélération pour une liste de valeurs et tracé
alpha0=0;alpha1=360;pas=0.1;alpha=np.arange(alpha0,alpha1,pas);acc=acceleration(alpha)
plt.plot(alpha, acc) ; plt.title("accélération en m/s² en fonction de alpha (°)")
plt.legend ; plt.grid(True) ; plt.show()

# saisie de l'intervalle de recherche de racine et précision imposée
a=float(input("valeur mini de alpha (°) \n ____: ")); b=float(input("valeur maxi de alpha (°) \n ____: "))
prec=0.01 # précision

# dichotomie, test et avertissement
A compléter
```

Cours : Newton



Soit la fonction f(x) continue. On recherche x* solution de l'équation f(x) = 0 dans l'intervalle à partir de a valeur x_0 proche de la racine.

On calcule $f(x_0)$. La tangente à f en $f(x_0)$ coupe l'axe des x en x_1. On calcule alors $f(x_1)$. La tangente à f en $f(x_1)$ coupe l'axe des x en x_2.

On continue ainsi jusqu'à ce que $(x_{i+1}-x_i)<\varepsilon$ où ε est la précision voulue.

Ici encore on part du principe que l'on dispose du tracé de la fonction et que l'on sait où chercher pour que la méthode converge.

Cette méthode nécessite de connaître la dérivée. En effet, l'équation de la tangente en $f(x_0)$ par exemple s'écrit : $y(x) = f'(x_0)\, x + b$.

Avec $f(x_0) = f'(x_0)\, x_0 + b$, on trouve $b = f(x_0) - f'(x_0)\, x_0$.

L'équation est donc finalement $y(x) = f'(x_0)\, x + f(x_0) - f'(x_0)\, x_0 = f'(x_0)\,(x - x_0) + f(x_0)$

Cette tangente coupe l'axe des x en x_1 tel que $0 = f'(x_0)\,(x_1 - x_0) + f(x_0)$

Finalement, si on démarre en $x_0 = a$, la suite est de la forme :

$$x_1 = \frac{f'(x_0)\cdot x_0 - f(x_0)}{f'(x_0)} = -\frac{f(x_0)}{f'(x_0)} + x_0$$

Le problème est évidemment lorsque l'on travaille sur une série de valeur sans connaître ni la fonction ni sa dérivée.

$$x_{n+1} = -\frac{f(x_n)}{f'(x_n)} + x_n$$

En slide suivante on donne la fonction et un exemple (le principe est à connaitre).

Question (optionnelle, il n'y aura pas de correction donnée) : Appliquer la méthode de Newton à partir du fichier "BOMaxpid.csv" qui donne la vitesse du moteur pour un échelon de tension et tracer $\omega_{moteur}(t)$. On cherche ici à annuler l'accélération, vous devez donc travailler sur la dérivée de $\omega_{moteur}(t)$. Il faut donc au préalable définir une fonction dérivation numérique. Pour ouvrir le fichier "BOMaxpid.csv" vous aurez besoin de la partie suivante : « 3.Importation d'un fichier de mesure ».

Recherche par la méthode de Newton de la solution de l'équation f(x)=0

```
def solveNewton(f,df,a,eps):
    c = a-f(a)/df(a)
    while abs(c-a)>eps:
        a = c
        c = c-f(c)/df(c)
    return c
```

Cette fonction a pour entrées :

- f, function : fonction à valeur de IR dans IR
- df, function : dérivée de f à valeur de IR dans IR.
- a, flt : solution initiale
- eps,flt : tolérance de la résolution

Et pour sortie :

- flt : solution de la fonction

Pour la dérivée, soit on la connait déjà (exemple ci-dessous), soit on utilise la fonction suivante :

```
def derive_fonctions(f,x,eps):
    return (f(x+eps)-f(x))/(eps)
```

Exemple si on connait déjà la dérivée :

```
def f_dicho(x):
    from math import sqrt
    return x*x-math.sqrt(2)
def df_dicho(x):
    return 2*x
```

Cours : fonction « fsolve »

« Fsolve » est une fonction présente dans la bibliothèque scipy. Elle permet de trouver la racine d'une fonction, elle s'écrit, pour une fonction f(x) dont on souhaite connaître la valeur de x telle que f(x)=0 :

scipy.optimize.`fsolve(f,[x0,x1])`

Avec x0 et x1 correspondant aux bornes dans lesquelles fsolve va chercher une solution.

Exemple avec la fonction accélération précédente :

```
import numpy as np ; import matplotlib.pyplot as plt; import scipy.optimize
N=8000 ; e = 0.02 ; L = 0.03 ; w=N*np.pi/30 # paramètres
def acceleration(alpha): # définition de l'accélération
            alpha=alpha*np.pi/180 ; beta=np.arcsin(-e/L*np.sin(alpha)); dep=e*np.cos(alpha)+L*np.cos(beta)
            betaprim=-e*w/L*(np.cos(alpha)/np.cos(beta))
            betasec=(betaprim**2*np.sin(beta)+e/L*w**2*np.sin(alpha))/np.cos(beta)
            depprim=-e*w*np.sin(alpha)-L*betaprim*np.sin(beta)
            depsec=-e*w**2*np.cos(alpha)-L*betasec*np.sin(beta)-L*betaprim**2*np.cos(beta)
            return(depsec) # retourne l'accélération du piston pour une valeur de alpha

# calcul de l'accélération pour une liste de valeurs et tracé
alpha0=0;alpha1=360;pas=0.1;alpha=np.arange(alpha0,alpha1,pas);acc=acceleration(alpha)
plt.plot(alpha, acc) ; plt.title("accélération en m/s² en fonction de alpha (°)")
plt.legend ; plt.grid(True) ; plt.show()

# saisie de l'intervalle de recherche de racine et précision imposée
a=float(input("valeur mini de alpha (°) \n ____: ")); b=float(input("valeur maxi de alpha (°) \n ____: "))
prec=0.01 # précision

# fsolve
x = (scipy.optimize.fsolve(acceleration,[a,b]))
print("la solution est ",round(x[0],3))
```

3.Importation d’un fichier de mesure

20 minutes

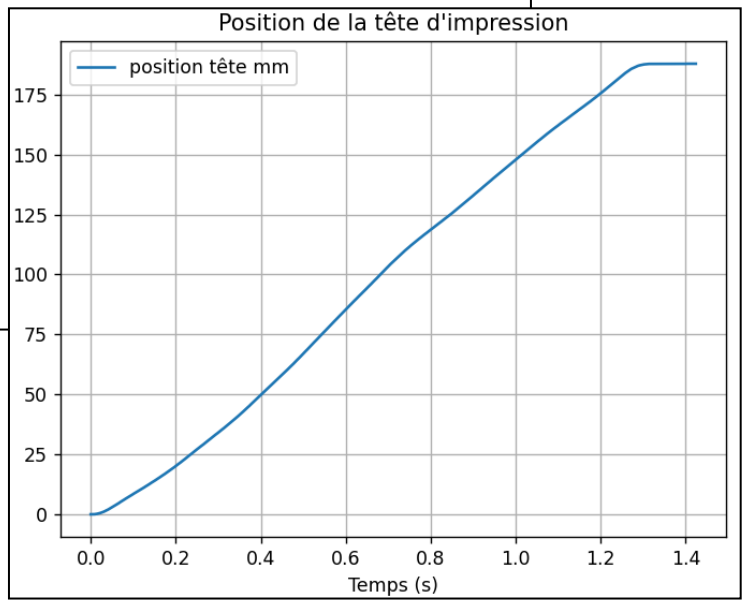
Exemple - Imprimante

Nous utiliserons pour l'exercice le fichier : "BO200imprimante.csv". Ce fichier est le résultat d'un essai en boucle ouverte avec une consigne en échelon de 200 mm.

Question 1 : Ouvrir le fichier avec un éditeur de texte afin de voir comment il est écrit et décrivez sa structure (colonne, ligne).

Question 2 : Avec Python, lire le fichier, convertir les colonnes temps et position en tableaux de flottants (`astype(float)`) en fonction de l'entête et tracer la position en fonction du temps.

```
import numpy as np ; import matplotlib.pyplot as plt
# import direct connaissant les entêtes
mesures = np.loadtxt('BO200imprimante.csv', delimiter=';', skiprows=1, dtype=str)
mesures = np.char.replace(mesures, ',', '.');
mesures = mesures.astype(float)
temps = mesures[:,0]; position = mesures[:,1]; vitesse = mesures[:,2]
# Affichage
plt.plot(temps, position, label="position tête mm")
plt.xlabel("Temps (s)")
plt.title("Position de la tête d'impression")
plt.grid(True);
plt.legend();
plt.savefig("courbes.png",dpi=125);
plt.show()
```



Exercice 3 - Importation mesures Maxpid

On utilise le fichier : "BFtrapeze90_maxpid.csv" : essai avec consigne de position de 90° en trapèze : accélération = 10 rd/s², vitesse constante = 1 rd/s, décélération = -10 rd/s².

Question 1 : Ouvrez le fichier avec un éditeur de texte et décrivez sommairement sa structure.

Question 2 : Tracez la position du bras et la vitesse de rotation du moteur en fonction du temps (vous préciserez les unités sur le tracé).

Vous devriez obtenir la courbe suivante :

La correction est donnée sur cahier de prépa.

4.Résolution d’équation différentielle 1er ordre : Euler

30 minutes

Cours : méthode d'Euler explicite

On souhaite résoudre une équation différentielle du type :

$$\tau.\frac{dy(t)}{dt} + y(t) = y_f$$

On pose alors :

- τ : constante de temps de l'équation différentielle (notée « `tau` » dans le programme)
- y_0 : valeur initiale de y(t) (on suppose que $t_0 = 0$)
- y_f, t_f : valeur finale de y(t) et de t
- nb : nombre d'échantillons pour la simulation

La résolution utilise l'approximation de la dérivé :

$$\frac{dy(t_i)}{dt} \approx \frac{y(t_{i+1}) - y(t_i)}{t_{i+1} - t_i}$$

En réutilisant l'équation différentielle donnée, cela donne :

$$\tau\,.\frac{y(t_{i+1}) - y(t_i)}{t_{i+1} - t_i} + y(t_i) = y_f$$

En posant le pas $p = t_{i+1} - t_i$:

$$\tau\,.y(t_{i+1}) = \tau\,.y(t_i) + p.\left(y_f - y(t_i)\right)$$

Soit

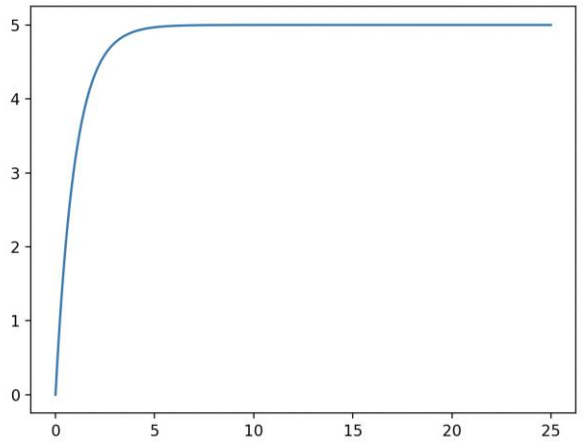
$$\boldsymbol{y(t_{i+1}) = y(t_i) + \frac{p}{\tau}.\left(y_f - y(t_i)\right)}$$

Cela donne dans python :

```
def euler_explicite(tau,y0,yf,tf,nb):
    t = 0
    y = y0
    pas = tf / nb
    res = []
    while t < tf:
        res.append((t,y))
        y = y + pas*(yf-y)/tau
        t = t + pas
    return res
```

Exemple :

```
res = euler_explicite(1,0,5,25,1000)
x,y = [],[]
for i in range(len(res)):
    x.append(res[i][0])
    y.append(res[i][1])
plot(x,y)
```



Exercice 4 : moteur Parvex assimilé à un 1er ordre

On donne les valeurs suivantes : Tension maxi : 40 V ; vitesse maxi : 6600 tr/mn

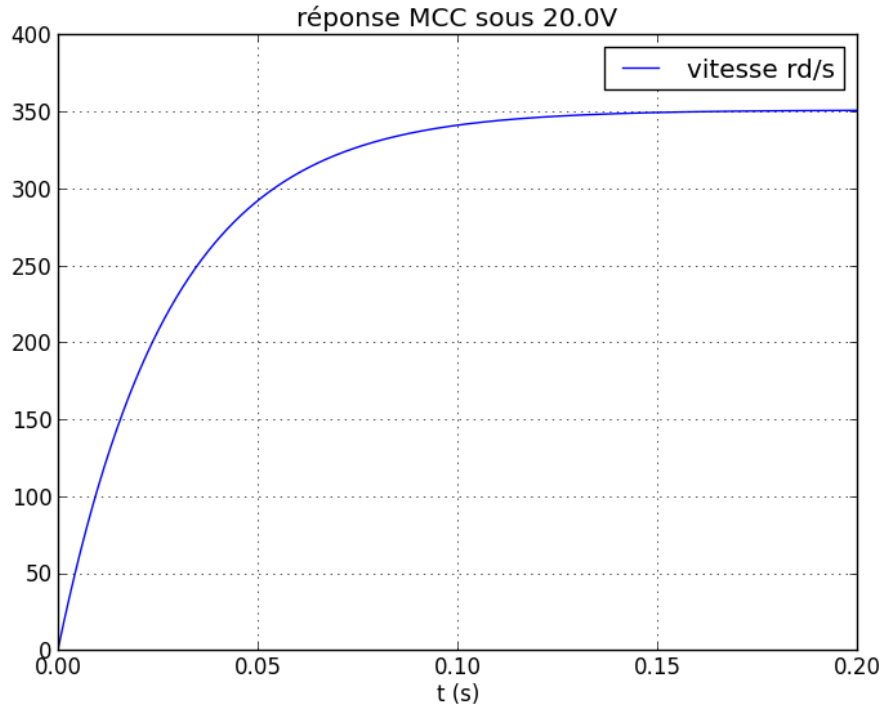
R = 4.68 Ohm ; Ke = 0.057 V.s ; Kc = 0.057 N.m/A ; J = 1.95 10-5 kg.m² (on néglige le couple resistant sec et fluide, ainsi que l'inductance),

Question 1 : Rappelez les quatre équations d'un moteur à courant continu et proposez une mise en forme de l'équation différentielle de l'équation de vitesse en fonction de u(t) tension d'entrée. Vous devriez obtenir une forme du type : $C.\frac{d\omega(t)}{dt} + B.\omega(t) = A.u(t)$

Question 2 : Proposez un programme qui permet de tracer la réponse en vitesse à un échelon de tension de 20 V.

Vous devriez obtenir la courbe suivante :

La correction est donnée sur cahier de prépa.



5.Suppression d’un bruit de mesure : moyennes et correcteurs

60 minutes

Cours : moyenne glissante arithmétique

On part d'une liste de valeurs de **n** éléments $\mathbf{y_i}$ d'indice **i** (variant de 0 à n-1) et on arrive à une liste de valeur ma_i (« ma » pour moyenne arithmétique). Chacun de ces ma_i est une moyenne d'un nombre N de y_i (N est la largeur de la fenêtre, et celle-ci glisse tout le long de la liste).

Prenons le cas N = 3 : on ne peut pas calculer les termes ma_0 et ma_{n-1}. On commence donc le calcul à partir du moment où ma_i est défini et la liste des ma_i sera plus courte que celle des y_i. Si on veut une liste de ma_i de même longueur que celle des y_i, on peut affecter y_i aux ma_i non définis au début et à la fin.

y_0	y_1	…	y_i	…	y_{n-2}	y_{n-1}
$ma_0 = y_0$	$ma_1 = \frac{y_0 + y_1 + y_2}{3}$	…	$ma_i = \frac{y_{i-1} + y_i + y_{i+1}}{3}$	…	$ma_{n-2} = \frac{y_{n-3} + y_{n-2} + y_{n-1}}{3}$	$ma_{n-1} = y_{n-1}$

Dans notre exemple on aurait alors $ma_0 = y_0$ et $ma_{n-1} = y_{n-1}$. Avec une assez grande valeur de N, la courbe est mieux lissée mais peut s'éloigner trop de la courbe de mesure, il faut ajuster.

Cours : moyenne glissante pondérée

Sur la même base de départ, au lieu d'une moyenne, on réalise une pondération des termes au fur et à mesure que l'on avance dans la liste. Cette pondération est caractérisée par le poids des mesures précédentes (me_{i-1}) par rapport à la mesure actuelle (y_i). On définit le coefficient de pondération : $\alpha = \frac{2}{N+1}$.

Plus α est petit (N grand), moins les dernières mesures comptent par rapport aux précédentes. Avec N petit, la courbe est peu filtrée, avec N grand, on s'éloigne trop de la courbe : N $\in$ [10 ; 30] donne souvent un bon résultat. Le modèle de calcul est donné ci-dessous pour N (α) quelconque.

y_0	y_1	…	y_i	…	y_{n-2}	y_{n-1}
$me_0 = y_0$	$me_1 = \alpha . y_1 + (1-\alpha) . me_0$	…	$me_i = \alpha . y_i + (1-\alpha) . me_{i-1}$	…	$me_{n-2} = \alpha . y_{n-2} + (1-\alpha) . me_{n-3}$	$ma_{n-1} = y_{n-1}$

Le résultat est une liste de me_i plus courte que celle des y_i car le premier élément ma_0 n'est pas défini. Ici encore on peut choisir d'avoir une liste de me_i plus courte que celles des y_i ou d'ajouter un premier terme ($me_0 = y_0$).

Exercice 5.a : vitesse tête de l'imprimante

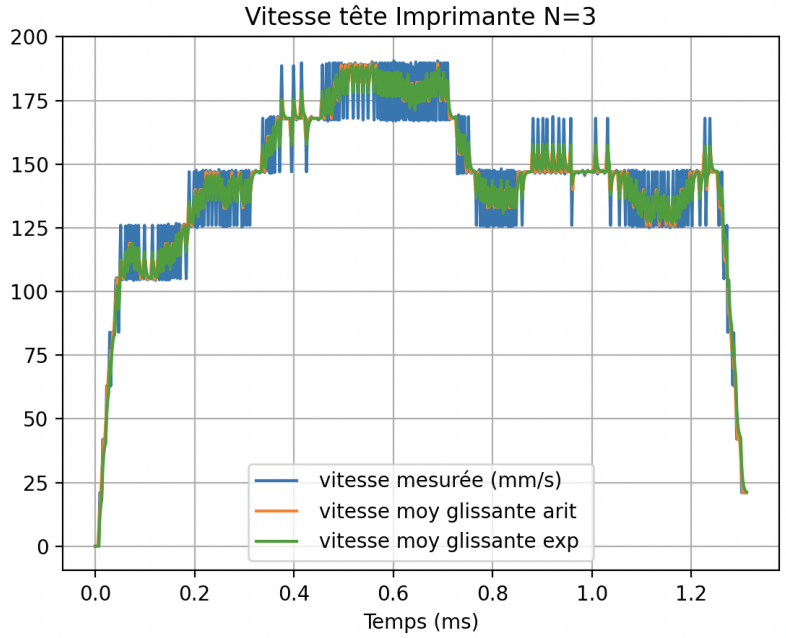
Question 1 : Importez le temps et la vitesse à partir du fichier *"BO200imprimante.csv"*

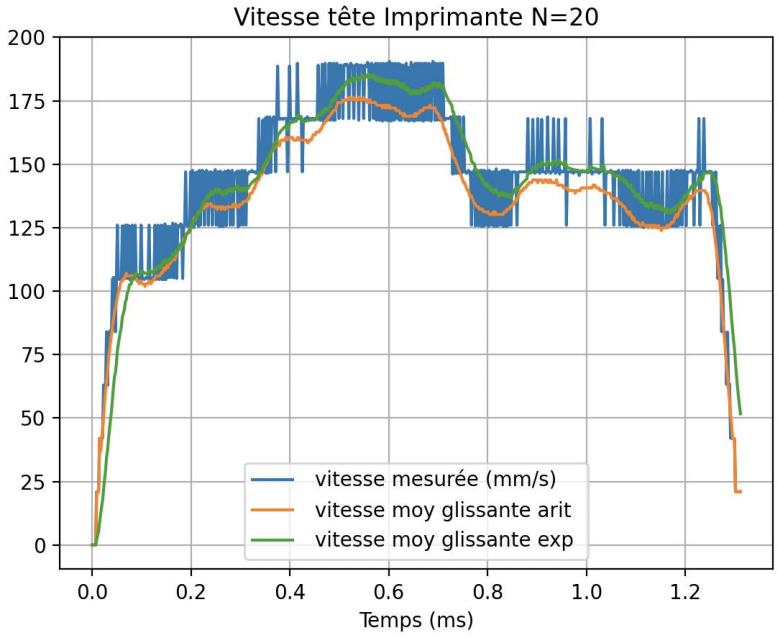
Question 2 : Proposez un programme qui permet de visualiser les résultats de la mesure de vitesse filtrées par les deux méthodes.

Question 3 : Faites varier N pour voir son influence.

Vous devriez obtenir les courbes ci-dessous.

La correction est donnée sur cahier de prépa.

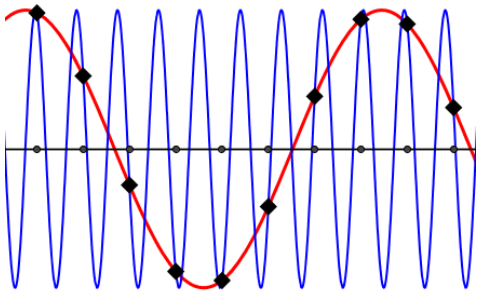




Cours : correcteur numérique 1/4

La fréquence d'échantillonnage d'un système d'acquisition doit être choisie pour éviter le phénomène dit de « repliement ». Ce dernier est susceptible de générer de faux signaux lorsque l'entrée analogique est sous échantillonnée.

En prenant l'exemple ci-contre, pour la période d'échantillonnage T_e = 1 s (point noir), les points associés à un signal de fréquence 0.1 Hz (courbe rouge) sont confondus avec ceux d'un signal de fréquence 0.9 Hz (courbe bleu).



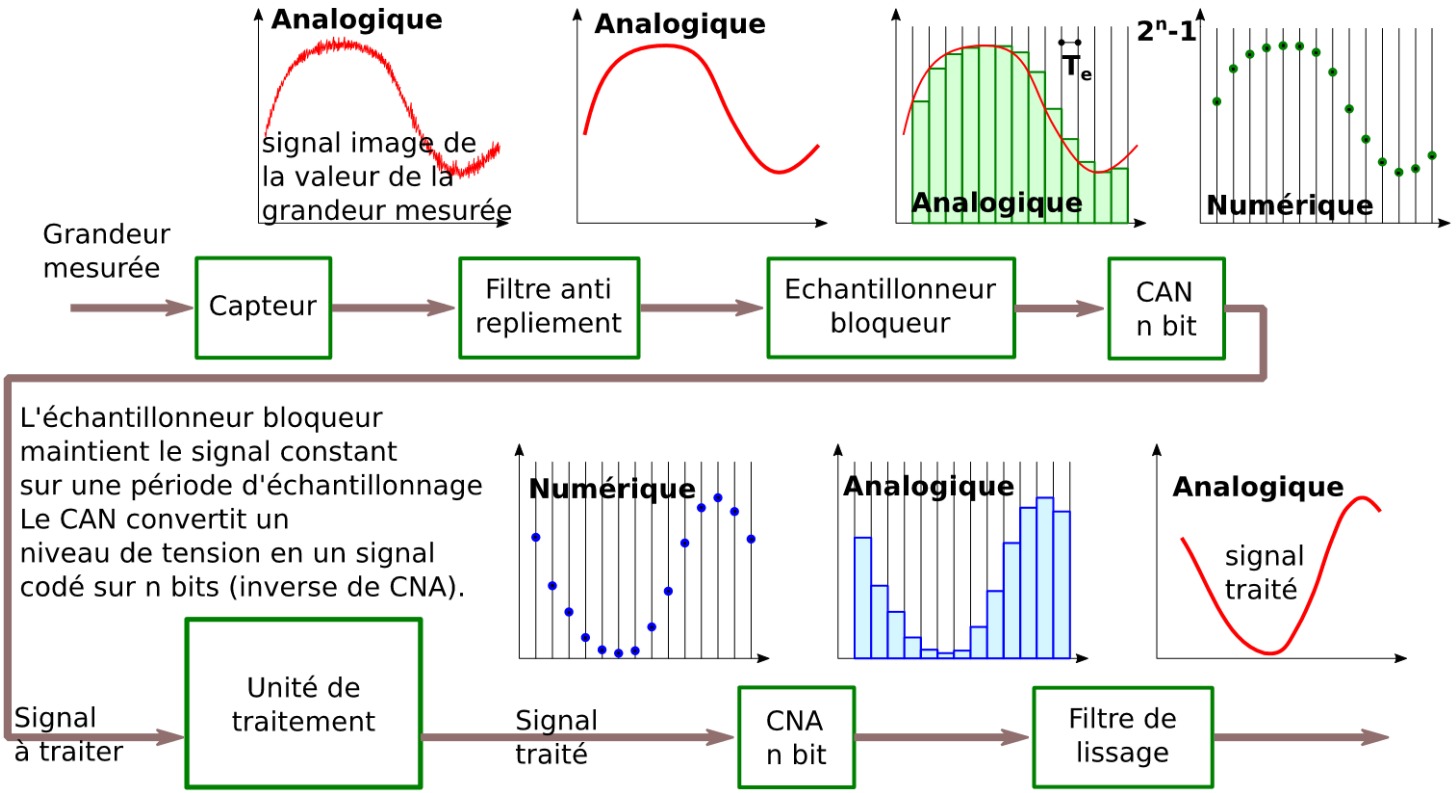
Le phénomène de repliement peut se produire dès lors que les signaux on des composantes dont les fréquences dépassent la moitié de la fréquence d'échantillonnage. Soit un signal x(t) compris dans la bande de fréquence [0 ; f_{max}]. On démontre (**Théorème de Nyquist-Shannon**) que pour pouvoir reconstruire (interpoler) le signal à partir des échantillons, la fréquence d'échantillonnage doit respecter $f_e > 2\ f_{max}$ (avec $f_e = 1/T_e$ et T_e période de l'échantillonnage. $f_e/2$ est appelée fréquence de Nyquist).

Afin d'éviter le phénomène de repliement en raison du bruit (en général un signal parasite à haute fréquence), on filtre le signal. Soit parce qu'on ne peut pas atteindre physiquement la fréquence d'échantillonnage qui serait nécessaire pour que les parasites ne faussent pas l'échantillonnage, ou parce que l'on ne peut pas traiter et transmettre le signal assez vite compte tenu de la quantité de donnée, ou tout simplement pour optimiser la chaine de traitement.

L'idée est donc d'éliminer les composantes du signal à traiter de fréquence supérieures à f_{max} donc à la fréquence de Nyquist = $f_e/2$. Un filtre idéal aurait un gain constant jusqu'au maximum des fréquences qui nous intéressent, et un gain nul au-delà. On pourrait tenir le même raisonnement pour un filtre passe bande dont le gain serait nul avant et après la bande de fréquence à traiter.

Cours : correcteur numérique 2/4

Modèle d'une chaîne d'acquisition numérique



Cours : correcteur numérique 3/4

Filtre du 1 er ordre

L'équation différentielle associée à ce filtre est : $\tau \frac{dml(t)}{dt} + ml(t) = y(t)$

L'équation discrétisée donne : $\tau \frac{ml_{i+1} - ml_i}{T_e} + ml_i = y_i$

Avec :

- T_e période d'échantillonnage
- $\omega_0 = \frac{1}{\tau}$: pulsation de cassure

On a donc : $ml_{i+1} = ml_i + \frac{T_e}{\tau}(y_i - ml_i)$

Tout comme les moyennes glissantes, le résultat est une liste de ml_i plus courte que celle des y_i car le premier élément ml_0 n'est pas défini. Ici encore on peut choisir d'avoir une liste de me_i plus courte que celles des y_i ou d'ajouter un premier terme ($ml_0 = y_0$).

Si $f_c \uparrow$, $\tau \downarrow$ et le signal est moins filtré. Si $T_e \uparrow$, le signal est mieux filtré mais s'éloigne du signal réel (ou mesuré).

Cours : correcteur numérique 4/4

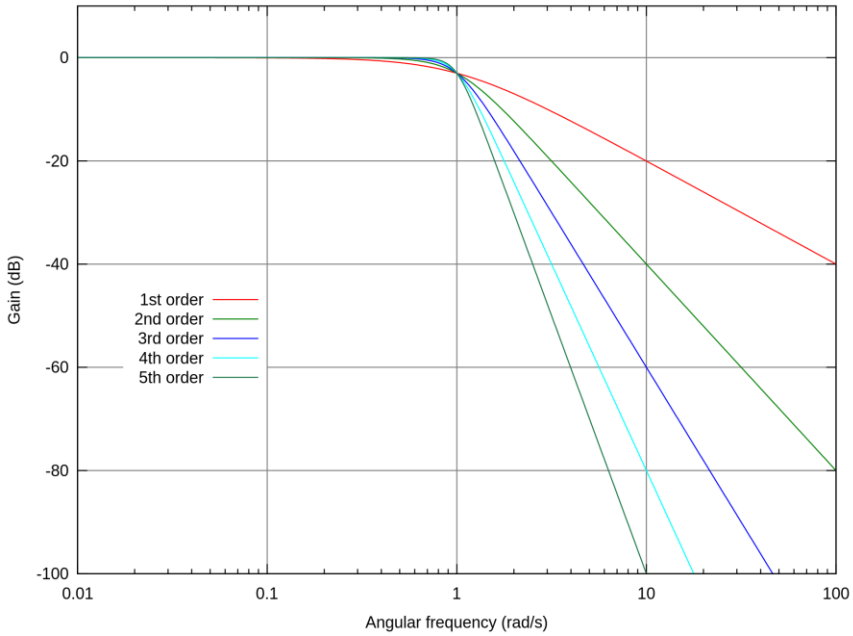
Filtre de Butterworth (physicien britannique 20ème siècle)

Ces filtres sont aussi des passe-bas, d'ordre 1, 2, 3, etc. On se limitera à l'étude de l'ordre 2. Ces filtres sont obtenus par combinaison de 1er et 2ème ordre. On donne ci-dessous les dénominateurs pour les deux premiers ordres pour $w_0 = 1$ rad/s :

Ordre	Dénominateur	Caractéristiques
1	1+p	$w_0 = 1$ rad/s
2	$1+1.4142p+p^2$	$w_0 = 1$ rad/s, $\xi = 0.7$

La particularité de ce filtre est que chaque polynôme a la même pulsation de cassure, qui correspond aussi à la pulsation de coupure du filtre à -3dB (la somme des gains à cette pulsation égale -3dB).

Par ailleurs le gain est pratiquement constant avant cette pulsation (pas de dépassement visible malgré les facteurs d'amortissements < 0.7).



Ce filtre est implémenté dans le module Scipy. On l'appelle ainsi :

```
from scipy import signal
# filtre de butterworth en utilisant le module scipy
Te = 0.002 ; fc = 40 ; fnyq = 1/2/Te ; ordre = 1 # inititalisation et choix de l'ordre
b, a = signal.butter(ordre, fc/fnyq, 'low', analog=False) # calcul du filtre
f_fb1 = signal.filtfilt(b, a, vitesse) # Application du filtre
```

Exercice 5.b : filtrage vitesse tête imprimante

Pour notre exemple, on travaille sur la base d'un fichier (BO200_imprimante.csv) donnant la vitesse d'un mobile, cette vitesse est calculée à partir d'une acquisition de position et on a une valeur toutes les 0.002 s, correspondant au pas de calcul.

On considère qu'il s'agit d'un fichier de mesure avec T_e = 0.002 (s) soit f_e = 500 (Hz). La règle de Shannon implique donc que l'on filtre tout ce qui se trouve au-delà de $f_{Nyquist}$ = 250 Hz pour ne pas avoir de problème de repliement.

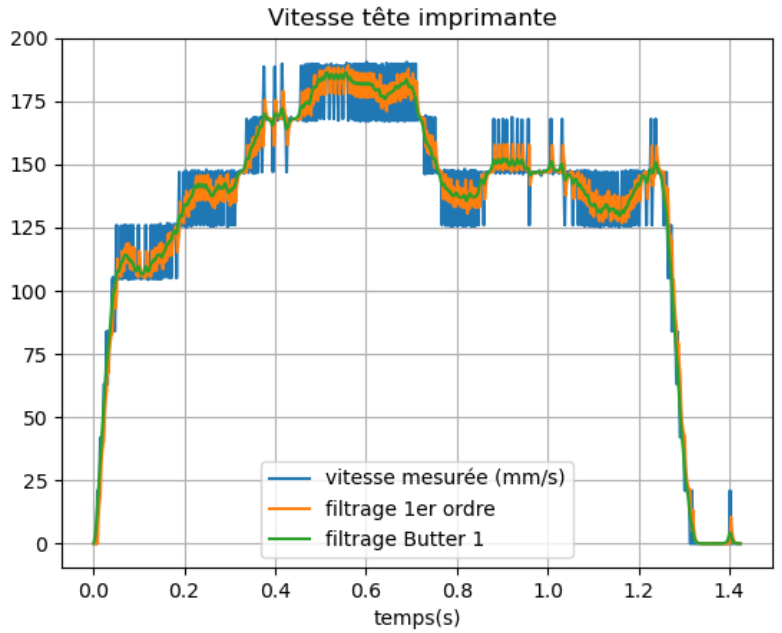
Question 1 : Importez le temps et la vitesse à partir du fichier "BO200_imprimante.csv"

Question 2 : Proposez un programme qui permet de superposer la mesure non filtrée aux mesures filtrées par un filtre du 1er ordre standard et du 1er ordre Butterworth (on prendra une fréquence de coupure `fc = 40HZ`)

Question 3 : Faites une comparaison qualitative entre les deux filtres.

Vous devriez obtenir les courbes ci-contre.

La correction est donnée sur cahier de prépa.



6.Dérivation et intégration numérique

45 minutes

Cours : dérivation & intégration numérique

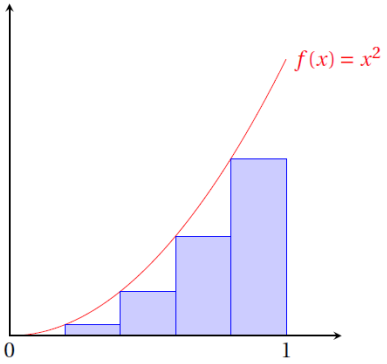
Dérivation numérique : méthode basique

Dans le cas où l'on souhaite connaître la vitesse à partir du relevé des positions : $\mathrm{vit}_i = \frac{\mathrm{pos}_{i+1} - \mathrm{pos}_i}{T_{i+1} - T_i}$

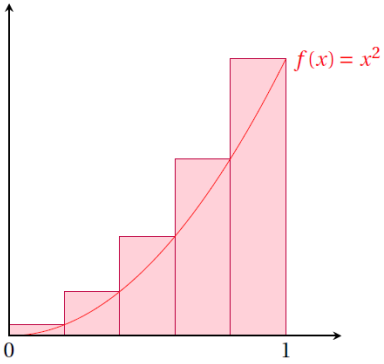
Intégration numérique : méthodes basiques

Si à l'inverse on souhaite connaître la position à partir du relevé des vitesses :

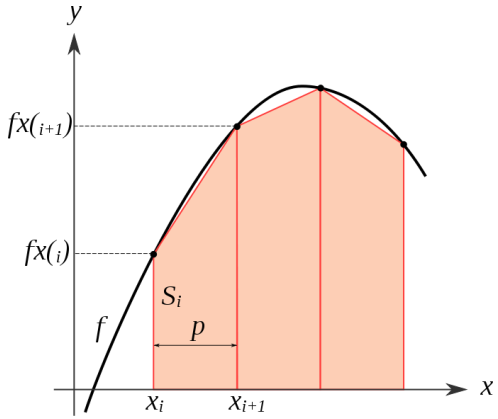
- Méthode des rectangles à droite : $\mathrm{pos}_i = \mathrm{vit}_{i+1}(T_{i+1} - T_i) + \mathrm{pos}_{i-1}$
- Méthode des rectangles à gauche : $\mathrm{pos}_i = \mathrm{vit}_i(T_{i+1} - T_i) + \mathrm{pos}_{i-1}$
- Méthode des trapèzes : $\mathrm{pos}_i = \frac{(\mathrm{vit}_i + \mathrm{vit}_{i+1})}{2}(T_{i+1} - T_i) + \mathrm{pos}_{i-1}$



Méthode des rectangles à gauche



Méthode des rectangles à droite



Méthode des trapèzes

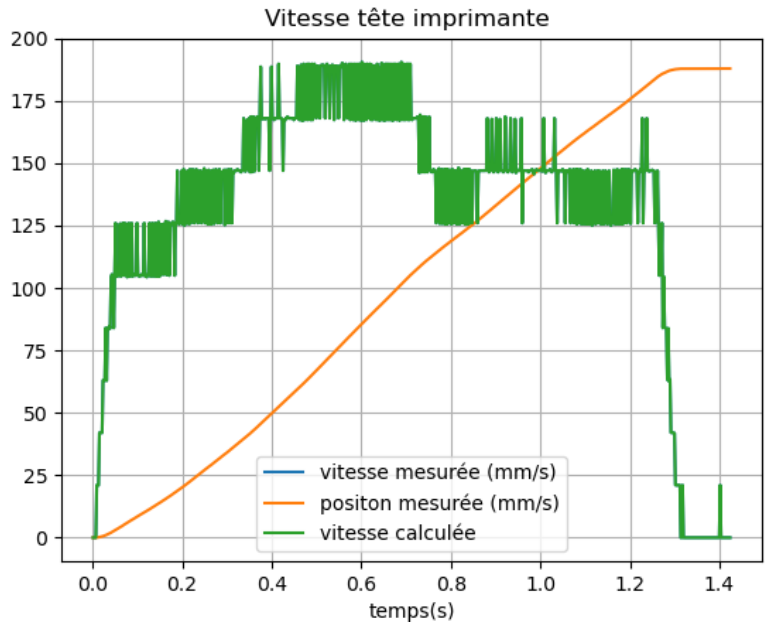
Exercice 6.a : dérivation position tête imprimante et bras maxpid

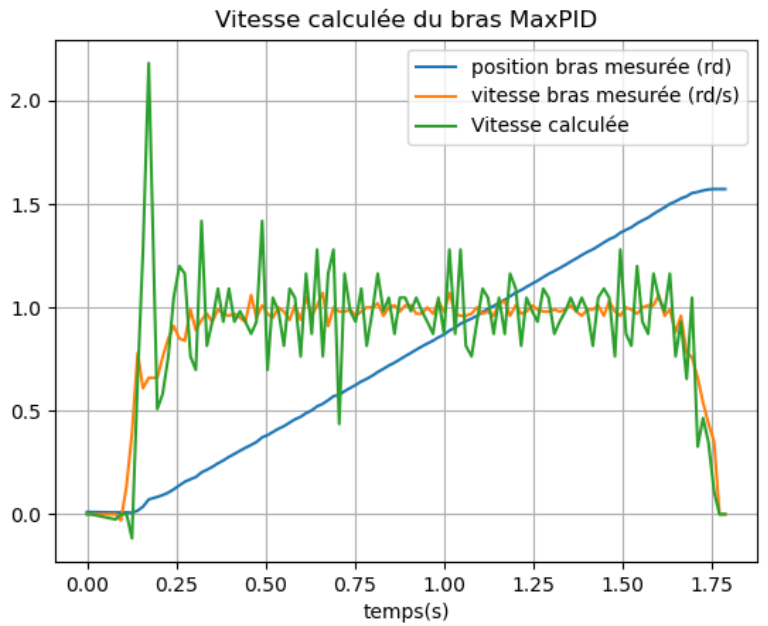
Question 1 : Proposez un programme qui, à partir du fichier "BO200imprimante.csv", permet de calculer la position de la tête à partir de la vitesse mesurée. Vous afficherez, sur un même graphe, en fonction du temps, la position et la vitesse de la tête, issues du fichier de mesure, ainsi que la vitesse calculée. Que pensez-vous de la vitesse mesurée ?

Question 2 : Proposez un programme qui, à partir du fichier "BFtrapeze90_maxpid.csv", permet de calculer la vitesse du bras à partir de la position mesurée. Vous afficherez, sur un même graphe, en fonction du temps, la position et la vitesse de la tête, issues du fichier de mesure, ainsi que la vitesse calculée. Concluez.

Vous devriez obtenir les courbes ci-dessous :

La correction est sur cahier de prépa.





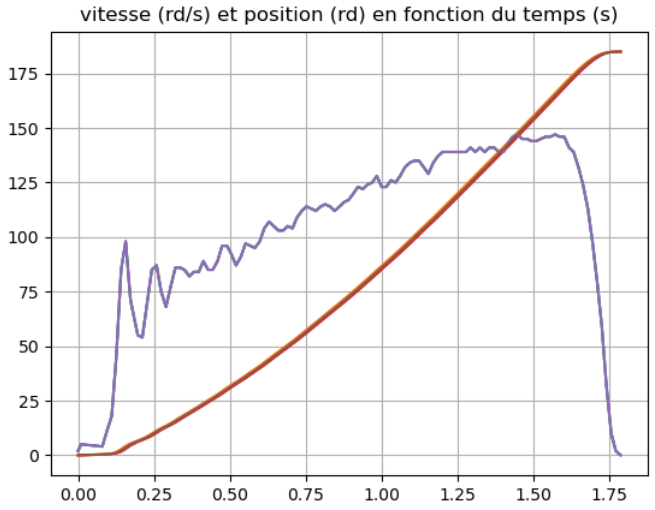
Exercice 6.b : intégration vitesse moteur maxpid

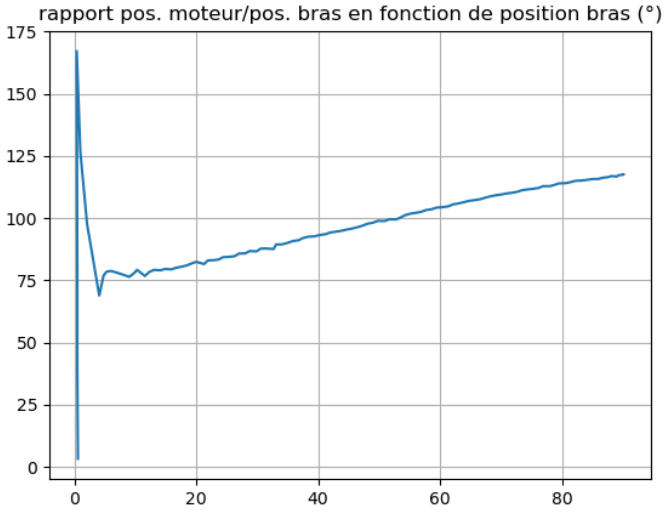
Question 1 : Analysez les données contenues dans le fichier "BFtrapeze90_maxpid.csv ». En créant trois fonctions "integ_def(x,y)", "integ_exc(x,y)" et "integ_tra(x,y)" qui calculeront numériquement ∫ydx par la méthodes des rectangles (droit et gauche) et par la méthode des trapèzes, proposer un programme qui permet de calculer numériquement la position du rotor à partir de la vitesse de celui-ci et du temps. Les 3 méthodes seront comparées sur un même graphe sur lequel figureront les positions calculées et la vitesse mesurée.

Question 2 : Tracez un graphe sur lequel sera affiché le rapport de position calculé du moteur sur la position mesurée du bras, en fonction de la position du bras (°). Il n'est pas improbable que vous rencontriez des problèmes (erreurs).

Vous devriez obtenir les courbes ci-dessous :

La correction est sur cahier de prépa.



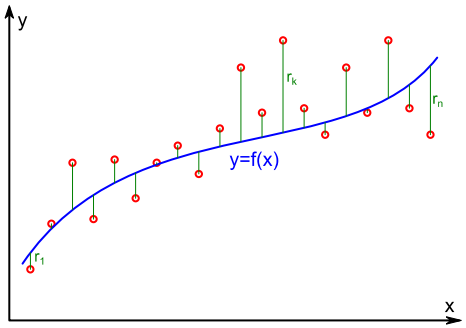


7.Régression linéaire : moindres carrés et IA

20 minutes

Cours : méthode des moindres carrés

On se limitera à un problème de deux dimensions, mais la méthode sera la même pour un problème de dimension n.



Les points (x_i,y_i) sont issus d'une mesure. Le modèle théorique associé au résultat de cette expérimentation est une fonction y(x, θ) où θ représente un ou plusieurs paramètres inconnus (les coefficients des puissances de x dans le cas d'un polynôme par exemple). Les paramètres θ optimaux sont ceux qui minimisent la somme suivante, appelée fonction « perte » ou « coût » $\mathrm{L}(\theta)$:

$$\mathrm{L}(\theta) = \sum_{i=1}^{n} \left(\mathrm{f}(\mathrm{x_i}, \theta) - \mathrm{y_i}\right)^2 = \sum_{i=1}^{n} \mathrm{r_i}^2(\theta)$$

Avec $r_i^2(\theta)$ l'écart entre la mesure yi et la prédiction du modèle (résidu). La fonction de coût $\mathrm{L}(\theta)$ utilisée ici est une mesure du carré de la distance entre les données expérimentales et le modèle théorique qui prédit ces données. On divise la fonction coût par $2.n$ pour normaliser les valeurs. La méthode des moindres carrés vise à minimiser $\mathrm{L}(\theta)$, d'où son nom. Dans notre cas (deux dimensions) on prendra pour fonction $f(x) = ax + b$ et on notera $(\hat{a}, \hat{b})$ le couple de valeur minimisant la fonction perte, soit :

$$(\hat{a}, \hat{b}) = \min_{(a,b)\in\mathbb{R}^2} \frac{1}{2.n} \sum_{i=1}^{n} (a.x_i + b - y_i)^2 = \min_{(a,b)\in\mathbb{R}^2} J(a,b)$$

Avec J appelée fonction coût :

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

Si on pose maintenant :

$$Y = X.\theta \qquad \theta = \begin{bmatrix} a \\ b \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \cdots & \cdots \\ x_n & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}$$

On obtient pour expression de J :

$$J(\theta) = \frac{1}{2.n} . (X.\theta - Y)^2$$

Cours : descente de gradient

On veut minimiser $J(\theta)$, fonction convexe, donc on annule ses dérivées partielles avec l'utilisation du gradient comme algorithme de minimisation :

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{n}.X^T.(X.\theta - Y)$$

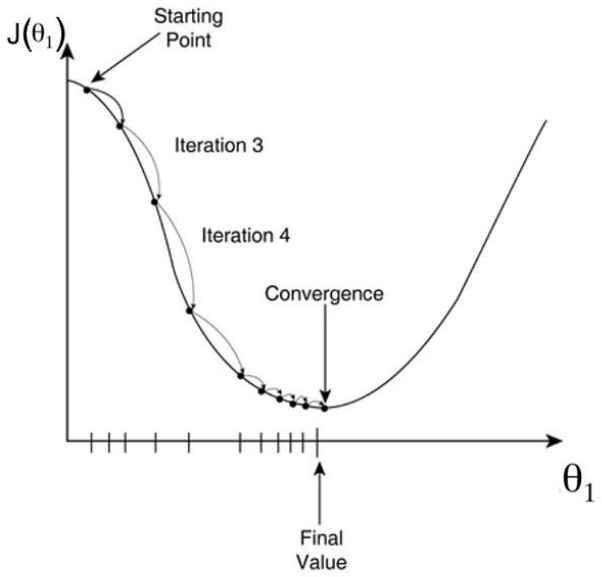
Cela donne un vecteur $\left[\frac{\partial J(a)}{\partial a} \quad \frac{\partial J(b)}{\partial b}\right]^T$ qui donnera notamment la direction de la pente de la fonction cout $\mathrm{L}(\theta)$. On obtient alors la fonction « descente de gradient » :

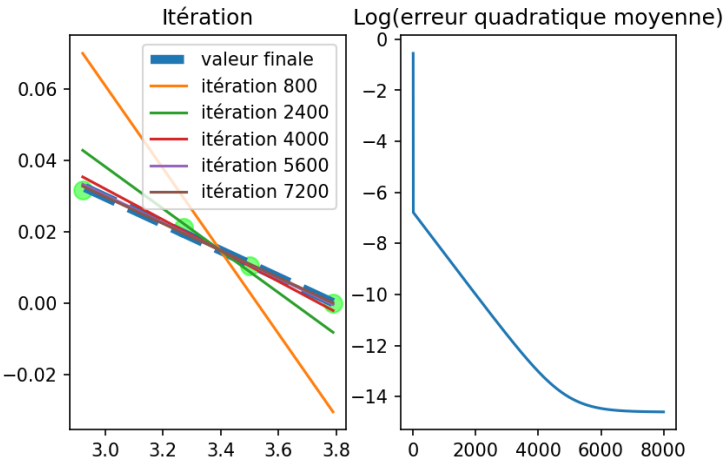
$$\theta = \theta - \alpha.\frac{\partial J(\theta)}{\partial \theta}$$

Deux paramètres interviennent : le nombre d'itérations et l'hyper paramètre α. Le premier nous fixera le nombre de fois où nous allons réitérer le calcul de la fonction descente de gradient, le second donne le « pas » de calcul (appelé également taux d'apprentissage ou encore « learning rate » pour les bilingues). Si le pas est trop important, le résultat final risque d'être approximatif et si le pas est trop faible, le programme risque de ne pas converger.

On donne ci-contre un exemple (dont le code est donné slide suivante) permettant à partir de 4 points (points vert sur le graphe de gauche) de déterminer la valeur de a et b minimisant la fonction coût (ici la fonction quadratique moyenne). On remarque qu'à partir de 6000 itérations, la fonction cout ne baisse quasiment plus : la fonction a converger.

Il n'y a pas d'exercice à traiter dans cette partie, cependant il faut savoir compléter et expliquer le code de la descente de gradient (slide suivante) et savoir utiliser le module `scikit-learn` (slide suivante également).





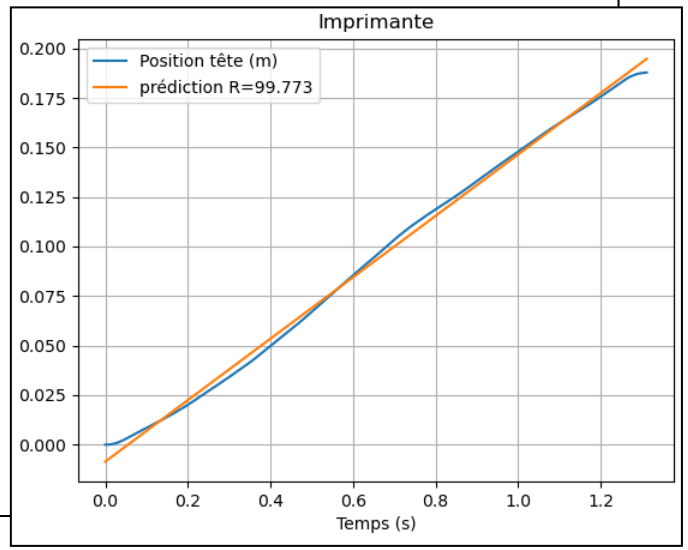
Exemple fonction descente gradient

```
def model(X, THETA): #fonction Y=X.THETA
    return X.dot(THETA)

def perte(X, y, THETA): #fonction perte = erreur quadratique moyenne
    n = len(y)
    return 1/(2*n) * np.sum((model(X, THETA) - y)**2)

def gradient(X, y, THETA):
    n = len(y)
    return 1/n * X.T.dot(model(X, THETA) - y)

def descente_gradient(X, y, THETA, alpha, nb_iterations):
    evolution_perte = np.zeros(nb_iterations)
    evolution_perte_log = np.zeros(nb_iterations)
    THETA_liste=[]
    for i in range(0, nb_iterations):
        THETA_liste.append(THETA) #enregistre les différentes valeurs de theta
        THETA = THETA - alpha * gradient(X, y, THETA) # mise à jour du parametre theta
        evolution_perte[i] = perte(X, y, THETA)
        evolution_perte_log[i] = log(evolution_perte[i])
    return THETA, evolution_perte_log, THETA_liste
```

Exemple fonction régression linéaire avec `scikit-learn`

Le module `scikit-learn` de python est dédié à l'apprentissage et comprend les régressions linéaires, les knn, K-Moyennes, etc. Voici un exemple de ce que peut donner la méthode du gradient avec l'exemple de la tête d'imprimante.

```
import numpy as np;import matplotlib.pyplot as plt;
from sklearn.linear_model import LinearRegression # Chargement de la fonction de linéarisation
mesures = np.loadtxt('BO200imprimante.csv', delimiter=';', skiprows=1, dtype=str)
mesures = np.char.replace(mesures, ',', '.');
mesures = mesures.astype(float)
temps = mesures[:,0]; position = 0.001*mesures[:,1]

#données
x = Temps.reshape((-1, 1))
y = Position
#instancier modèle
model_linReg = LinearRegression()
#entrainement du modèle
model_linReg.fit(x, y)

#précision du modèle + transformation en caractères pour l'affichage
precision = model_linReg.score(x, y)
R=round((precision*100),3)
coefR=str(R)

#prédiction
prediction = model_linReg.predict(x)
plt.plot(temps, position, label="position tête m")
plt.plot(temps, prediction, label="prédiction R="+coefR)
plt.xlabel("Temps (s)")
plt.title("Position de la tête d'impression")
plt.grid(True);plt.legend();plt.show()
```



8.Kmean

40 minutes

Cours : K-mean (par l'exemple)

On souhaite installer 3 dépôts de bus dans le département afin de développer les transports en commun intra-départementaux. L'idée est simple : minimiser la distance entre les dépôts de bus et les principales agglomérations.

Vous disposez de la base de donnée gouvernementale (2020) des communes de plus de 1000 habitants dans le tableau « ardeche_sup_1000.csv » , comportant :

- Col 0 : le code postal,
- Col 1 : longitude en ° (coordonnée X), Col2 : latitude en° (coordonnée Y)
- Col 3 : nombre d'âmes.

On choisit la distance euclidienne : $$d = \sqrt{(X_i - X_{cj})^2 + (Y_i - Y_{cj})^2}$$

(X_i,Y_i) coordonnées ville $i \in \{1,94\}$ villes, (X_{cj},Y_{cj}) coordonnées centroïde $j \in \{1,3\}$.

Méthode

<u>Initialisation</u> : On choisit arbitrairement k centroïdes (k = 3) qui seront les centres de k clusters.

<u>Boucle :</u>

- Chaque ville est affectée dans le cluster du centroïde qui est le plus proche.
- Calcul des nouveaux centroïdes (barycentre) de chacun des clusters que l'on vient de former.
- On recommence jusqu'à ce qu'à ce qu'il y ait convergence : les centroïdes ne changent pas entre deux itérations. Prévoyez une sortie de boucle sur une autre condition que l'immobilité des centroïdes car la convergence n'est pas garantie.

Exercice 8 : K-mean

Pour afficher les emplacements de la carte, chargez le programme :

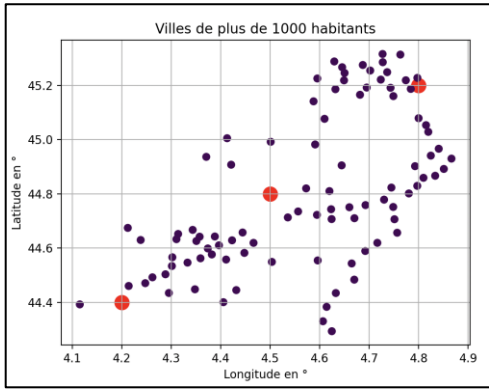
```
import numpy as np ; import matplotlib.pyplot as plt

# import direct connaissant les entêtes (N°dep,longitude,latitude,Nbre habitants)
ardeche = np.loadtxt('ardeche_sup_1000.csv', delimiter=';', skiprows=1, dtype=str)
ardeche = np.char.replace(ardeche, ',', '.'); ardeche=ardeche.astype(float)

# initialise la classe à 0 pour chaque ville et initialise arbitrairement 3 centroïde
colk=np.ones((len(ardeche[:,1]),1))
Cent=np.array([[4.2,44.4],[4.5,44.8],[4.8,45.2]])
def affichage(Dep,Clas,CdG,n): # fonction affichage des villes et centroïdes
            plt.figure(n)
            plt.scatter(CdG[:,0],CdG[:,1],c="red",s=150);plt.scatter(Dep[:,1],Dep[:,2],c=Clas);
            plt.xlabel("Longitude en °");plt.ylabel("Latitude en °");
            plt.title("Villes de plus de 1000 habitants");plt.grid(True);plt.show()
affichage(ardeche,colk,Cent,1) # affiche la carte et centroïdes initiaux
```

Question 1 : Ouvrez le fichier « kmoyenne-ardeche initial.py » et affichez la carte. Vous repérerez la plage de (X,Y) de la carte.

Question 2 : Définissez une fonction classe(Dep,Clas,CdG) qui affecte à chaque ville son centroïde le plus proche et retourne (Clas).

Dep : tableau des villes (94 lig x 4 col), Clas : tableau (1 col) des classes (c'est-à-dire « 1 » pour le centroïde 1, « 2 » pour le centroïde 2, etc.) de chaque ville, CdG tableau des centroïdes (3 lig x 2 col)

Question 3 : Définissez une fonction nouvcent(Dep,Clas) qui calcule le centroïde d'une classe (barycentre) et qui renvoie CdG.



Exercice 8 : K-mean

Question 4 : Ecrivez le programme qui permet de déterminer la position finale de chaque centroïde. La condition d'arrêt est que les centroïdes ne bouge plus, celle-ci peut s'écrire : np.array_equal(Centsuivant,Centprécédent). Il faut prévoir une sortie au cas où cela ne converge pas, par exemple en limitant le nombre d'itération. On souhaite voir une carte à chaque itération pour observer le mouvement des centroïdes et l'évolution des classes. Affichez les coordonnées finales des centroïdes (pas sur la carte mais pour les connaître).

Question 5 : Utilisez votre programme pour tester le nombre d'itérations et la position finale des centroïdes en fonction des positions initiales de ces derniers. Vous pouvez aussi tester des positions initiales de centroïdes de façon qu'il n'y ait pas un cluster vide au départ (ce qui risque de faire planter le code à cause d'une division par zéro).

Bonus : Proposez une amélioration qui permettra de tracer la distance moyenne des villes à leur centroïde en fonction des itérations : on devrait retrouver la méthode du coude.

Vous devriez obtenir le graphe ci-contre :

La correction est sur cahier de prépa.