Ingénierie Numérique et IA

TD

v1.1 Ⓟ

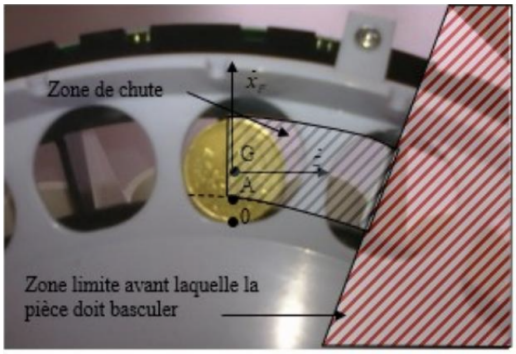
Institution Sainte Marie – 2 Rue de l'Abbaye - 92160 Antony

Table des matières

1 Résolution d'une équation différentielle

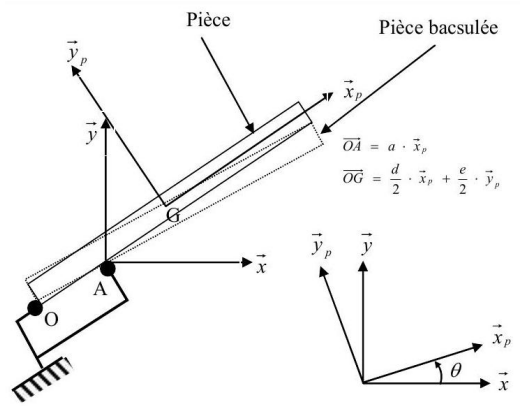
D'après un exercice basé sur la Trieuse de monnaie, sujet de TP classique, rédigé par Cedric Lusseau.

1.1 Présentation



La pièce est entrainée par le plateau tournant. Elle est inclinée de 40° par rapport à l'horizontale. Quand la pièce arrive dans la zone de chute, elle n'est plus maintenue dans sa partie haute. Celle-ci va alors basculer. Il est nécessaire que ce basculement s'effectue avant d'arriver dans la zone limite. Il est donc nécessaire de connaître le temps que la pièce va mettre à basculer.

1.2 Modélisation



Hypothèses :

- Le mouvement pendant le basculement est une rotation autour de $(A, \vec{z})$
- Pas de frottement en A
- On néglige les effets de la rotation du plateau. On partira alors de la position statique
- Les liaisons sont parfaites
- Pièce de masse m
- Épaisseur «e» négligeable
- R est le repère lié au socle

Dans toute cette partie nous considérons la pièce au début du basculement c'est-à-dire qu'il n'y a aucun effort en 0 .

Critère de Basculement

On va considérer que la pièce a basculé lorsque celle-ci a tourné d'un angle de telle sorte que l'extrémité de la pièce a parcourue une distance égale à son épaisseur (figure ci-dessus). Ainsi, il sera considéré que la pièce à suffisamment basculé lorsque celle a parcouru un angle de 0.3 rad .

La matrice d'inertie en G est de la forme

$$I(G,P) = \begin{bmatrix} A & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & A \end{bmatrix}_B$$

Question 1 Justifier la forme de la matrice

Question 2 Par application du PFD, avec la stratégie que vous voulez, établir l'équation de mouvement et la mettre sous la forme $\ddot{\theta} + C \cdot \cos\theta = 0$:

Question 3 À l'aide des données suivantes : calculer C

- $m = 5.74$ g
- $d = 22.25$ mm
- $a = 3$ mm
- $A = 7.104146875e-07$ kg $\cdot$ m^2 Rappel : $m\dfrac{d^2}{2}$ pour un disque.

1.3 Résolution numérique de l'équation

L'idée ici est de créer un vecteur $Y = \begin{pmatrix}\theta\\ \dot{\theta}\end{pmatrix}$ qui sous sa forme dérivée va être le vecteur $\dot{Y} = \begin{pmatrix}\dot{\theta}\\ \ddot{\theta}\end{pmatrix}$
Le problème peut alors se mettre sous la forme $\dot{Y} = f(Y,t)$. En effet nous avons $Y[0] = \theta$ et $Y[1] = \dot{\theta}$

Il est aussi possible d'écrire $\dot{Y} = \begin{pmatrix}\dot{\theta}\\ \ddot{\theta}\end{pmatrix} = \begin{pmatrix}\dot{\theta}\\ -C\cdot\cos(\theta)\end{pmatrix} = \begin{pmatrix}Y[1]\\ -C\cdot\cos(Y[0])\end{pmatrix}$ ce qui donne bien une expression de la forme $\dot{Y} = f(Y,t)$.

Par conséquent la fonction recherchée est $(Y,t) = \begin{pmatrix} Y[1] \\ -C\cdot\cos(Y[0]) \end{pmatrix}$.

Cette fonction va prendre en argument un vecteur Y et une liste t et va renvoyer le vecteur :

$$f(Y,t) = \begin{pmatrix} Y[1] \\ -C \cdot \cos(Y[0]) \end{pmatrix}$$

Question 4 Écrire la fonction $f(Y,t)$ en python

Question 5 La plage de temps est donnée par `t= np.linspace(0,0.05,1000)`. Donner le pas de temps h.

Question 6 Écrire la relation de récurrence qui permet de résoudre $\dot{Y} = f(Y,t)$. L'expression de Y à un instant t sera notée Y_i et l'expression de Y à un instant t + h sera notée Y_{i+1}.

Les conditions initiales sont données par Y = [[0.9], [0]]. Ainsi, le résultat est sous la forme de liste de liste. Y[0] représente la liste des positions et Y[1] la liste des vitesses de rotation.

Question 7 A l'aide d'une boucle for, calculer l'évolution de Y pour la plage t et tracer la courbe de l'évolution de la position.

Une pièce est un cylindre fin, un disque. il s'agit d'un solide de révolution généré autour de l'axe $(G,\vec{y_p})$ ce qui justifie la forme de la matrice.

On veut appliquer un TMD en A en projection autour de $\vec{z}$ car le problème est plan (dans le plan $(A,\vec{y},\vec{x})$) et que le contact ponctuel en A entre le support et la pièce ne génère aucun moment autour de $\vec{z}$ en A.

Ainsi, le moment du poids en A est égal au moment dynamique de la pièce calculé en A. Soit après calcul :

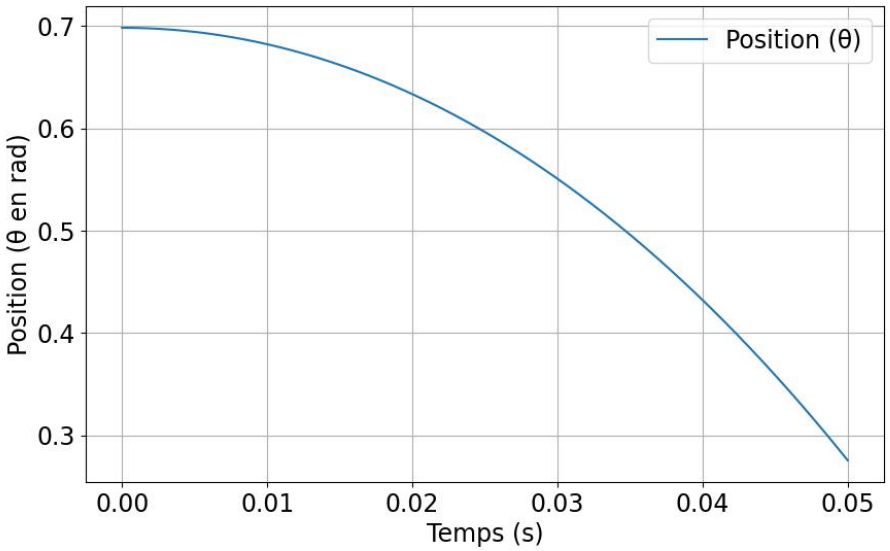
$$(A + m\cdot(\frac{d}{2} - a)^2)\ddot{\theta} = -(\frac{d}{2} - a)\cos\theta \cdot m \cdot g$$

Il vient :

$$C = \frac{(\frac{d}{2} - a)\cdot m \cdot g}{(A + m\cdot(\frac{d}{2} - a)^2)} \simeq 420 s^{-2}$$

```python
import numpy as np
import matplotlib.pyplot as plt
m = 5.74e-3  # kg
g = 9.81    # m.s-2
d = 22.25e-3  # m
a = 3e-3    # m
A = 7.104146875e-07  # kg.m^2
# Calcul de la constante C
C = (d/2 - a) * m * g / (A + m * (d/2 - a)**2)
# Plage de temps
t = np.linspace(0, 0.05, 1000)
h = t[1] - t[0]
# Fonction dérivée
def f(Y, t, C):
    theta = Y[0]
    theta_dot = Y[1]
    theta_dot_deriv = theta_dot
    theta_ddot_deriv = -C * np.cos(theta)
    return np.array([theta_dot_deriv, theta_ddot_deriv])
# Conditions initiales
Y = np.zeros((2, len(t)))
```

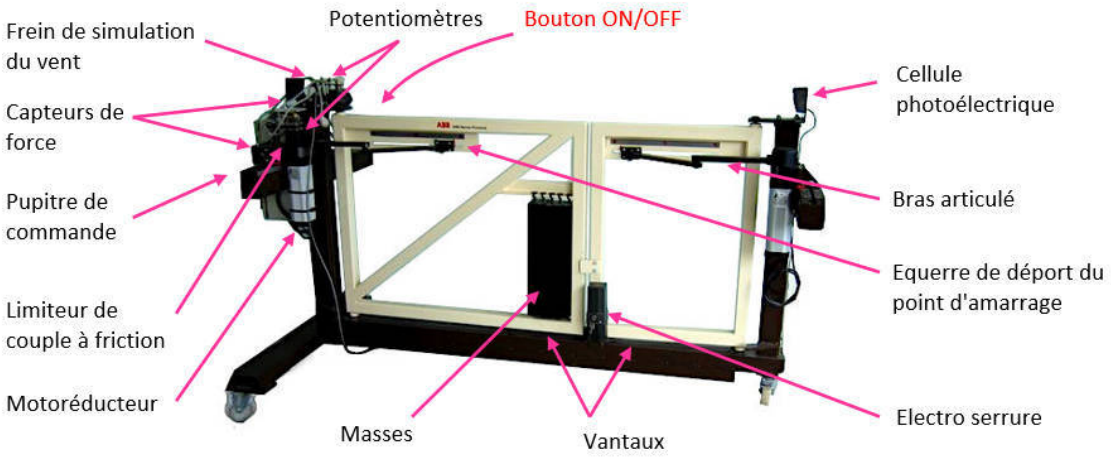
```python
Y[:, 0] = [40 * np.pi / 180, 0]
# Boucle de résolution numérique (méthode d'Euler explicite)
for i in range(len(t) - 1):
    Y[:, i + 1] = Y[:, i] + h * f(Y[:, i], t[i], C)
# Tracé de l'évolution de la position
plt.figure(figsize=(10, 6))
plt.plot(t, Y[0], label='Position')
plt.xlabel('Temps (s)')
plt.ylabel('Position (en rad)')
plt.legend()
plt.grid(True)
plt.savefig('position_evolution.png')
plt.show()
```



2 Traitement numérique de mesures

2.1 Présentation du problème

Un essai de lâcher sur le portail domoticc permet d'en déduire après traitement des mesures l'inertie du vantail autours de l'axe de rotation.



Pour cela il faut obtenir l'accélération à partir de la mesure de position avec une courbe très bruitées qu'il va falloir filtrer. Une fois le vantail « lâché », l'écriture du théorème du moment dynamique autour de l'axe de rotation donne $I_{(O,vantail)}\ddot{\theta} = -C_r$.

Le code qui permet de tracer la courbe est le suivant :

```
import matplotlib.pyplot as plt

# Lire les données du fichier
def lire_donnees(fichier):
    with open(fichier, 'r') as f:
        lignes = f.readlines()

    T = []
    Y = []
    for ligne in lignes:
        ligne = ligne.strip().replace(",", ".")
        temps, position = ligne.split("\t")
        T.append(float(temps))
        Y.append(float(position))

    return T, Y

# Chemin vers le fichier

fichier = r'G:\Mon Drive\CPGE\2024\Exercices\IA\Codes\Dom.txt'
T, Y = lire_donnees(fichier)

plt.clf()
plt.plot(T,Y,label="Position")
plt.xlabel("Temps en s")
plt.ylabel("Angle en deg")
plt.legend()
plt.grid()
plt.show()
```

Question 1 Écrire une fonction dérivation deriv(L, T) qui à partir de deux listes L et T renvoie la liste dérivée des éléments de L par rapport à la discrétisation de T. Appliquer cette fonction pour tracer l'évolution de l'accélération et de la vitesse angulaire.

```
def deriv(L, T):
    dL = []
    for i in range(1, len(L)):
        dL.append((L[i] - L[i-1]) / (T[i] - T[i-1]))
    dL.insert(0, 0)
    return dL
```

La courbe d'accélération obtenue est difficilement exploitable car la dérivation successive sur un pas a engendré un bruit. Il va donc falloir faire un traitement numérique de la courbe.

2.2 Réalisation du filtre numérique d'ordre 1.

Le concept est d'obtenir une liste de signal filtré `LF` à partir d'une liste de non filtrée `L` et d'une liste de temps `T`.

La relation est de la forme $\tau\frac{dLF(t)}{dt} + LF(t) = L(t)$.

Question 2 Écrire la relation de récurrence qui permet d'obtenir la liste `LF` en utilisant la méthode d'Euler explicite.

La relation de récurrence pour le filtre d'ordre 1 est donnée par :

$$LF_{n+1} = \frac{\tau}{\tau + \Delta t} LF_n + \frac{\Delta t}{\tau + \Delta t} L_n$$

Question 3 Écrire la fonction `filtre(L,tau, T)` qui prend en argument deux listes de même taille L et T ainsi qu'un réel tau et qui renvoie la liste L filtrée. Tracer la courbe. La valeur de τ sera prise égale au pas d'échantillonnage.

```
def filtre(L, tau, T):
    LF = [L[0]]
    for i in range(1, len(L)):
        dt = T[i] - T[i-1]
        LF_next = (tau * LF[-1] + dt * L[i]) / (tau + dt)
        LF.append(LF_next)
    return LF
```

2.3 Réalisation d'un filtre par moyenne glissante

La moyenne glissante est une moyenne qui au lieu d'être calculée sur l'ensemble des n valeurs d'un échantillonnage, est calculée tour à tour sur chaque sous-ensemble de N valeurs consécutives ($N \leqslant n$) ; le sous-ensemble utilisé pour calculer chaque moyenne « glisse » sur l'ensemble des données. On appelle N , l'ordre de la moyenne glissante.

Par exemple, le tableau suivant montre les moyennes mobiles simples sur 3 valeurs, pour une série de 7 mesures.

Mesures	2	3	5	8	8	7
Moyennes glissantes	néant	néant	$(2+3+5)/3$ 3,3333	$(3+5+8)/3$ 5,3333	$(5+8+8)/3$ 7	$(8+8+7)/3$ 7,6666

Une formule permettant de calculer une moyenne mobile est

$$\bar{x}_n = \frac{1}{N}\sum_{k=0}^{N-1} x_{n-k} \quad \text{ou} \quad \bar{x}_n = \bar{x}_{n-1} + \frac{x_n - x_{n-N}}{N}$$

Question 4 Écrire une fonction `filtre_mg(L,N)` qui prend en argument une liste L ainsi que l'ordre de la moyenne glissante N et qui retourne la liste L filtrée. Tester sur la courbe d'accélération et afficher le résultat obtenu pour différentes valeurs de N.

```python
def filtre_mg(L, N):
    LF = []
    for i in range(len(L)):
        start = max(0, i - N + 1)
        LF.append(sum(L[start:i+1]) / (i - start + 1))
    return LF
```

Question 5 Tracer le résultat du filtre d'ordre 1 et de la moyenne glissante sur le même graphique.

Utilisez le code suivant pour tracer les courbes :

```python
plt.figure(figsize=(12, 6))
plt.plot(T, acceleration, label="Accélération")
plt.plot(T, acceleration_filtree, label="Accélération filtrée (ordre 1)")
plt.plot(T, acceleration_mg, label="Accélération filtrée (moyenne glissante)")
plt.xlabel("Temps (s)")
plt.ylabel("Accélération (deg/s^2)")
plt.legend()
plt.grid()
plt.show()
```

Les lois de Coulomb indiquent que le couple résistant et donc la décélération devrait être constante. Même avec le filtre, il réside tout de même quelques oscillations. Il va falloir calculer la valeur moyenne et l'écart type sur la plage utile.

2.4 Calcul de la valeur moyenne et de l'écart type et réponse à la problématique

Le calcul de la moyenne arithmétique des nombres d'une liste est donné par $\overline{\mathrm{x}} = \frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{x_i}$.

Question 6 Écrire une fonction moyenne et calculer la valeur moyenne sur la zone de lâcher.

```python
def moyenne(L):
    return sum(L) / len(L)
```

Le calcul de la variance des nombres d'une liste est donné par $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$.

Le calcul de l'écart type des nombres d'une liste est donné par $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}$.

Question 7 Écrire et calculer l'écart type sur la zone de lâcher.

```
def ecart_type(L):
    m = moyenne(L)
    return (sum((x - m) ** 2 for x in L) / len(L)) ** 0.5
```

Question 8 Estimer un encadrement du moment d'inertie à partir des valeurs de moyenne et d'écart type.

- Moyenne de l'accélération : $\ddot{\theta}_{\text{moy}} = 165.02\,\text{rad/s}^2$
- Écart type de l'accélération : $\sigma_{\ddot{\theta}} = 180.08\,\text{rad/s}^2$
- Couple résistant : $C_r = 28\,\text{Nm}$

L'encadrement du moment d'inertie I est donné par les formules suivantes :

$$I_{\min} = \frac{-C_r}{\ddot{\theta}_{\text{moy}} + \sigma_{\ddot{\theta}}}$$

$$I_{\max} = \frac{-C_r}{\ddot{\theta}_{\text{moy}} - \sigma_{\ddot{\theta}}}$$

En utilisant les valeurs fournies, nous obtenons :

$$I_{\min} = \frac{-28}{165.02 + 180.08} = \frac{-28}{345.10} \approx -0.081\,\text{kg} \cdot \text{m}^2$$

$$I_{\max} = \frac{-28}{165.02 - 180.08} = \frac{-28}{-15.06} \approx 1.86\,\text{kg} \cdot \text{m}^2$$

L'encadrement du moment d'inertie I est donc :

$$-0.081\,\text{kg} \cdot \text{m}^2 \leq I \leq 1.86\,\text{kg} \cdot \text{m}^2$$

2.5 Calcul de l'accélération à partir de la régression de la courbe de vitesse

Comme la dérivation numérique a tendance à bruiter les courbes, une autre idée consiste à travailler sur la courbe de vitesse et pas d'accélération.

Question 9 Tracer la courbe de vitesse et repérer les indices correspondant à la zone utile.

Utilisez le code suivant pour tracer la courbe de vitesse :

```
plt.figure(figsize=(12, 6))
plt.plot(T, vitesse, label="Vitesse")
plt.xlabel("Temps (s)")
plt.ylabel("Vitesse (deg/s)")
plt.legend()
plt.grid()
plt.show()
```

La courbe de vitesse est clairement moins bruitée que la courbe d'accélération. Il reste à faire une régression linéaire sur la deuxième partie de la courbe.

La régression linéaire consiste à chercher les paramètres a et b définissant la droite $y = ax + b$ qui passe au plus près d'un ensemble de points (x_k, y_k). Les paramètres a et b sont déterminés par la méthode des moindres carrés qui consiste, dans le cas d'une régression linéaire, à minimiser la quantité

$$Q(a,b) = \sum_{k=1}^{N} (y_k - ax_k - b)^2$$

Le minimum de $Q(a,b)$ est obtenu lorsque ses dérivées par rapport à a et b sont nulles. Il faut donc résoudre le système à deux équations deux inconnues suivant

$$\begin{cases} \frac{\partial Q(a,b)}{\partial a} = 0 \\ \frac{\partial Q(a,b)}{\partial b} = 0 \end{cases} \Leftrightarrow \begin{cases} -2\sum_{k=1}^{N} x_k (y_k - ax_k - b) = 0 \\ -2\sum_{k=1}^{N} (y_k - ax_k - b) = 0 \end{cases}$$

Les solutions de ce système sont

$$a = \frac{N\sum_{k=1}^{N} x_k y_k - \sum_{k=1}^{N} x_k \sum_{k=1}^{N} y_k}{N\sum_{k=1}^{N} x_k^2 - \left(\sum_{k=1}^{N} x_k\right)^2} \quad b = \frac{\sum_{k=1}^{N} x_k^2 \sum_{k=1}^{N} y_k - \sum_{k=1}^{N} x_k \sum_{k=1}^{N} x_k y_k}{N\sum_{k=1}^{N} x_k^2 - \left(\sum_{k=1}^{N} x_k\right)^2}$$

Question 10 Créer une fonction `moindrecarre(X,Y)` qui prend en argument deux listes X et Y et renvoie les coefficients (a,b) de la droite des moindres carrés.

```python
def moindrecarre(X, Y):
    n = len(X)
    sumX = sum(X)
    sumY = sum(Y)
    sumXY = 0
    sumX2 = 0

    for i in range(n):
        sumXY += X[i] * Y[i]
        sumX2 += X[i] ** 2

    a = (n * sumXY - sumX * sumY) / (n * sumX2 - sumX ** 2)
    b = (sumX2 * sumY - sumX * sumXY) / (n * sumX2 - sumX ** 2)

    return a, b
```

Question 11 Appliquer la méthode des moindres carrés à la zone utile et en déduire le coefficient directeur puis l'accélération moyenne.

Appliquez la fonction `moindrecarre` à la zone utile pour obtenir les coefficients de la droite de régression linéaire. On trouve :

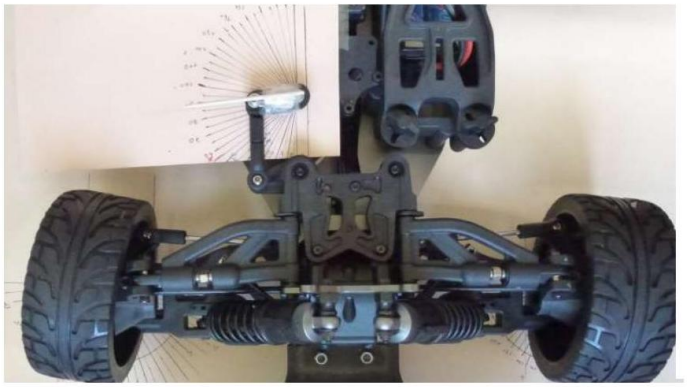
- a le coeff directeur : $a = 166.2$ deg/s^2
- une accélération moyenne : $acc = 165.9$ deg/s^2

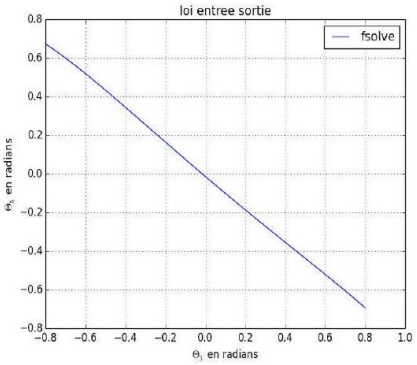
Autant dire que les deux méthodes donnent des résultats proches.

Recherche de zéro d'une équation

2.6 Présentation :

Le sujet porte sur la direction d'une voiture radio commandée et la relation qu'il va y avoir entre l'angle de commande et l'angle d'inclinaison d'une roue.





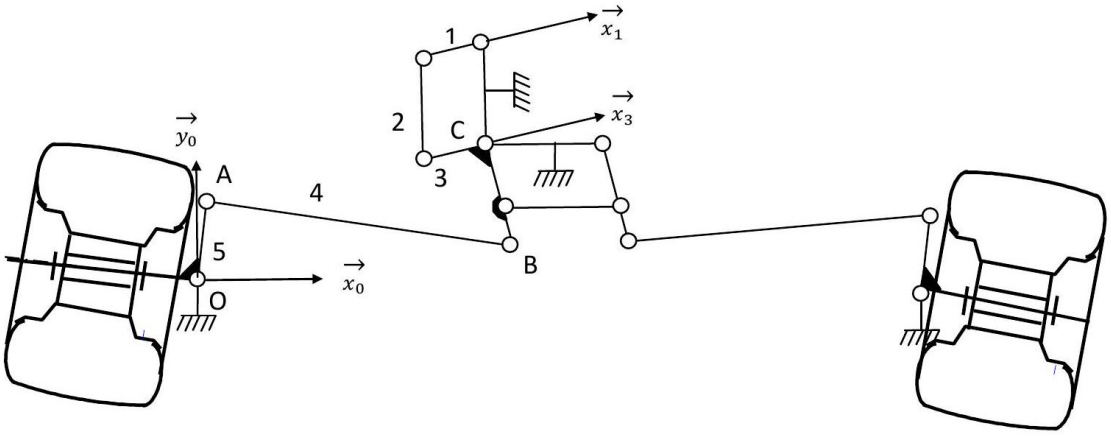
Plan :

- Écrire la fermeture géométrique et mettre le résultat sous la forme d'un problème de recherche de zéro.
- Résolution avec fsolve
- Résolution par la méthode de dichotomie
- Résolution par la méthode de Newton
- Résolution par combinaison des méthodes.

2.7 Recherche de la loi entrée sortie géométrique

Question 1 A partir de de la modélisation géométrique, montrer que la relation entre le mouvement d'entrée qui est l'angle du servomoteur θ_1 et l'angle d'une des roues θ_5 est

$$(a \cdot \sin\theta_5 + c + b \cdot \sin\theta_3)^2 + (-a \cdot \cos\theta_5 + d - b \cdot \cos\theta_3)^2 - L^2 = 0$$



Remarque

la géométrie des pièces $1, 2, 3, 0$ formant un parallélogramme déformable, $\theta_1 = \theta_3$

Solide	Repère associé	Paramètres géométriques
S_0	$R_0\,(0, \overrightarrow{x_0}, \overrightarrow{y_0}, \overrightarrow{z_0})$	$\overrightarrow{OC} = c \cdot \overrightarrow{x_0} + d \cdot \overrightarrow{y_0}$ avec $c = 75$ mm; $d = 40$ mm
S_5	$R_5\,(0, \overrightarrow{x_5}, \overrightarrow{y_5}, \overrightarrow{z_0})$	$\overrightarrow{OA} = a \cdot \overrightarrow{y_5}$ avec $a = 23$ mm $\theta_5 = (\overrightarrow{x_0}, \overrightarrow{x_5}) = (\overrightarrow{y_0}, \overrightarrow{y_5})$
S_3	$R_3\,(C, \overrightarrow{x_3}, \overrightarrow{y_3}, \overrightarrow{z_0})$	$\overrightarrow{CB} = -b \cdot \overrightarrow{y_3}$ avec $b = 32$ mm $\theta_3 = \left(\overrightarrow{0_0}, \overrightarrow{x_3}\right) = (\overrightarrow{y_0}, \overrightarrow{y_3})$
S_4		$\|\overrightarrow{AB}\| = L$ avec $L = 75$ mm

L'équation trouvée $(a \cdot \sin\theta_5 + c + b \cdot \sin\theta_3)^2 + (-a \cdot \cos\theta_5 + d - b \cdot \cos\theta_3)^2 - L^2 = 0$ est non linéaire. Afin de pouvoir tracer la courbe une résolution numérique est nécessaire. Sur le véhicule il est observé la variation de l'angle $\theta_3 \in [-0,8; 0,8]$ en rad. Il faut donc trouver les valeurs de θ_5 correspondantes.Le problème numérique est donc la recherche de zéro pour une fonction continue non linéaire. Pour cela, vous allez utiliser :

- la fonction fsolve de la bibliothèque numpy
- la méthode de dichotomie
- la méthode de Newton.

2.8 Utilisation de fsolve de la bibliothèque scipy de Numpy

```
import numpy as np
from scipy.optimize import fsolve
import matplotlib.pyplot as plt

# Définir la fonction dont on cherche la racine
def f(theta5, theta3):
    a, b, c, d, L = 23, 32, 75, 40, 75
    return (a * np.sin(theta5) + c + b * np.sin(theta3))**2 + \
           (-a * np.cos(theta5) + d - b * np.cos(theta3))**2 - L**2

# Valeurs de theta3
abscisse = np.linspace(-0.8, 0.8, 100)
ordonnee_scipy = []

# Résolution avec fsolve
for theta3 in abscisse:
    solution, = fsolve(f, 0, args=(theta3))
    ordonnee_scipy.append(solution)

plt.figure(figsize=(10, 6))
```

```
plt.plot(abscisse, ordonnee_scipy, label='fsolve', linestyle='-')
#plt.plot(abscisse, ordonnee_dichotomie, label='Dichotomie', linestyle='-')
#plt.plot(abscisse, ordonnee_newton, label='Newton', linestyle='-.')
plt.legend()
plt.grid(True)
plt.xlabel('Theta 3 en rad')
plt.ylabel('Theta 5 en rad')
plt.title('Comparaison des méthodes de recherche de zéro')
plt.show()
```

Question 2 Vérifier et exécuter le code.

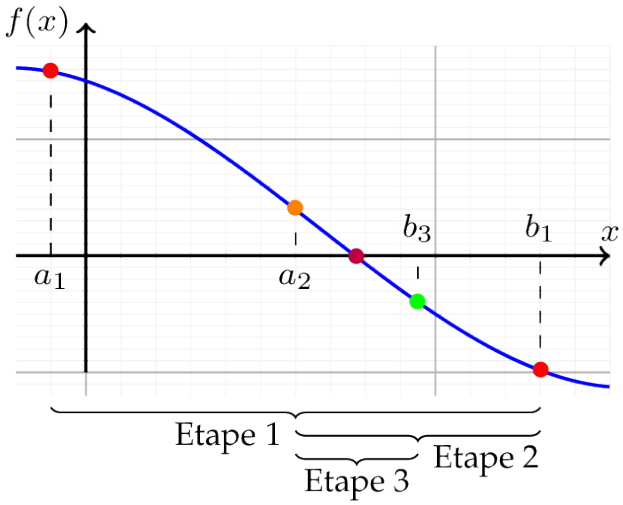
2.9 Méthode de dichotomie

Question 3 Ajouter au code précédent le tracé de la courbe obtenue à partir d'une recherche par dichotomie. La précision sera de 10^{-2}.

Rappel

Méthode par dichotomie
La méthode de dichotomie, également connue sous le nom de méthode de bissection, est un algorithme simple et efficace pour trouver une racine d'une fonction continue $f(x)$ sur un intervalle $[a, b]$. Cette méthode repose sur le théorème des valeurs intermédiaires, qui garantit l'existence d'une racine dans l'intervalle si $f(a)$ et $f(b)$ ont des signes opposés.

1. **Initialisation** :
   - Choisissez un intervalle initial $[a, b]$ tel que $f(a) \cdot f(b) < 0$. Cela garantit qu'il existe au moins une racine dans cet intervalle.
2. **Itération** :
   - Calculez le point milieu c de l'intervalle :
   $$c = \frac{a+b}{2}$$
   - Évaluez $f(c)$.
   - Si $f(c) = 0$, alors c est une racine et l'algorithme s'arrête.
   - Si $f(a) \cdot f(c) < 0$, alors la racine est dans l'intervalle $[a, c]$. Mettez à jour $b = c$.
   - Sinon, la racine est dans l'intervalle $[c, b]$. Mettez à jour $a = c$.
3. **Répétition** :
   - Répétez le processus jusqu'à ce que la largeur de l'intervalle $[a, b]$ soit inférieure à une tolérance donnée ou jusqu'à ce qu'un nombre maximum d'itérations soit atteint.



Avantages et inconvénients :

- **Simplicité** : L'algorithme est facile à comprendre et à implémenter.
- **Robustesse** : La méthode est robuste et garantit la convergence si la fonction est continue sur l'intervalle et change de signe.

- **Convergence lente** : Comparée à d'autres méthodes comme la méthode de Newton, la dichotomie peut nécessiter plus d'itérations pour atteindre une précision donnée.
- **Nécessite des signes opposés** : La méthode ne fonctionne que si la fonction change de signe sur l'intervalle initial.

La méthode de dichotomie est particulièrement utile pour les fonctions où les dérivées ne sont pas facilement calculables ou lorsque la simplicité de l'algorithme est préférée. Elle est souvent utilisée comme point de départ pour des méthodes de recherche de racines plus avancées.

2.10 Méthode de Newton

Question 4 Écrire la fonction qui permet de calculer la dérivée.

Question 5 Ajouter au code précédent le tracé de la courbe obtenue à partir d'une recherche par la méthode de Newton. On demande 2 itérations.

Rappel

Méthode de Newton

La méthode de Newton est un algorithme itératif utilisé pour trouver des approximations successives des racines (ou zéros) d'une fonction réelle $f\, x :\to f(x)$. Cette méthode est particulièrement efficace lorsque la fonction est dérivable et que l'on peut calculer sa dérivée.

Avantages et inconvénients :

- **Convergence rapide** : La méthode converge très rapidement lorsque l'estimation initiale est proche de la racine.
- **Précision** : Elle est très précise pour les fonctions dérivables et lisses.

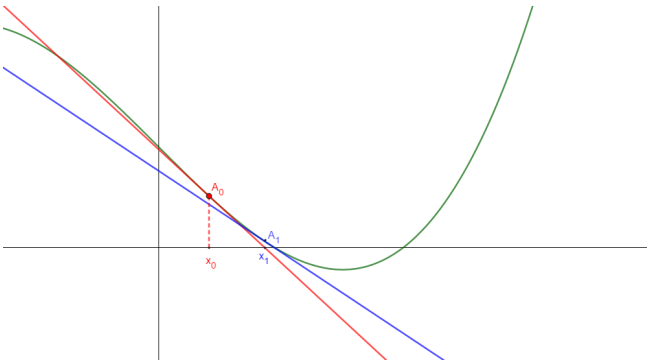
- **Dépendance à la dérivée** : La méthode nécessite le calcul de la dérivée de la fonction, ce qui peut être complexe ou impossible pour certaines fonctions.
- **Sensibilité à l'estimation initiale** : Si l'estimation initiale est trop éloignée de la racine, la méthode peut diverger ou converger vers une autre racine.
- **Problèmes avec les points stationnaires** : La méthode peut rencontrer des problèmes si la dérivée est nulle ou très proche de zéro.

La méthode de Newton est largement utilisée dans les domaines de l'ingénierie, de la physique et des mathématiques pour résoudre des équations non linéaires. Elle est particulièrement utile lorsque l'on a besoin d'une solution précise et que la fonction est bien définie et dérivable.

1. **Initialisation** :
   - Choisissez une estimation initiale x_0 proche de la racine.
2. **Itération** :
   - Pour chaque itération n, calculez la nouvelle approximation x_{n+1} en utilisant la formule :

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

   où $f'(x_n)$ est la dérivée de $f(x)$ évaluée en x_n.
3. **Répétition** :
   - Répétez le processus jusqu'à ce que la différence entre deux itérations successives soit inférieure à une tolérance donnée ou jusqu'à ce qu'un nombre maximum d'itérations soit atteint.



Question 6 Comparer les différentes méthodes et donner le coefficient de linéarité de la loi entrée - sortie.

```python
# Résolution par dichotomie

def dichotomie(f,a,b,eps=1e-2):
    if f(b)*f(a)>=0:return None
    while b-a > eps:
        c = (a+b)/2.0
        if f(a) * f(c) < 0: b = c
        else: a = c
```

```
    return a
ordonnee_dichotomie = []
for theta3 in abscisse:
    a, b = -0.8, 0.8
    solution = dichotomie(lambda x: f(x, theta3), a, b)
    ordonnee_dichotomie.append(solution)

# Résolution par la méthode de Newton
def deriv(f, x, theta3, h=1e-5):
    return (f(x + h, theta3) - f(x - h, theta3)) / (2 * h)

def newton(f, df, x0, theta3, tol=1e-5, max_iter=2):
    x = x0
    for _ in range(max_iter):
        fx = f(x, theta3)
        dfx = df(x, theta3)
        if abs(fx) < tol:
            return x
        if dfx == 0:
            raise ValueError("Dérivée nulle, la méthode de Newton échoue.")
        x -= fx / dfx
    return x

ordonnee_newton = []
for theta3 in abscisse:
    solution = newton(f, lambda x, theta3: deriv(f, x, theta3), 0, theta3)
    ordonnee_newton.append(solution)
```

3 Intelligence Artificielle

Version de V.Simoes d'après une source de A.Caignot, V. Crespel, D. Violeau, R. Marrou

3.1 Vrai ou Faux

Question 1 L'intelligence artificielle permet de reproduire le comportement du vivant de manière autonome.

Vrai. L'IA peut imiter certaines capacités humaines comme la reconnaissance d'images ou la prise de décision.

Question 2 L'apprentissage automatique consiste à réaliser des prédictions à partir de beaucoup de données d'entrée.

Vrai. Le machine learning utilise des données pour faire des prédictions ou des classifications.

Question 3 Il faut uniquement des données d'entrée pour qu'un algorithme supervisé puisse organiser les données en groupe ou prévoir des valeurs de sortie.

Faux. Un algorithme supervisé nécessite des données d'entrée et les valeurs de sortie associées pour l'entraînement.

Question 4 Un algorithme de classification consiste à classer les données d'entrée selon leur valeur.

Vrai. La classification est une tâche de machine learning où les données sont réparties en catégories.

Question 5 La phase d'inférence correspond à l'étape où l'on prédit des sorties pour de nouvelles entrées sur le modèle obtenu après la phase d'apprentissage.

Vrai. L'inférence est l'application du modèle entraîné à de nouvelles données.

Question 6 La régression linaire multivariée consiste à déterminer une valeur de sortie comme étant la somme pondérée de plusieurs attributs d'entrée.

Vrai. La régression linéaire multivariée modélise la relation entre plusieurs variables indépendantes et une variable dépendante.

Question 7 L'algorithme k-NN permet de traiter des problèmes de classifications uniquement.

Faux. k-NN peut être utilisé pour la classification et la régression.

Question 8 Les réseaux de neurones déterminent autant de poids et de biais qu'il y a de neurones dans le modèle.

Faux. Les réseaux de neurones déterminent des poids et biais pour chaque connexion entre les neurones, pas seulement pour chaque neurone.

3.2 Type d'apprentissage

Pour chacune des exemples ci-dessous, préciser si le problème est de type supervisé ou non supervisé et si c'est un problème de régression ou de classification.

Question 9 On cherche à déterminer à partir d'images de fleurs dans quelle famille elles appartiennent. On dispose pour cela de plusieurs images dont on connait la famille d'appartenance.

Supervisé, classification. On a des étiquettes (familles de fleurs) pour les images d'entraînement.

Question 10 A partir de données telles que les habitudes d'achat, la localisation géographique, les habitudes de déplacement d'un titulaire de carte, ainsi que les données en temps réel sur l'utilisation des cartes telles que ce qu'ils essaient d'acheter, où ils essaient de l'acheter et ce qu'ils ont acheté le même jour, des fabricants de cartes bancaires souhaitent repérer si une transaction est frauduleuse ou non.

Supervisé, classification. On cherche à classer les transactions comme frauduleuses ou non.

Question 11 Un algorithme trie automatiquement les photos en fonction des personnes qui les constituent.

Non supervisé, classification. L'algorithme doit découvrir les groupes sans étiquettes préalables.

Question 12 A partir de données de pluviométrie, de température, de type de sol associé à un rendement obtenu pour des parcelles agricoles, on cherche à prévoir le rendement d'une parcelle agricole quelconque.

Supervisé, régression. On prédit une valeur continue (rendement) à partir de données d'entrée.

Question 13 En fonction de sa taille, on cherche à savoir si une tumeur est bénigne ou non.

Supervisé, classification. On classe les tumeurs en bénignes ou malignes.

3.3 Choix d'algorithmes d'apprentissage automatique

Pour chacun des exemples ci-dessous, préciser quelles méthodes d'apprentissage il est possible d'utiliser pour résoudre le problème. Préciser la nature des données d'entrée et de sortie.

Question 14 On dispose d'enregistrements sonores et les textes associés. On souhaite déterminer le texte correspondant à un nouvel enregistrement sonore.

Méthodes : Réseaux de neurones récurrents (RNN). Données d'entrée : signaux audio. Données de sortie : texte transcrit.

Question 15 On dispose de données d'angles d'un robot 2 axes et les couples des 2 moteurs correspondant. On cherche à déterminer les couples pour d'autres positions.

Méthodes : Régression linéaire, réseaux de neurones. Données d'entrée : angles. Données de sortie : couples des moteurs.

Question 16 On dispose de données concernant des patients atteints de diabètes, leur âge, leur sexe, l'indice de masse corporelle, la moyenne de la pression dans le sang et 6 autres données relatives au sang. L'objectif est de prédire le taux d'évolution de la maladie après un une année.

Méthodes : Régression linéaire, réseaux de neurones. Données d'entrée : caractéristiques des patients. Données de sortie : taux d'évolution de la maladie.

Question 17 On cherche à trier des fruits selon leur maturité (mûr ou pas mûr) à partir de base de données de compositions de couleur RVB et du type de fruit.

Méthodes : k-NN, réseaux de neurones. Données d'entrée : compositions de couleur RVB. Données de sortie : état de maturité.

3.4 Structure d'un réseau de neurones

On considère un réseau de neurones sans couche cachée pour résoudre un problème constitué de 6 attributs et de 3 grandeurs de sortie à prédire qui peuvent prendre n'importe qu'elles valeurs réelles entre 0 et 1 . On dispose de 1000 échantillons pour lesquels on connait les valeurs de sortie.

Question 18 Indiquer s'il s'agit d'un problème de classification ou de régression.

Régression. Les valeurs de sortie sont continues.

Question 19 Combien de neurones faut-il utiliser ?

3 neurones de sortie, car il y a 3 grandeurs de sortie.

Question 20 Préciser le nombre d'inconnues de poids et biais nécessaires pour résoudre ce problème.

Poids : 6 (entrées) * 3 (sorties) = 18. Biais : 3 (un par sortie). Total : 21.

Question 21 Combien d'échantillons utiliserez-vous pour la phase d'apprentissage et pour la phase de validation ?

Par exemple, 800 pour l'apprentissage et 200 pour la validation.

Question 22 Quel type de fonction d'activation est pertinente compte-tenu des sorties souhaitées ?

Sigmoïde, car les sorties sont entre 0 et 1.

3.5 Remplissage d'un réseau de neurones

On cherche à prédire un facteur de qualité (compris entre 0 et 1) d'un alliage de magnésium pour différents facteurs de propriétés (lié à la vitesse de refroidissement, à la teneur en magnésium et à la teneur en aluminium). Pour cela, on utilise un réseau constitué de 3 entrées, d'une couche cachée de neurones et d'une donnée de sortie.

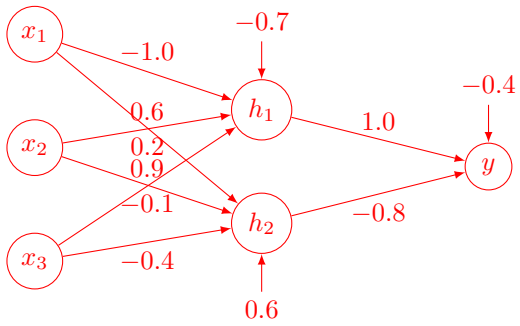
Nous avons comme dataset les données d'entrées qui correspondent aux valeurs $X = (2.1, 1.7, 1.3)$ et la donnée de sortie à la valeur $y = 0,71$.

La fonction d'activation utilisée sera la sigmoïde. On se donne initialement des valeurs aléatoires entre -1 et 1 pour les différents poids et biais. Par exemple :

$$\omega = \begin{pmatrix} -1.0 & 0.6 & 0.9 \\ 0.2 & -0.1 & -0.4 \end{pmatrix} \text{ et } b = \begin{pmatrix} -0.7 \\ 0.6 \end{pmatrix}. \text{ Ainsi que } \omega = \begin{pmatrix} 1 \\ -0.8 \end{pmatrix} \text{ et } b = -0.4 \text{ pour la couche de sortie.}$$

Question 23 Dessinez le réseau de neurone et compléter les différentes valeurs de sortie des neurones après une passe d'apprentissage.

Le dessin du réseau de neurones et les valeurs de sortie peuvent être calculées en utilisant les poids et biais initiaux et la fonction d'activation sigmoïde.



Question 24 Que vaut l'erreur quadratique après une passe d'apprentissage ?

L'erreur quadratique peut être calculée comme la différence au carré entre la sortie prédite et la sortie réelle.

Entrées pondérées pour la couche cachée :

$$z_1 = (-1.0 \cdot 2.1) + (0.6 \cdot 1.7) + (0.9 \cdot 1.3) - 0.7 = -0.61$$

$$z_2 = (0.2 \cdot 2.1) + (-0.1 \cdot 1.7) + (-0.4 \cdot 1.3) + 0.6 = 0.33$$

Sorties des neurones de la couche cachée après activation sigmoïde :

$$h_1 = \frac{1}{1 + e^{-(-0.61)}} = 0.352$$

$$h_2 = \frac{1}{1 + e^{-(0.33)}} = 0.582$$

Entrée pondérée pour la couche de sortie :

$$z_{\text{output}} = (1.0 \cdot 0.352) + (-0.8 \cdot 0.582) - 0.4 = -0.513$$

Sortie finale après activation sigmoïde :

$$y_{\text{output}} = \frac{1}{1 + e^{-(-0.513)}} = 0.374$$

Erreur quadratique

L'erreur quadratique après une passe d'apprentissage est calculée comme suit :

$$\text{Erreur quadratique} = (0.71 - 0.374)^2 = 0.113$$

Question 25 Quels sont les étapes suivantes de l'algorithme ?

Les étapes suivantes incluent la rétropropagation pour ajuster les poids et biais, puis une nouvelle passe d'apprentissage.

Après plusieurs itérations, on obtient les coefficients suivants :

$$\omega = \begin{pmatrix} -0.1 & 0.7 & -0.8 \\ 0.5 & -0.4 & -0.7 \end{pmatrix} \text{ et } b = \begin{pmatrix} -0.9 \\ 0.7 \end{pmatrix}. \text{ Ainsi que } \omega = \begin{pmatrix} 0.3 \\ 0.7 \end{pmatrix} \text{ et } b = 0.6 \text{ pour la couche de sortie.}$$

Question 26 Quel valeur obtient-on en sortie avec ces nouveaux poids et biais ?

La sortie peut être calculée en utilisant les nouveaux poids et biais et la fonction d'activation sigmoïde.

Après plusieurs itérations, les nouveaux poids et biais sont :

$$\omega = \begin{pmatrix} -0.1 & 0.7 & -0.8 \\ 0.5 & -0.4 & -0.7 \end{pmatrix}, \quad b = \begin{pmatrix} -0.9 \\ 0.7 \end{pmatrix}, \quad \omega = \begin{pmatrix} 0.3 \\ 0.7 \end{pmatrix}, \quad b = 0.6$$

On dispose d'une nouvelle donnée d'entrée test avec $X_{\text{test}} = (1.8, 1.5, 1.4)$ et la donnée de sortie à la valeur $y_{\text{test}} = 0.93$.

Question 27 Déterminer la valeur prédite par l'algorithme et conclure sur la justesse de ce dernier.

Pour la donnée test $X_{\text{test}} = (1.8, 1.5, 1.4)$, les calculs de la sortie prédite avec les nouveaux poids et biais sont les suivants :

1. **Entrées pondérées pour la couche cachée** :

$$z_1 = -1.15, \quad z_2 = 0.02$$

2. **Sorties des neurones de la couche cachée après activation sigmoïde** :

$$h_1 = 0.240, \quad h_2 = 0.505$$

3. **Entrée pondérée pour la couche de sortie** :

$$z_{\text{output}} = 1.026$$

4. **Sortie finale après activation sigmoïde** :

$$y_{\text{output}} = 0.736$$

Et enfin :

- **Sortie prédite** : 0.736
- **Sortie réelle** : 0.93

L'algorithme prédit une valeur de 0.736, ce qui est inférieur à la sortie réelle de 0.93. Cela indique que le modèle pourrait nécessiter un ajustement supplémentaire ou plus d'itérations d'entraînement pour améliorer sa précision.

Exercices en 3 parties adaptées du sujet CCinp - PSI - 2024

Contrôle du correcteur de facteur de puissance par Intelligence Artificielle

Présentation du problème

Dans cette partie, nous explorons l'utilisation d'un correcteur de facteur de puissance pour minimiser les pertes en ligne dans l'installation électrique d'un petit studio. Le schéma ci-dessous illustre le dispositif complet :

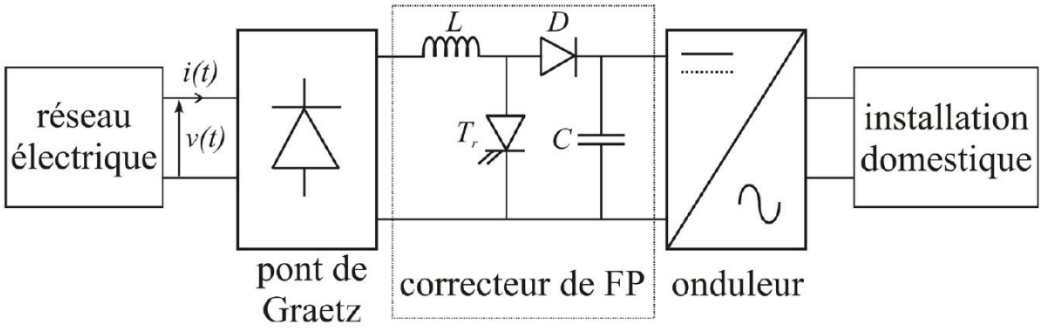


FIGURE 1 – Correcteur de facteur de puissance utilisé pour une installation domestique

Le dispositif régule la tension aux bornes du condensateur C pour maintenir un fonctionnement similaire à celui décrit dans la partie II. L'installation comprend :

- Un ensemble box Internet-télévision.
- Un système d'éclairage.
- Un radiateur électrique.
- Un chauffe-eau.

Question 1 Les différents éléments de cette installation sont-ils câblés en série ou en dérivation ? Justifier ce choix.

Les éléments sont généralement câblés en dérivation (parallèle) pour permettre à chaque appareil de fonctionner indépendamment des autres et de recevoir la tension nécessaire.

Question 2 Quelle est l'allure des courants absorbés par le radiateur électrique et le chauffe-eau lorsqu'ils sont connectés directement au réseau électrique ?

Les courants absorbés par le radiateur électrique et le chauffe-eau sont continus et sinusoïdaux, typiques des charges résistives.

Chaque élément de l'installation a trois modes de fonctionnement, représentés par un niveau x_i (0, 1, ou 2). Le tableau suivant montre la correspondance entre x_i et les modes de fonctionnement :

	box-TV x_1	éclairage x_2	radiateur x_3	chauffe-eau x_4
$x_i = 0$	éteint 0 W	éteint 0 W	éteint 0 W	éteint 0 W
$x_i = 1$	veille 100 W	puissance moitié 50 W	puissance moitié	maintient à température 50 W
$x_i = 2$	allumé 200 W	puissance maximale 100 W	puissance maximale	puissance maximale

FIGURE 2 – Modes de fonctionnement

Question 3 Donner un ordre de grandeur de la puissance électrique maximale consommée par un radiateur électrique et un chauffe-eau correspondant aux besoins d'un studio.

Un radiateur électrique peut consommer jusqu'à 2000 W, et un chauffe-eau peut aller jusqu'à 3000 W, selon la taille et les besoins spécifiques.

Question 4 Combien y a-t-il de combinaisons possibles entre les différents modes de fonctionnement de chacun des quatre éléments ?

Il y a $3^4 = 81$ combinaisons possibles, car chaque élément a 3 modes de fonctionnement.

Le paramètre Δ est ajusté pour minimiser les pertes totales. Le tableau suivant montre les valeurs optimales de Δ pour différentes combinaisons :

	box-TV x_1	éclairage x_2	radiateur x_3	chauffe-eau x_4	Δ_{opt}(V)
Données d'entraînement	0	0	0	1	-1
	0	0	1	1	-1
	0	0	2	0	-1
	0	0	0	2	-1
	0	0	1	2	-1
	0	1	0	0	0,18
	2	2	0	0	0,32
	2	2	2	0	0,49
	2	2	2	2	-1
Données test	0	0	1	0	-1
	0	0	2	1	-1
	0	0	2	2	-1
	1	0	0	0	0,22
	2	2	1	0	0,42

FIGURE 3 – Valeurs optimales du paramètre Δ pour quatorze combinaisons différentes

Question 5 Dans le tableau, expliquer pourquoi $\Delta_{\text{opt}} = -1$ pour toutes les combinaisons telles que $x_1 = 0$ et $x_2 = 0$.

Lorsque $x_1 = 0$ et $x_2 = 0$, les appareils consommant des courants impulsionnels sont éteints, donc il est préférable de désactiver le correcteur de facteur de puissance pour minimiser les pertes.

Question 6 Commenter le choix $\Delta_{\text{opt}} = -1$ pour la combinaison $(x_1, x_2, x_3, x_4) = (2, 2, 2, 2)$.

Pour cette combinaison, tous les appareils fonctionnent à pleine puissance, ce qui peut entraîner des pertes importantes dans le correcteur. Il est donc plus efficace de le désactiver.

Contrôle du paramètre Δ avec un réseau de neurones

L'objectif est de mettre en place un contrôleur capable de fournir la consigne Δ_{opt} en fonction des combinaisons (x_1, x_2, x_3, x_4). Le contrôleur doit permettre :

- De désactiver le correcteur de facteur de puissance (cas $\Delta = -1$).
- De fixer la valeur du paramètre Δ lorsque le correcteur est activé.

Question 7 Proposer un mode de communication pour que les différents appareils électriques puissent transmettre leurs modes de fonctionnement au contrôleur.

Les appareils peuvent communiquer via un réseau local (Wi-Fi, Bluetooth) en envoyant des signaux indiquant leur état de fonctionnement.

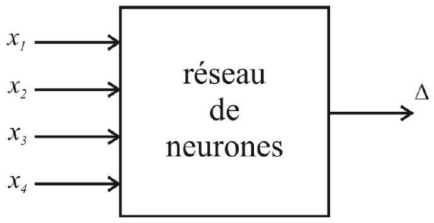


FIGURE 4 – Entrées-sortie du réseau de neurones

Question 8 Quels sont les avantages et les inconvénients du choix d'un réseau de neurones pour modéliser la commande ?

Avantages : Capacité à apprendre des relations complexes, adaptabilité. **Inconvénients :** Nécessite beaucoup de données, complexité de mise en œuvre, manque de transparence.

Remarque : 9 données pour un jeu d'entrainement, on est pas loin d'une blague... Clairement insuffisant !

Question 9 L'apprentissage supervisé est-il pertinent ? Justifiez votre réponse.

Oui, car nous avons des données étiquetées, (couples entrée/sortie connus !) ce qui permet au réseau de neurones d'apprendre à prédire Δ en minimisant l'erreur.

Définition du perceptron

Le réseau de neurones est un perceptron multicouche avec :

- Une couche d'entrée de 4 neurones.
- Une couche cachée de 4 neurones.

- Une couche de sortie d'un neurone.

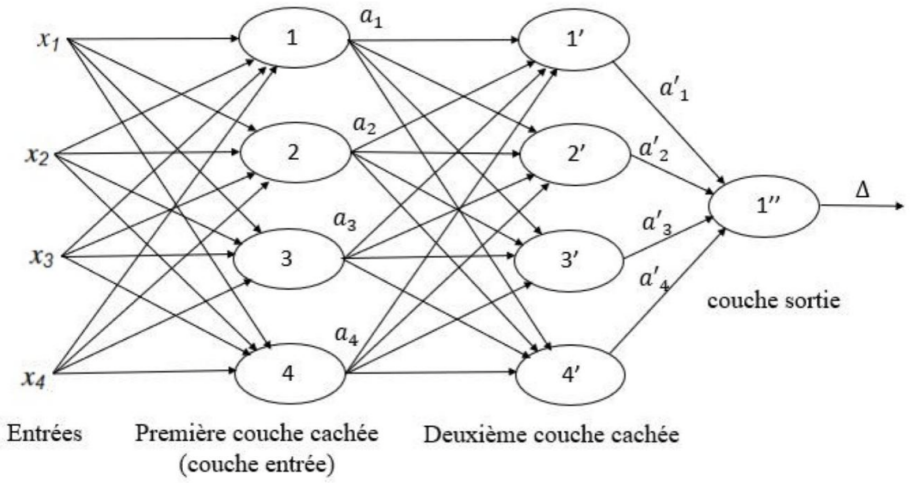


FIGURE 5 – Structure du réseau de neurones choisi

Chaque neurone utilise une fonction d'activation sigmoïde :

$$f(x) = \frac{1}{1+e^{-x}}$$

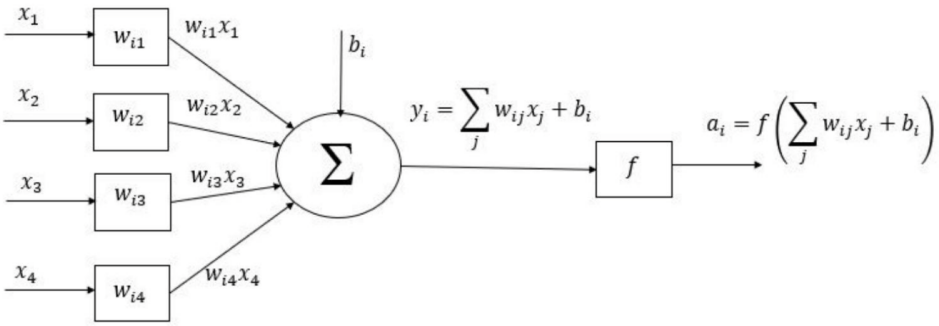


FIGURE 6 – Structure d'un neurone de la première couche

Question 10 Calculer la dérivée de la fonction f notée $f'(x)$ et écrire une fonction Python `f_prime` qui prend en argument x et qui renvoie la dérivée $f'(x)$.

Par dérivation de la fonction $f(x) = \frac{1}{1+\mathrm{e}^{-\mathrm{x}}}$, on obtient : $f'(\mathrm{x}) = \frac{\mathrm{e}^{-\mathrm{x}}}{(1+\mathrm{e}^{-\mathrm{x}})^2}$

```
1 import numpy as np
2
3 def f_prime(x) :
4     return np.exp(-x)/(1+np.exp(-x))**2$
```

Phase d'inférence

La phase d'inférence consiste à calculer la sortie Δ à partir d'un vecteur d'entrée connu :

$$\Delta = f\left(W_3\left(f\left(W_2 f\left(W_1 X + B_1\right) + B_2\right)\right) + b_3\right)$$

Question 11 Donner les dimensions des matrices W_2, W_3, B_2, et b_3.

- $W_2 : 4 \times 4$
- $W_3 : 1 \times 4$
- $B_2 : 4 \times 1$
- $b_3 : 1 \times 1$

Question 12 Donner l'expression de la matrice W_1 qui vérifie :

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = W_1 \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

La matrice W_1 est une matrice 4×4 où chaque élément w_{ij} représente le poids entre l'entrée j et le neurone i de la première couche cachée.

$$W_1 = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} \\ w_{3,1} & W_{3,2} & w_{3,3} & w_{3,4} \\ w_{4,1} & w_{4,2} & w_{4,3} & w_{4,4} \end{bmatrix}$$

Question 13 Écrire en Python la fonction `inference_couche` qui prend en argument le vecteur d'entrées noté X, la matrice des poids notée W, le vecteur de biais noté B, et qui renvoie le vecteur des sorties de la couche noté A.

Voici la fonction Python :

```python
import numpy as np

def inference_couche(X, W, B):
    Y = np.dot(W, X) + B # calul de la somme biaisée du neurone
    A = 1 / (1 + np.exp(-Y)) # calcul de l'activation, ici sigmoïde (return f(Y) si f définie)
    return A
```

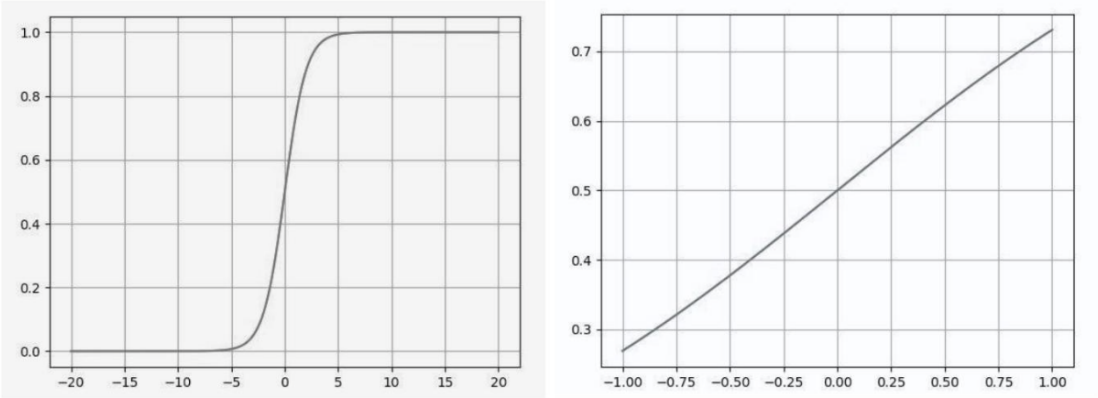


FIGURE 7 – Représentation graphique de la fonction sigmoïde f

Question 14 Calculer $A = f(Y)$ dans le cas où :

$$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}, \quad X = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad W = \begin{bmatrix} 0.3 & 0.2 & 0.25 & -0.04 \\ 0.4 & -0.3 & 0.6 & -0.3 \\ -0.4 & 0.3 & -0.6 & -0.4 \\ -1 & -0.75 & -0.1 & -0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Calculons d'abord Y :

$$Y = WX + B = \begin{bmatrix} 0.2 \\ -0.3 \\ 0.3 \\ -0.75 \end{bmatrix}$$

Ensuite, appliquons la fonction d'activation f :

$$\mathbf{A} = f(Y) = \begin{bmatrix} \frac{\mathbf{1}}{\mathbf{1+e^{-0,2}}} \\ \frac{\mathbf{1}}{\mathbf{1+e^{0,3}}} \\ \frac{\mathbf{1}}{\mathbf{1+e^{-0,3}}} \\ \frac{\mathbf{1}}{\mathbf{1+e^{0,75}}} \end{bmatrix}$$

$$\mathbf{A} = f(Y) = \begin{bmatrix} 0.55 \\ 0.42 \\ 0.57 \\ 0.33 \end{bmatrix}$$

Rétropropagation

La rétropropagation ajuste les poids et les biais pour minimiser l'erreur entre la sortie calculée et la sortie cible.

Question 15 Montrer que $\dfrac{\partial \mathcal{L}}{\partial w_{ij}} = \dfrac{\partial \mathcal{L}}{\partial a_i} f'(y_i) x_j$.

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i} \cdot \frac{\partial a_i}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i} \cdot \frac{\partial f\left(y_i\right)}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i} \cdot \frac{\partial f\left(\sum_{k=1}^{n}\left(w_{ik} \cdot x_k\right) + b_i\right)}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i} \cdot \frac{\partial\left(\sum_{k=1}^{n}\left(w_{ik} \cdot x_k\right) + b_i\right)}{\partial w_{ij}} \cdot f'\left(\sum_{k=1}^{n}\left(w_{ik} \cdot x_k\right) + b_i\right)$$

où n est le nombre d'entrées du iième neurone de la couche considérée.

Or : $\frac{\partial\left(\sum_{k=1}^{n}\left(w_{ik} \cdot x_k\right) + b_i\right)}{\partial w_{ij}} = x_j$ Donc : $\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i} \cdot x_j . f\left(\sum_{k=1}^{n}\left(w_{ik} x_k\right) + b_i\right)$ avec : $y_i = \sum_{k=1}^{n}\left(w_{ik} \cdot x_k\right) + b_i$ On en déduit :

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i} \cdot x_j, f'\left(y_i\right)$$

Remarque : En utilisant la règle de la chaîne, nous avons directement :

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} = \frac{\partial \mathcal{L}}{\partial a_i} f'(y_i) x_j$$

Question 16 Donner la relation matricielle entre $\frac{\partial \mathcal{L}}{\partial W}$, $\frac{\partial \mathcal{L}}{\partial A}$, $f'(Y)$, et X, puis entre $\frac{\partial \mathcal{L}}{\partial W}$, $\frac{\partial \mathcal{L}}{\partial A}$, W, X, et B.

$$\left(\frac{\partial \mathcal{L}}{\partial A} * Y\right) \cdot X^T = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial a_i} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial a_m} \end{bmatrix} * f' \left(\begin{bmatrix} y_1 \\ \vdots \\ \vdots \\ y_i \\ \vdots \\ \vdots \\ y_m \end{bmatrix} \right) \cdot \begin{bmatrix} \mathrm{x_1} \dots & \mathrm{x_j} \dots \mathrm{x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial a_i} f'(y_i) \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial a_m} f'(y_m) \end{bmatrix} \cdot \begin{bmatrix} \mathrm{x_1} \dots & \mathrm{x_j} \dots \mathrm{x_n} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) \cdot x_1 & \dots & \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) \cdot x_j & \dots & \frac{\partial \mathcal{L}}{\partial a_1} f'(y_1) \cdot x_n \\ \vdots & & \vdots & & \\ \frac{\partial \mathcal{L}}{\partial a_i} f'(y_i) \cdot x_1 & \dots & \frac{\partial \mathcal{L}}{\partial a_i} f'(y_i) \cdot x_j & \dots & \frac{\partial \mathcal{L}}{\partial a_i} f'(y_i), x_n \\ \vdots & & \vdots & & \vdots \\ \frac{\partial \mathcal{L}}{\partial a_m} \cdot f'(y_m) \cdot x_1 & \dots & \frac{\partial \mathcal{L}}{\partial a_m} \cdot f'(y_m) \cdot x_j & \dots & \frac{\partial \mathcal{L}}{\partial a_m} \cdot f'(y_m) \cdot x_n \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_{11}} & \dots & \frac{\partial \mathcal{L}}{\partial w_{1j}} & \dots & \frac{\partial \mathcal{L}}{\partial w_{1n}} \\ \vdots & \vdots & \vdots & & \\ \frac{\partial \mathcal{L}}{\partial w_{i1}} & \dots & \frac{\partial \mathcal{L}}{\partial w_{ij}} & \dots & \frac{\partial \mathcal{L}}{\partial w_{in}} \\ \vdots & & \vdots & \vdots & \\ \frac{\partial \mathcal{L}}{\partial w_{m1}} & \dots & \frac{\partial \mathcal{L}}{\partial w_{mj}} & \dots & \frac{\partial \mathcal{L}}{\partial w_{mn}} \end{bmatrix}$$

La relation matricielle est :

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial A} f'(Y) X^T$$

et

$$\frac{\partial \mathcal{L}}{\partial W} = f'(W.X + B).X^T$$

Question 17 Écrire la fonction Python `retropropagation_couche` qui prend en argument la valeur d'entrée d'une couche, notée X, la matrice de poids d'une couche, notée W, le vecteur de biais de la couche, noté B, le vecteur d'erreur en sortie de la couche, noté Ea, et la valeur du coefficient d'apprentissage, notée α.

Voici la fonction Python :

```python
import numpy as np

def retropropagation_couche(X, W, B, Ea, alpha):
    f_prime_Y = f_prime(Y)
    E_W = np.dot(F_prime_Y*Ea,X.T)
    E_B = F_prime_Y*Ea
    E_X = np.dot(W,F_prime_Y)*Ea

    W -= alpha * E_W
    B -= alpha * E_B

    return W, B, E_X
```

Analyse des résultats

L'évolution de la fonction de coût total L_{tot} pendant la phase d'entraînement est montrée ci-dessous :

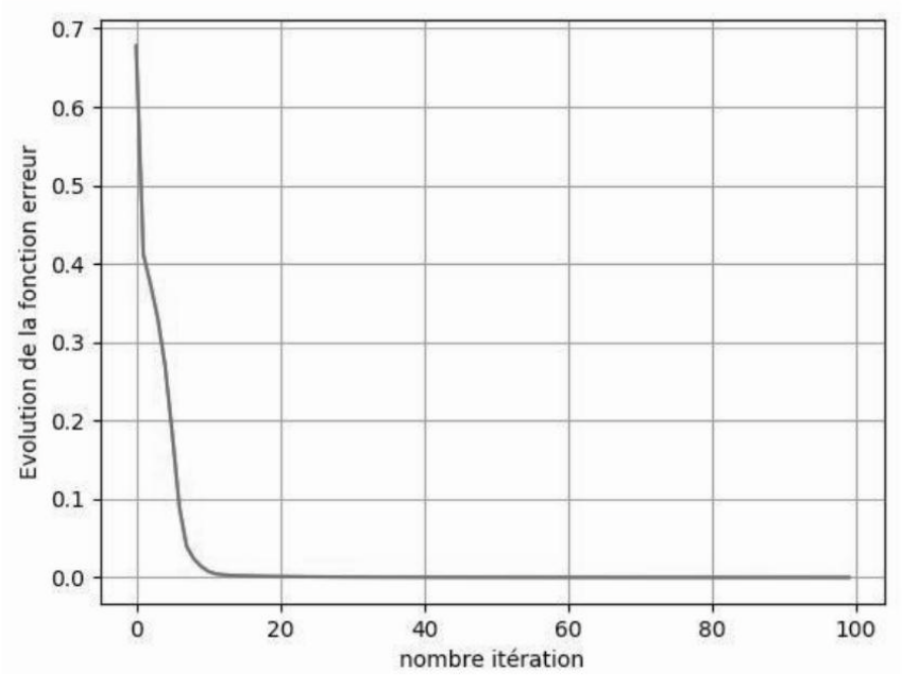


FIGURE 8 – Entraînement du modèle

	box-TV x_1	éclairage x_2	radiateur x_3	chauffe-eau x_4	Δ_{opt} (V)	Δ (V)
Données d'entraînement	0	0	0	1	-1	- 0,981
	0	0	1	1	-1	- 0,979
	0	0	2	0	-1	- 0,961
	0	0	0	2	-1	- 0,983
	0	0	1	2	-1	- 0,980
	0	1	0	0	0,18	0,178
	2	2	0	0	0,32	0,341
	2	2	2	0	0,49	0,474
	2	2	2	2	-1	- 0,970
Données test	0	0	1	0	-1	- 0,919
	0	0	2	1	-1	- 0,977
	0	0	2	2	-1	- 0,978
	1	0	0	0	0,22	0,015
	2	2	1	0	0,42	0,412

FIGURE 9 – Valeurs optimales du paramètre Δ pour quatorze combinaisons différentes

Question 18 Analyser les résultats. Ce réseau de neurones est-il capable de bien prédire la valeur optimale de Δ ?

Le réseau de neurones semble capable de prédire correctement Δ pour 4 valeurs sur 5 sur les données d'entraînement. Pour la dernière on es très loin de la valeur optimale (0.22 vs 0.015). 1 sur 5 c'est 20%, la remarque précédente sur le manque de données est toujours valide...

Remarque : une réponse argumentée et chiffrée devait donner des points ici... Que vous tranchiez pertinent

ou non.

Question 19 Pour améliorer le modèle, est-il préférable d'augmenter le nombre d'itérations de l'entraînement ou d'augmenter le nombre de données de test ? Justifier votre réponse.

On voit sur la courbe d'évolution de la fonction d'erreur que celle-ci se stabilise très rapidement dès 10 à 15 itérations. Il n'est donc pas nécessaire d'augmenter le nombre d'itération pour améliorer les performances de l'algorithme de prédiction.

En revanche, il était assez évident qu'avec seulement 9 données d'entraînement (sic.) l'apprentissage ne pouvait pas être efficace. Il est donc préférable d'augmenter le nombre de données d'entrainement. (Et pas de données test comme le suggère l'énoncé).