Partie 1 - Sciences de l'ingénieur

Robot humanoïde

CORRIGÉ

Sous-partie 1

Question 1.1 **Trois éléments d'interactivité :**

- **deux caméras figurent des yeux et permettent d'effectuer de la reconnaissance faciale ou d'objet ;**
- **deux antennes animées permettent de traduire un état émotionnel ;**
- **les mouvements réalisables évoquent ceux d'un humain.**

Question 1.2 **Voir le tracé de $\ddot{\varphi}_{\text{tête}}(t)$ pour la loi ❶ sur le document réponse DR1**.

DR1

Éléments de calculs :

- d'après le chronogramme de la vitesse ❶ du DR1 **pour 0 s ≤ t < 0,3 s**, $\omega_{\text{tête}}$ croît linéairement si bien que $\ddot{\varphi}_{\text{tête}}$ est constante, de valeur :

$$\ddot{\varphi}_{\text{tête}} = \frac{\mathrm{d}\omega_{\text{tête}}}{\mathrm{d}t} = \frac{4{,}7}{0{,}3} = \mathbf{15{,}7\ rad{\cdot}s^{-2}}\ ;$$

- pour **0,3 s ≤ t < 0,6 s**, $\omega_{\text{tête}}$ est constante et $\ddot{\varphi}_{\text{tête}} = \mathbf{0\ rad{\cdot}s^{-2}}$;
- pour **0,6 s ≤ t < 0,9 s**, $\omega_{\text{tête}}$ décroît linéairement et $\ddot{\varphi}_{\text{tête}}$ est constante de valeur :

$$\ddot{\varphi}_{\text{tête}} = \frac{\mathrm{d}\omega_{\text{tête}}}{\mathrm{d}t} = \frac{-4{,}7}{0{,}3} = \mathbf{-15{,}7\ rad{\cdot}s^{-2}}.$$

La loi de mouvement ❶ ne correspond pas à un mouvement fluide car l'accélération correspondante présente des discontinuités aux instants 0 s, 0,3 s, 0,6 s et 0,9 s.

La loi de mouvement ❷ correspond à un mouvement fluide car l'accélération correspondante ne présente pas de discontinuité.

Question 1.3 **Étapes de la démonstration :**

a) D'après l'énoncé $\boldsymbol{k \times I_m - k_f \times \omega_m - C_{\text{sec}} = 0}$, d'où :

$$I_m = \frac{k_f}{k} \times \omega_m + \frac{C_{\text{sec}}}{k}.$$

b) Par **identification avec $I_m = 0{,}001 \times \omega_m + 0{,}04$** (voir la figure 8) il vient :

$$\frac{k_f}{k} = \mathbf{0{,}001}\ \text{A·s·rad}^{-1} \text{ et } \frac{C_{\text{sec}}}{k} = \mathbf{0{,}04}\ \text{A}.$$

Application numérique :

$$k_f = 0{,}001 \times k = 0{,}001 \times 0{,}4 = \mathbf{4 \times 10^{-4}\ N{\cdot}m{\cdot}s{\cdot}rad^{-1}}\ ;$$

$$C_{\text{sec}} = 0{,}04 \times k = 0{,}04 \times 0{,}4 = \mathbf{0{,}016\ N{\cdot}m}.$$

Question 1.4 **Le réglage 2 est le seul réglage qui satisfait les exigences 1.1.3 et 1.1.4 : l'erreur statique est inférieure à 2° et la réponse ne présente pas d'oscillations.**

Le réglage 3 engendre une erreur statique supérieure à 2°.

Le réglage 1 engendre des oscillations.

Question 1.5 L'étude a montré :

- que **les moteurs employés permettent de réaliser des mouvements fluides** (exigence 1.1.2) ;
- que **le réglage 2 de l'asservissement permet de positionner la tête avec précision** (erreur statique inférieure à 2°, exigence 1.1.3) **et sans oscillations** (exigence 1.1.4).

La solution technique utilisée pour mouvoir la tête est validée.

Sous-partie 2

Question 1.6 Voir le diagramme de bloc interne de la chaîne de puissance de la pince complété sur le document réponse **DR2**.

Question 1.7

$$\boldsymbol{\alpha} = 142 \times \frac{18}{36} \mathbf{= 71{,}0°}.$$

$$\boldsymbol{h_{pince}} = 84{,}9 \times \sin(31{,}4°) + 82 \times \sin(71° - 31{,}4°) \mathbf{= 96{,}5\ mm}.$$

h_{pince} > 90 mm, l'exigence 1.2.3 (ouverture minimale de 90 mm) est respectée.

Question 1.8 $\sum \overrightarrow{\mathcal{M}_{O_1}}(\overrightarrow{F_{ext}}) = \vec{0}$, d'où :

$$\overrightarrow{O_1O_1} \wedge \overrightarrow{F_{(0\to2)}} + \overrightarrow{O_1A} \wedge \overrightarrow{A_{(S\to2)}} + \overrightarrow{O_1I} \wedge \overrightarrow{D_{(1\to2)}} = \vec{0} ;$$

$$\Rightarrow \vec{0} + \begin{pmatrix} 49{,}23 \\ 2{,}11 \\ 0 \end{pmatrix} \wedge \begin{pmatrix} \left\|\overrightarrow{A_{(S\to2)}}\right\| \times \cos\beta \\ -\left\|\overrightarrow{A_{(S\to2)}}\right\| \times \sin\beta \\ 0 \end{pmatrix} + \begin{pmatrix} -18 \\ 0 \\ 0 \end{pmatrix} \wedge \begin{pmatrix} 14{,}8 \\ -40{,}7 \\ 0 \end{pmatrix} = \vec{0} ;$$

$$\Rightarrow \begin{pmatrix} 0 \\ 0 \\ -48{,}28 \times \left\|\overrightarrow{A_{(S\to2)}}\right\| \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 732{,}6 \end{pmatrix} = \vec{0}.$$

Sur $\overrightarrow{z_0}$:

$$-48{,}28 \times \left\|\overrightarrow{A_{(S\to2)}}\right\| + 732{,}6 = 0 ;$$

$$\Rightarrow \left\|\overrightarrow{A_{(S\to2)}}\right\| \mathbf{= 15{,}2\ N}.$$

D'après le principe des actions mutuelles : $\overrightarrow{A_{(S\to2)}} = -\overrightarrow{A_{(2\to S)}}$

D'où :

$$\left\|\overrightarrow{\boldsymbol{A_{(2\to S)}}}\right\| = \left\|\overrightarrow{A_{(S\to2)}}\right\| \mathbf{= 15{,}2\ N}.$$

Question 1.9 Quantum :

$$\boldsymbol{q} = \frac{U_{PE}}{2^{24}} = \frac{3}{2^{24}} \mathbf{= 1{,}79 \times 10^{-7}\ V}.$$

Pour $F_{capteur}$ = 5 N :

$$U_d = F_{capteur} \times k_{capteur} = 5 \times 3{,}14 \times 10^{-4} = 1{,}57 \times 10^{-3}\ \text{V} ;$$

$$U = U_d \times A_d = 1{,}57 \times 10^{-3} \times 128 = 0{,}201\ \text{V} ;$$

$N = \boldsymbol{N_{max}} = \mathrm{E}(\frac{U}{q}) \approx$ **1 123 849**. (toute valeur entre 1 118 000 et 1 128 000 acceptée)

Question 1.10 L'étude a montré :

- que **la pince a une ouverture minimale de 96,5 mm** ;
- que **la pince peut exercer une force de serrage suffisante** avec une force de serrage de 15 N ;
- que **l'effort exercé par la pince sur l'objet peut être limité par logiciel**.

La capacité de la pince du robot Reachy à saisir et maintenir un objet de la vie courante sans l'endommager est validée.

Sous-partie 3

Question 1.11 θ**[687] = 195°**.

***r*[687] = 1,01 m**.

Question 1.12 Trame, octet 2 → 7 bits de poids faibles de θ_code[1389] : $101\ 1100_2$.

Trame, octet 3 → 8 bits de poids forts de θ_code[1389] : $1010\ 0110_2$.

$$\theta\_code[1389] = 101\ 0011\ 0101\ 1100_2.$$

Trame, octet 4 → 8 bits de poids faibles de *r_code*[1389] : $0110\ 0001_2$.

Trame, octet 5 → 8 bits de poids forts de *r_code*[1389] : $0000\ 0101_2$.

$$r\_code[1389] = 0000\ 0101\ 0110\ 0001_2.$$

$\theta\_code[1389] = 21340_{10} \rightarrow \theta[1389] = 360 - \theta\_code[1389] / 64 = 26,6°$.

$r\_code[1389] = 1377_{10} \rightarrow r[1389] = r\_code[1389] / 4000 = 0,344$ **m**.

Question 1.13 $$n = 360 / R_{lidar} = 360 / 0,24 = 1500.$$

$$\Delta t = \frac{60}{n \times N_{lidar}} = \frac{60}{1500 \times 600} = 6,67 \times 10^{-5}\text{s}.$$

Question 1.14 5 octets de données par paquet de données.

1 octet de donnée → 10 bits transmis.

Durée totale d'émission d'un paquet de données :

$$\frac{5 \times 10}{D_{liaison}} = \frac{5 \times 10}{1 \times 10^6} = 5 \times 10^{-5}\text{s}.$$

La durée totale d'émission d'un paquet de données, égale à 5×10^{-5}s, est inférieure à l'intervalle de temps $\Delta t = 6,67 \times 10^{-5}$ s qui sépare deux points de mesure consécutifs. La valeur du débit est donc suffisamment élevée pour permettre le transfert des paquets de données avec la résolution et la fréquence de rotation choisies.

Question 1.15 DR3 **Voir** les **instructions en Python** complétées dans le document réponse **DR3**.

L'étude a montré :

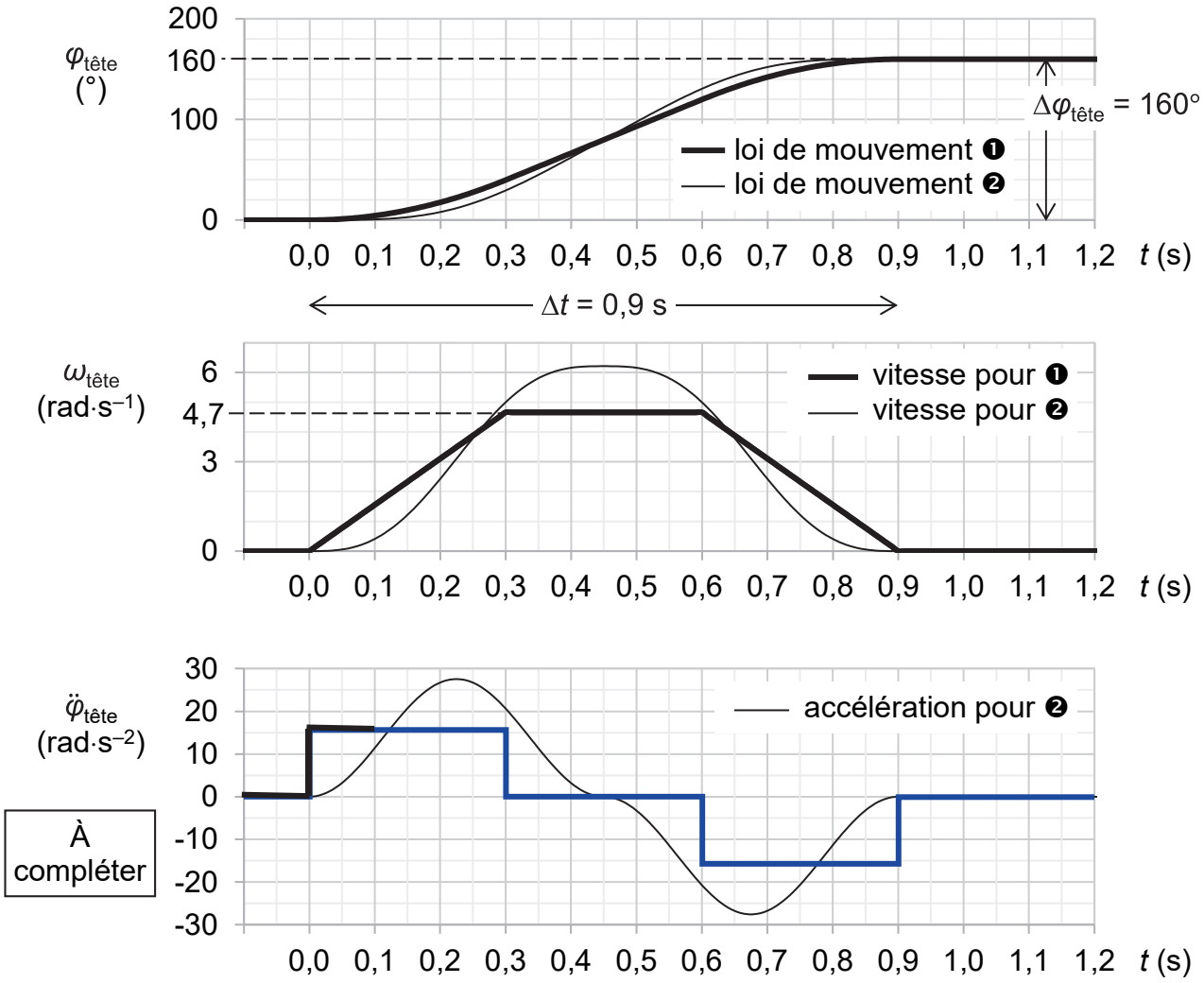
- que le **lidar** permet de localiser les points obstacles avec une **résolution faible** (0,24° soit 1500 points sur 360°) à une **fréquence de balayage élevée** (600 min^{-1} soit 360° en 0,1 s) ;

– que l'**algorithme anticollision détecte les points critiques** et permettra donc d'adapter la vitesse de déplacement **lorsque au moins un point obstacle présente un risque de collision**.

Les solutions matérielle et logicielle mises en œuvre pour éviter les collisions sont validées.

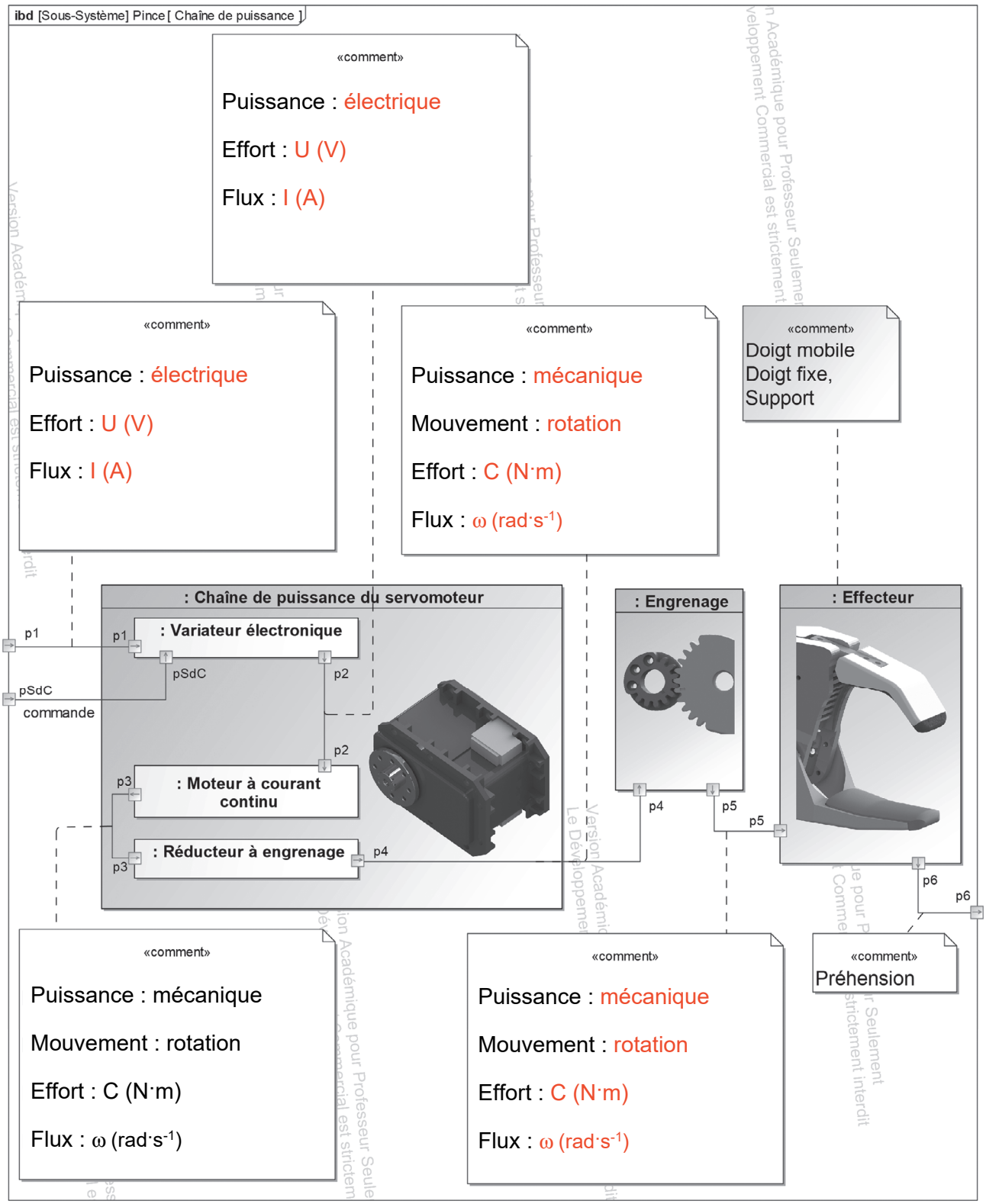
Question 1.2

Tracer l'accélération pour la loi ❶ sur le graphique marqué « à compléter ».



Question 1.6

Les zones à compléter sont identifiées par les pointillés (. . . . .).



Question 1.15

Les zones à compléter sont identifiées par les pointillés (. . . . . .)

```
# n est le nombre de points de mesure du lidar sur 360°.
# La zone critique est un disque de 0,55 m de rayon.
# La zone proche est un anneau de rayons 0,55 m et 0,70 m.
# dist_c = [] crée une liste vide nommée dist_c.
# dist_c.append(x) ajoute l'élément x à la fin dist_c.
# dist_c + dist_p ajoute la liste dist_p à la fin de dist_c.

i = 0 ; n_critiques = 0 ; n_proches = 0
dist_c = [] ; angle_c = [] ; dist_p = [] ; angle_p =[]

while i........ < n........ :
    if dist[i] < 0,55..... :
        dist_c.append(dist[i]) ; angle_c.append(angle[i])
        n_critiques = n_critiques + 1
    elif dist[i] < 0,70..... :
        dist_p.append(dist[i]) ; angle_p.append(angle[i])
        n_proches = n_proches + 1
    i = i + 1
dist = dist_c + dist_p ; angle = angle_c + angle_p
```