

TIPE 2022

Orthèse médicale

Promotion 2022

Vincent Marais n°15091

Julien Védani



Figure – Julien Védani France Télévisions

Problématique

Problématique : Comment la commande de l'exosquelette peut-elle assister correctement son utilisateur lorsqu'il marche ?



Figure – Julien Védani au pôle saint Hélier

Phases de mon TIPE

① Phase de Recherche

② Phase de modélisation de la commande de l'exosquelette

Bastien Fraudet



Figure – <https://b-temia.com/keego/> + Contact : Bastien Fraudet

Qu'est-ce que le mouvement de flexion lors de la marche ?

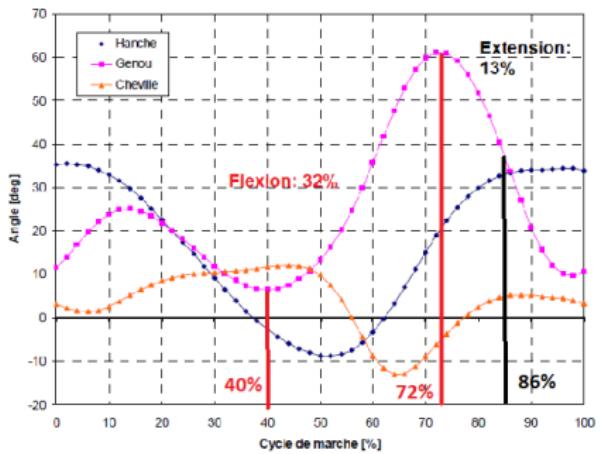
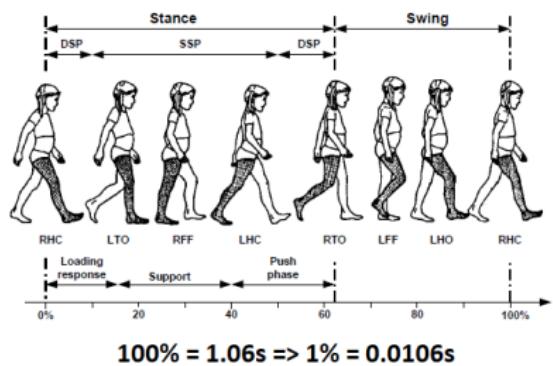


Figure – Cycle de la marche / Amplitude de l'angle du genou, de la cheville et de la hanche (Carl Schmitt 20 avril 2007)

Couple pour la mise en flexion du genou

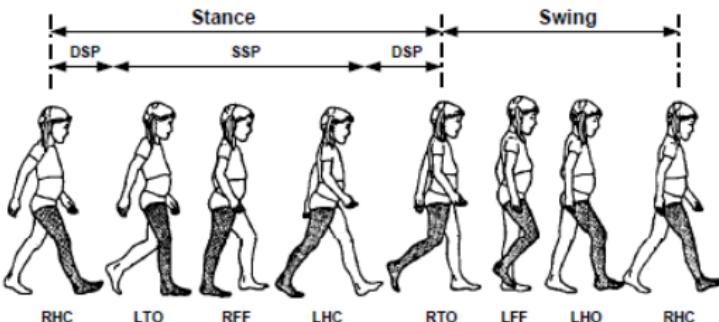
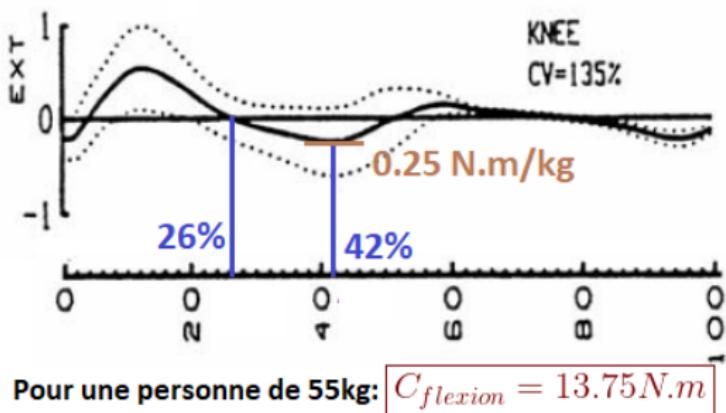


Figure – Thèse marche (Carl Schmitt 20 avril 2007)

Phase de modélisation de la commande

- ① Dimensionner le motoréducteur
 - ② Asservir le motoréducteur
 - ③ Réaliser la consigne du motoréducteur

Choix du moteur sur le site MAXON :

| Caractéristiques moteur | VALEUR |
|---------------------------|---|
| Tension nominale | $U = 12V$ |
| Vitesse nominale | $\omega_m = 8410 \text{tr}.\text{min}^{-1}$ |
| Couple nominal | $C_{mot,e} = 174 \text{mN.m}$ |
| Courant nominal | $I = 16.2A$ |
| Résistance aux bornes | $R = 0.101\Omega$ |
| Inductance | $L = 0.0266 \text{mH}$ |
| Constante de couple | $K_c = 11.5 \times 10^{-3} \text{Nm.A}^{-1}$ |
| Constante de vitesse | $K_e = 11.5 \times 10^{-3} \text{V.s.rad}^{-1}$ |
| Moment d'inertie du rotor | $J_{rotor} = 119 \times 10^{-7} \text{kg.m}^2$ |

Fig : EC 45 Ø45 mm, à commutation électronique, 150 Watt

Choix du réducteur

| Caractéristiques réducteur | VALEUR |
|-------------------------------|--|
| Rapport de réduction | $r = 1/\rho = 1 : 126$ |
| Rendement | $\eta = 72\%$ |
| Couple continu maximal | $C_{red} = 15.0 N.m$ |
| Moment d'inertie du réducteur | $J_{red} = 14 \times 10^{-7} \text{ kg.m}^2$ |

Fig : Réducteur planétaire GP 42 C Ø42 mm, 3.0–15.0 Nm

On veut au minimum que notre motoréducteur délivre un couple :

$$C_{mot,s} = 15 \text{ N.m} / \eta = 0.72$$

$$\left\{ \begin{array}{l} \frac{\omega_{red}}{\omega_{mot}} = 1/\rho_{min} \\ \frac{P_{sortie}}{P_{entrée}} = \frac{C_{mot,s} \times \omega_{red}}{C_{mot,e} \times \omega_{mot}} = \eta \end{array} \right. \quad (1)$$

Donc :

$$\rho_{min} = 119.73$$

Assister en effort ⇒ Asservissement en courant du moteur

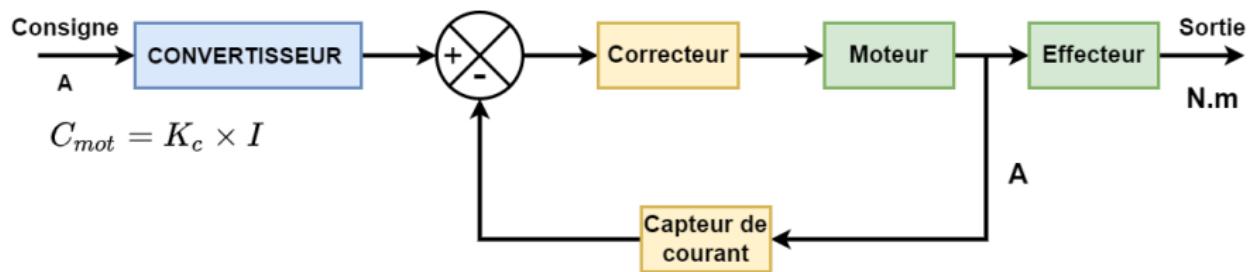


Figure – Schéma de principe de l'asservissement en courant du moteur

| CRITÈRE de stabilité | VALEUR |
|----------------------|---------------------------|
| Marge de Gain | $M_G \geq 10dB$ |
| Marge de phase | $M_\varphi \geq 45^\circ$ |

Asservissement de courant (Consigne)

| CRITÈRE | VALEUR |
|---------------------|-----------------------|
| Durée de simulation | $T = 0.1696s$ |
| Pente de la rampe | $P = 77.709 A/s$ |
| Ecart de traînage | $\varepsilon_T < 5\%$ |

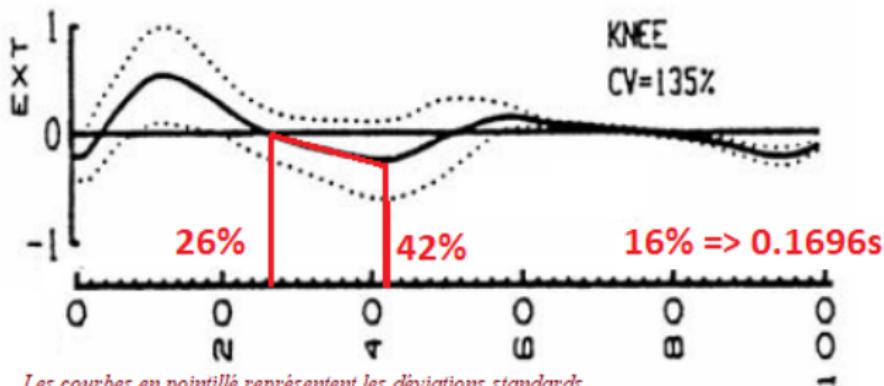


Figure – Moments articulaires, Cadence de marche naturelle, 19 sujets (Carl Schmitt 20 avril 2007)

Schéma Bloc asservissement de courant : sans correction

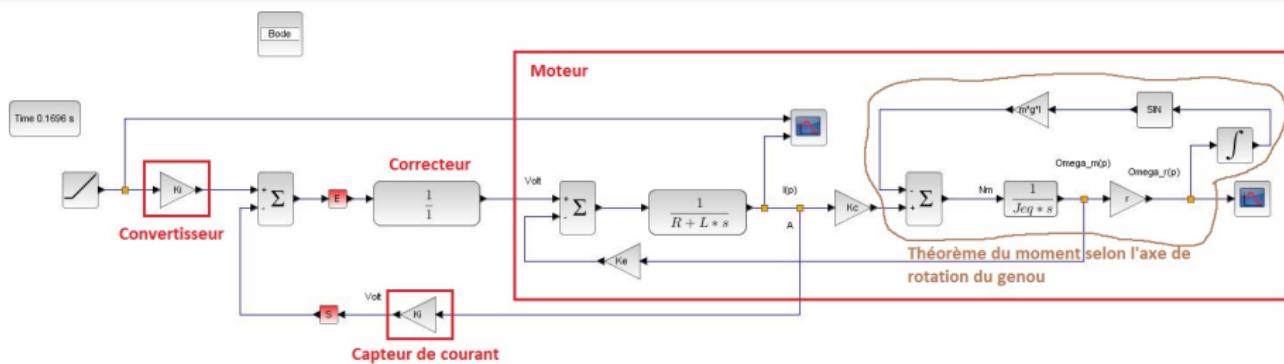


Figure – Schéma bloc sans correction sur scilab-xcos

| Constantes | VALEUR |
|--|--|
| Gain du Capteur de courant | $K_i = 0.1V/A$ |
| Moment d'inertie équivalent | $J_{eq} = 2.905 \times 10^{-5} kg.m^2$ |
| Masse de {jambe + pied} | $m = 3.355 kg$ |
| Distance genou centre de gravité {jambe + pied} | $l = 24cm$ |

Réponse temporelle asservissement de courant : sans correction

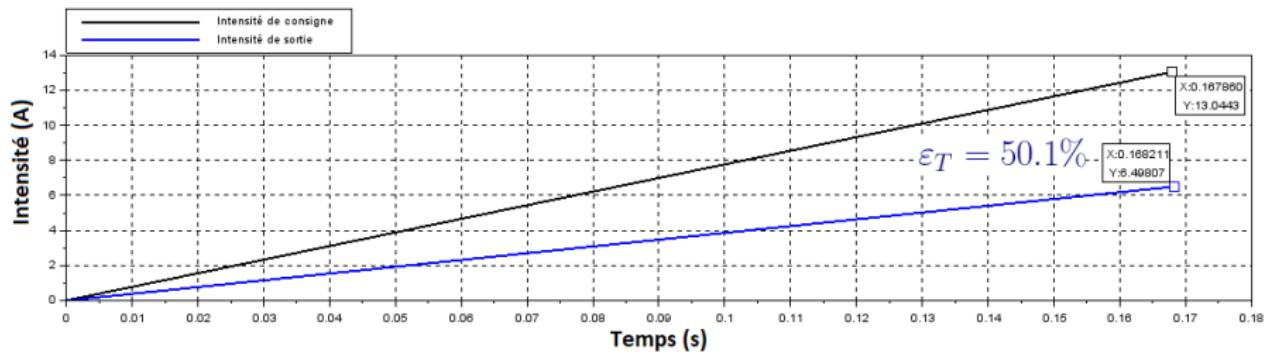


Figure – Couple de sortie = 6.71 N.m d'après la relation (1)

Schéma bloc asservissement de courant corrigé (PI)

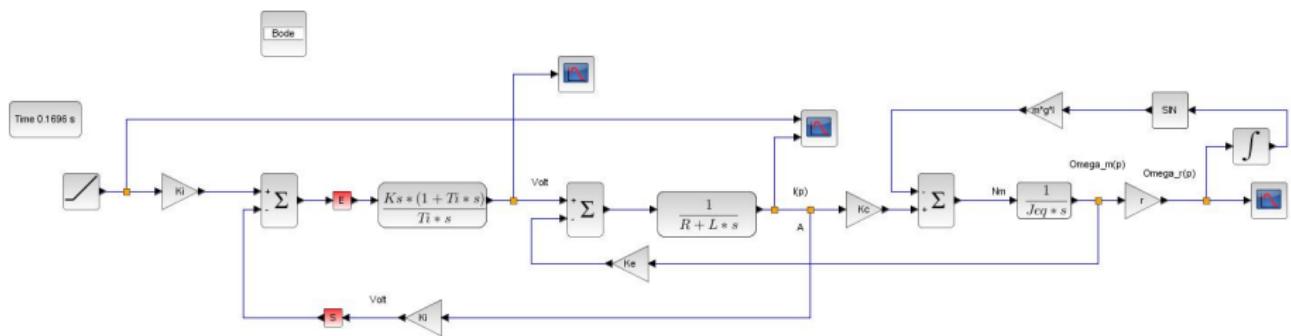


Figure – Schéma bloc avec correction sur scilab-xcos

Correcteur PI :

$$\frac{K_s(1 + T_i * s)}{T_i * s}$$

avec $K_s = 1$ et $T_i = 1s$

Bode et réponse temporelle asservissement de courant (PI)

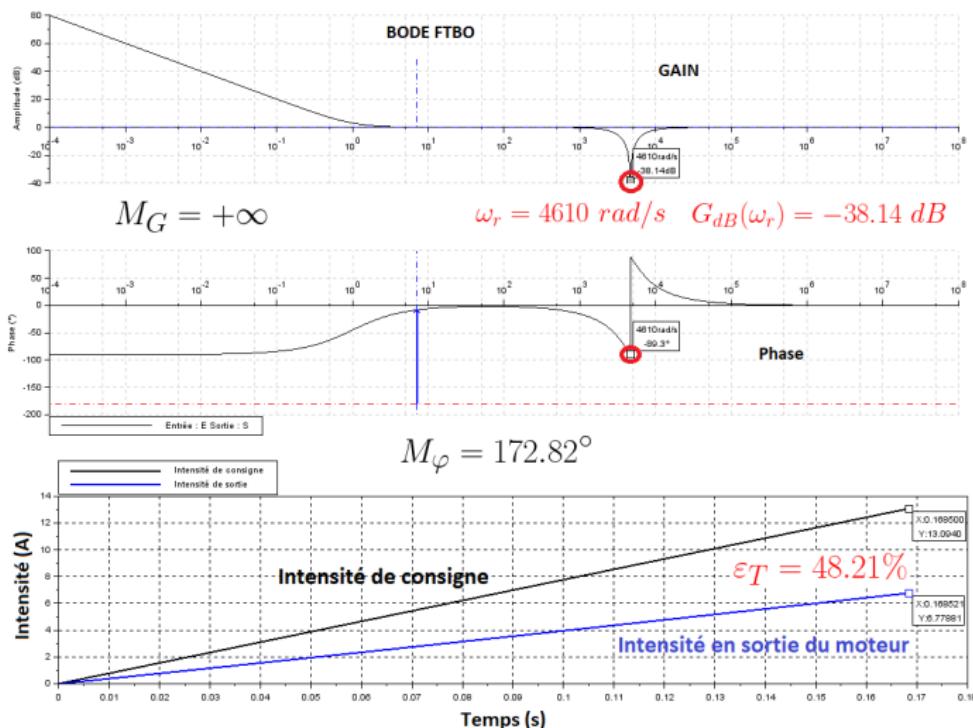


Figure – $C_{mot,s} = 7 \text{ N.m} / \text{Réglage : } K_s = 10^{38.14/20} = 80.72$

Bilan asservissement de courant

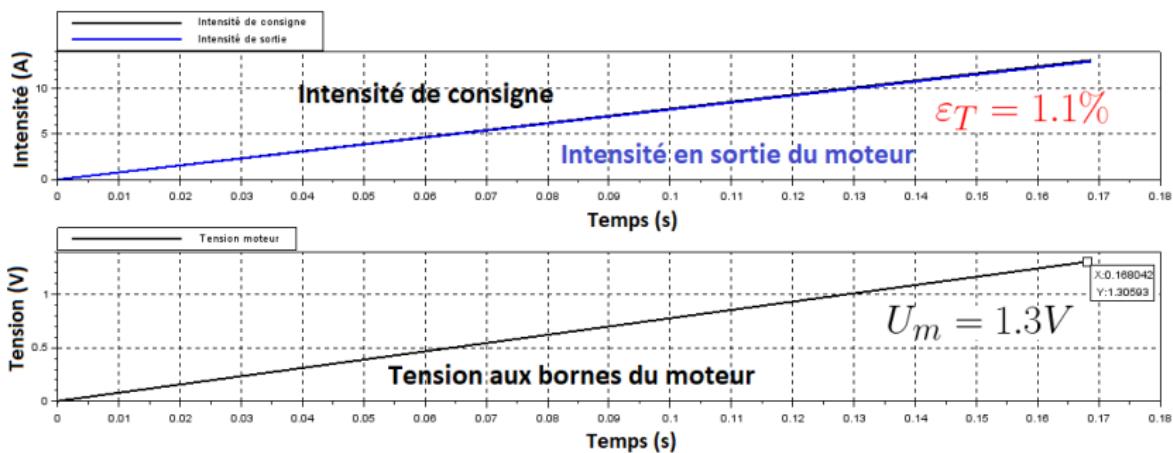


Figure – $C_{mot,s} = 13.52 N.m$

| CRITÈRE | VALEUR |
|-------------------|-----------------------|
| M_φ | 90.7° |
| Ecart de traînage | $\varepsilon_T < 5\%$ |
| M_G | $+\infty$ |

Comment déterminer la consigne en effort du motoréducteur adaptée à l'utilisateur ?

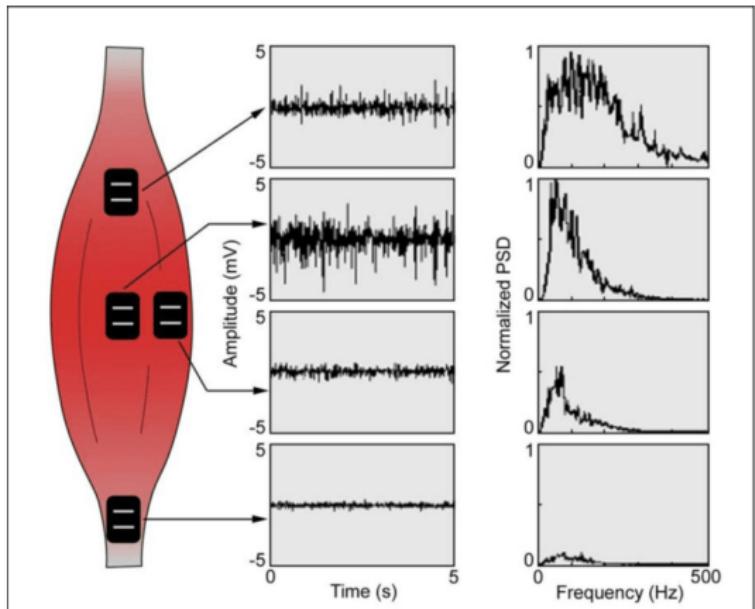
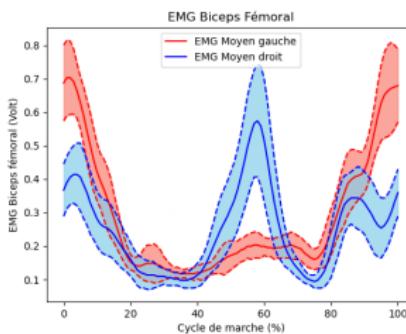


Figure – Signaux électromyographiques (EMG) bruts + Capteurs électromyographiques (Viet Anh Dung Cai ,2011)

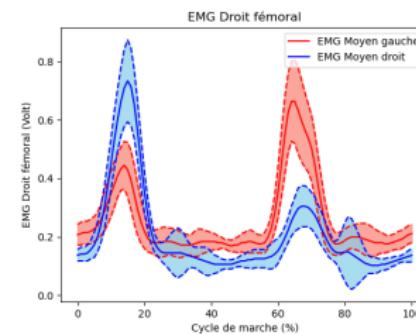
Signaux EMG fournis par Mathieu LEMPEREUR



Biceps Fémoral



Droit Fémoral



Vaste Latéral

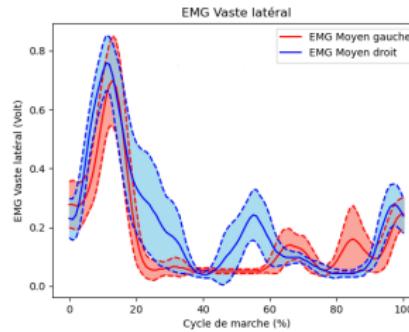


Figure – Signaux EMG (Volt) des différents muscles en fonction du cycle de marche (%) d'une femme souffrant sclérose en plaque

Regression

Soit τ le moment du genou droit et u les signaux EMG moyen du droit fémoral côté droit.

Dataset : (τ, u)

Modèle : $\tau = f(u) = ?$

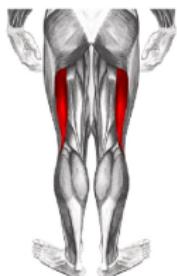
Fonction coût (MSE) : $J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left(f(u^{(i)}) - \tau^{(i)} \right)^2$

Algorithme de minimisation : Déterminer $\frac{\partial J}{\partial \theta}(\theta) = 0$

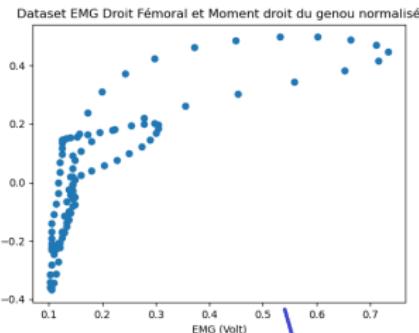
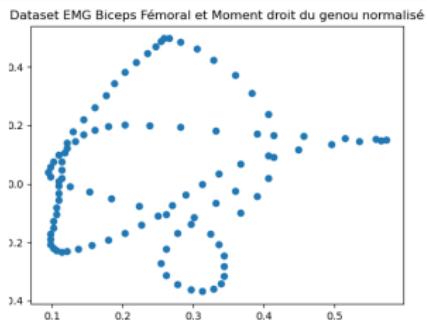
- Descente du gradient
- Méthode des Equations Normales

Coefficient de détermination : R^2

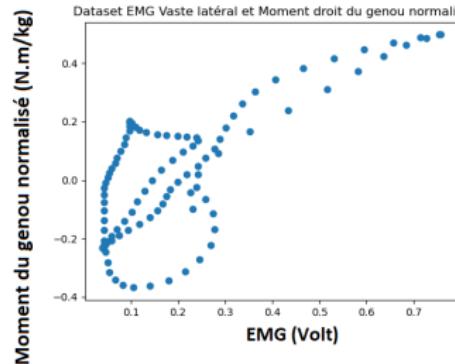
Analyse de la base de donnée



Biceps Fémoral



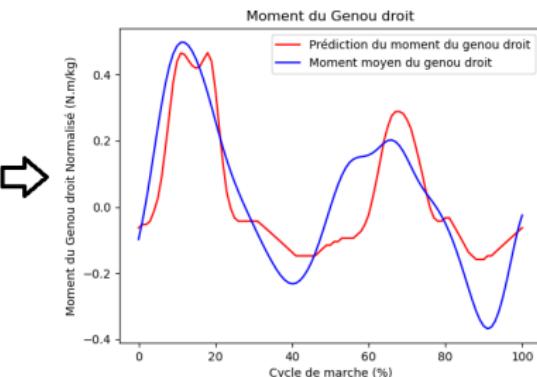
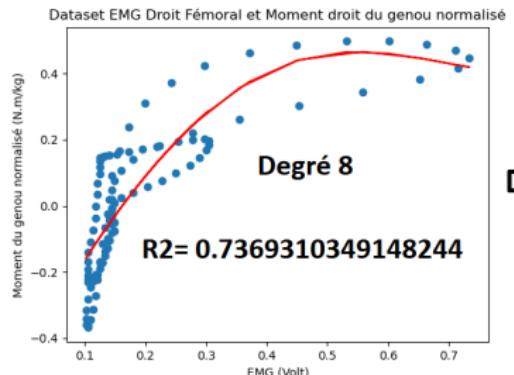
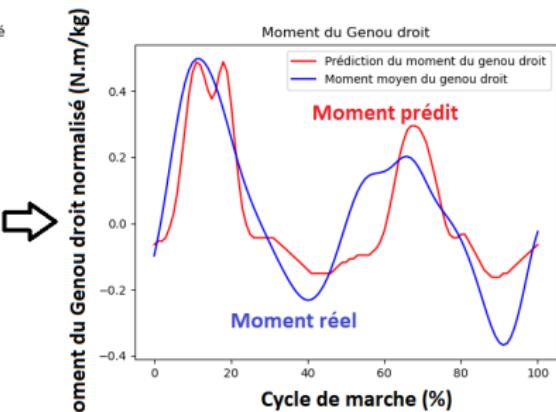
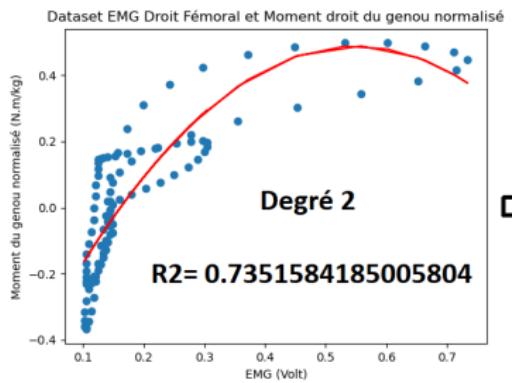
Vaste Latéral



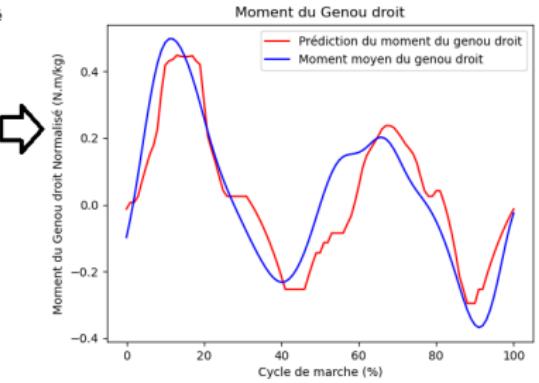
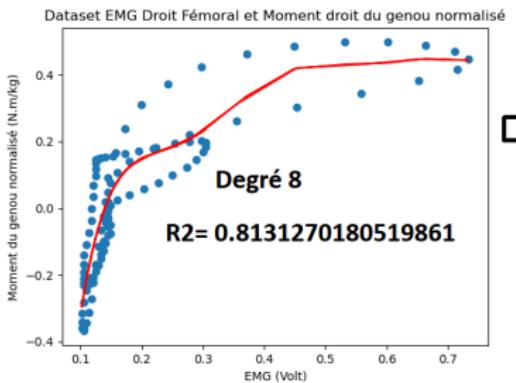
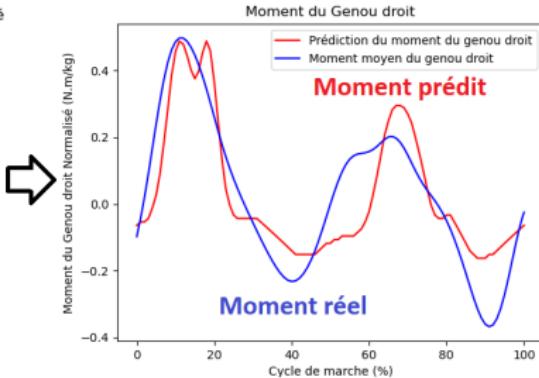
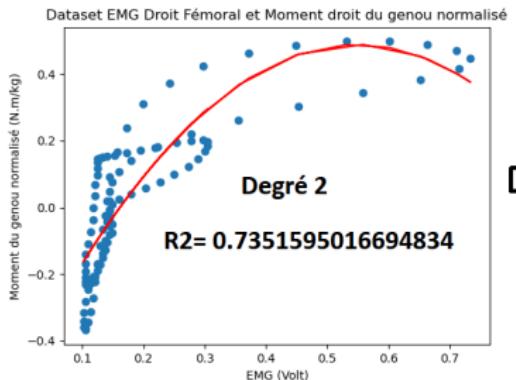
$$\tau = f(u) = \sum_{i=0}^n a_i u^i$$

Figure – Nuage de points du moment du genou normalisé (N.m/kg) en fonction du signal EMG (Volt)

Moment prédit avec la descente du gradient



Moment prédit avec la méthode des équations normales



Degré du modèle polynômial avec R^2 le plus proche de 1

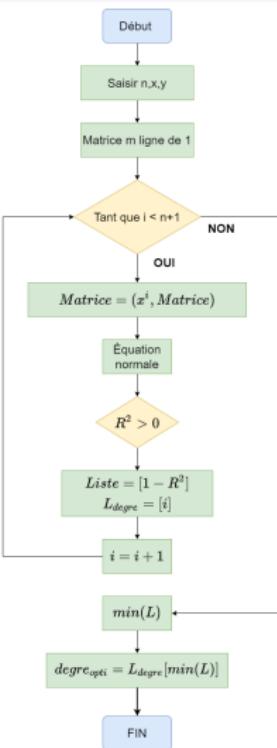


Figure – Algorithme de la fonction degré_polynome_opti

Modèle polynômial Final

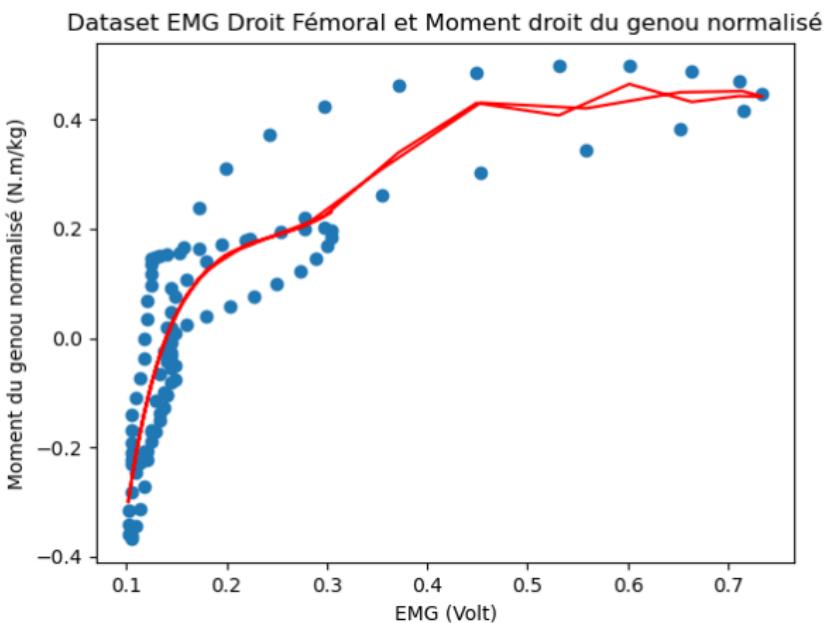
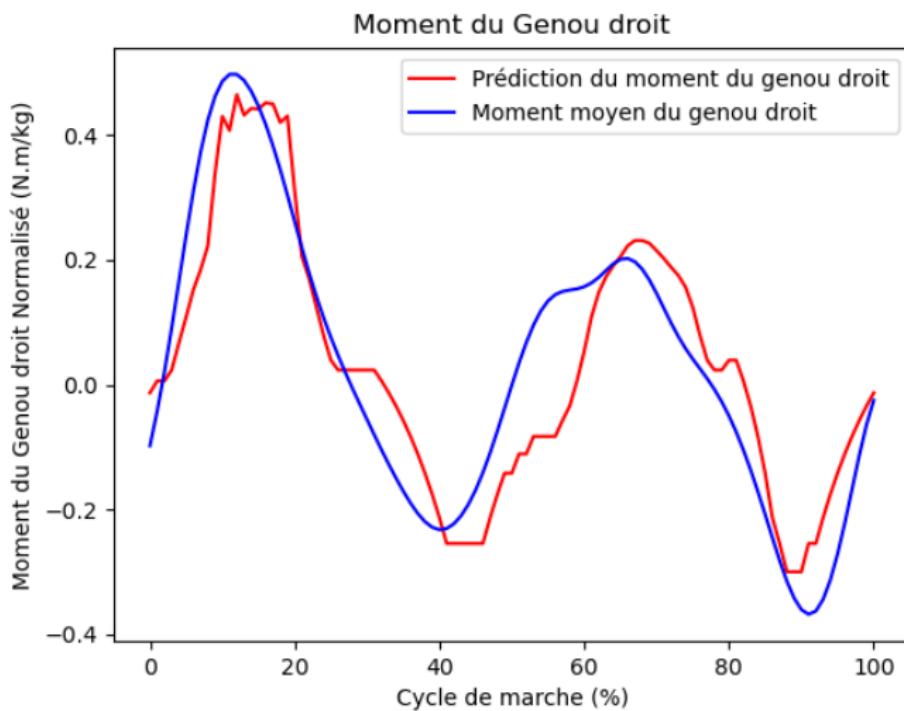


Figure – $R^2 = 0.8137128655340371$, Polynôme de degré 19

Prédiction Moment grâce aux signaux EMG



Figure

Conclusion

Annexe : Complément sur la marche

| | |
|------------------------------------|---------------------------------|
| <i>Stance:</i> | <i>appui</i> |
| <i>Swing:</i> | <i>oscillation</i> |
| <i>DSP (double support phase):</i> | <i>double support</i> |
| <i>SSP (single support phase):</i> | <i>appui unilatéral</i> |
| <i>RHC (right heel contact):</i> | <i>contact talon droit</i> |
| <i>LTO (left toe off):</i> | <i>envol pied gauche</i> |
| <i>RFF (right foot flat):</i> | <i>pied droit à plat</i> |
| <i>LHC (left heel contact):</i> | <i>contact talon gauche</i> |
| <i>RTO (right toe off):</i> | <i>envol pied droit</i> |
| <i>LFF (left foot flat):</i> | <i>pied gauche à plat</i> |
| <i>LHO (left heel off):</i> | <i>décollement talon gauche</i> |
| <i>Loading response:</i> | <i>réception du poids</i> |
| <i>Support:</i> | <i>appui intermédiaire</i> |
| <i>Push phase:</i> | <i>phase de poussée</i> |

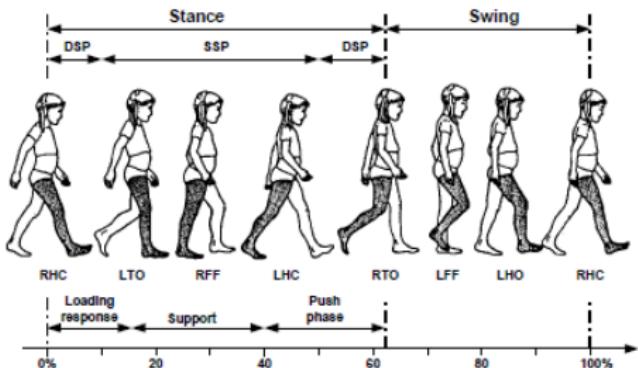
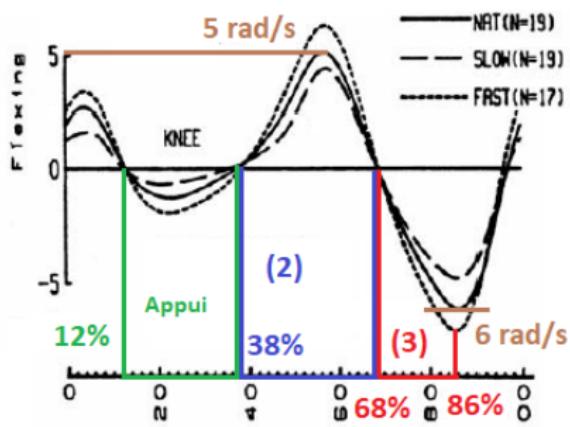


Figure 3.7 Phases du cycle de marche [POPOVIC 2000]

Figure – Thèse marche (Carl Schmitt 20 avril 2007)

| Caractéristiques | Femme | Homme | Enfants (5 ans) |
|---------------------------|-----------------|-----------------|-----------------|
| Vitesse [m/s] | 1.4 ± 0.15 | 1.5 ± 0.2 | 1.05 ± 0.05 |
| Durée du cycle [s] | 1.03 ± 0.08 | 1.06 ± 0.09 | 0.77 ± 0.06 |
| Cadence [pas/min.] | 117 ± 9 | 113 ± 8 | 153 ± 11 |
| Longueur de la foulée [m] | 1.37 ± 0.1 | 1.60 ± 0.18 | 0.86 ± 0.08 |
| Durée phase d'appui [s] | 0.64 ± 0.06 | 0.65 ± 0.07 | non mesurée |

Annexe : Vitesse de rotation du genou



(2): Flexion (3): Extension

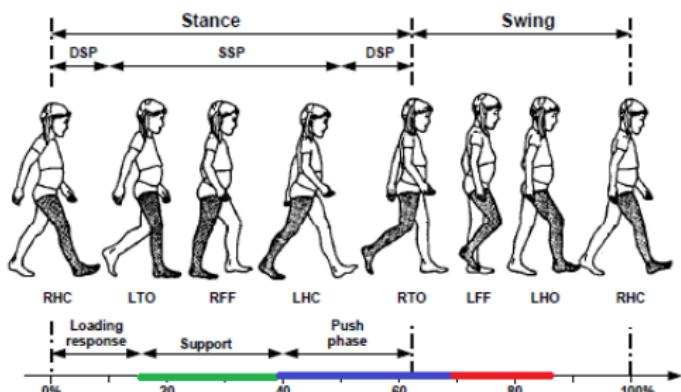


Figure – Thèse marche (Carl Schmitt 20 avril 2007)

Annexe : Caractéristiques du moteur

EC 45 Ø45 mm, à commutation électronique, 150 Watt

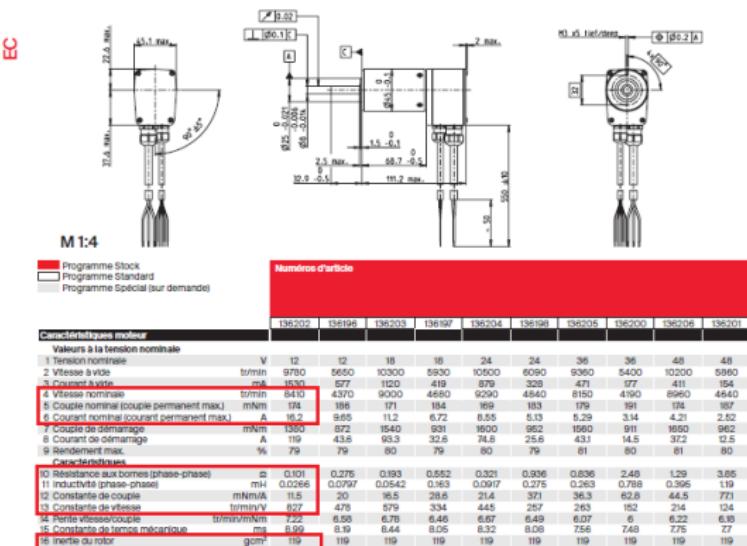


Figure –

https://www.maxongroup.com/medias/sys_master/root/8841251389470/FR-230.pdf

Annexe : Perturbation

Isolement : { Jambe + pied (fixe) = S de masse $m = 55kg$ }

Bilan des actions mécaniques :

- Poids de l'ensemble {jambe + pied } G :

$$\{P\} = \begin{Bmatrix} 0 \\ - \\ -m_{jb+pied} \times g \end{Bmatrix}_{G(\vec{x}_0, \vec{y}_0, \vec{z}_0)}$$

- Pivot en (O, \vec{y}_0) :

$$\begin{Bmatrix} x_0 & - \\ - & 0 \\ z_0 & - \end{Bmatrix}_{O(\vec{x}_0, \vec{y}_0, \vec{z}_0)}$$

- Couple moteur : $C_{mot,s}\vec{y}_0$
- On néglige le poids de l'ensemble {moteur + réducteur }

Inertie équivalente

Isolement : { Jambe + pied (fixe) + motoréducteur }

Mouvement :

- Mouvement de rotation de l'ensemble {jambe + pied} autour de l'axe (O, \vec{y}_0)
- Pas de translation de l'ensemble {jambe + pied} par rapport au sol

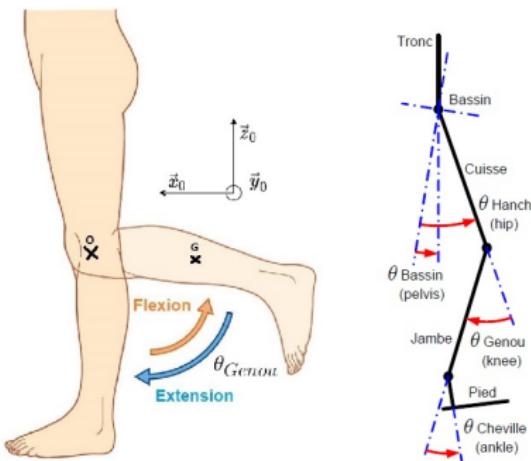


Figure – Angles articulaires de la jambe

Inertie équivalente

Bilan d'énergie cinétique :

$$E_c(S/sol) = E_c(\text{moteur}/sol) + E_c(\text{red}/sol) + E_c(\text{Jambe}/sol)$$

$$\frac{1}{2}J_{eq}\omega_{mot}^2 = \frac{1}{2}J_{mot}\omega_{mot}^2 + \frac{1}{2}J_{red}\omega_{mot}^2 + \frac{1}{2}J_{Jambe}\omega_{red}^2$$

Donc

$$J_{eq} = J_{mot} + J_{red} + r^2J_{jambe} = 2.905 \times 10^{-5} \text{ kg.m}^2$$

Référence capteur de courant : <https://www.gotronic.fr/art-capteur-de-courant-acs712-vma323-31191.htm> (Capteur de courant ACS712 VMA323)

Annexe : Perturbation

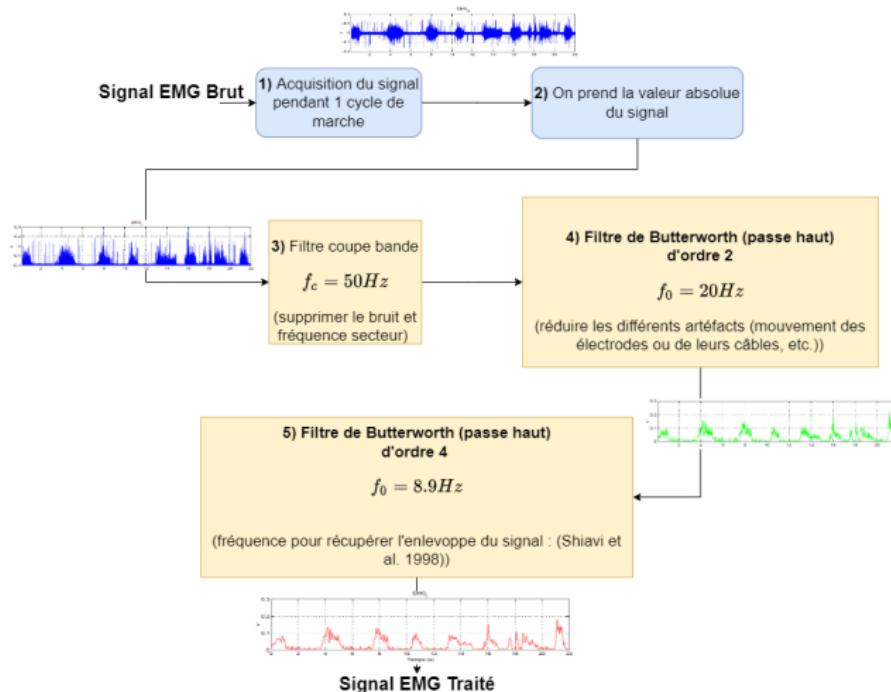
Changement de point en O : $\overrightarrow{OG} = -\frac{H}{2} [\sin(\theta_{Genou}) \vec{x}_0 + \cos(\theta_{Genou}) \vec{z}_0]$

$$\vec{M}(O, \vec{P}) = \vec{M}(G, \vec{P}) + \overrightarrow{OG} \wedge \vec{R}(\vec{P}) = -m_{jb+pied} g \frac{H}{2} \sin(\theta_{Genou}) \vec{y}_0$$

Théorème du moment dynamique en selon (O, \vec{y}_0) :

$$J_{eq} \frac{d\omega_{mot}}{dt} = C_{mot,s} - m_{jb+pied} g \frac{H}{2} \sin(\theta_{Genou})$$

Annexe : Traitement des signaux EMG



Figure

Annexe : Notation Régression sous forme de matrice

Soit $f(u) = \sum_{i=0}^n a_i u^i$

- $Y = (\tau)_{1 < i < n}$ le vecteur du moment du genou droit (target variable)
- X la matrice du signal EMG du droit fémoral côté droit (Feature)

$$X = \begin{bmatrix} (u^{(1)})^n & \dots & 1 \\ \dots & \dots & \dots \\ (u^{(m)})^n & \dots & 1 \end{bmatrix}$$

- θ vecteur des paramètres du modèle f :

$$\theta = \begin{pmatrix} a_n \\ a_{n-1} \\ .. \\ .. \\ a_0 \end{pmatrix}$$

Annexe : Notation Régression sous forme de matrice

Fonction coût (MSE) : $J(\theta) = \frac{1}{2m} \sum_{i=1}^m (X\theta - Y)^2$

Algorithme de minimisation :

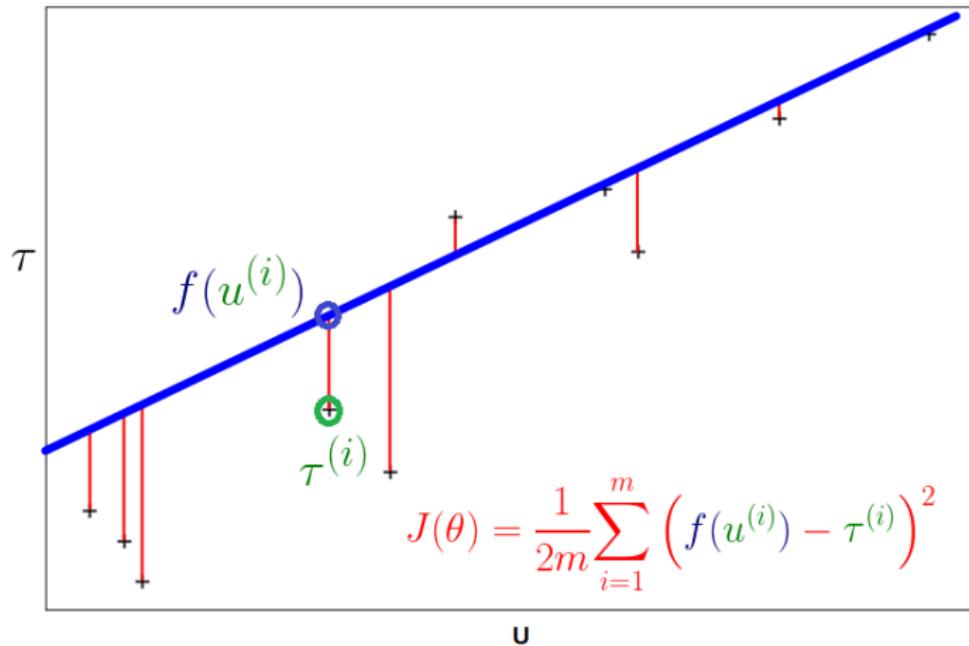
- Le gradient : $\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T (X\theta - Y)$
- Descente du gradient : $\theta_{i+1} = \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta}$
- Équation normale : $\theta = (X^T X)^{-1} X^T Y$

Coefficient de détermination linéaire : R^2

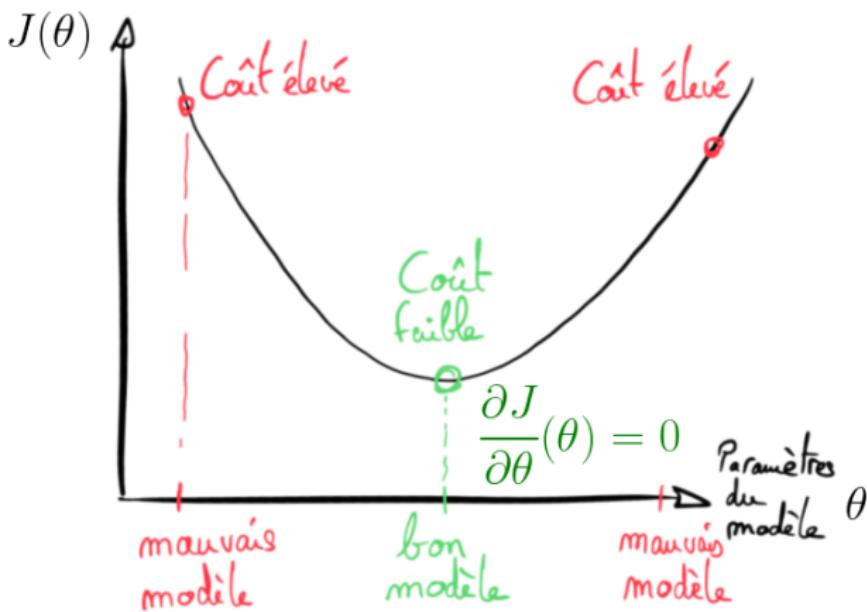
$$R^2 = 1 - \frac{\left(\sum_{i=1}^m \tau^{(i)} - f(u^{(i)}) \right)^2}{\left(\sum_{i=1}^m \tau^{(i)} - \tau_{moy} \right)^2}$$

- Moment du genou réel à l'instant du cycle i : $\tau^{(i)}$
- Moment du genou prédit à l'instant du cycle i : $f(u^{(i)})$
- τ_{moy} : moyenne des moments du genou

MSE



Explication Descente du gradient et la méthode des équations normales



Annexe : Programme EMG

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import os
import numpy as np
import btk #conda install -c conda-forge btk
import matplotlib.pyplot as plt
# from mpi_toolkits.mplot3d import Axes3D

## Dataset d'une femme de 74.5kg atteint de sclérose en plaque

def read_c3d(filename):
    reader = btk.btkAcquisitionFileReader()
    reader.Setfilename(filename)
    reader.Update()
    acq = reader.GetOutput()
    return acq

file = "c:\\\\Users\\\\vince\\\\Desktop\\\\LAM_Cath_Session_01-btk_Moyenne.c3d"
acq = read_c3d(file)
file = "c:\\\\Users\\\\vince\\\\Desktop\\\\LAM Cath Session 01 - btk_Ecart-type.c3d"
acql = read_c3d(file)
## Angle du genou

LeftKneeAnglesMean = acq.GetPoint("LKneeAngles").GetValues()
LeftKneeAnglesStd = acql.GetPoint("LKneeAngles").GetValues()
RightKneeAnglesMean = acq.GetPoint("RKneeAngles").GetValues()
RightKneeAnglesStd = acql.GetPoint("RKneeAngles").GetValues()
plt.plot(range(101),LeftKneeAnglesMean[:,0],'r', label='Angle moyen gauche')
plt.plot(range(101),LeftKneeAnglesMean[:,0]+LeftKneeAnglesStd[:,0],'r--')
plt.plot(range(101),LeftKneeAnglesMean[:,0]-LeftKneeAnglesStd[:,0],'r--')
plt.fill_between(range(101),LeftKneeAnglesMean[:,0]-LeftKneeAnglesStd[:,0],LeftKneeAnglesMean[:,0]
+LeftKneeAnglesStd[:,0],alpha=0.7,color='salmon')
plt.plot(range(101),RightKneeAnglesMean[:,0],'b', label='Angle moyen droit')
plt.plot(range(101),RightKneeAnglesMean[:,0]+RightKneeAnglesStd[:,0],'b--')
plt.plot(range(101),RightKneeAnglesMean[:,0]-RightKneeAnglesStd[:,0],'b--')
plt.fill_between(range(101),RightKneeAnglesMean[:,0]-RightKneeAnglesStd[:,0],RightKneeAnglesMean[:,0]
+RightKneeAnglesStd[:,0],alpha=0.7,color='skyblue')
plt.title('Angle F/E Genou')
plt.xlabel('Cycle de marche (%)')
plt.ylabel('Angle du genou (°)')
plt.legend()
plt.show()
```

Annexe : Programme EMG

```

## Moment du genoux
LeftKneeMomentsMean = acq.GetPoint("LKneeMoment").GetValues()
LeftKneeMomentsStd = acq.GetPoint("LKneeMoment").GetValues()
RightKneeMomentsMean = acq.GetPoint("RKneeMoment").GetValues()
RightKneeMomentsStd = acq.GetPoint("RKneeMoment").GetValues()
plt.plot(range(101),LeftKneeMomentsMean[:,0]*74.5/1000,'r', label='Couple Moyen gauche')
plt.plot(range(101),(LeftKneeMomentsMean[:,0]+LeftKneeMomentsStd[:,0])*74.5/1000,'r--')
plt.plot(range(101),(LeftKneeMomentsMean[:,0]-LeftKneeMomentsStd[:,0])*74.5/1000,'r--')
plt.fill_between(range(101),(LeftKneeMomentsMean[:,0]-LeftKneeMomentsStd[:,0])*74.5/1000,
(LeftKneeMomentsMean[:,0]+LeftKneeMomentsStd[:,0])*74.5/1000, alpha=0.7,color='salmon')
plt.plot(range(101),RightKneeMomentsMean[:,0]*74.5/1000,'b', label='Couple Moyen droit')
plt.plot(range(101),(RightKneeMomentsMean[:,0]+RightKneeMomentsStd[:,0])*74.5/1000,'b--')
plt.plot(range(101),(RightKneeMomentsMean[:,0]-RightKneeMomentsStd[:,0])*74.5/1000,'b--')
plt.fill_between(range(101),(RightKneeMomentsMean[:,0]-RightKneeMomentsStd[:,0])*74.5/1000,
(RightKneeMomentsMean[:,0]+RightKneeMomentsStd[:,0])*74.5/1000, alpha=0.7,color='skyblue')
plt.title('Moment F/E Genou') # Moment en N.m mais initialement c'était en N.mm/kg
plt.xlabel('Cycle de marche (%)')
plt.ylabel('Moment du genou (N.m)')
plt.legend()
plt.show()

## EMG Biceps fémoral
LeftIJMean = acq.GetAnalog('Biceps Femoris Gauche').GetValues() + 2*acq.GetAnalog('Biceps Femoris
Gauche').GetOffset()
LeftIJStd = acq.GetAnalog('Biceps Femoris Gauche').GetValues() + 2*acq.GetAnalog('Biceps Femoris
Gauche').GetOffset()
RightIJMean = acq.GetAnalog('Biceps Femoris Droit').GetValues() + 2*acq.GetAnalog('Biceps Femoris
Droit').GetOffset()
RightIJStd = acq.GetAnalog('Biceps Femoris Droit').GetValues() + 2*acq.GetAnalog('Biceps Femoris
Droit').GetOffset()
plt.plot(range(101),LeftIJMean[:,0],'r', label='EMG Moyen gauche')
plt.plot(range(101),LeftIJMean[:,0]+LeftIJStd[:,0],'r--')
plt.plot(range(101),LeftIJMean[:,0]-LeftIJStd[:,0],'r--')
plt.fill_between(range(101),LeftIJMean[:,0]-LeftIJStd[:,0],LeftIJMean[:,0]+LeftIJStd[:,0],
alpha=0.7,color='salmon')
plt.plot(range(101),RightIJMean[:,0],'b', label='EMG Moyen droit')
plt.plot(range(101),RightIJMean[:,0]+RightIJStd[:,0],'b--')
plt.plot(range(101),RightIJMean[:,0]-RightIJStd[:,0],'b--')
plt.fill_between(range(101),RightIJMean[:,0]-RightIJStd[:,0],RightIJMean[:,0]+RightIJStd[:,0],
alpha=0.7,color='skyblue')
plt.title('EMG Biceps Femoral')
plt.xlabel('Cycle de marche (%)')
plt.ylabel('EMG Biceps fémoral (Volt)')
plt.legend()
plt.show()

```

Annexe : Programme EMG

```
## EMG: Vaste latéral
LeftVLMean = acq.GetAnalog('Vastus Lateralis Gauche').GetValues() + 2*acq.GetAnalog('Vastus Lateralis Gauche').GetOffset()
LeftVLStd = acq1.GetAnalog('Vastus Lateralis Gauche').GetValues() + 2*acq1.GetAnalog('Vastus Lateralis Gauche').GetOffset()
RightVLMean = acq.GetAnalog('Vastus Lateralis Droit').GetValues() + 2*acq.GetAnalog('Vastus Lateralis Droit').GetOffset()
RightVLStd = acq1.GetAnalog('Vastus Lateralis Droit').GetValues() + 2*acq1.GetAnalog('Vastus Lateralis Droit').GetOffset()
plt.plot(range(101),LeftVLMean[:,0],'r', label='EMG Moyen gauche')
plt.plot(range(101),LeftVLMean[:,0]+LeftVLStd[:,0],'r--')
plt.plot(range(101),LeftVLMean[:,0]-LeftVLStd[:,0],'r--')
plt.fill_between(range(101),LeftVLMean[:,0]-LeftVLStd[:,0],LeftVLMean[:,0]+LeftVLStd[:,0],alpha=0.7,color='salmon')
plt.plot(range(101),RightVLMean[:,0],'b', label='EMG Moyen droit')
plt.plot(range(101),RightVLMean[:,0]+RightVLStd[:,0],'b--')
plt.plot(range(101),RightVLMean[:,0]-RightVLStd[:,0],'b--')
plt.fill_between(range(101),RightVLMean[:,0]-RightVLStd[:,0],RightVLMean[:,0]+RightVLStd[:,0],alpha=0.7,color='skyblue')
plt.title('EMG Vaste latéral')
plt.xlabel('Cycle de marche (%)')
plt.ylabel('EMG Vaste latéral (Volt)')
plt.legend()
plt.show()
```

Annexe : Programme EMG

```

## EMG: Droit fémoral
LeftRFMean = acq.GetAnalog('Rectus Femoris Gauche').GetValues() + 2*acq.GetAnalog('Rectus Femoris Gauche').GetOffset()
LeftRFStd = acq1.GetAnalog('Rectus Femoris Gauche').GetValues() + 2*acq1.GetAnalog('Rectus Femoris Gauche').GetOffset()
RightRFMean = acq.GetAnalog('Rectus Femoris Droit').GetValues() + 2*acq.GetAnalog('Rectus Femoris Droit').GetOffset()
RightRFStd = acq1.GetAnalog('Rectus Femoris Droit').GetValues() + 2*acq1.GetAnalog('Rectus Femoris Droit').GetOffset()
plt.plot(range(101),LeftRFMean[:,0], 'r', label='EMG Moyen gauche')
plt.plot(range(101),LeftRFMean[:,0]+LeftRFStd[:,0], 'r--')
plt.plot(range(101),LeftRFMean[:,0]-LeftRFStd[:,0], 'r--')
plt.fill_between(range(101),LeftRFMean[:,0]-LeftRFStd[:,0],LeftRFMean[:,0]+LeftRFStd[:,0],alpha=0.7,color='salmon')
plt.plot(range(101),RightRFMean[:,0], 'b', label='EMG Moyen droit')
plt.plot(range(101),RightRFMean[:,0]+RightRFStd[:,0], 'b--')
plt.plot(range(101),RightRFMean[:,0]-RightRFStd[:,0], 'b--')
plt.fill_between(range(101),RightRFMean[:,0]-RightRFStd[:,0],RightRFMean[:,0]+RightRFStd[:,0],alpha=0.7,color='skyblue')
plt.xlabel('Cycle de marche (%)')
plt.ylabel('EMG Droit fémoral (Volt)')
plt.legend()
plt.title('EMG Droit fémoral')
plt.show()

# Pré-analyse dataset
Biceps_Femoral= RightIJMean
Droit_Femoral=RightRFMean
Vaste_lateral=RightVLMean
Angle_droit=np.delete(RightKneeAnglesMean,[1,2],1)
Moments_droit=np.delete(RightKneeMomentsMean,[1,2],1) # On supprime les colonnes [1,2] , 1: colonne et si on met 0 on supprime les lignes 1 et 2
Moments_droit=np.multiply(Moments_droit, 10**-3) ## On normalise Nm/kg les données

```

Annexe : Programme EMG

```
## Affichage de la base de donnée Moment
plt.scatter(Droit_Femoral,Moments_droit)
plt.xlabel('EMG (Volt)')
plt.ylabel('Moment du genou normalisé (N.m/kg)')
plt.plot()
plt.title('Dataset EMG Droit Fémoral et Moment droit du genou normalisé')
plt.show()

plt.scatter(Biceps_Femoral,Moments_droit)
plt.xlabel('EMG (Volt) ')
plt.ylabel('Moment du genou normalisé (N.m/kg)')
plt.plot()
plt.title('Dataset EMG Biceps Fémoral et Moment droit du genou normalisé')
plt.show()

plt.scatter(Vaste_lateral,Moments_droit)
plt.xlabel('EMG (Volt)')
plt.ylabel('Moment du genou normalisé (N.m/kg)')
plt.plot()
plt.title('Dataset EMG Vaste latéral et Moment droit du genou normalisé')
plt.show()

## Affichage de la base de donnée Angles
plt.scatter(Droit_Femoral.Angle_droit)
plt.xlabel('EMG ')
plt.ylabel('Angle du genou')
plt.plot()
plt.show()

plt.scatter(Biceps_Femoral,Angle_droit)
plt.xlabel('EMG ')
plt.ylabel('Angle du genou')
plt.plot()
plt.show()

plt.scatter(Vaste_lateral,Angle_droit)
plt.xlabel('EMG ')
plt.ylabel('Angle du genou')
plt.plot()
plt.show()
```

Annexe : Programme EMG

```
## On crée la matrice X avec le modèle tau = u^a*exp(b-u*c)
X_1=np.hstack(((np.log(Biceps_Femoral), np.ones(Biceps_Femoral.shape), -Biceps_Femoral ))) # ln(u) / 1
/-u
print(X_1)
X_2=np.hstack(((np.log(Droit_Femoral), np.ones(Droit_Femoral.shape), -Droit_Femoral ))) # ln(u) / 1 /
-/u
print(X_2)
X_3=np.hstack(((np.log(Vaste_lateral), np.ones(Vaste_lateral.shape), -Vaste_lateral ))) # ln(u) / 1 /
-/u
print(X_3)
print(X_3.T)

## On crée la matrice X avec le modèle tau = a + bu^2 + c*exp(u)
X_12=np.hstack((( np.ones(Biceps_Femoral.shape), Biceps_Femoral**2, np.exp(Biceps_Femoral) )))
X_22=np.hstack((( np.ones(Droit_Femoral.shape), Droit_Femoral**2, np.exp(Droit_Femoral) )))
X_32=np.hstack((( np.ones(Vaste_lateral.shape), Vaste_lateral**2, np.exp(Vaste_lateral) )))

##Modèle polynomial
X_13=np.hstack((( Biceps_Femoral**4 , Biceps_Femoral**3, Biceps_Femoral**2, Biceps_Femoral,
np.ones(Biceps_Femoral.shape) )))
X_23=np.hstack((( Droit_Femoral**4 , Droit_Femoral**3, Droit_Femoral**2, Droit_Femoral,
np.ones(Droit_Femoral.shape) )))
X_33=np.hstack((( Vaste_lateral**4 , Vaste_lateral**3, Vaste_lateral**2, Vaste_lateral,
np.ones(Vaste_lateral.shape) )))
```

Annexe : Programme EMG

```

## ## On crée le vecteur theta
theta = np.random.randn(3,1) # [a,b,c]
print(theta)
## Modèle

def F_expo(X,theta): # F = exp(X * \theta) ou tau = u^a*exp(b-u*c)
    return np.exp(X.dot(theta))

def modèle(X,theta): # tau = a + bu^2 + c*exp(u) + Modèle polynomial
    return X.dot(theta)

## Fonction MSE

def MSE_1(X,y,theta): # tau = u^a*exp(b-u*c)
    m= len(y)
    return 1/(2*m)*np.sum((F_expo(X,theta) - y)**2)

def MSE_2(X,y,theta): # tau = a + bu^2 + c*exp(u) + Modèle polynomial
    m= len(y)
    return 1/(2*m)*np.sum((modèle(X,theta) - y)**2)

## Gradient

def grad_1(X, y, theta):
    m = len(y)
    return 1/m * X.T.dot(F_expo(X, theta) - y)

def grad_2(X, y, theta):
    m = len(y)
    return 1/m * X.T.dot(modèle(X, theta) - y)

## Descente du gradient 1: u^a*exp(b-u*c)
def gradient_descent_1(X, y, theta, learning_rate, n_iterations):
    # création d'un tableau de stockage pour enregistrer l'évolution du Cout du modèle
    Cout_temporel = np.zeros(n_iterations)

    for i in range(0, n_iterations):
        theta = theta - learning_rate * grad_1(X, y, theta) # mise à jour du paramètre theta (formule du gradient descent)
        Cout_temporel[i] = MSE_1(X, y, theta) # on enregistre la valeur du Cout au tour i dans Erreur_temporel[i]

    return theta, Cout_temporel

```

Annexe : Programme EMG

```
## Descente du gradient 2: a + bu^2 + c*exp(u) + Modèle polynomial
def gradient_descent_2(X, y, theta, learning_rate, n_iterations):
    # création d'un tableau de stockage pour enregistrer l'évolution du Cout du modèle
    Erreur_tempo = np.zeros(n_iterations)

    for i in range(0, n_iterations):
        theta = theta - learning_rate * grad_2(X, y, theta) # mise à jour du parametre theta (formule
        du gradient descent)
        Erreur_tempo[i] = MSE_2(X, y, theta) # on enregistre la valeur du Cout au tour i dans
        Erreur_tempo[i]

    return theta, Erreur_tempo
## R2
def rcarre(y, pred):
    u = ((y - pred)**2).sum() #residu de la somme des carres
    v = ((y - y.mean())**2).sum()# somme totale des carres
    return 1 - u/v

## Indice mini d'une liste
def Indice_min_liste(L):
    N=len(L)
    min=L[0]
    indice_min=0
    for i in range (0,N):
        if min>=L[i]:
            min=L[i]
            indice_min=i
    return indice_min
```

Annexe : Programme EMG

```
## Meilleur learning rate
def meilleur_learning_rate(x,y,learning_rate_max, n_iteration):
    i=0.001
    L_ecart=[]
    L_learning=[]
    X=np.hstack(((np.log(x), np.ones(x.shape), -x ))) # ln(u) / 1 / -u
    theta=np.random.rand(3,1)

    while i < learning_rate_max:
        theta_final=gradient_descent_l(X, y, theta, learning_rate, n_iterations)[0]
        predictions = F_expo(X, theta_final)
        if rcarre(y, predictions) > 0:
            L_ecart.append([1-rcarre(y, predictions)])
            L_learning.append(i)
        i+=0.001
    Indice = Indice_min_liste(L_ecart)
    return L_learning[Indice] #
meilleur_learning_rate(x=Droit_Femoral,y=Moments_droit,learning_rate_max = 1, n_iteration =3000)

## Analyse final modèle a + bu^2 + c*exp(u) (Descente du gradient)
n_iterations = 1000
learning_rate = 0.001

theta_final, Erreur_tempo = gradient_descent_l(X_22, Moments_droit, theta, learning_rate,
n_iterations)

print(theta_final) # voici les paramètres du modèle une fois que la machine a été entraînée

# création d'un vecteur prédictions qui contient les prédictions de notre modèle final
predictions = modèle(X_22, theta_final)

# Affiche les résultats de prédictions (en rouge) par rapport à notre Dataset (en bleu)
plt.scatter(Droit_Femoral, Moments_droit)
plt.plot(Droit_Femoral, predictions, c='r')
plt.show()
```

Annexe : Programme EMG

```
## Analyse final modèle a + bu^2 + c*exp(u) (Descente du gradient)
n_iterations = 1000
learning_rate = 0.001

theta_final, Erreur_tempo = gradient_descent_1(X_22, Moments_droit, theta, learning_rate,
n_iterations)

print(theta_final) # voici les paramètres du modèle une fois que la machine a été entraînée

# création d'un vecteur prédictions qui contient les prédictions de notre modèle final
predictions = modèle(X_22, theta_final)

# Affiche les résultats de prédictions (en rouge) par rapport à notre Dataset (en bleu)
plt.scatter(Droit_Fémoral, Moments_droit)
plt.plot(Droit_Fémoral, predictions, c='r')
plt.show()

## Analyse final modèle tau = u^a*exp(b-u*c) (Descente du gradient)
n_iterations = 3000
learning_rate = 0.001

theta_final, Erreur_tempo = gradient_descent_1(X_2, Moments_droit, theta, learning_rate, n_iterations)

print(theta_final) # voici les paramètres du modèle une fois que la machine a été entraînée

# création d'un vecteur prédictions qui contient les prédictions de notre modèle final
predictions = F_expo(X_2, theta_final)

# Affiche les résultats de prédictions (en rouge) par rapport à notre Dataset (en bleu)
plt.scatter(Droit_Fémoral, Moments_droit)
plt.plot(Droit_Fémoral, predictions, c='r')
plt.xlabel('EMG (Volt)')
plt.ylabel('Moment du genou normalisé (N.m/kg)')
plt.title('Dataset EMG Droit Fémoral et Moment droit du genou normalisé')

plt.show()

print('R2=', rcarre(Moments_droit, predictions))

plt.plot(range(3000), Erreur_tempo) # Evolution de l'erreur
plt.show()
```

Annexe : Programme EMG

```

## Analyse final modèle polynomial (Descente du gradient)
theta = np.random.randn(5,1) # [a,b,c]

n_iterations = 10000
learning_rate = 1.5 # Avec 1.5 on a R2= 0.73695233867375076

theta_final, Erreur_tempo = gradient_descent_2(X_23, Moments_droit, theta, learning_rate,
n_iterations)

print(theta_final) # voici les paramètres du modèle une fois que la machine a été entraînée

# Crée un vecteur prédictions qui contient les prédictions de notre modèle final
predictions = modèle(X_23, theta_final)

# Affiche les résultats de prédictions (en rouge) par rapport à notre Dataset (en bleu)
plt.scatter(Droit_Fémoral, Moments_droit)
plt.plot(Droit_Fémoral, predictions, c='r')
plt.xlabel('EMG (Volt)')
plt.ylabel('Moment du genou normalisé (N.m/kg)')
plt.title('Dataset EMG Droit Fémoral et Moment droit du genou normalisé')
plt.show()
print('R2=', rcarre(Moments_droit, predictions))

```

Annexe : Programme EMG

```

## Déterminer le degré du modèle polynomial le plus optimal (Descente de gradient)
def polynome_opti_grad(n,x,y,learning_rate,n_iterations): # n degré max du polynôme
    X=np.array(np.ones(x.shape))
    L_ecart=[]
    L_degre=[]
    L_r2=[]
    theta=np.random.randn(2,1)

    for i in range(1,n+1):
        X=np.hstack((X*x**i, X))

        theta_final=gradient_descent_2(X, y, theta, learning_rate, n_iterations)[0]
        predictions = model(X, theta_final)
        if rcarre(y, predictions) > 0:
            L_ecart.append([1 - rcarre(y, predictions)])
            L_degre.append(i)
            L_r2.append(rcarre(y, predictions))

    min_ecart=Indice_min_liste(L_ecart)
    degré_poly_opti=L_degre[min_ecart]
    P = creation_polygone(x, degré_poly_opti)
    return degré_poly_opti, P, L_ecart, L_degre, L_r2 #
polynome_opti_grad(10,Droit_Fémoral,Moments_droit, 1.5, 10000)

## Polynome création
def creation_polygone(x,n):
    L=[]
    X=np.array(np.ones(x.shape))
    for i in range (1,n+1):
        X=np.hstack((X*x**i, X))
        L.append(i)
    return X

```

Annexe : Programme EMG

```

##Equation normale final Droit_Femoral (Sans polynome opti)
X_p=np.hstack((( Droit_Femoral**8, Droit_Femoral**7, Droit_Femoral**6, Droit_Femoral**5,
Droit_Femoral**4, Droit_Femoral**3, Droit_Femoral**2, Droit_Femoral, np.ones(Droit_Femoral.shape) )))

#X_p=np.hstack(( Droit_Femoral**2, Droit_Femoral, np.ones(Droit_Femoral.shape) ))

theta_final_normale= np.linalg.pinv(X_p.T.dot(X_p)).dot(X_p.T).dot(Moments_droit)

predictions = modele(X_p, theta_final_normale)
plt.scatter(Droit_Femoral, Moments_droit)
plt.plot(Droit_Femoral, predictions, c='r')
plt.xlabel('EMG (Volt)')
plt.ylabel('Moment du genou normalisé (N.m/kg)')
plt.title('Dataset EMG Droit Fémoral et Moment droit du genou normalisé')
plt.show()

print('R2=', rcarre(Moments_droit, predictions))

## Déterminer le degré du modèle polynomial le plus opti (Eq Normale)
def polynome_opti_normal(n,x,y): # n degré max du polynôme
    X=np.array(np.ones(x.shape))
    L_ecart=[]
    L_degre=[]
    L_r2=[]

    for i in range(1,n+1):
        X=np.hstack((x**i, X))

        theta_final=np.linalg.pinv(X.T.dot(X)).dot(X.T).dot(y)
        predictions = modele(X, theta_final)
        if rcarrefy(predictions) > 0:
            L_ecart.append([1-rcarrefy(y, predictions)])
            L_degre.append(i)
            L_r2.append(rcarrefy(y, predictions))

    min_ecart=Indice_min_liste(L_ecart)
    degré_poly_opti= L_degre[min_ecart]
    P = creation_polynome(x, degré_poly_opti)
    return degré_poly_opti , P, L_ecart, L_degre, L_r2 #
polynome_opti_normal(10,Droit_Femoral,Moments_droit)

```

Annexe : Programme EMG

```
## Equation normale final Droit Femoral (Avec polynome opti)
X_p=polynome_opti_normal(101,Droit_Femoral,Moments_droit)[1]
theta_final_normale= np.linalg.pinv(X_p.T.dot(X_p)).dot(X_p.T).dot(Moments_droit)

predictions = modele(X_p, theta_final_normale)
plt.scatter(Droit_Femoral, Moments_droit)
plt.plot(Droit_Femoral, predictions, c='r')
plt.xlabel('EMG (Volt)')
plt.ylabel('Moment du genou normalisé (N.m/kg)')
plt.title('Dataset EMG Droit Femoral et Moment droit du genou normalisé')
plt.show()

print('R2=', rcarre(Moments_droit, predictions))
|
## Représentation final Moment predit et moment réel
plt.plot(range(101), predictions, 'r', label='Prédiction du couple' )
plt.plot(range(101),RightKneeMomentsMean[:,0]*10**-3,'b', label='Couple Moyen droit')
plt.title('Moment du Genou droit')
plt.xlabel('Cycle de marche (%)')
plt.ylabel('Moment du Genou droit Normalisé (N.m/kg)')
plt.legend()
plt.show()
```

Annexe : Références bibliographiques

Références bibliographiques :

- ① Ameli, Santé,
<https://www.ameli.fr/assure/sante/themes/sclerose-en-plaques>
- ② Pôle Saint Hélier, Information sur le pôle saint Helier,
<https://www.pole-sthelier.fr/pole-saint-helier/actualites/65-lexosquelette-keeogo-est-arrive.html>
- ③ Carl Schmitt, Orthèses fonctionnelles à cinématique parallèle et
sérieuse pour la rééducation des membres inférieurs, PRÉSENTÉE le
20 avril 2007 (Thèse marche)
- ④ Guillaume Saint-Cirgue, Apprendre_le_ML_en_une_semaine, 2019

Annexe : Références bibliographiques

Références bibliographiques :

- ① Saber MEFOUED, Commande Robuste Référence Intention d'une Orthèse Active pour l'Assistance Fonctionnelle aux Mouvements du Genou, Soutenue le : 12/12/2012 (Orthèse Lissi)
- ② Khalil Ullah, Asif Khan, Ihtesham-ul-Islam and Mohammad A. U. Khan Electromyographic (EMG) signal to joint torque processing and effect of various factors on EMG to torque model, Accepted 8 November, 2011 (Modèle signaux EMG)
- ③ Viet Anh Dung Cai. Contribution à l'étude d'exosquelettes isostatiques pour la rééducation fonctionnelle, application à la conception d'orthèses pour le genou.. Automatique / Robotique. Université Pierre et Marie Curie - Paris VI, 2011. Français. tel-00641503 (Interet des signaux EMG)