MPSI – PCSI

# Sciences Industrielles de l'Ingénieur

## Séquence 9

## Analyser et modéliser pour décrire le comportement d'un système

Pierre-Louis CHIAMBARETTO – Alexandre GARCIA – Sébastien GERGADIER

# TD 1 – Télescope constitué de deux satellites[1]

**Q1.** Que représentent les valeurs 0 et 1 que peuvent prendre les variables internes **cap1** et **cap2** ?

---

La variable **cap1** correspond à l'état du capteur 1. Si il fonctionne correctement, la machine d'états reste dans l'état **actif1** et la variable **cap1** reste à 1. Si un défaut est détecté sur le capteur 1 (**def1**=1), on active l'état **Défaut1**. La variable **cap1** passe à 0. Puis on réactive indéfiniment l'état **Défaut1** si **def1** reste égale à 1. Idem pour la variable **cap2**.

---

**Q2.** Préciser le rôle des variables internes **i1** et **i2** ?

---

Les variables **i1** et **i2** sont des compteurs qui permettent une petite temporisation avant de remettre en service un capteur.

---

**Q3.** Donner l'expression de la variable **mesure** dans les cas suivants :
- Si les deux capteurs ne sont pas en défaut ;
- Si le capteur 1 présente un défaut ;
- Si le capteur 2 présente un défaut ;
- Si les deux capteurs sont en défaut.

---

- Si les deux capteurs ne sont pas en défaut : $mesure = \dfrac{capteur1 + capteur2}{2}$ ;
- Si le capteur 1 présente un défaut : $mesure = capteur2$ ;
- Si le capteur 2 présente un défaut : $mesure = capteur1$ ;
- Si les deux capteurs sont en défaut : $mesure = 1000$.

---

**Q4.** Quelle durée s'écoule entre l'instant où **def1** passe de la valeur 1 à la valeur 0 et l'instant où la mesure issue du capteur 1 est exploitée ?

---

Le pas de calcul est de 0.01 s. La variable **i1** doit passer de 1 à 10 en l'incrémentant à chaque pas de calcul. La machine d'états repassera à l'état **actif1** 0.09 s après le passage à 0 de la variable **def1**. Les mesures étant effectuée tous les centièmes de secondes, la mesure suivante sera 0.01 s plus tard.

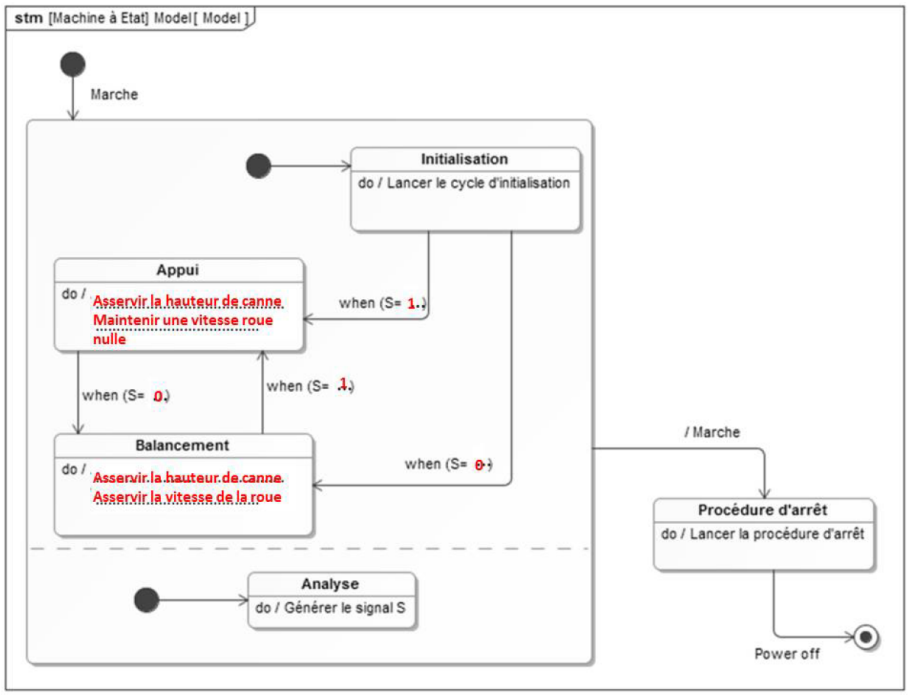
Au final, la mesure du capteur 1 sera exploitée 0.1 s après le passage à 0 de la variable **def1**.

---

1. Extrait X-ENS - PSI - 2015

# TD 2 – Canne motorisée[1]

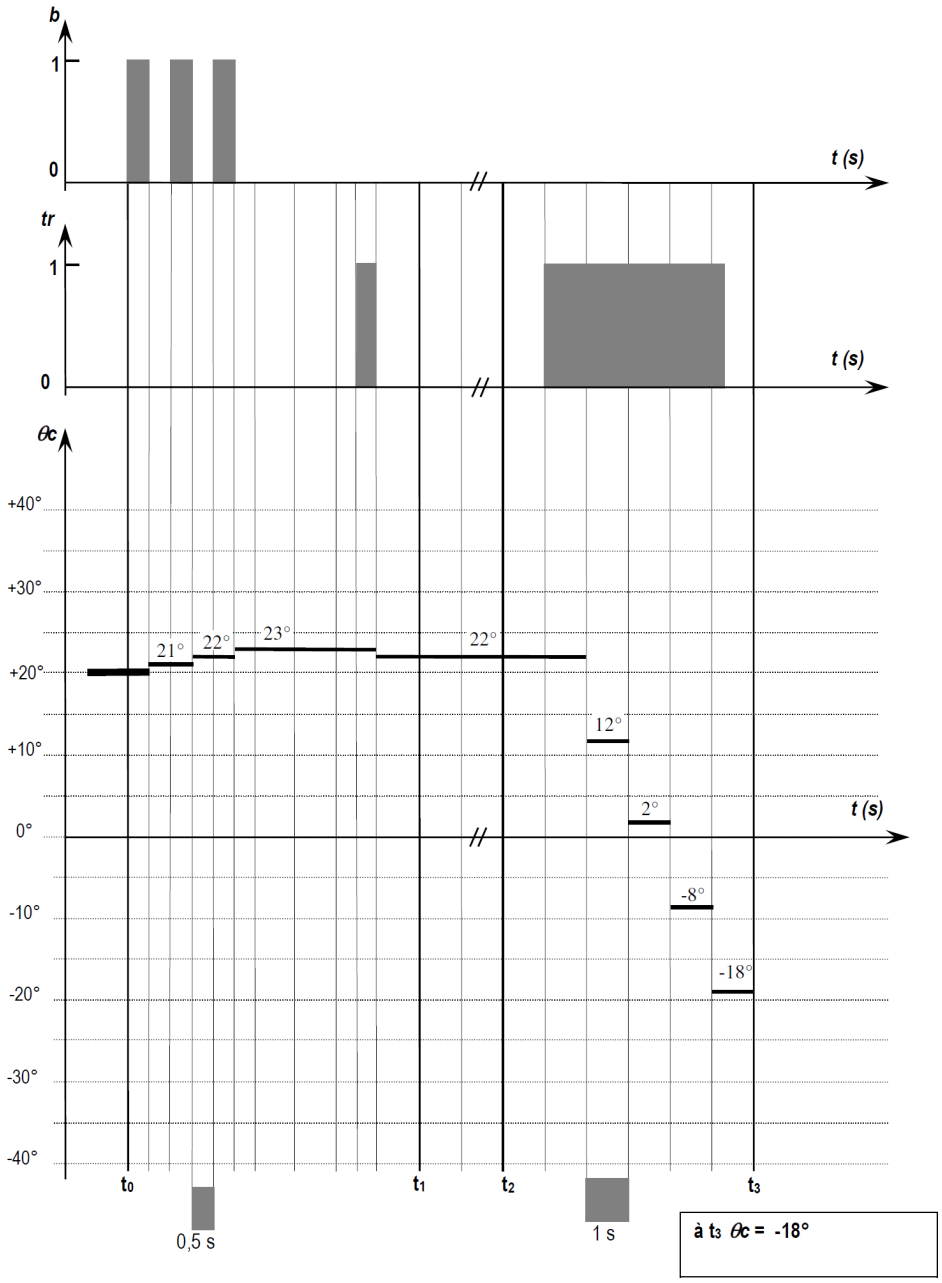
**Q1.** Compléter le diagramme d'états fourni en page suivante afin de gérer les asservissements en phase de balancement et d'appui. Pour cela, préciser les valeurs (0 ou 1) de la variable $S$ et les activités associées aux états des phases d'appui et de balancement.




1. Extrait CCINP - PSI - 2018

# TD 3 – Quille pendulaire[1]

**Q1.** À l'instant $t_0$, la situation du graphe de commande est telle que les états Attente (Attente consigne, attente Incréments et Attente_hydrau) soient actifs et la quille est alors inclinée de $+20°$. Le navigateur donne une série d'impulsions sur $b$ et $tr$ conformément au chronogramme donné sur la figure **??**. En analysant le modèle de commande, compléter le graphe des évolutions temporelles de la consigne angulaire $\theta_C$ donné sur le document réponse jusqu'à l'instant $t_3$ et donner la valeur obtenue pour $\theta_C$, et ce, sans se préoccuper de la façon dont la partie opérative réagit à cette consigne.




1. Extrait CCMP - PSI - 2014

**Q2.** La situation de départ est telle que les états Attente (Attente consigne, attente Incréments et Attente_hydrau) soient actifs et la quille est alors inclinée de +40°. Le navigateur donne la consigne de virement de bord, *vir*.

— Donner la liste des états successivement actifs présentées par le modèle de commande jusqu'au retour dans la situation identique à celle de départ. La situation initiale est notée (Attente consigne, attente Incréments, Attente_hydrau) ;

— En considérant que la chaîne de commande de la quille est précise, donner la valeur angulaire que représente $mes(\theta)$ en fin de ce cycle.

---

— (Attente consigne, attente Incréments, Attente_hydrau) → (Virement, attente Incréments, Déblocage) → (Virement, attente Incréments, Attente retour de quille) → (Virement, attente Incréments, Asservissement) → (Virement, attente Incréments, Blocage) → (Virement, attente Incréments, Fin_mouvements) → (Attente consigne, attente Incréments et Attente_hydrau).

— $mes(\theta)$ = -40°.

# TD 4 – Commande générique d'ouvrants pilotés d'automobiles[1]

**Q1.** Déterminer l'expression littérale de la résultante selon $\vec{z}$ de l'action mécanique du joint inférieur sur la vitre au cours du déplacement de celle-ci.

---

On décompose un effort infinitésimal du joint inférieur sur la vitre par la somme d'un effort normal et d'un effort tangentiel :

$$\vec{R}_{\text{joint inférieur}\to\text{vitre}} \cdot \vec{z} = 2\int (\overrightarrow{dN} + \overrightarrow{dT}) \cdot \vec{z}$$

L'effort normal est une charge linéique suivant $\vec{x}$, l'effort tangentiel peut s'exprimer en utilisant le modèle de Coulomb au glissement. Il s'oppose au mouvement de la vitre ($\pm\vec{z}$) :

$$\overrightarrow{dN} = \pm p \cdot dl\,\vec{x} \qquad \overrightarrow{dT} = \pm 2f \cdot dN\,\vec{z} = \pm 2fp \cdot dl\,\vec{z}$$

$$\vec{R}_{\text{joint inférieur}\to\text{vitre}} \cdot \vec{z} = \pm 2 \int_0^L 2fpdl = \pm 2fpL$$

---

**Q2.** Représenter l'évolution au cours du temps de la résultante des efforts résistants selon $\vec{z}$ de l'ensemble des joints sur la vitre (2 joints verticaux de hauteur $H$ et un joint horizontal de longueur $L$). Donner les valeurs numériques minimale et maximale de cet effort.

---

Préalablement, il faut calculer l'effort global des joints (inférieur et latéraux) sur la vitre. Mais on a déjà fait celui du joint inférieur sur la vitre, il ne nous reste plus qu'à calculer l'effort global des joints latéraux sur la vitre.

On décompose un effort infinitésimal des joints latéraux sur la vitre par la somme d'un effort normal et d'un effort tangentiel :

$$\vec{R}_{\text{joints latéraux}\to\text{vitre}} \cdot \vec{z} = 2\int (\overrightarrow{dN} + \overrightarrow{dT}) \cdot \vec{z}$$

L'effort normal des joints latéraux est une charge linéique suivant $\vec{x}$, l'effort tangentiel peut s'exprimer en utilisant le modèle de Coulomb au glissement. Il s'oppose au mouvement de la vitre ($\pm\vec{z}$) :

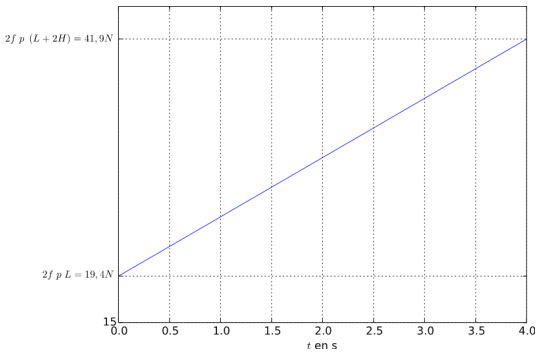
$$\overrightarrow{dN} = \pm p \cdot dz\,\vec{x} \qquad \overrightarrow{dT} = \pm f \cdot dN\,\vec{z} = \pm fp \cdot dz\,\vec{z}$$

En conséquence, en ne conservant que la composante suivant -$\vec{z}$ (vitre montante) (hypothèse : 4 contacts sur la hauteur de part et d'autre de la vitre, et à gauche et à droite de la vitre) :

$$\vec{R}_{\text{joints latéraux}\to\text{vitre}} \cdot \vec{z} = 4\int fpdz = 4fpz \quad \text{avec } z \in [0, H]$$

La résultante globale des actions mécaniques de frottement des joints sur la vitre s'exprime donc par :

$$\vec{R}_{\text{joints}\to\text{vitre}} \cdot \vec{z} = 2fpL + 4fpz \quad \text{avec } z \in [0, H]$$

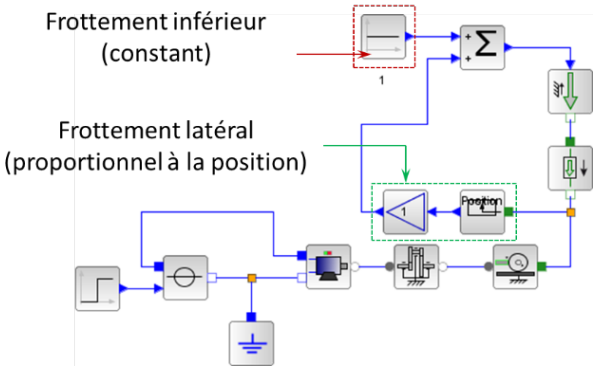



On obtient alors, puisque la vitesse de montée est supposée constante ($z = V \times t$) :

---

**Q3.** Indiquer, en entourant sur la figure **??**, l'action du joint horizontal inférieur (en rouge) et l'action des joints verticaux latéraux (en vert).

---




---

**Q4.** Compléter, en réalisant un tracé de couleur bleue, le schéma-blocs multiphysique de la figure **??** pour prendre en compte cet obstacle. Une palette composée de constituants standards est donnée sur la figure **??**.

---

---

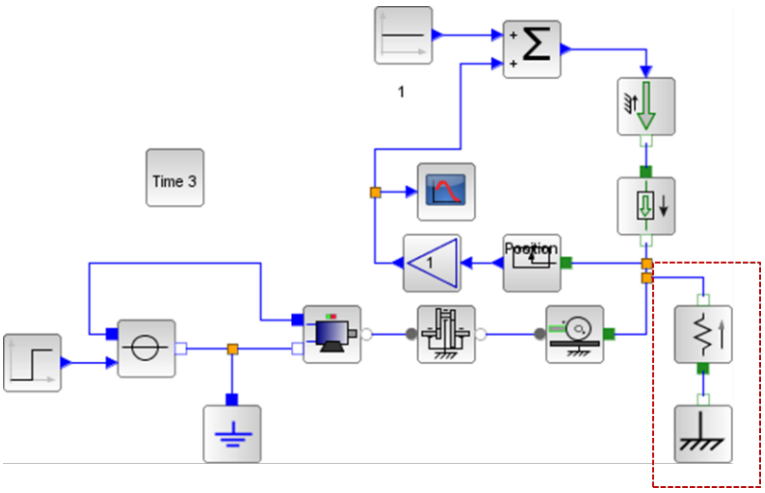
**Q5.** Dans chacune des situations, relever le temps au bout duquel la vitre atteint la position maximale définie dans le diagramme des exigences. Commenter l'influence de l'effort résistant sur la vitesse en régime permanent et en régime transitoire. Justifier votre choix entre un modèle sans effort résistant et un modèle avec prise en compte de l'effort résistant.

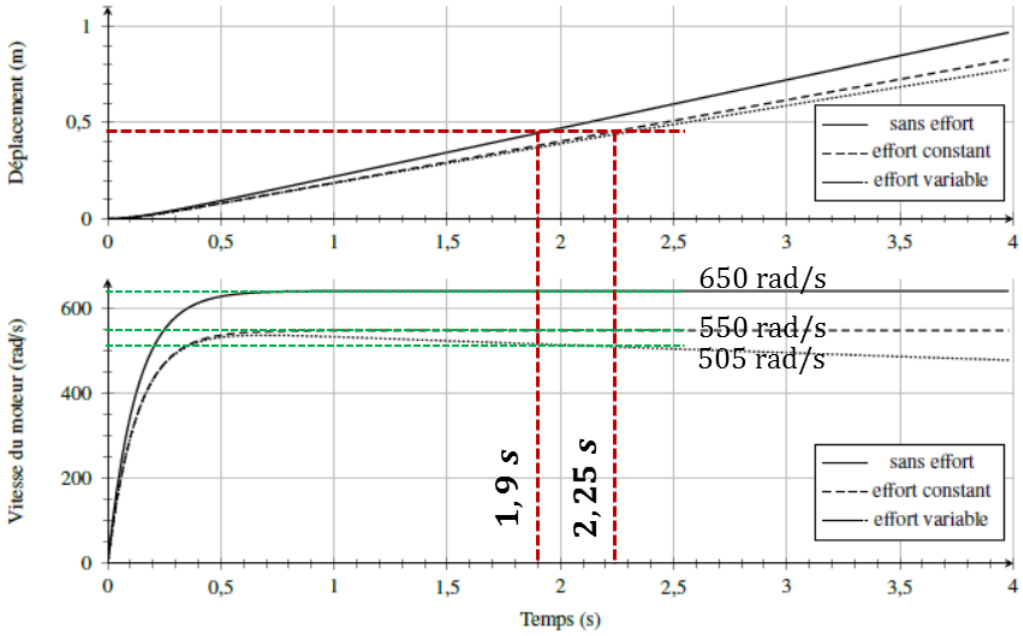
---

Dans le cas où il n'y a pas d'effort résistant, le temps de déplacement est de 1.9 s, la vitesse atteinte par le moteur en régime permanent est de 650 rad/s.

Dans le cas où il y a un effort résistant, le temps de déplacement est de 2.25 s, la vitesse atteinte par le moteur en régime permanent est de 550 rad/s (effort constant).

1. Extrait CCINP - PSI - 2017




Dans les deux cas la durée de fermeture est inférieure à celle souhaitée dans le cahier des charges ce qui est donc satisfaisant.

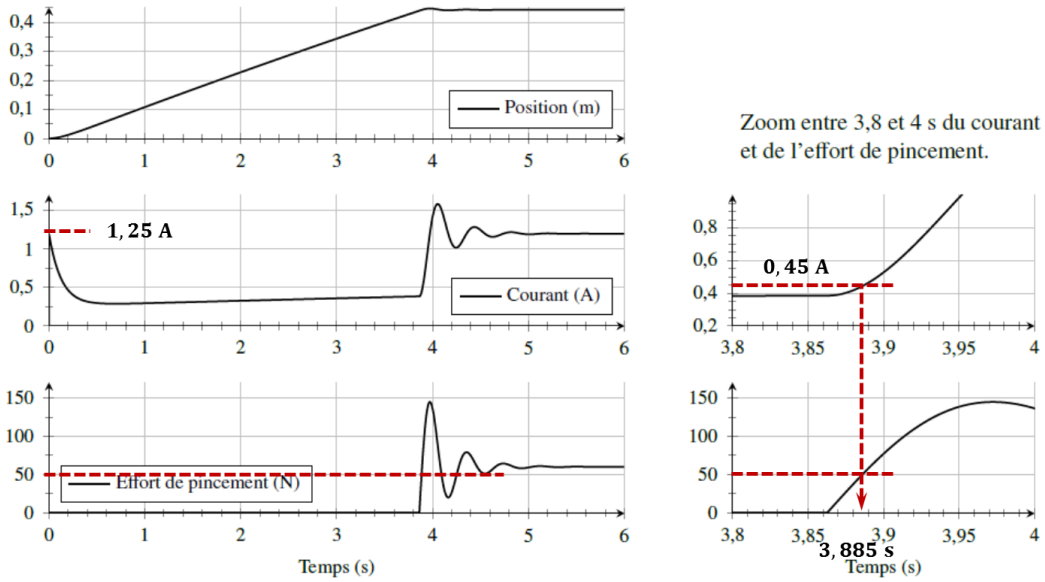
Dans les modèles avec prise en compte des frottements la vitesse de rotation du moteur en régime permanent est bien inférieure à celle obtenue sans tenir compte du frottement. Dans le cas d'un effort constant la vitesse atteinte en régime permanent est d'environ 465 rad/s et est constante.

Dans le cas d'un effort variable et à la fin du régime permanent, la vitesse de rotation du moteur est la même que dans le cas précédent puis diminue linéairement au cours du temps. La prise en compte du frottement avec un effort variable dans la modélisation présente une influence non négligeable et il est préférable d'en tenir compte.

---

**Q6.** Déterminer l'intervalle de temps où l'effort est inférieur à la force maximale admissible donnée par la législation (diagramme des exigences de la figure 2). En déduire la variation de courant sur cet intervalle et la comparer à celle obtenue au démarrage. Conclure sur la fiabilité de la mesure de courant pour repérer précisément un obstacle.

---




L'effort reste inférieur à celui de la législation pendant 3.885 s. À ce stade le courant consommé est de 0.45 A à comparer à 1.25 A lors du démarrage du moteur. La méthode de détection de l'effort de pincement par la mesure de courant ne semble donc pas adaptée puisque le courant en régime transitoire est plus important que celui mesuré lors d'un pincement. Avec une méthode de seuillage on n'arriverait donc pas à distinguer la phase de démarrage du moteur et le pincement.

---

**Q7.** Quels sont les intérêts d'utiliser deux capteurs à effet Hall placés en quadrature ?

---

L'utilisation de deux capteurs en quadrature a pour avantage d'une part de multiplier par 2 la résolution du capteur. D'autre part, l'utilisation de deux capteurs permet la détection du sens de rotation du rotor.

---

**Q8.** Déterminer le plus petit déplacement de la vitre en mm qu'il est possible de mesurer avec ce capteur.

---

On peut mesurer 1/8ème de tour soit 0.785 rad. D'après la valeur de $r$ donnée initialement, on a un plus petit déplacement mesurable de $\Delta z$=0.3 mm/impulsion.

---

**Q9.** En prenant une raideur d'obstacle $k$ = 20 N/mm correspondant à la dernière phalange de l'auriculaire, combien d'impulsions auront été comptées à partir du moment où la phalange commence à être écrasée jusqu'à ce que l'effort dans la phalange soit de 50 N (diagramme des exigences, figure 2) ? Commenter ce résultat.

---

Pour atteindre un effort résistant $F_r$=50 N, il faut que la vitre se déplace de la distance $d = \frac{F_r}{k}$ = 2.5 mm. Cela correspond à $N_{imp} = \frac{d}{\Delta z}$ = 8.3 impulsions. On pourra prendre 8 impulsions pour avoir une légère marge de sécurité.

---

**Q10.** En supposant que le moteur tourne parfaitement à la vitesse nominale de 300 rad/s,

déterminer le nombre d'impulsions moyen $N_{moy}$ mesuré à chaque période d'échantillonnage.

---

$T_{ech}$, la période d'échantillonnage vaut 10 ms ;
$\Delta\theta$, le pas angulaire par impulsion en rad/impulsion vaut $\Delta\theta = \dfrac{2\pi}{8}$ ;
$\Delta t_{imp}$, la durée d'une impulsion (en s/impulsion) vaut $\Delta t_{imp} = \dfrac{\Delta\theta}{\omega}$ ;
Le nombre d'impulsion moyen par durée d'échantillonnage est donné par $N_{moy} = \dfrac{T_{ech}}{\Delta t_{imp}}$.
On a alors $N_{moy} = \dfrac{8 \cdot \omega \cdot T_{ech}}{2\pi} \approx 3.82$ impulsions.

---

**Q11.** Déterminer les deux valeurs extrêmes de rotation du moteur en tours/min.

---

$$N_{moy}^{min} = 3 \text{ et } N_{moy}^{max} = 4$$

On trouve donc :

$$\omega_{min} = \frac{\pi N_{moy}^{min}}{4T_{ech}} = 235.6 \text{ rad/s}$$

$$\omega_{max} = \frac{\pi N_{moy}^{max}}{4T_{ech}} = 314.2 \text{ rad/s}$$

---

**Q12.** Conclure quant à la pertinence de l'utilisation de la variation de la vitesse pour obtenir un résultat fiable pour la détection. Au vu de la simulation de la figure 8, commenter également l'hypothèse de vitesse constante avant détection d'obstacle.

---

La variation de vitesse mesurée est assez importante. On obtient une erreur de mesure de :

$$\frac{\omega_{max} - \omega_{min}}{\omega} = \frac{\pi}{4.T_{ech}\omega} = 22.4\%$$

De plus, ces résultats ne sont valables qu'à vitesse constante alors que les résultats de la simulation dans le cas d'un effort variable montrent que la vitesse décroît linéairement avec une décélération égale à environ 6.25 rad/s$^2$. Sur une durée d'échantillonnage (égale à 10 ms) la vitesse décroit en théorie à 0.0625 rad/s ce qui reste faible devant une variation de vitesse due à la précision de $\omega_{max} - \omega_{min} = 78$ rad/s.

Cette méthode est donc peu satisfaisante.

---

**Q13.** Donner l'expression des deux conditions notées « transition 1 » et « transition 2 » permettant de passer de l'état montée à l'état arrêt directement.

---

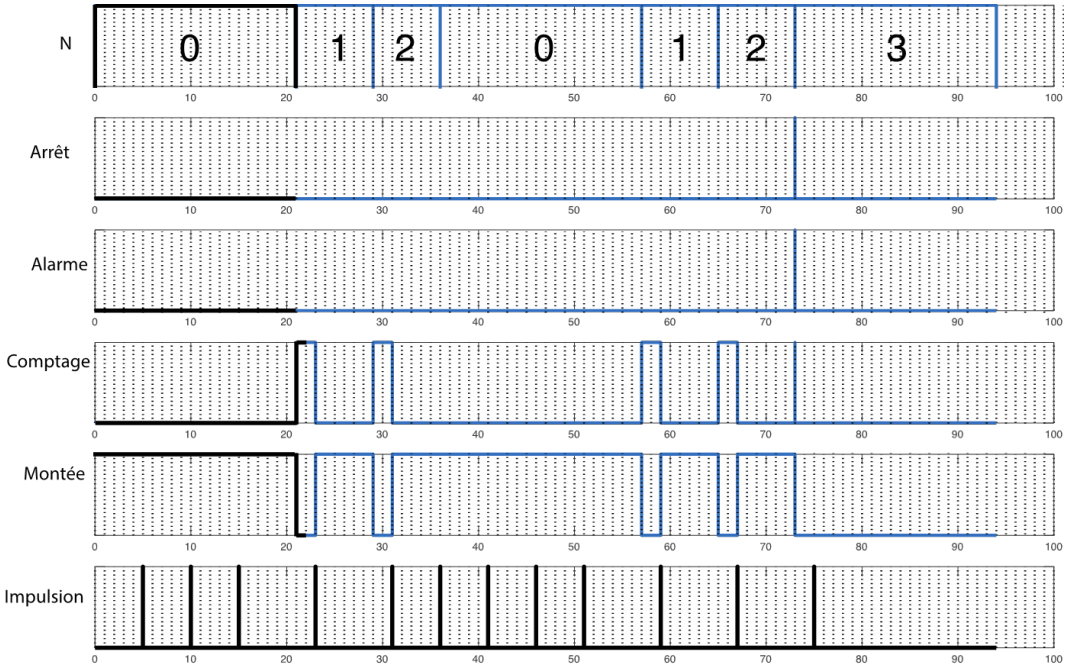
On passe de la montée à l'arrêt :

— quand on arrive en haut : `transition1 : fin course = 1` ;
— quand on ré-appuie sur le bouton haut : `bouton haut = 1` ;

---

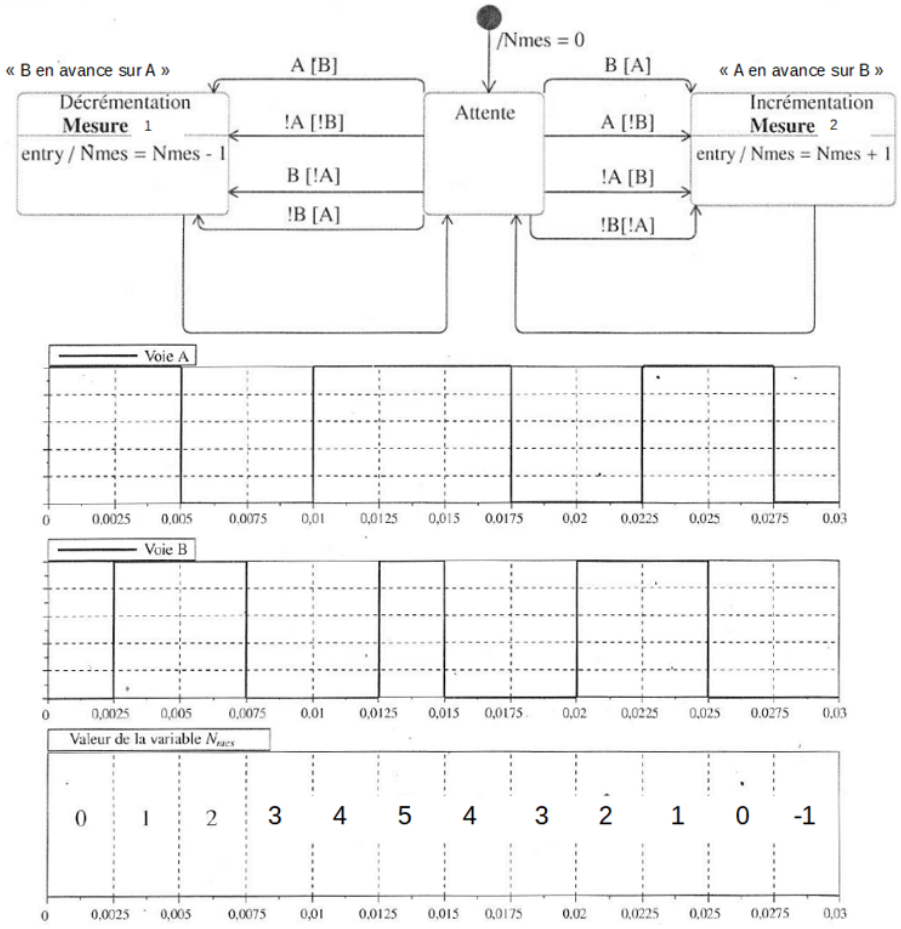
**Q14.** Compléter le chronogramme de la figure **??** en indiquant par des créneaux les durées pendant lesquelles un état est activé et l'évolution du contenu de la variable $N$. La durée de l'alarme et de l'arrêt est supposée très faible et sera représentée par un Dirac (une impulsion).

Dans le chronogramme partiel proposé, on remarque que N est incrémenté (et donc l'état comptage est activé) après une impulsion et au-delà de 6 secondes. On en déduit donc que `t=prediction()` = 6 s.



# TD 5 – Gestion d'un codeur incrémental[1]

**Q1.** Compléter sur la figure **??**, le chronogramme donnant l'évolution de la valeur $N_{mes}$ renvoyée par le compteur. Indiquer sur le diagramme d'états (figure 5) à quel numéro de mesure (mesures numérotées sur la figure 4) correspond chacun des états.




1. Extrait CCINP - PSI - 2019

# TD 6 – Régressions linéaires

**Q1.** Télécharger les fichiers `TD6_Reg_Lin.py` et `diabeteOK.csv` et compléter, dans la Zone 1 du fichier Python, le chemin absolu où se trouve le fichier de la base de données.

---

La variable **cap1**

---

**Q2.** Analyser le code Python, dans la Zone 2, relatif à la définition de la fonction `lecture_data`. Expliquer le rôle de cette fonction, et ce qu'elle renvoie.

---

---

**Q3.** En complétant la Zone 2, procéder à l'appel de la fonction `lecture_data` afin de stocker, dans une variable `X` **toutes** les caractéristiques des patients, et dans une variable `Y` les mesures quantitatives de la progression de la maladie.

---

---

**Q4.** Analyser le code fourni dans la Zone 3, et le compléter afin de :

- disposer dans la variable `X_train` des caractéristiques qui seront utilisées pour l'apprentissage ;
- disposer dans la variable `X_test` des caractéristiques qui seront utilisées pour le test ;
- disposer dans la variable `Y_train` des mesures quantitatives de la progression de la maladie qui seront utilisées pour l'apprentissage ;
- disposer dans la variable `Y_test` des mesures quantitatives de la progression de la maladie qui seront utilisées pour le test.

---

---

```
# Importation du module dédié à la régression linéaire
from sklearn import linear_model

# Création du modèle d'une régression linéaire
modele=linear_model.LinearRegression()
# Lancement de la phase d'apprentissage sur les données d'
   apprentissage data_train et Y_train
modele.fit(data_train,Y_train)
# Affichage des poids associés à chaque caractéristique (ai)
print(modele.regr.coef_) # Liste des poids ai avec i!=0
print(modele.intercept_) # Ordonnée à l'origine a0
# Affichage de l'ordonnée à l'origine
# Prévision à partir du modèle sur la donnée new_data
modele.predict(new_data)
```

**Q5.** Dans la Zone 4, compléter le code permettant de déclarer le modèle sous le nom `regr_diabete`, d'exécuter la phase d'apprentissage sur les données d'apprentissage (`X_train` et `Y_train`) et d'afficher les poids relatifs aux différentes caractéristiques et l'ordonnée à l'origine.

**Q6.** Compléter, dans la Zone 5, le code permettant de prévoir les mesures quantitatives de la progression de la maladie sur les données de test (`X_test`), notées `Y_pred`. Les afficher et les comparer aux valeurs des données de test `Y_test`.

La détermination des métriques permettant de caractériser les performances de la modélisation linéaire est possible en utilisant directement les fonctions associées au module `sklearn`.

```
# Importation des fonctions liées à l'évaluation des métriques
# Pour la détermination de la MSE
from sklearn.metrics import mean_squared_error
# Pour la détermination du coefficient de détermination R2
from sklearn.metrics import r2_score
```

**Q7.** Compléter, dans la Zone 6, le code permettant de déterminer et d'afficher les métriques RMSE (racine carrée normalisée de MSE) et coefficient de détermination, à partir de notre modèle de régression linéaire multivarié sur les 10 caractéristiques des patients.

**Q8.** Conclure quant à la satisfaction ou non du cahier des charges.

# Et le coût de la vie...

**Q1.** Télécharger le fichier `cost-of-living-2016.txt` et compléter, dans la Zone 1 du fichier `TD6_Reg_Lin.py` relative au cout de la vie, le chemin absolu où se trouve le fichier de la base de données.

---

---

**Q2.** Dans la Zone 2, et en vous inspirant de la définition de la fonction `lecture_data`, définir une fonction `lecture_data2` prenant en arguments :

- la variable `nom`, du type chaîne de caractères, correspondant au nom de la base de données ;
- la variable `liste_colonne_car`, correspondant à la liste des index des caractéristiques utiles de la base de données pour prévoir la grandeur de sortie ;
- la variable `liste_colonne_Y`, correspondant à la liste des index des grandeurs de sortie de la base de données.

Elle renvoie 2 tableaux, `X` et `Y`, du type tableau, contenant respectivement les caractéristiques des données et les grandeurs de sortie.

---

---

**Q3.** En complétant la Zone 2, procéder à l'appel de la fonction `lecture_data2` afin de stocker, dans une variable `X` les caractéristiques pertinentes, et dans une variable `Y` les valeurs de la grandeur `IndexCourses`.

---

---

**Q4.** En vous inspirant des questions 4, 5, 6 et 7 de l'exercice précédent sur la maladie du diabète, compléter, dans la Zone 3, les codes permettant de créer les modèles de prévision des grandeurs de sortie `IndexCourses`, `IndexLocation`, `IndexPrixRestaurant` et `IndexPouvAchatLocal` et d'évaluer leurs coefficients de détermination respectifs. On utilisera 80% des données pour l'apprentissage, et donc 20% pour le test.

---

$R^2$ IndexCourses : 0.8971289765195721
$R^2$ IndexLocation : 0.9602848296525663
$R^2$ IndexPrixRestaurant : 0.8717683576501172
$R^2$ IndexPouvAchatLocal : 0.5010198853435603

---

**Q5.** Conclure quant à la satisfaction du cahier des charges imposant un coefficient de détermination supérieur à 0.85.

---

Les prédictions sont globalement satisfaisantes, car elles sont supérieures à 0.85, sauf pour l'index du Pouvoir d'Achat Local

---

**Q6.** Pour les éventuels modèles ne respectant pas le cahier des charges, proposer (sans les mettre en œuvre) des solutions qui permettraient d'améliorer la modélisation de ces grandeurs.

---

La modélisation de la grandeur du pouvoir d'achat local est très insuffisante (coefficient de détermination de l'ordre de 0.5). Plusieurs pistes d'améliorations peuvent être envisagées :

- Ajouter des caractéristiques pertinentes pour la modélisation, et supprimer celles n'apportant que très peu d'informations ;
- Mettre en place une régression polynomiale multivariée.

---

# TD 7 – Classification de vins par algorithme des $k$-means

```
from sklearn.cluster import KMeans
model=KMeans(n_clusters=3) # On définit le modèle de
    classification et l'hyperparamètre (ici 3)
model.fit(X) # On entraine le modèle sur des données d'
    apprentissage X
model.predict(Xnew) # Utilisation du modèle pour faire la pré
    vision de Xnew
model.cluster_centers_ # Permet de renvoyer les coordonnées des k
     centroides
model.inertia_ # On peut évaluer l'efficacité du modèle
```

**Q1.** Quel type d'apprentissage doit être mis en œuvre ? Justifier votre réponse.

---

Apprentissage non-supervisé, car les données ne sont pas étiquetées. En effet, nous ne disposons par, pour chacune des 178 bouteilles, du cépage associé.

---

```
np.loadtxt(nom,delimiter,skiprows)
```

**Q2.** Ouvrir, avec le bloc-Notes (ou équivalent sous Mac), le fichier de données `wine.txt` et déterminer le séparateur (délimiteur) de colonne utilisé, et si la première ligne est celle d'une donnée ou celle des caractéristiques des bouteilles de vin. Dans la zone prévue à cet effet (Zone 1), compléter le code Python permettant de charger le fichier de données `wine.txt` dans la variable `X`.

---

Le séparateur est la virgule, et la première ligne est celle des noms associés aux caractéristiques des bouteilles de vin.

---

```
X=np.loadtxt('wine.txt',delimiter=',',skiprows=1)
```

**Q3.** Dans la zone 2, compléter le code permettant d'extraire les données relatives aux colonnes `Nonflavanoid_Phenols` et `Proline`, respectivement dans les variables `X1` et `X2`. Exécuter la Zone 2, et visualiser la figure 1. Est-il si évident que 3 cépages sont présents à partir de ces 2 critères ?

```
X1=X[:,4] # Colonne "Nonflavanoid_Phenols"
X2=X[:,7] # Colonne "Proline"
```

---

Non, il n'est pas évident d'identifier 3 cépages sur ces 2 critères. Heureusement, il y en a 8...

---

**Q4.** Soit une grandeur $x$ dont l'intervalle de variation est $[min, max]$. Déterminer l'équation de $x_{norm}$, en fonction de $s$, $min$ et $max$ afin que l'intervalle de variation de $x_{norm}$ soit l'intervalle

[0, 1]. Dans la Zone 3, compléter le code de la fonction `normalise(X)` qui retourne le tableau `X` dont les valeurs de chaque colonne sont comprises entre 0 et 1. Elle devra aussi renvoyer les listes `Mini` et `Maxi` contenant respectivement les minimas et maximas de chaque critère. Procéder à l'appel de cette fonction sur les données d'apprentissage, et vérifier le bon fonctionnement de la normalisation (par exemple, en traçant sur une figure 2, l'équivalent de la figure 1).

$$x_{norm} = \frac{x - min}{max - min}$$

```
def normalise(data):
    m=len(data) # Nombre de lignes du data_set
    n=len(data[0]) # Nombre de colonnes du data_Set
    data_norm=np.zeros((m,n)) # Création d'une matrice de même
   taille que Data_Set
    Mini=[] # Liste des minimas de chaque colonne
    Maxi=[] # Liste des maximas de chaque colonne
    for j in range(n):
        mini=min(data[:,j])
        maxi=max(data[:,j])
        Mini.append(mini)
        Maxi.append(maxi)
        for i in range(m):
            data_norm[i,j]=(data[i,j]-mini)/(maxi-mini)
    return data_norm,Mini,Maxi

Xnorm,Mini,Maxi=normalise(X)
X1=Xnorm[:,4] # Colonne "Nonflavanoid_Phenols"
X2=Xnorm[:,7] # Colonne "Proline"
plt.figure(2)
plt.grid()
plt.xlabel('Nonflavanoid_Phenols')
plt.ylabel('Proline')
plt.title('Répartition des données normalisées')
plt.scatter(X1,X2,c='black')
plt.show()
```

**Q5.** Compléter, dans la Zone 4, le code permettant de définir le modèle de classification avec $k$=3 (sous le nom `classifieur_wine`), d'exécuter l'apprentissage et d'afficher les coordonnées des $k$ centroïdes.

```
# Création du modèle
classifieur_wine=KMeans(n_clusters=3)
# Lancement de la phase d'apprentissage
classifieur_wine.fit(Xnorm)
# Affichage des 3 centroides
print(classifieur_wine.cluster_centers_)
```

Renvoie : `[[1.29298387e+01 2.50403226e+00 2.11112903e+00 1.58403226e+00 3.88387097e-01 5.65032258e+00 8.83967742e-01 7.28338710e+02] [1.38044681e+01 1.88340426e+00 2.86723404e`

```
3.01425532e+00 2.85319149e-01 5.70255319e+00 1.07829787e+00 1.19514894e+03] [1.25166667e
2.49420290e+00 2.07072464e+00 1.75840580e+00 3.90144928e-01 4.08695652e+00 9.41159420e-0
4.58231884e+02]]
```

Les valeurs que vous obtenez peuvent être différentes, car l'initialisation des centroïdes est aléatoire.

---

**Q6.** Procéder à la prévision de la classe d'appartenance de la nouvelle donnée `New_Data=np.array([[13.4],[1.8],[3.5],[2.5],[0.15],[5],[1.01],[890]])`, et afficher la classe d'appartenance.

```
New_Data=np.array
   ([[13.4],[1.8],[3.5],[2.5],[0.15],[5],[1.01],[890]])
New_Data_Norm=np.array([(New_Data[i]-Mini[i])/(Maxi[i]-Mini[i])
   for i in range(len(New_Data))]).reshape(1,8)
print(classifieur_wine.predict(New_Data_Norm))
```

---

Renvoie : `[2]`. Mais vous pouvez avoir autre chose, encore en raison de l'initialisation aléatoire...

---

**Q7.** Compléter la Zone 5 afin de générer la liste des inerties pour $k \in [1, 10]$ et de tracer l'évolution de l'inertie en fonction de $k$ sur la figure 3.

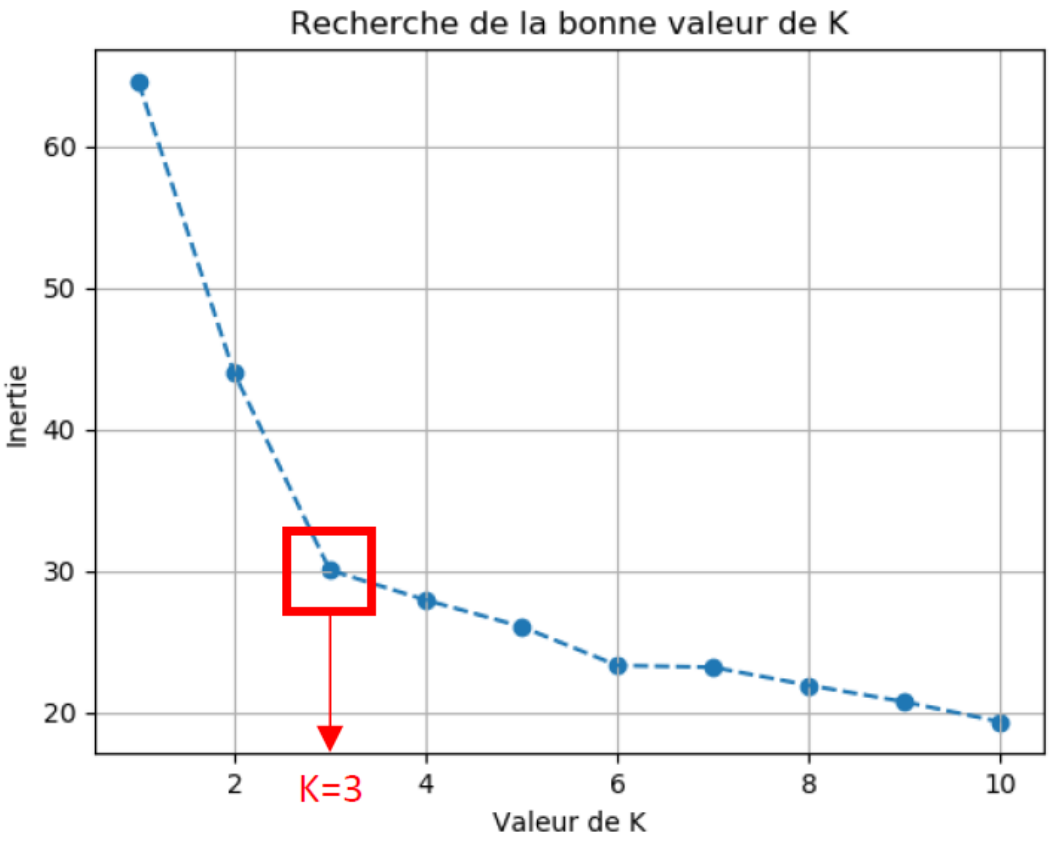
```
liste_k=[]
liste_inertie=[]
for k in range(1,11):
    classifieur_wine=KMeans(n_clusters=k)
    classifieur_wine.fit(Xnorm)
    liste_inertie.append(classifieur_wine.inertia_)
    liste_k.append(k)

plt.figure(3)
plt.plot(liste_k, liste_inertie,'--')
plt.title('Inertie du modèle en fonction de k')
plt.xlabel('Nombre de classes k')
plt.ylabel('Inertie')
plt.grid()
plt.show()
```

**Q8.** Identifier le coude sur cette figure, et valider le choix de $k$=3.

---

Le choix de $k$=3 semble donc tout à fait pertinent.

---


Recherche de la bonne valeur de K
Inertie
60
50
40
30
20
2
K=3
4
6
8
10
Valeur de K

# TD 8 – Algorithme $k$-NN et réseaux de neurones

**Q1.** Est-ce une régression ou une classification qu'il est nécessaire de mettre en place ? Est-ce un apprentissage supervisé ou non-supervisé ? Justifier vos réponses.

---

Classification et apprentissage supervisé, car on veut la classe d'appartenance et on bénéficie pour l'apprentissage des enregistrements étiquetés.

---

**Q2.** Dans la Zone 1 du fichier Python, compléter le code permettant déterminer et d'afficher le nombre d'enregistrement de chaque classe.

---

---

**Q3.** Peut-on considérer qu'il y a un biais dans la base de données vis-à-vis du critère retenu ci-dessus ?

---

`histo=[5244, 6695, 5075, 3626]`, donc pas de biais.

---

```
# Fonction pour la déclaration de l'objet classifieur par kNN
from sklearn.neighbors import KNeighborsClassifier
# Fonction pour séparation données d'apprentissage et de test
from sklearn.model_selection import train_test_split
# Fonction pour le tracé de la matrice de confusion
from sklearn.metrics import confusion_matrix
# Fonction pour l'évaluation des métriques (sensibilité et spé
    cificité)
from sklearn.metrics import classification_report
```

**Q4.** Analyser le code Python et indiquer le rôle des lignes de code Python de la Zone 2.

---

---

**Q5.** Exécuter le code de la Zone 2 et de la Zone 3. Que représente le contenu de la variable `Y_kNN_predit` ?

---

---

**Q6.** Exécuter le code de la Zone 4, et visualiser la figure 1 relative à la matrice de confusion pour la classification par les $k$ plus proches voisins (avec $k$=3). La classification des données de tests est-elle parfaite ? Justifier votre réponse.

**Q7.** Compléter le tableau ci-dessous en indiquant, pour chaque classe, le nombre d'enregistrement considéré comme Vrai Positif (VP), Faux Positif (FP), Vrai Négatif (VN) et Faux Négatif (FN).

| | Classe 0 | Classe 1 | Classe 2 | Classe 3 |
|---|---|---|---|---|
| VP | | | | |
| FP | | | | |
| VN | | | | |
| FN | | | | |

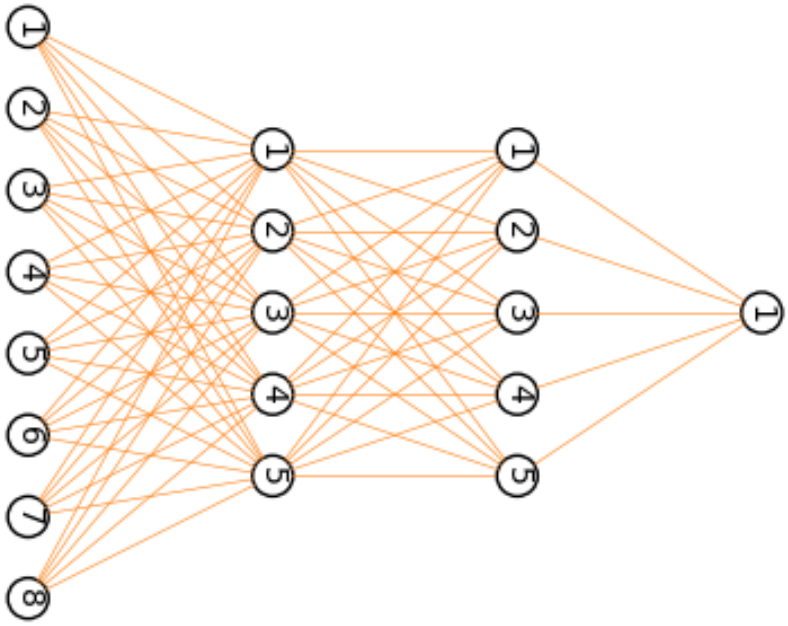
**Q8.** Déterminer pour chaque classe, la sensibilité et la sensibilité, en complétant le tableau ci-dessous.

| | Classe 0 | Classe 1 | Classe 2 | Classe 3 |
|---|---|---|---|---|
| Sensibilité | | | | |
| Spécificité | | | | |

**Q9.** Exécuter le Zone 5 du code Python, et visualiser les performances de la classification fournies par Python, à l'aide de l'instruction `classification_report`. Comparer et valider les valeurs calculées à la question 8.

**Q10.** Conclure sur les performances du classifieur par l'approche des $k$ plus proches voisins vis-à-vis du cahier des charges.

**Q11.** Identifier, sur la FIGURE **??**, la couche d'entrée, la couche de sortie, et les éventuelles couches cachées. Dans le cas de couches cachées, déterminer le nombre de neurones par couches cachées.



**Q12.** Après analyse de la FIGURE **??**, justifier la présence de 8 neurones sur la partie gauche. De même, justifier la présence d'un neurone sur la partie droite. Que représente les 8 entrées des neurones de gauche ? Que représente la sortie du neurone de droite ?

**Q13.** Analyser les arguments de la fonction permettant la déclaration d'un réseau de neurones (fonction `MLPClassifier`), et comparer les à l'analyse de la question 11. Quelle fonction d'activation est utilisée dans ce classifieur ?

On note `hidden_layer_sizes=(5,5)`, qui correspond à 2 couches cachées, composées chacune de 5 neurones, comme nous l'avions identifié à la Q11. La fonction d'activation est la fonction tangente hyperbolique (`activation='tanh'`).

**Q14.** Exécuter le code de la Zone 7 permettant la prévision sur les données de test et de tracer la matrice de confusion. Comparer les performances de ce classifieur à celles attendues dans le cahier des charges et à celles du classifieur par l'algorithme du $k$-NN.

**Q15.** Modifier les paramètres du classifieur par réseau de neurones tels que décrit dans le tableau ci-dessous, et évaluer les performances en termes de classification (sensibilité moyenne – *recall avg* et spécificité moyenne – *precision avg*).

| | (5,5) | (10,10) | (10,10,10) | (100,100) |
|---|---|---|---|---|
| `relu` | recall avg :<br>*precision avg* : | *recall avg* :<br>*precision avg* : | *recall avg* :<br>*precision avg* : | *recall avg* :<br>*precision avg* : |
| `tanh` | *recall avg* :<br>*precision avg* : | *recall avg* :<br>*precision avg* : | *recall avg* :<br>*precision avg* : | *recall avg* :<br>*precision avg* : |
| `logistic` | *recall avg* :<br>*precision avg* : | *recall avg* :<br>*precision avg* : | *recall avg* :<br>*precision avg* : | *recall avg* :<br>*precision avg* : |

**Q16.** Conclure quant au(x) paramètre(s) du classifieur à réseau de neurones à adopter afin de satisfaire le cahier des charges. Les critères retenus seront bien évidemment les performances de classification et l'espace mémoire de stockage des différents poids et biais associés à chaque neurone.

**Q17.** Dresser une synthèse sur le classifieur à adopter afin de respecter le cahier des charges.

# TD 9 – Mise en évidence du phénomène de sur-apprentissage

```
# Importation du module dédié à la régression linéaire
from sklearn import linear_model
# Création du modèle d'une régression linéaire
modele=linear_model.LinearRegression()
# Lancement de la phase d'apprentissage sur les données d'
   apprentissage X et Y
modele.fit(X,Y)
# Prévision à partir du modèle sur la donnée new_data
modele.predict(new_data)

# Importation des fonctions liées à l'évaluation des métriques
# Pour la détermination de la MSE et du coefficient de dé
   termination R2
from sklearn.metrics import mean_squared_error,r2_score
# Affichage des métriques
print(mean_squared_error(Yreel,Ypredit))
print(r2_score(Yreel,Ypredit))
```

**Q1.** En vous inspirant des questions réalisées dans le TD6, construire, dans la Zone 2, le modèle de la régression linéaire monovariable, et afficher la MSE et le coefficient de détermination de cette modélisation. Conclure quant à la satisfaction du cahier des charges.

**Q2.** Afin de visualiser la courbe représentative du modèle linéaire, compléter la Zone 3 pour procéder au tracé.

**Q3.** Exécuter le code de la Zone 4, et observer le modèle obtenu par régression polynomiale de degré 4. Compléter cette Zone 4 afin de calculer et afficher la MSE et le coefficient de détermination $R^2$. Comparer ces valeurs à celles obtenues à la Q1, et au cahier des charges.

**Q4.** En vous inspirant fortement du code de la Zone 4, saisir dans la Zone 5 le code permettant de déterminer, de tracer la régression polynomiale monovariable de degré 15 et les 2 métriques (MSE et $R^2$). Comparer ces valeurs.

**Q5.** Qu'observez-vous sur la figure du tracé du polynôme de degré 15 ? Cela vous paraît-il intéressant ? Quel nom donne-t-on à ce phénomène ? Conclure quant au modèle à adopter pour modéliser au mieux le comportement du capteur.