La marche : visualisation et mesure de la dissymétrie du corps humain

Comment visualiser et mesurer la dissymétrie du corps humain pendant la marche ?

La marche

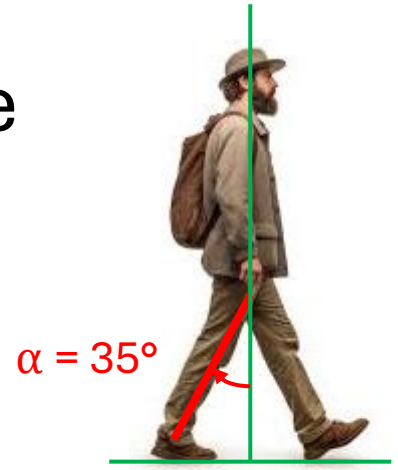
- *Transition* écologique

MINISTÈRE
DE LA TRANSITION
ÉCOLOGIQUE
Liberté
Égalité
Fraternité

- *Transformation* du mouvement

translation

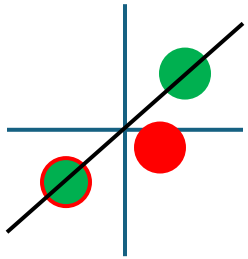
- *Conversion* de la position angulaire en information numérique



© Ministère de la transition écologique, Freepik

La dissymétrie du corps

- *Définition* (non-symétrie)



- *Grandeur* observable et mesurable

- *Permet la correction* de défauts posturaux pendant la marche

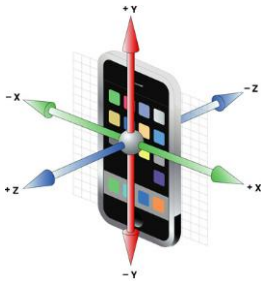
© Freepik

Choix du système expérimental

- *Critères* : coût, performance, utilisation facile

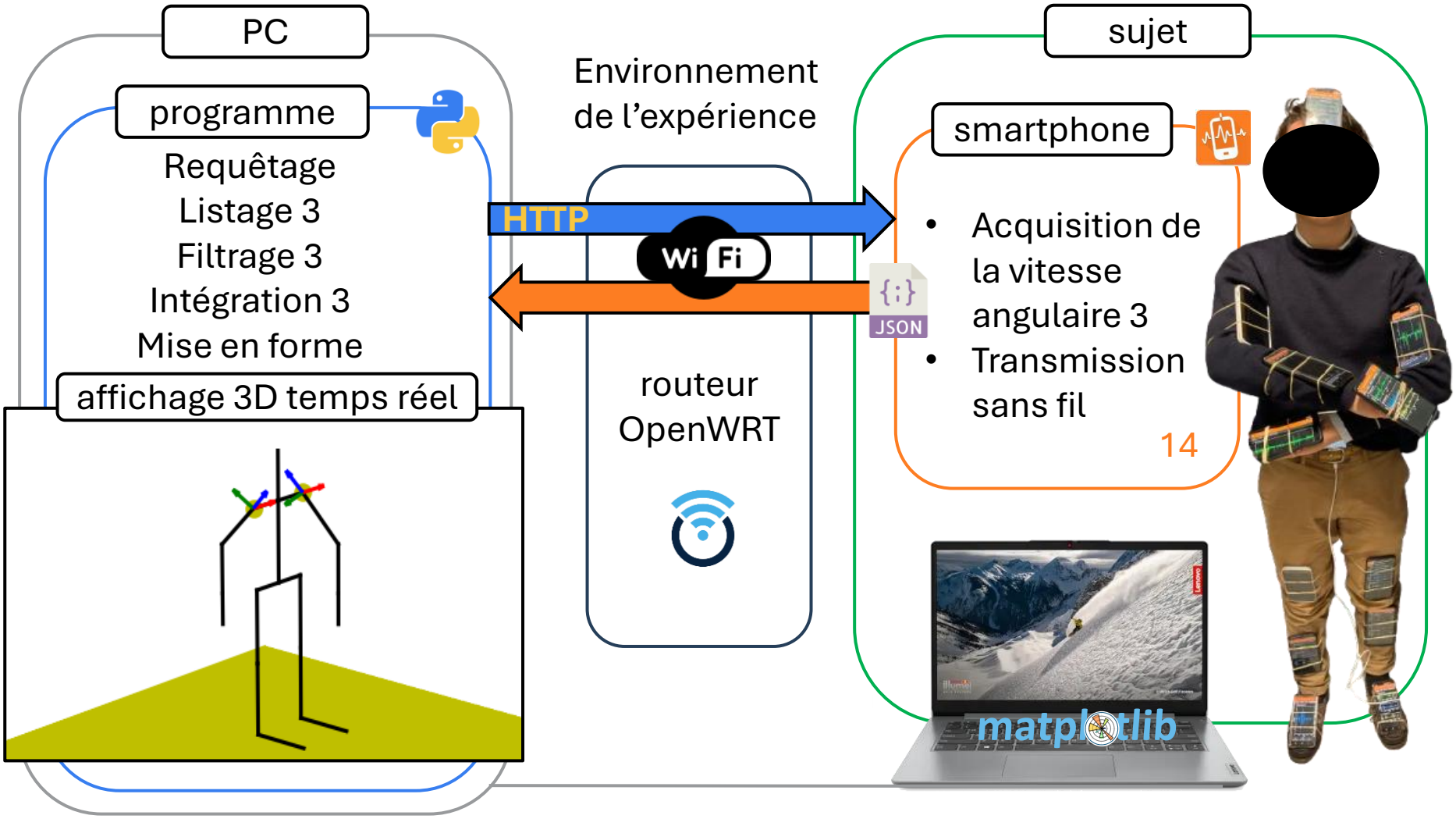
➢ *Capteurs* standards ou smartphones phyphox

➢ *Calcul et affichage* embarqués ou déportés



© Lenovo, SweetRandomScience, Phyphox

Fonctionnement expérimental



© Istock, Matplotlib, OpenWRT, Lenovo

Structure code-expérience

- Bibliothèques
- Fonctions
- Classes
- Script d'affichage en temps réel parallèle aux calculs en continu

```
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import requests as r
import threading as th
import numpy as np
```

requete()
integrale()
decomp_vecteurs()
rotations()
dissymetrie()
epsilon()
moy_gliss()
cpt
sim
pt
sld
rep
animate()

Récupération des données

- *Requête* trame http  http://192.0.0.1/get?cmd=gyr_time&gyrX&gyrY&gyrZ
- *Réponse* fichier json 
- *Gestion* des erreurs 

```python
def requete(IPPORT,souhait=["gyr_time","gyrX","gyrY","gyrZ"]):
    demande,reponse = "",[]
    for k in souhait: demande+=k+"&"
    requete = "http://"+IPPORT+"/get?"+demande
    try:
        data = r.get(url=requete).json()
        for k in souhait:
            reponse += [data["buffer"][k]["buffer"][0]]
    except Exception as e: pass # print(e)
    return reponse
```

- *Performance* : 25±15ms/requête   réseau (wifi 6) : routeur, PC et smartphone

Filtrage des données de vitesse

- *Filtre passe-bande* (deux filtres successifs)

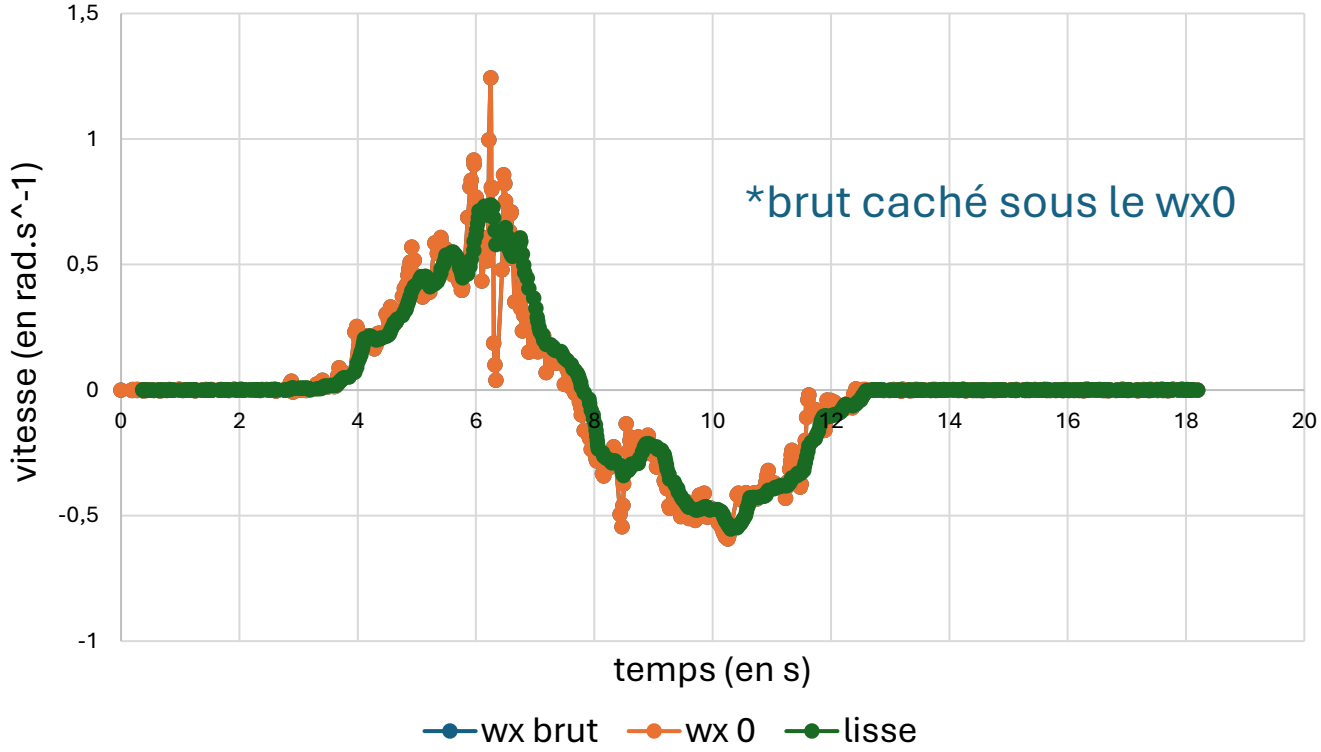
➢ *Saturation* (~passe haut) $\epsilon(x) = \begin{cases} 0 \text{ si } |x| < 10^n, n \in N \\ x \; sinon \end{cases}$

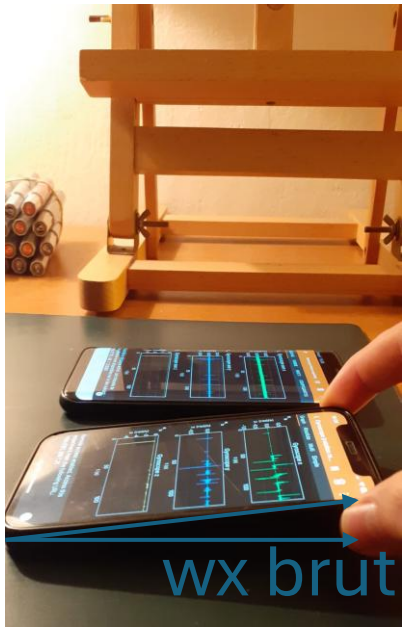
➢ *Moyenne glissante* (passe bas)

```
def moy_gliss(d,L,n=10):
    l = len(L)
    if l == n:
        for i in range(1,n-1): L[i+1]=L[i]
        L[1]= d
        return sum(L)/l
    else:
        L.append(d)
        return sum(L)/(l+1)
```

Graphe du filtrage des données

- 18s d'essai en déplaçant angulairement un smartphone-capteur sur un axe



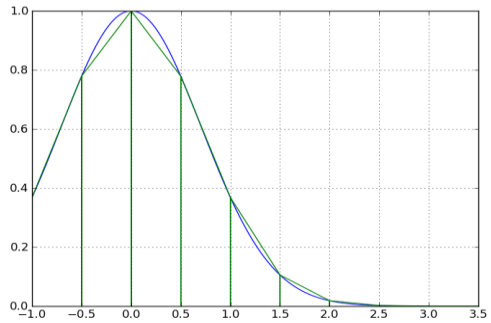


© Excel

Equations horaires et intégrations

- *Transformation* : vitesse angulaire en position angulaire

$$\int \omega = \alpha, (\omega\ en\ rad.s^{-1}\ et\ \alpha\ en\ rad)$$



- *Intégration* numérique : méthode des trapèzes

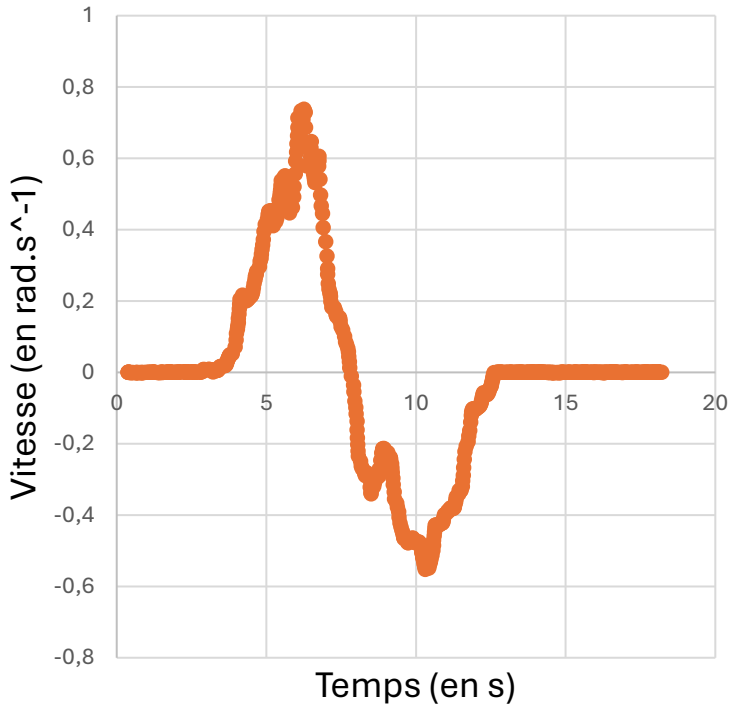
```
def integrale(L,Lt,V_init=0): return V_init + np.trapezoid(L,x=Lt)

Lt.append(R[0]),LW.append(R[-3:])
self.A = [integrale([LW[-2][coord],LW[-1][coord]],Lt[-2:],self.A[coord]) for coord in range(3)]
```

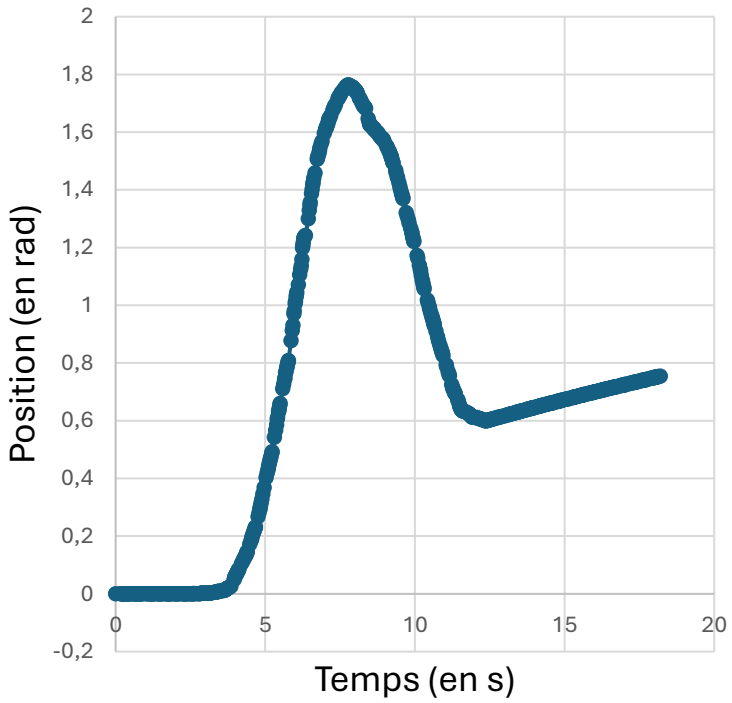
- *Performance* : 17±0,2ms
  AMD Ryzen 5800u @4.4Ghz

```
time.perf_counter()
```

Graphes des équations horaires

- Vitesse angulaire, intégration, position angulaire (essai angulaire comme précédemment)



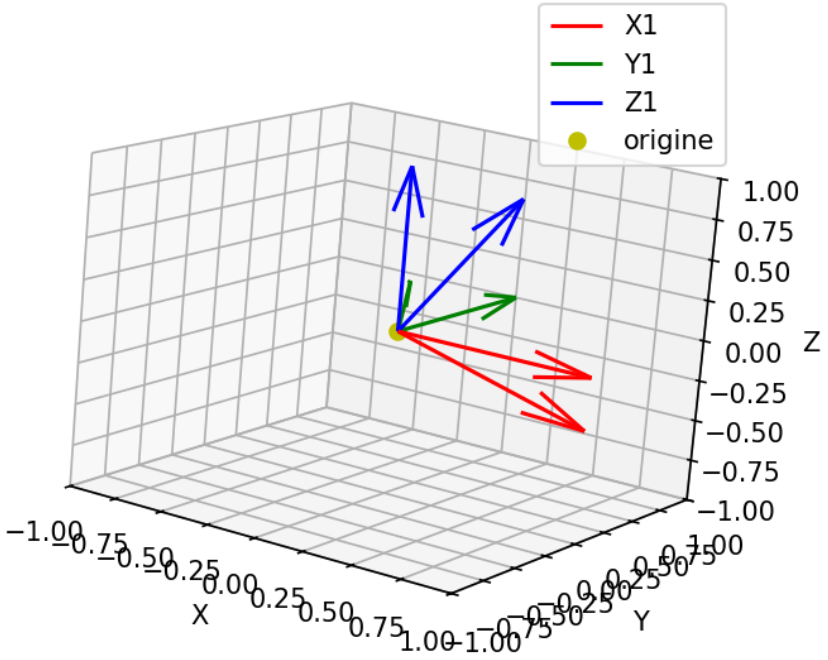


Représentation position angulaire

- *Repère orthonormé* pour chaque capteur

$$M = R_\theta P^{-1} R_\phi P$$

$$\begin{pmatrix} \cos\phi & 0 & 0 \\ -\sin\theta\cos\phi & \cos\theta & -\sin\theta\cos\phi \\ \cos\theta\sin\phi & \sin\theta & \cos\phi\cos\theta \end{pmatrix}$$

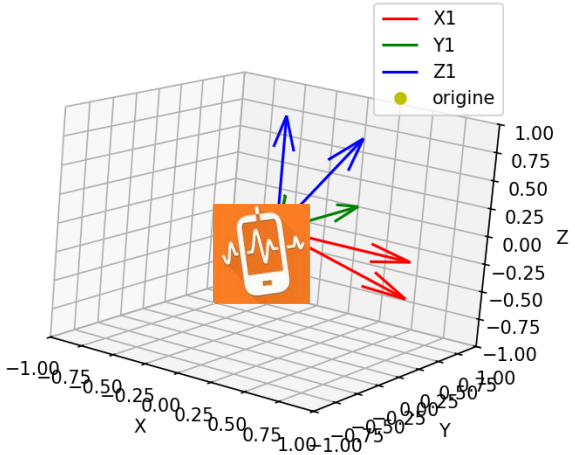


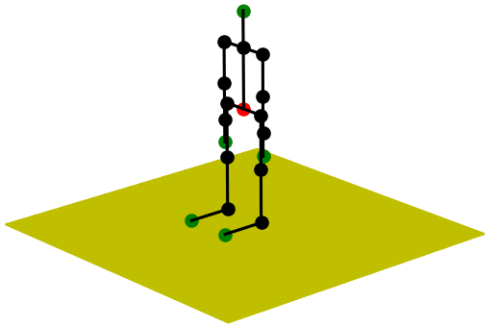

```
M = rotations(self.A[0],self.A[1])
e = 0.15
ax_x = M @ np.array([1, 0, 0]) * e
ax_y = M @ np.array([0, 1, 0]) * e
ax_z = M @ np.array([0, 0, 1]) * e
```

Représentation position spatiale

- Changement de coordonnées : sphériques à cartésiennes

```
X = [x,y,z] # position 3D
A = [ax,ay,az] # postion angulaire
W = [wx,wy,wz] # vitesse angulaire
```





$$pos_x = x_p + N\cos(\alpha_1)\cos(\alpha_2)$$

$$pos_y = y_p + N\cos(\alpha_1)\sin(\alpha_2)$$

$$pos_z = z_p + N\sin(\alpha_1)$$

```
def decomp_vecteurs(point_liaison,norme,a1,a2):
    x = point_liaison.X[0] + norme*np.cos(a1)*np.cos(a2)
    y = point_liaison.X[1] + norme*np.cos(a1)*np.sin(a2)
    z = point_liaison.X[2] + norme*np.sin(a1)
    return [x,y,z]
```

Calcul des positions spatiales

- *Cascade* de calculs de position des liaisons à partir d'une position initiale fixe : le bassin

$$Composition\ des\ mouvements\ \forall n \in N,\ \overrightarrow{A(n+1/0)} = \overrightarrow{A(n+1/n)} + \overrightarrow{A(n/0)}$$

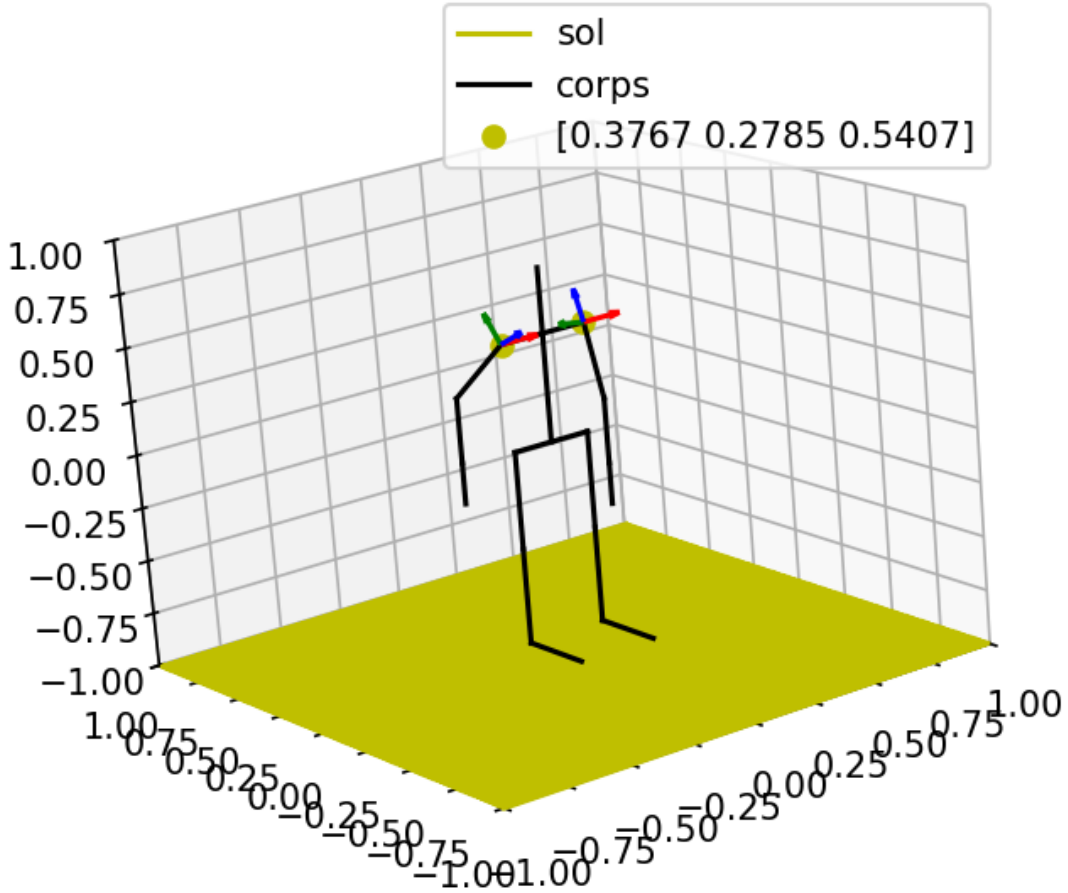
```
bas_dos = pt([0,0,0],"ro")
haut_dos = pt([bas_dos.X[0],bas_dos.X[1],bas_dos.X[2]+dos])
colonne = sld(bas_dos,haut_dos)
```

- *Construction et mise à jour temps réel* squelette

$$\forall k_{capteur} \in Squelette_n\ ,on\ a:\ X_k = \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix}, \mathrm{A}_k = \begin{pmatrix} \theta_k \\ \phi_k \\ \gamma_k \end{pmatrix} et\ les\ extrémités$$

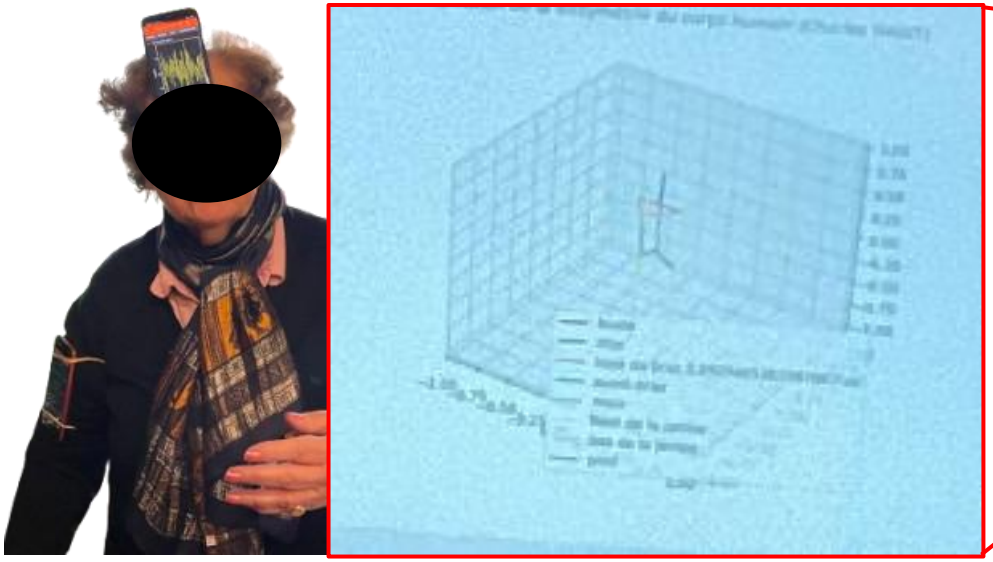
Résultat de la visualisation

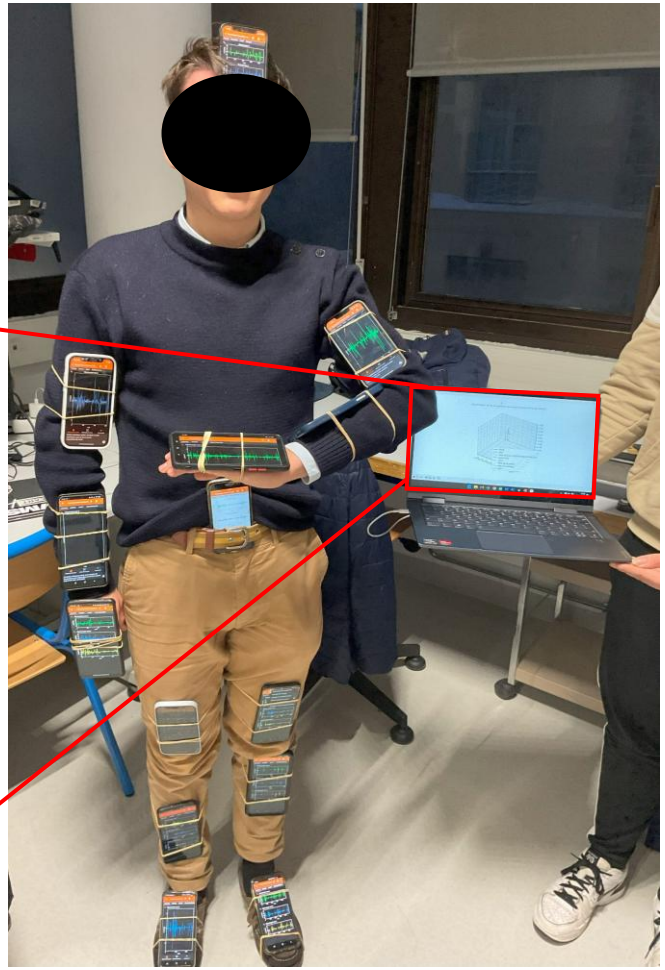
- *Essai* d'observation des épaules



Utilisation de la visualisation

- *Visualisation en temps réel* d'une position fixe après un déplacement du corps





Mesure de la dissymétrie

- *Comparaison* entre deux angles

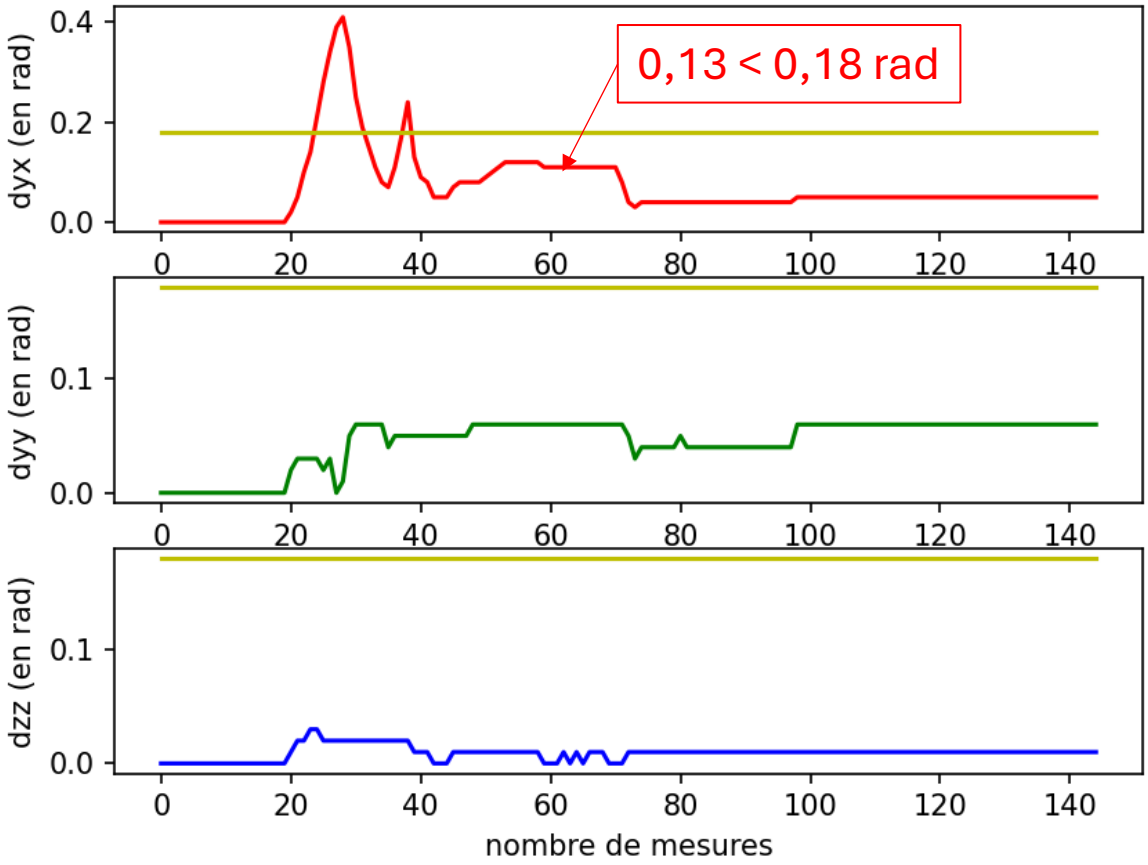
  $D = |\alpha_g - \alpha_d|$ avec (α_g et α_d en rad)

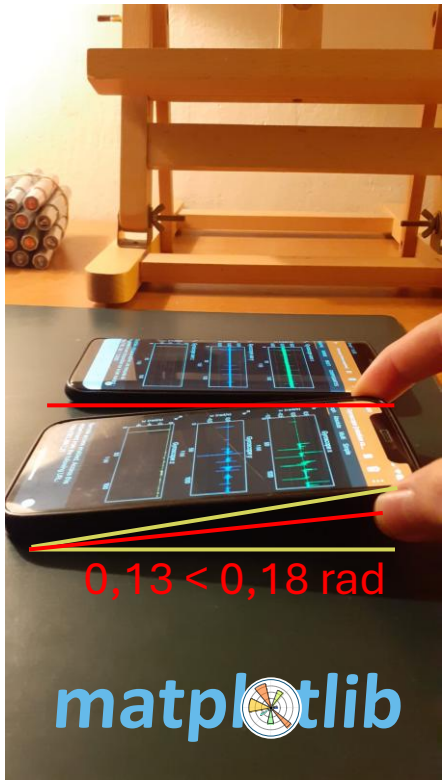
- *Définition* du critère de performance :

  expérimentalement environ 10° (0,18 rad) sur chaque angle caractéristique

Relevé de la dissymétrie

- *Essai* réalisé en temps réel (2 capteurs)



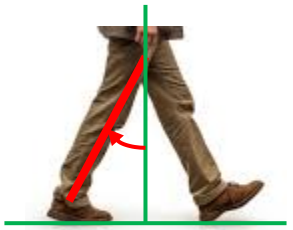


Simulation de la marche idéale ?

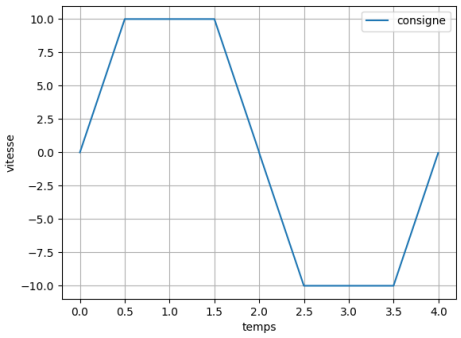
- *Mesure* des maxima et minima angulaires de chaque liaison sphérique

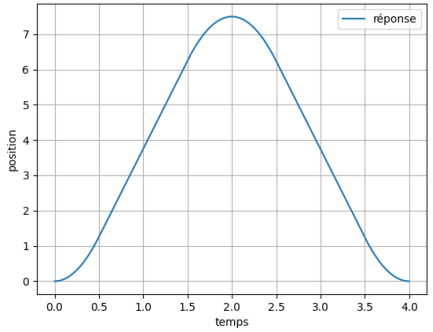
- *Synchronisation* corporelle

réglage
amplitude
temps



- Création des *consignes* trapèzes de vitesse et classe python mimant la réponse capteur



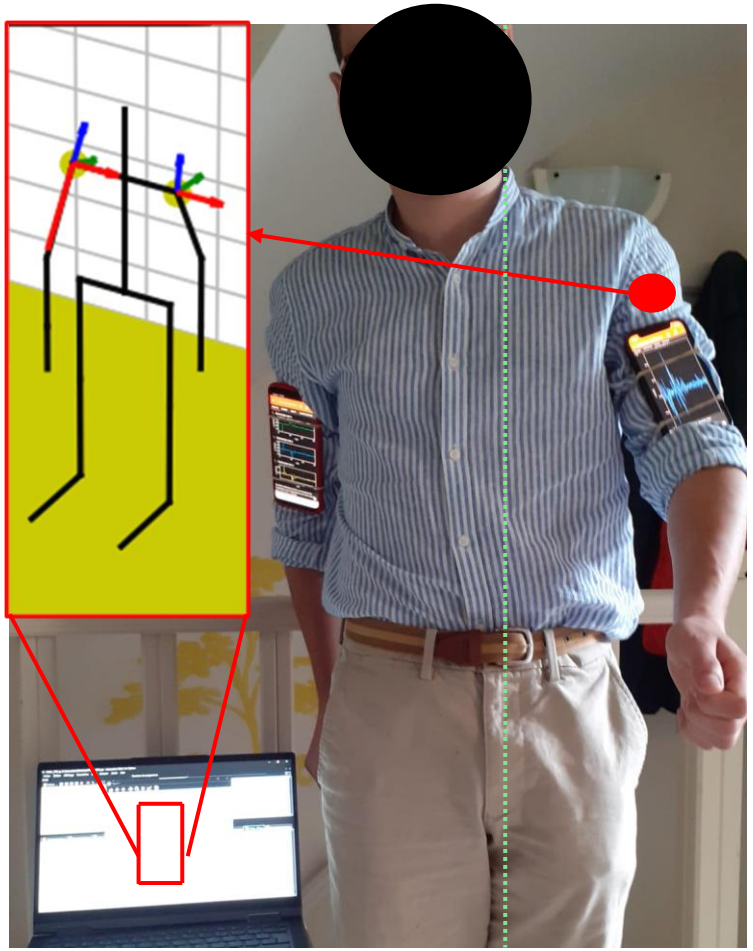


Création du guide correctif

- *Simulation* par symétrie axiale (suivi guide rouge par le bras avant)

- *Unique calcul* par axe (pas de pertes de perf.)

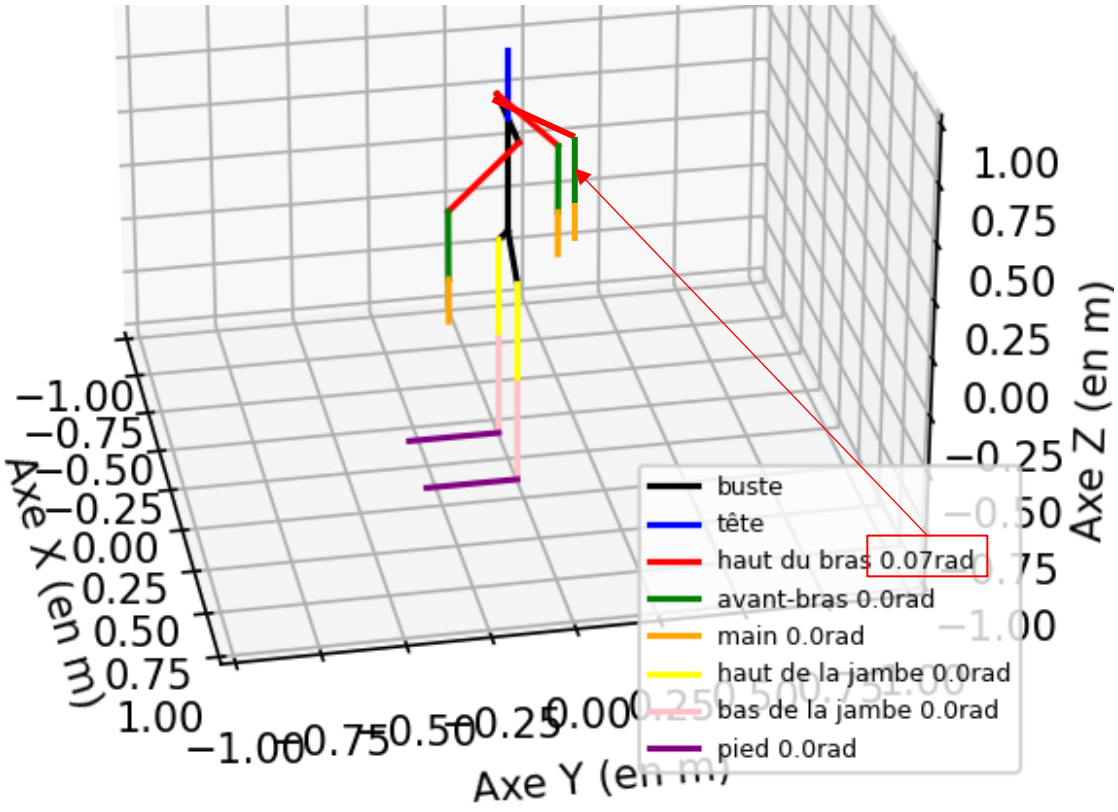
➢ *Copie* classe python



```
coude_g_bis.X = decomp_vecteurs(epaule_g_bis,hautbras,-copie.A[0]-np.pi/2,copie.A[1]+np.pi/2)
```

Utilisation du système expérimental

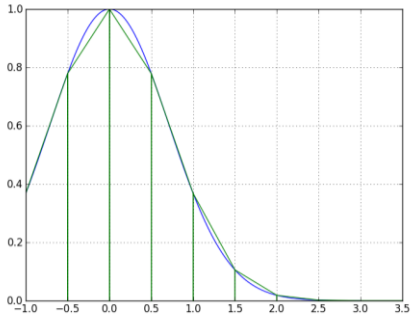
- Correction écart : réel + simulation bras gauche



Ecarts : réel, mesure et simulation

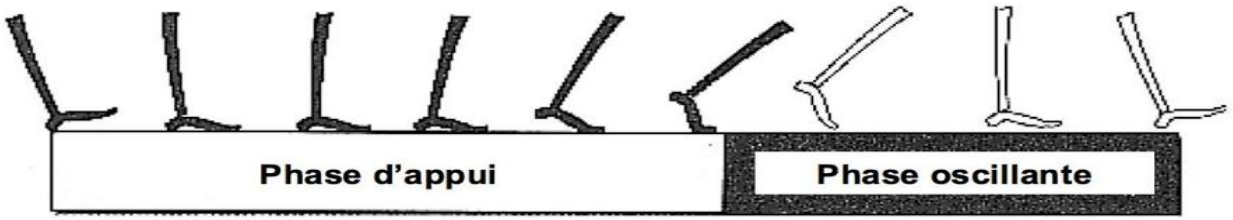
- Non symétrie de la marche réelle
- Dérive des capteurs (non-retour)
- Calcul numérique (discontinuité)
- Lenteur du système avec 14 téléphones (4 fps)

OdG relevé
1 minute
20Hz



Améliorations et conclusion

- Performances expérimentales
- Fixation physique des capteurs
- Modèle biomécanique de la marche



- Retour aux objectifs & problématique

© David Gasq

Annexes

- Codes : visualisations, mesures, simulations
- Démonstrations : affichages 3D
- Etc…