MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE LA JEUNESSE
ET DES SPORTS

Liberté
Égalité
Fraternité

Rapport du jury

Concours : AGREGATION EXTERNE

Section : sciences industrielles de l'ingénieur

Option : ingénierie informatique

Session 2023

Rapport de jury présenté par : Alix Munier Kordon, présidente du jury
Professeur des universités

Sommaire

Avant-propos

Le concours 2023 était la septième session de l'agrégation de sciences industrielles de l'ingénieur (SII) de l'option « ingénierie informatique ». Ce rapport est dans la continuité de ceux des sessions précédentes.

L'ingénierie informatique appliquée aux systèmes analyse et résout des problèmes scientifiques et technologiques communs à l'informatique générale et spécifiques à l'informatique embarquée (software et hardware). Ces problèmes s'inspirent de défis sociétaux liés à l'utilisation massive d'objets connectés (IoT) à faibles empreintes. Les spécificités et contraintes sont nombreuses et variées en fonction des applications comme le temps réel, la miniaturisation, l'autonomie et la consommation énergétiques, les interfaces homme-machine, la numérisation de grandeurs analogues à des grandeurs physiques (capteurs), le traitement de la data, les réseaux informatiques et de télécommunication, la cyber sécurité, la fiabilité, la sûreté de fonctionnement, etc.

Les attentes du concours de l'agrégation SII sont définies par l'arrêté du 28 décembre 2009 fixant les sections et les modalités d'organisation des concours de l'agrégation. Les concours de recrutement d'enseignants n'ont pas pour seul objectif de valider les compétences scientifiques et technologiques des candidats ; ils doivent aussi valider les compétences professionnelles qui sont souhaitées par l'État employeur qui recrute des professeurs. L'excellence scientifique et la maîtrise disciplinaire sont indispensables pour présenter le concours, mais pour le réussir, les candidats doivent aussi faire preuve de qualités didactiques et pédagogiques et de bonnes aptitudes à communiquer.

Les trois épreuves d'admissibilité sont construites de manière à évaluer un spectre large de compétences scientifiques et technologiques ; la première épreuve est commune aux quatre options de l'agrégation SII, les deux autres spécifiques à l'option.
Les trois épreuves d'admission sont complémentaires des épreuves d'admissibilité ; la première épreuve d'admission est commune aux quatre options, les deux autres spécifiques à l'option. Elles permettent l'évaluation des compétences pédagogiques des futurs professeurs et s'appuient sur le référentiel des compétences professionnelles des métiers du professorat et de l'éducation (publié au BOEN du 25 juillet 2013). Elles comportent un entretien avec le jury qui permet d'évaluer la capacité du candidat à s'exprimer avec clarté et précision, à réfléchir aux enjeux scientifiques, technologiques, didactiques, épistémologiques, culturels et sociétaux que revêt l'enseignement du champ disciplinaire du concours. Ces épreuves d'admission, dont le coefficient total est le double de celui des épreuves d'admissibilité, ont eu une influence significative sur le classement final.

Les candidats et leurs formateurs sont invités à lire avec application les commentaires et conseils donnés dans ce rapport et dans ceux des sessions antérieures afin de bien appréhender les compétences ciblées. La préparation à ces épreuves commence dès l'inscription au concours. Proposer une séquence pédagogique à partir d'activités expérimentales ne s'improvise pas et nécessite une préparation rigoureuse. De même, la qualité du dossier dépend de la pertinence du choix du support. Elle impose aux futurs professeurs de s'engager, dès le début de leur carrière, dans un processus de rapprochement avec le monde de l'entreprise et de la recherche.
Ces épreuves permettent *« également d'évaluer la capacité du candidat à prendre en compte les acquis et les besoins des élèves, à se représenter la diversité des conditions d'exercice de son métier futur, à en connaître de façon réfléchie le contexte dans ses différentes dimensions (classe, équipe éducative, établissement, institution scolaire, société) et les valeurs qui le portent, dont celles de la République ».*
Les thématiques de la laïcité et de la citoyenneté trouvent toute leur place lors des entretiens avec le jury ; en effet, la mission première que fixe la Nation à ses enseignants est de transmettre et faire partager aux élèves les valeurs et principes de la République ainsi que l'ensemble des dispositions de la charte de la laïcité.

Pour cette septième session, 17 postes sont ouverts pour l'agrégation sciences industrielles de l'ingénieur option ingénierie informatique. Parmi les 152 inscrits, 62 candidats ont été présents aux trois épreuves d'admissibilité. Le nombre d'inscrits est stable par rapport à l'année dernière ; l'édition 2022 avait connu une baisse significative du nombre d'inscrits ; une raison possible pourrait être la création de l'agrégation d'Informatique qui a pu drainer une partie des candidats.

Cette session s'est révélée être, comme les précédentes, d'un très bon niveau. Les candidats ont su démontrer un sens de la pédagogie et une posture professionnelle compatibles avec l'exercice des missions d'enseignant ; le jury les en félicite. Cependant, le jury n'a pas souhaité pouvoir l'ensemble des postes : en effet, 15 postes sur 17 ouverts au concours ont été pourvus. Le jury n'a pas jugé opportun de pourvoir les deux postes restants compte-tenu du niveau académique des candidats suivants.

L'agrégation est un concours prestigieux de recrutement de cadres de catégorie A de la fonction publique qui impose de la part des candidats un comportement et une présentation irréprochables.

Pour conclure cet avant-propos, le jury souhaite que ce rapport soit une aide efficace aux futurs candidats. Tous sont invités à se l'approprier par une lecture attentive.

Remerciements

Le lycée La Martinière Monplaisir à Lyon a accueilli les épreuves d'admission de cette session 2023 des quatre options de l'agrégation externe section sciences industrielles de l'ingénieur.

Les membres du jury tiennent à remercier le proviseur du lycée, ses collaborateurs et l'ensemble des personnels pour la qualité de leur accueil et l'aide efficace apportée tout au long de l'organisation et du déroulement de ce concours qui a eu lieu dans d'excellentes conditions. Nous remercions en particulier chaleureusement Frédéric Janin-Gadoux, directeur délégué aux formations du lycée pour son investissement et son dévouement pour tous les aspects logistiques et organisationnels.

Résultats statistiques

Session	Inscrits	Nombre de postes	Présents aux trois épreuves d'admissibilité	Admissibles	Présents aux épreuves d'admission	Admis
2017	264	15	106	35	33	15
2018	280	12	105	28	23	12
2019	240	14	104	35	31	15*
2020	202	15	82	28	25	15
2021	210	15	92	36	33	15
2022	150	15	63	31	27	17**
2023	**152**	**17**	**62**	**34**	**31**	**15**

* un candidat a été inscrit sur liste complémentaire
** deux candidats ont été inscrits sur liste complémentaire

Statistiques des notes obtenues à l'admissibilité à la session 2023

Moyenne obtenue par le premier candidat admissible	**18,42**
Moyenne obtenue par le dernier candidat admissible	**6,76**
Moyenne des candidats présents	7,58
Moyenne des candidats admissibles	10,38
Écart-type des candidats présents	3,9
Écart-type des candidats admissibles	2,89

Statistiques des notes obtenues à l'admission à la session 2023

Moyenne obtenue par le premier candidat admis	**17,6**
Moyenne obtenue par le dernier candidat admis	**9,16**
Moyenne des candidats présents	10,9
Moyenne des candidats admis	12
Écart-type des candidats présents	2,3
Écart-type des candidats admis	1,7

Épreuve d'admissibilité de sciences industrielles de l'ingénieur

A. Présentation de l'épreuve

Arrêté du 28 décembre 2009 modifié

- Durée totale de l'épreuve : 6 heures
- Coefficient 1

L'épreuve est commune à toutes les options. Les candidats composent sur le même sujet au titre de la même session, quelle que soit l'option choisie.
Elle a pour but de vérifier que le candidat est capable de mobiliser ses connaissances scientifiques et techniques pour conduire une analyse systémique, élaborer et exploiter les modèles de comportement permettant de quantifier les performances globales et détaillées d'un système des points de vue matière, énergie et information afin de valider tout ou partie de la réponse au besoin exprimé par un cahier des charges. Elle permet de vérifier les compétences d'un candidat à synthétiser ses connaissances pour analyser et modéliser le comportement d'un système pluritechnologique automatique.

B. Sujet

Le sujet est disponible en téléchargement sur le site du ministère à l'adresse :
https://media.devenirenseignant.gouv.fr/file/Agreg_externe/00/6/s2023_agreg_externe_sii_1_1430006.pdf

Ce sujet porte sur le bâtiment de la Philharmonie de Paris. Son édification a débuté en septembre 2009 et s'est achevée en 2015 dans le parc de la Villette à Paris.

C. Éléments de correction

PARTIE 1 – Activité *Maestra, Maestro !* à la Philharmonie des enfants

Question 1 : La taille de la vidéo après compression est : 3840×2160×3×25×120/5000 ≈ 15 Mo.

Question 2 : Le débit de transmission est d'environ $\frac{1\text{ bit}}{100\mu\text{s}}$ = 10 000 bits/s. La durée totale de la transmission est de :

$$\frac{(12\times 8\text{ bits}+12\times 3\text{ bits})}{10\,000\text{ bits/s}} \approx 0{,}0132\text{ s.}$$

Question 3 : Pour le checksum, on applique l'opération $\otimes$ entre chaque octet :

$$0xAB = (1010\ 1011)_2$$
$$0x00 = (0000\ 0000)_2$$
$$0x52 = (0101\ 0010)_2$$
$$0x42 = (0100\ 0010)_2$$
$$0xE4 = (1110\ 0100)_2$$

On obtient :

$$1010\ 1011 \oplus 0000\ 0000 \oplus 0101\ 0010 \oplus 0100\ 0010 \oplus 1110\ 0100 = (0101\ 1111)_2 = (5F)_{16}.$$

Les deux caractères XX sont donc 5F si la lecture de la trame est correcte.

Question 4 : On relève du bit faible au bit fort : début de l'écran 11 (2 bits de STOP de l'octet précédent) puis 0 (un bit de START)
11^{eme} octet X : 1010 1100
11 (END) 0 (START)
12^{eme} octet X : 0110 0010
11 (END) 1 (aucune donnée)

11^{eme} octet lecture sérielle avec LSB en premier : $(0011\ 0101)_2 = (35)_{16}$ = caractère 5 ;
12^{eme} octet : $(0100\ 0110)_2 = (46)_{16}$ = caractère F.

On retrouve bien le résultat de la **question 3**.

Question 5 : Le calcul du *checksum* permet de fiabiliser les échanges entre le bracelet et le lecteur par exemple en cas de bruits parasites.

Question 6 : La taille de la vidéo compressée (15 Mo) est bien inférieure à 20 Mo, la durée totale de transmission du numéro du bracelet au lecteur est inférieure à 50 ms (13 ms) et enfin le calcul du *checksum* permet d'éviter les erreurs de lecture. L'ensemble des critères de l'exigence Id 1.1.1 est donc vérifié.

Question 7 : `distance = dx*sqrt((p1[0]-p0[0])**2+(p1[1]-p0[1])**2)`

Question 8 : $I(x+\delta x,y+\delta y,t+\delta t) \approx I(x,y,t)+\frac{\partial I}{\partial x}\delta x+\frac{\partial I}{\partial y}\delta y+\frac{\partial I}{\partial t}\delta t.$

Question 9 :

$$A = \begin{bmatrix} I_x(p_0) & I_y(p_0) \\ I_x(p_1) & I_y(p_1) \\ \dots & \dots \\ I_x(p_{n^2-1}) & I_y(p_{n^2-1}) \end{bmatrix},\ P = \begin{bmatrix} -I_t(p_0) \\ -I_t(p_1) \\ \dots \\ -I_t(p_{n^2-1}) \end{bmatrix}.$$

Question 10 :

```
for i in range(n) :
      for j in range(n) :
            It[i,j] = (I2[i,j] - I1[i,j]) / dt
# OU It[:,:] = (I2[:,:] - I1[:,:]) / dt
return It
```

Question 11 :

```
    for j in range(n) :
        for i in range(n) :
            pts.append(mat[i,j])
    # OU for i in range(n) :
    #         for j in range(n) :
    #             pts.append(mat[i,j])
    # OU            pts.append(mat[j,i])
    # OU pts = [0]*(n*n)
    #    k = 0
    #    for i in range(n) :
    #         for j in range(n) :
    #             pts[k] = mat[i,j]
    #             k += 1

    return pts
```

Question 12 :

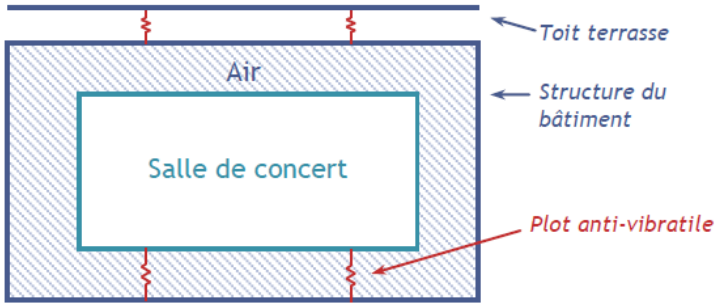
```
def Lucas_Kanade(Ix,Iy,It) :
    ptsIx = liste(Ix)
    ptsIy = liste(Iy)
    ptsIt = liste(It)
    # calcul de Vx Vy : A V = P
    A = zeros((len(ptsIx),2))
    P = zeros((len(ptsIx),1))
    for i in range(len(ptsIx)):
    # complexité de la boucle : O(n^2) (dimension de ptsIx : n^2)
        A[i,0] = ptsIx[i] # dimension de A : n^2×2
        A[i,1] = ptsIy[i]
        P[i]   = - ptsIt[i] # dimension de P : n^2×1
    At = transpose(A) # complexité : O(2n^2)
    C  = dot(At,A) # complexité O(4n^2)
    C  = linalg.inv(C) # complexité O(8)=O(1) - dimension de C est 2×2
    C  = dot(C,np.transpose(A)) # complexité O(4n^2)
    V  = dot(C,P) # complexité O(2n^2)
    return V
```

On ajoute des complexités d'ordre $O(n^2)$, la complexité de la fonction est donc $O(n^2)$.

Question 13 : La complexité de la méthode différentielle est quadratique tandis que celle de la méthode fréquentielle (transformée de Fourier d'une image) est $O(n^4)$. La méthode différentielle est donc efficace et l'exigence Id 1.2.1 est vérifiée. Il faudra tout de même vérifier que la taille de l'image $I(x,y,t)$ ne soit pas trop importante (en terme de nombre de pixels, sinon les calculs de traitement d'image seront longs en $O(n^2)$) et que le nombre d'images par seconde reste suffisant (pour garantir que δt entre deux images consécutives soit suffisamment petit et l'approximation de Taylor pertinente).

PARTIE 2 – Étude acoustique de la Grande salle Pierre Boulez

Question 14 :



Question 15 : $T_r = 0{,}16\frac{V}{A_0}$ donc $A_0 = 0{,}16\frac{V}{T_r}$.

Calcul du volume de la salle V : $V = 58\times46\times22 = 58\,696\ m^3$.

Calcul de l'aire d'absorption équivalente A_0 : $A_0 = \frac{0{,}16\times58\,696}{3} = 3130{,}45\ m^2$.

Calcul du coefficient d'absorption acoustique moyen : $A_0 = \sum_i \alpha_i S_i = \alpha_0(S_0+S_B)$ avec S_0 les surfaces de la salle sans les balcons.

Calcul de S_0 : $S_0 = 2\times(58\times46+58\times22+46\times22) = 9912\ m^2$.

On en déduit : $\alpha_0 = \frac{A_0}{S_0+S_B} = \frac{3130{,}45}{9912+3400} = 0{,}23$.

Question 16 : Calcul de l'aire d'absorption équivalente de la salle comble :
$A_c = \alpha_0\times(S_0+S_B-S_{sp})+\alpha_{sp}\times S_{sp}$ avec S_{sp} la surface des spectateurs telle que $S_{sp} = 2400\times0{,}5 = 1200\ m^2$.
D'où $A_C = 0{,}23\times(9912+3400-1200)+0{,}8\times1200 = 3745{,}76\ m^2$.

Calcul du temps de réverbération de la salle comble :

$$T_{rc} = \frac{0{,}16\times V}{A_c} = \frac{0{,}16\times58\,696}{3745{,}76} = 2{,}5\ s.$$

Question 17 : La canopée joue le rôle de réflecteur. Plus l'effectif de musiciens sera réduit, plus la position de la canopée doit être faible. Cela permet de favoriser les réflexions du son des musiciens vers le public et la scène.

Question 18 : Dans le cas de la musique amplifiée, il est nécessaire de limiter le temps de réverbération. En effet, une réverbération longue peut entraîner des effets gênants : les panneaux absorbants en velours permettront d'augmenter la surface d'absorption équivalente et donc diminuer le temps de réverbération.

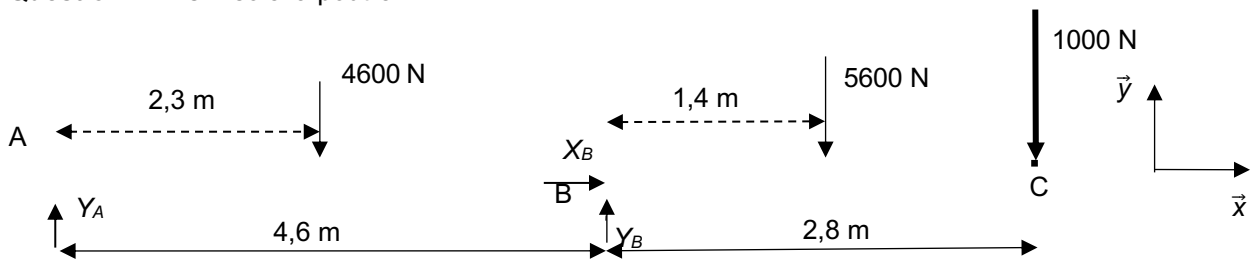
Question 19 : Le BIM permet de suivre en continu les modifications apportées par tous les acteurs sur la conception de la grande salle de concert. En ce sens, le BIM permet de donner une dimension supplémentaire aux études acoustiques illustrées sur les figures 14 et 15. L'acoustique n'est pas seulement une question de qualité d'aménagement mais la somme des choix opérés en structure, dans les équipements techniques, les matériaux…Par exemple, les acousticiens peuvent intégrer dans le BIM les caractéristiques acoustiques des différentes parois ou revêtements afin de vérifier la compatibilité avec d'éventuelles modifications d'architecture ou de matériaux.

PARTIE 3 – Dimensionnement de la canopée

Question 20 : Par associativité sur l'intégrale de surface, la section de l'IPE 200 est un rectangle de 100x200 moins deux rectangles de 47,2x166(47,2 x183)

$$I_{Gz}=\frac{100\times200^3}{12}-2\frac{47{,}2\times183^3}{12}=18\,455\,902\ \text{mm}^4.$$

Question 21 : On isole la poutre :



Le théorème de la résultante statique donne :

$$X_B = 0$$
$$Y_A - 4600 + Y_B - 5600 - 1000 = 0 \Longrightarrow Y_A + Y_B = 11\,200\ \text{N}.$$

Le théorème du moment statique en B donne :

$$-4{,}6\vec{x}\wedge Y_A\vec{y} - 2{,}3\vec{x}\wedge-4600\vec{y} + 2{,}8\vec{x}\wedge-1000\vec{y} + 1{,}4\vec{x}\wedge-5600\vec{y} = \vec{0}.$$
$$\Longrightarrow -60-4{,}6Y_A = 0 \Longrightarrow Y_A = -13{,}04\text{N}.$$

On en déduit : Y_B = 11 213,04 N.

Question 22 : On détermine l'expression du moment fléchissant puis on calcule sa valeur en *B*.
On isole le tronçon entre *A* et *B* à gauche, l'équilibre de cette portion nous donne :

$$M_{fZ}\vec{z}-x\vec{x}\wedge-13\vec{y}-\frac{x}{2}\vec{x}\wedge-1000x\vec{y}=\vec{0} \Longrightarrow M_{fZ} = -13x-500x^2.$$

En *B* : x = 4,6 m donc M_{fZ} = -10 639,8 Nm ≈ -10 640 Nm

Question 23 : Nature de la contrainte en *B* : contrainte normale.
Contrainte normale Maxi en *B* pour y_{max}=100 mm :

$$\sigma^B_{Max}=\frac{M_{fzMax}}{I_{Gz}}y_{Max}=\frac{10{,}64\ .\ 10^6}{20\ .\ 10^6}100=53{,}2\ \text{MPa}.$$

Question 24 : Conclusion sur le choix du matériau : La contrainte normale maxi est très inférieure à R_e = 235 MPa. Le coefficient de sécurité est supérieur à 4 ce qui permet de conclure que le matériau retenu convient.

Question 25 : C1 : Canopée (masse en translation verticale)
C2 : Arbre de synchronisation
C3 : Arbre de transmission
C4 : Treuil (tambour et réducteur)
C5 : Poulie de mouflage

Question 26 : Dans le bloc « Corde », pour les positions négatives la force est proportionnelle au déplacement, de manière cohérente avec la raideur imposée (l'effort vaut bien 100 N pour un déplacement de 1 m). Cependant, à la différence d'un bloc ressort pour lequel cette loi serait valable pour toutes positions, pour les positions positives du bloc Corde la force est nulle. Cela permet de modéliser le fait qu'un câble ne peut pas être soumis à de la compression. La force nulle correspond au moment où le câble est détendu. Ainsi il est pertinent d'utiliser un bloc Corde pour modéliser un câble afin de ne pas ajouter de raideur de compression qui n'a pas lieu d'être dans le cas de câble.

Question 27 : La raideur de l'arbre en torsion est telle que $M_t = k_t \cdot \Delta\theta$
Or, pour un arbre soumis à de la torsion l'angle unitaire de torsion est donné par
$\alpha = \frac{M_t}{I_0 \cdot G}$ avec $I_0 = \frac{\pi \cdot \left(D^4 - (D-2 \cdot e)^4\right)}{32}$
Par ailleurs $\alpha = \frac{\Delta\theta}{L}$
Ainsi $k_t = \frac{M_t}{\Delta\theta} = \frac{I_0 \cdot G}{L} = \frac{\pi \cdot \left(D^4 - (D-2 \cdot e)^4\right) \cdot G}{32 \cdot L}$

AN $k_t = 4970$ N·m·rad^{-1}

Question 28 : Les câbles sont soumis à de la traction. La raideur du câble est telle que $F_{cable} = k_c \cdot \Delta l$
où Δl représente l'allongement du câble. Pour un câble de longueur L

$$F_{cable} = S \cdot \sigma = S \cdot E \cdot \varepsilon = S \cdot E \cdot \frac{\Delta l}{L}$$

Par identification $k_c = \frac{E \cdot S}{L}$
AN $k_c = 2{,}52 \cdot 10^6$ N·m^{-1}

Question 29 : D'après les figures, sans arbre de synchronisation, la tension dans le câble 4 devient nulle (ce qui est cohérent car l'un des couples moteurs a été imposé à 0 Nm) ce qui peut être problématique au regard de l'enroulement au niveau du tambour. L'arbre de synchronisation permet donc de vérifier qu'aucun câble ne se détende et de répartir les efforts dans les deux cinématiques en cas de désynchronisation mécanique des moteurs ou de défaillance.
Dans le cas où un arbre de synchronisation est présent, la tension maximale relevée est de $4 \cdot 10^4$ N, or la charge maximale admissible est de $34{,}8 \cdot 10^4$ N. La tension est donc bien inférieure à un cinquième de la charge maximale admissible.

Question 30 : 4 périodes correspondent à 0,9 s, ainsi la période d'oscillation vaut 0,225 s et la fréquence 4,4 Hz.

Si les moteurs tournent à leur vitesse nominale, la ligne d'arbre tourne alors à une vitesse de 198 tr·min^{-1} soit 3,3 tr·s^{-1}. Les à-coups générés par les joints de cardan ont une fréquence de 6,6 Hz. Il n'y a donc pas de risque de mise en résonance à la vitesse de rotation nominale.
S'il y avait un risque, quelques solutions (non exhaustives) sont possibles :
- ajouter des amortisseurs (élastomères) dans la chaîne de transmission ;
- modifier les joints de cardan par des joints tripodes ;
- choisir des câbles/arbres plus gros, afin de modifier la raideur.

Question 31 : Chaque câble étant mouflé une fois,
$v = \frac{R}{2} \cdot \omega_{tambour} = \frac{R}{2} \cdot r_1 \cdot r_2 \cdot \omega_m$ donc $r_g = \frac{v}{\omega_m} = \frac{R}{2} \cdot r_1 \cdot r_2 = 1{,}58 \cdot 10^{-4}$ m·rad^{-1}
La vitesse de rotation nominale du moteur étant de 1445 tr·min^{-1} :

$$v = 0{,}024 \text{ m·s}^{-1}$$

Pour satisfaire l'exigence d'une vitesse maximale de déplacement de 0,02 m·s^{-1} le moteur ne doit pas tourner à sa vitesse nominale. Grâce au variateur de fréquence cette exigence pourra être vérifiée.

Question 32 : Les câbles ont une masse linéique de 1,85 kg·m^{-1}. Dans le cas le plus défavorable la longueur d'un brin vaut 18 m (position « maintenance »). Seule la longueur de câble côté moteur est à prendre en compte, car l'autre brin est fixé au bâti. Pour 6 câbles, la masse vaut

$$M_{cables} = 6 \cdot 1{,}85 \cdot 18 = 200 \text{ kg}$$

La masse de l'habillage vaut $80 \cdot 175 = 14 \cdot 10^3$ kg. La masse totale de la canopée est donc

$$M=m_{equipement}+m_{structure}+m_{fers}+m_{habillage}=15\cdot10^3+7\cdot10^3+3{,}4\cdot10^3+14\cdot10^3=39{,}4\cdot10^3 \text{ kg}$$

La masse des câbles est bien négligeable au regard de la masse totale de la canopée de 40 tonnes.

Question 33 : Le centre de gravité total de la canopée vérifie :

$$M\cdot\overrightarrow{OG_t}=m_{equipement}\cdot\overrightarrow{OG_{equipement}}+m_{structure}\cdot\overrightarrow{OG_{structure}}+m_{fers}\cdot\overrightarrow{OG_f}+m_{habillage}\cdot\overrightarrow{OG_h}$$

La masse de l'équipement audiovisuel ainsi que la masse propre de la structure hexagonale étant supposées uniformément réparties sur la structure hexagonale, le centre de gravité de l'équipement et de la structure sont situés en O.

Ainsi :

$$M\cdot\overrightarrow{OG_t}=m_{fers}\cdot\overrightarrow{OG_f}+m_{habillage}\cdot\overrightarrow{OG_h}$$

En projetant dans le repère R :

$$x_t=\frac{m_{fers}\cdot x_f+m_{habillage}\cdot x_h}{M}=0{,}032 \text{ m}$$
$$y_t=\frac{m_{fers}\cdot y_f+m_{habillage}\cdot y_h}{M}=0{,}144 \text{ m}$$
$$z_t=\frac{m_{fers}\cdot z_f+m_{habillage}\cdot z_h}{M}=-0{,}179 \text{ m}$$

Comme $x_t>0$, $y_t>0$ et $y_t>x_t$, il est possible de déduire sans calcul que le câble situé en A_2 sera le plus sollicité (car le plus proche du centre de gravité). Il semble donc cohérent dans la suite d'étudier la Canopée soumise uniquement aux actions des câbles A_2, A_4 et A_6.

Question 34 : L'ensemble de la Canopée sauf les poulies est soumise à l'action de la pesanteur, et aux trois actions des poulies supposées verticales.

Le principe fondamental de la statique appliqué à cet ensemble s'écrit en O :

$$\begin{cases} -M\cdot g\cdot\vec{z}+\overrightarrow{F_2}+\overrightarrow{F_4}+\overrightarrow{F_6}=\vec{0} \\ \overrightarrow{OG}\wedge\vec{P}+\overrightarrow{OA_2}\wedge\overrightarrow{F_2}+\overrightarrow{OA_4}\wedge\overrightarrow{F_4}+\overrightarrow{OA_6}\wedge\overrightarrow{F_6}=\vec{0} \end{cases}$$

$$\begin{cases} -M\cdot g\cdot\vec{z}+F_2\cdot\vec{z}+F_4\cdot\vec{z}+F_6\cdot\vec{z}=\vec{0} \\ \left(x_G\cdot\vec{x}+y_G\cdot\vec{y}+z_G\cdot\vec{z}\right)\wedge(-M\cdot g\cdot\vec{z})+\left(\frac{R}{2}\cdot\vec{x}+\frac{\sqrt{3}\cdot R}{2}\cdot\vec{y}\right)\wedge F_2\cdot\vec{z}+(-R\cdot\vec{x})\wedge F_4\cdot\vec{z}+\left(\frac{R}{2}\cdot\vec{x}-\frac{\sqrt{3}\cdot R}{2}\cdot\vec{y}\right)\wedge F_6\cdot\vec{z}=\vec{0} \end{cases}$$

En résultante en projection sur $\vec{z}$ et en moment en O en projection sur $\vec{x}$ et $\vec{y}$

$$\begin{cases} F_2+F_4+F_6-M\cdot g=0 & (1) \\ -y_G\cdot M\cdot g+\frac{\sqrt{3}\cdot R}{2}\cdot F_2-\frac{\sqrt{3}\cdot R}{2}\cdot F_6=0 & (2) \\ x_G\cdot M\cdot g-\frac{R}{2}\cdot F_2+R\cdot F_4-\frac{R}{2}\cdot F_6=0 & (3) \end{cases}$$

Après résolution :

$$F_2=\frac{M\cdot g}{3}\left(1+\frac{x_G}{R}+\frac{\sqrt{3}\cdot y_G}{R}\right)$$
$$F_4=\frac{M\cdot g}{3}\left(1-\frac{2\cdot x_G}{R}\right)$$
$$F_6=\frac{M\cdot g}{3}\left(1+\frac{x_G}{R}-\frac{\sqrt{3}\cdot y_G}{R}\right)$$

L'application numérique donne $F_2=138\cdot10^3$ N

L'effort dans le câble est deux fois inférieur à cause du mouflage, donc

$$F_{cable2}=69\cdot10^3 \text{ N}$$

Question 35 : L'étude se place dans le cas d'un maintien en statique de la canopée. A la limite du déplacement descendant (cas le plus défavorable), les différents frottements auront tendance à s'opposer au mouvement, donc à diminuer l'effort nécessaire de maintien.

Question 36 : En supposant que les trois câbles moteurs sont soumis à une tension de $7 \cdot 10^4$ N, que les frottements dans toutes les liaisons sont négligeables (les rendements des réducteurs et renvois d'angle sont unitaires), que l'action de l'arbre de synchronisation sur le moteur considéré est nulle, et que les trois réducteurs sont identiques, le couple moteur vaut

$$C_m = 3 \cdot F_{cable2} \cdot r_g \cdot 2 = 3 \cdot F_{cable2} \cdot r_1 \cdot r_2 \cdot R$$

Le document technique indique le couple maximal en sortie du motoréducteur, il faut donc déterminer le couple en sortie du motoréducteur :

$$C_{red} = 3 \cdot F_{cable2} \cdot r_2 \cdot R = 3 \cdot 7 \cdot 10^4 \cdot \frac{0{,}282}{122{,}48} = 484 \text{ N·m}$$

Le document technique indique que le couple maximal en sortie du motoréducteur de maintien en statique est de 1,15·430 N·m soit 494 N·m, ce qui est supérieur au couple requis. L'exigence est donc respectée.

Question 37 : En supposant que seules les inerties des deux rotors ainsi que la masse de la canopée ne sont pas négligeables, l'énergie cinétique totale vaut :

$$E_C = \frac{1}{2} \cdot M \cdot v^2 + J_m \cdot \omega_m^2 = \frac{1}{2} \cdot \left(M + 2 \cdot \frac{J_m}{r_g^2} \right) \cdot v^2$$

Le théorème de l'énergie cinétique appliqué à l'ensemble des pièces en mouvement permet d'écrire :

$$\frac{dEc}{dt} = -M \cdot g \cdot v + \eta \cdot P_m$$

$$\left(M + 2 \cdot \frac{J_m}{r_g^2} \right) v \cdot \dot{v} = -M \cdot g \cdot v + \eta \cdot P_m$$

Ainsi

$$P_m = \frac{1}{\eta} \left[M \cdot g \cdot v + \left(M + 2 \cdot \frac{J_m}{r_g^2} \right) \cdot v \cdot \dot{v} \right]$$

La puissance est maximale en fin de phase d'accélération, avec $\dot{v} = 0{,}02$ m·s^{-2} et $v = 0{,}02$ m·s^{-1}.
Pour $\eta = 0{,}9$:

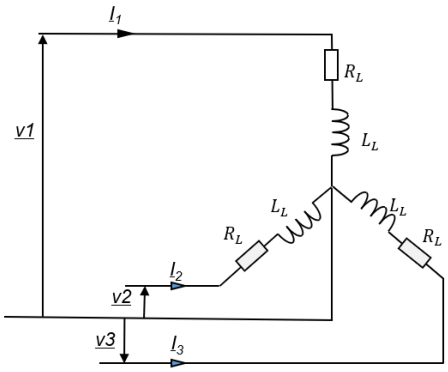
$$P_m = \frac{1}{0{,}9} \left[39 \cdot 10^3 \cdot 9{,}81 \cdot 0{,}02 + \left(39 \cdot 10^3 + 2 \cdot \frac{280 \cdot 10^{-4}}{\left(1{,}58 \cdot 10^{-4}\right)^2} \right) 0{,}02 \cdot 0{,}02 \right]$$

$$P_m = 9{,}51 \text{ kW}$$

Or, deux moteurs de 7,5 kW sont choisis, la puissance totale disponible est donc de 15 kW ce qui est supérieur à 9,5 kW. L'exigence 2.1.4.2 pourra donc être respectée.

PARTIE 4 – Dimensionnement du réseau d'alimentation du sous-ensemble scénique

Question 38 :



Question 39 :

$$\Delta V=0{,}02\cdot 230=\frac{\rho_c\cdot l_c}{S_c}\cdot I_L=300\cdot\frac{1{,}72\cdot 10^{-8}\cdot 230}{S_c} \qquad S_c=256\ mm^2$$

$$P_{fils}=3\cdot R_{fils}\cdot I^2=4.1\ kW$$

On choisit la section de 300 mm² selon tableau pour un courant maximum de 621 A.

Question 40 : Pour le module Lumière :

- la puissance active vaut : $P=3\cdot R_L\cdot I^2=432kW$,
- la puissance réactive vaut : $Q=3\cdot X_L\cdot I^2=243kVAr$, $S=495\ kVA$.

Question 41 : Pour le module audiovisuel :

- La puissance active vaut : $P_a=S\cdot\cos\varphi=200\ 10^3\cdot 0{,}8=160kW$,
- La puissance réactive vaut : $Q=S\cdot\sin\varphi=P_a\cdot\tan\varphi=120kVAr$.

Pour le module machinerie :

- la puissance active des 2 moteurs vaut : $P=2\cdot\frac{7{,}5\cdot 10^3}{0{,}9}=16{,}7kW$,
- la puissance réactive vaut : $Q=P\cdot\tan\varphi=9kVAr$.

Donc un total pour la machinerie : P=526,7kW et Q=147kVAr S=547 kVA.

Question 42 :

La puissance active totale vaut : $P_{charges}=16{,}7+432+510+160=1118{,}7kW$.
La puissance réactive totale vaut : $Q_{charges}=9+243+138+120=510kVAr$,

avec $P_{totale}=P_{charge}+3\cdot P_{fils}=1118{,}7+12{,}3\ kW$.

$$\eta=\frac{P_{charge}}{P_{totale}}=\frac{1118{,}7}{1130}=99\%.$$

Question 43 : Courant en A avant compensation : $I_A=\frac{\sqrt{P_{totale}^2+Q_{totale}^2}}{\sqrt{3}\cdot U_A}=1790$ A.

Le facteur de puissance vaut : $\tan\varphi=\frac{Q_{totale}}{p_{totale}}=0{,}45$ soit $\cos\varphi=0{,}9$.

Question 44 : L'échelle du tracé suggérée est la suivante : 100 kVAr (ou kW) pour 1cm.

Comme pour un condensateur la réactance vaut $X_c=\frac{1}{cw}$ on a $Q_c=3\cdot U_A{}^2cw$ ainsi on obtient une capacité de

$c=\frac{57\cdot 10^3}{3\cdot 400^2\cdot 314}=597\ \mu F$.

Question 45 : Courant en A' après compensation : $I'_A = \frac{P_{total}}{\sqrt{3}\cdot U_B \cdot \cos\varphi} = \frac{1131}{\sqrt{3}\cdot 400 \cdot 0{,}93} = 1788$ A.

La puissance apparente : $S'_A = \sqrt{P_{totale}^2 + (Q_{totale} - Q_c)^2} = 1218$ kVA.

Question 46 : Le courant primaire à vide I_{10} est faible devant le courant nominal, il est alors possible de négliger R_f et L_1.

Question 47 : Le raccordement en étoile permet de ramener le neutre ce qui est nécessaire si des équipements fonctionnent en monophasé.

- Le rapport m : $m = \frac{V_{20}}{U_{1n}} = \frac{\sqrt{3}\cdot 235}{20000} = 0.02$ pour un couplage triangle-étoile, à vide,
- La résistance $R_S = \frac{P_{cc}}{3I_{2n}^2} = 1{,}17$ mΩ,
- L'impédance $Z_S = \frac{V_{cc}\cdot V_{2n}}{I_{2n}} = 7{,}6$ mΩ (l'erreur sur Vcc = 6% a été prise en compte lors de la correction)
- La réactance $X_S = \sqrt{Z_S^2 - R_S^2} = 7{,}57$ mΩ.

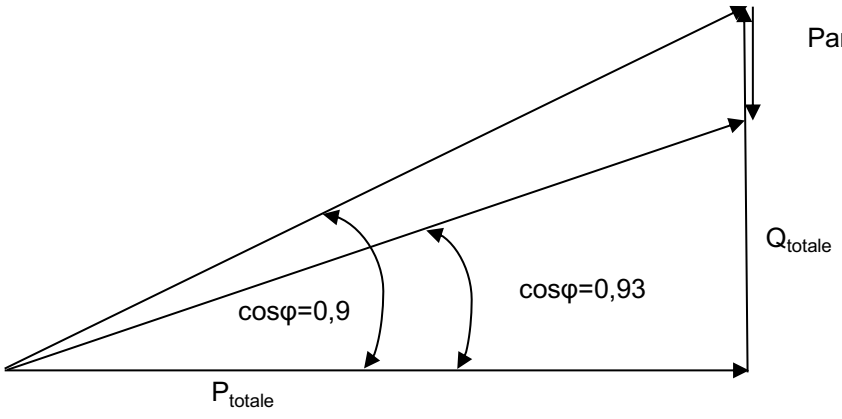
Question 48 : L'impédance magnétisante est négligée compte tenu de l'ordre de grandeur du courant consommé à vide.

Construction de Fresnel : $\underline{V_{2n}} = \underline{V_2} + \underline{I_{2n}}\cdot(R_S + jX_S)$ équation de Kapp

La chute de tension est déterminée par la relation $\Delta V_2 = I_{2n}\cdot R_S \cdot \cos\varphi + I_{2n}\cdot X_S \cdot \sin\varphi = 6.96$V.
Soit $\frac{\Delta V_2}{V_{2n}} = 3\% <$ 5% l'exigence 3.1.2.2 est vérifiée.

La puissance apparente : $S'_A = \sqrt{P_{totale}^2 + (Q_{totale} - Q_c)^2} = 1218$ kVA< 1250 kVA, l'exigence 3.1.3.1 est vérifiée.

PARTIE 5 – Synthèse



Question 49 :

1. La Philharmonie des enfants, et plus précisément l'attraction *Maestra, Maestro !* a pour vocation de faire découvrir la musique symphonique aux jeunes enfants. L'attraction doit permettre à

un jeune visiteur de se mettre dans la peau d'un chef d'orchestre (Id 1.1 sur le DT1). La lecture par technologie RFID du bracelet contenant les informations du visiteur (adresse e-mail) ainsi que la pertinence de l'algorithme de traitement d'image permettant de fournir deux inputs du logiciel de traitement vidéo ont été vérifiés.

2. Les qualités acoustiques de la Grande salle Pierre Boulez doivent être adaptés à différents styles musicaux (Id 2.1 sur le DT1). L'isolement de la salle Pierre Boulez par rapport à l'ensemble du bâtiment (calcul du temps de réverbération), le réglage en hauteur de la canopée ainsi que le dimensionnement de la canopée (motorisation, structure, tension maximale dans les câbles qui la soutiennent) ont été étudiés.
3. Le module scénique doit être alimenté (Id 3.1 sur le DT1). Le choix du transformateur par le calcul des différentes puissances actives et réactives mises en jeu a été validé. La chute de tension de moins de 5% sur la ligne d'alimentation a été vérifiée.

Question 50 :

1. Une étude thermique pourrait être mise en œuvre afin de respecter la norme en vigueur (modèle BIM, modèle multi-physique, choix des matériaux, isolants etc …) : exigence 4 du DT1 ;
2. Une étude de la régulation de la température et de l'hydrométrie dans la salle Pierre Boulez pourrait être mise en œuvre afin de vérifier l'exigence 5 du DT1 (asservissement à l'aide de capteurs de température et d'humidité).

D. Commentaires du jury

Cette épreuve, d'une durée de 6 heures, coefficient 1, est commune aux quatre options. Les candidats composent dans les mêmes conditions, sur le même sujet au titre de la même session quelle que soit l'option choisie. Conformément à l'arrêté du 28 décembre 2009 modifié, « cette épreuve a pour but de vérifier que le candidat est capable de mobiliser ses connaissances scientifiques et techniques pour conduire une analyse systémique, élaborer et exploiter les modèles de comportement permettant de quantifier les performances globales et détaillées d'un système des points de vue matière, énergie et information afin de valider tout ou partie de la réponse au besoin exprimé par un cahier des charges. Elle permet de vérifier les compétences d'un candidat à synthétiser ses connaissances pour analyser et modéliser le comportement d'un système pluritechnologique automatique ».

1. Présentation générale du sujet

Ce sujet porte sur le bâtiment de la Philharmonie de Paris, dans le parc de la Villette, dont l'édification, confiée aux Ateliers Jean Nouvel, s'est achevée en 2015. La Philharmonie propose plusieurs espaces, à vocation scénographique, acoustique, culturelle et pédagogique. Le sujet se focalise principalement sur l'espace éducatif (Philharmonie des enfants) et l'espace scénographique et acoustique principal (la Grande salle Pierre Boulez).

Les différentes parties proposent de vérifier plusieurs des exigences du cahier des charges de la Philharmonie des enfants et de la Grande salle Pierre Boulez :

– les exigences liées à l'attraction *Maestra, Maestro !* de la Philharmonie des enfants,
– les exigences liées à l'acoustique de la Grande salle Pierre Boulez ainsi qu'au dimensionnement de la structure et de la motorisation de la canopée, réflecteur acoustique principal de la salle,
– les exigences liées à l'alimentation électrique du sous-ensemble scénique de la Philharmonie.

Les objectifs de la **partie 1** sont de valider les exigences liées à l'attraction *Maestra, Maestro !*, notamment :

- d’analyser l’information transmise par un bracelet RFID situé sur le poignet de l’enfant, permettant d’enregistrer son activité et de la lui envoyer sur adresse e-mail (validation de l’exigence Id 1.1.1) ;
- de proposer une partie de l’algorithme en langage Python permettant d’interpréter les gestes de l’enfant imitant un chef d’orchestre sur un enregistrement vidéo et de valider l’efficacité de l’algorithme (exigence 1.2.1).

L’objectif de la **partie 2** est de vérifier les qualités acoustiques de la Grande salle et valider l’exigence Id 2.1. « les qualités acoustiques de la salle doivent être adaptées à différents genres musicaux ».

Les objectifs de la **partie 3** sont :
- de vérifier que les solutions envisagées pour la conception de la structure de la canopée permettent de valider les exigences de sécurité, en termes de contrainte (Id 2.1.1) et de tension maximale des câbles (Id 2.1.3.3.) ;
- de valider le dimensionnement de la motorisation en vérifiant les exigences de vitesse de translation (Id 2.1.4.2) et de maintien statique (Id 2.1.3.2).

L’objectif de la **partie 4** est de valider le choix du transformateur de la ligne d’alimentation du poste d’exploitation Scénique, notamment par :
- la caractérisation de la consommation énergétique du module Lumière ;
- le calcul du rendement énergétique de l’alimentation des modules du poste d’exploitation scénique dans sa globalité, et la validation de l’exigence correspondante (id 3.1.3.1) ;
- le dimensionnement de la batterie de condensateurs qui permet de diminuer les chutes de tension, et donc de limiter les surcoûts de l’installation en aval du transformateur (exigence Id 3.1.2.1) ;
- et finalement de valider le choix du transformateur à partir des charges sur la ligne d’alimentation ainsi que des pertes en ligne (Id 3.1.2.2).

Enfin l'objectif de la **partie 5** est de conclure en effectuant une synthèse sur le travail réalisé et des études complémentaires restant à mener.

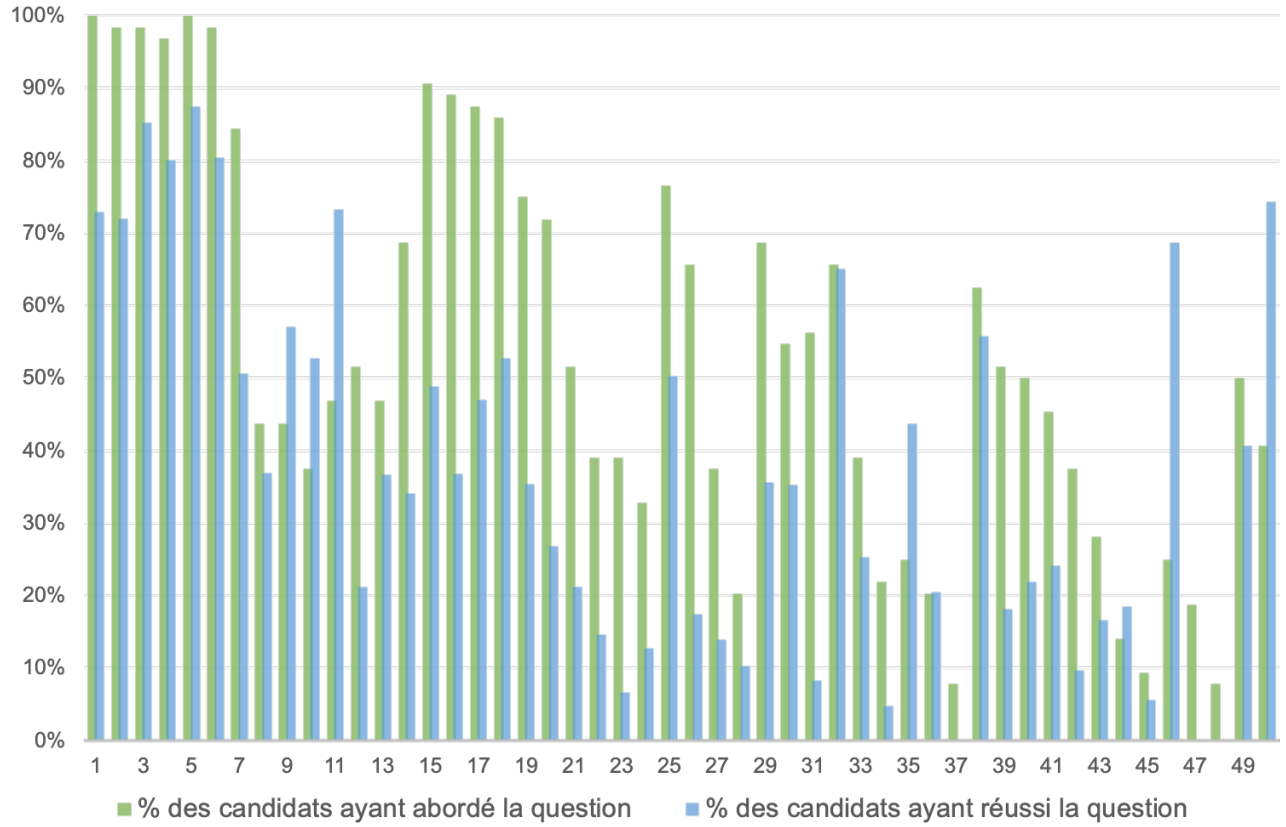
Bien que les 5 parties du sujet soient indépendantes, il était conseillé de traiter ce sujet dans l'ordre.

2. Analyse globale des résultats

L’analyse globale des résultats amène aux constats suivants :
– la première partie été abordée par une très grande majorité de candidats (95%). La première sous-partie était très accessible et a un taux de réussite de 60%. En revanche seuls 10% des candidats ont réussi correctement la deuxième sous-partie ;
– la deuxième partie, d’ordre général, a également été largement abordée par les candidats (88%) et a été traitée correctement par 32% des candidats. Elle a un taux de réussite de 41% ;
– la troisième partie a été abordée par 93% des candidats. Elle a un taux de réussite de seulement 24%;
– la quatrième partie a été abordée par 72% des candidats. Elle a un taux de réussite de seulement 26% ;
– la cinquième et dernière partie a été abordée par 37% des candidats et a un taux de réussite de 44%.

3. Analyse spécifique aux candidats de l’option ingénierie informatique

Les statistiques de réussite propres à chaque question du sujet sont données dans l’histogramme ci-dessous pour les candidats de l’option ingénierie informatique.



Le jury constate que toutes les parties ont été globalement abordées, il constate également que :

Partie 1 : la sous-partie 1.1, très simple, est la seule partie du sujet a avoir été abordée par la plupart des candidats de l'option. La sous-partie 1.2, qui traite d'un algorithme de traitement d'images très simplifiée en language Python a été réussi par moins de 50% des candidats. C'est trop peu pour une sous-partie qui concerne l'option ingénierie informatique. Même sans maitriser le langage Python, la sous-partie est abordable en utilisant la syntaxe proposée en annexe.

Partie 2 : la partie 2 qui traite de considérations simplifiées de l'acoustique de la grande salle Pierre Boulez a été globalement bien abordée, les modèles utiles à la résolution étaient donnés. Pour quelques questions ouvertes, il était plus difficile de proposer une réponse pertinente. Le jury rappelle qu'il faut systèmatiquement préciser les bonnes unités aux grandeurs physiques. Le temps de réverbération (en seconde) et l'aire d'absorption équivalente (en mètres carré) étaient trop souvent associés à la mauvaise unité.

Partie 3 : la sous-partie 3.1 qui traite du dimensionnement d'un fer rayonnant de la canopée en utilisant la résistance des matériaux a été très peu abordée par l'ensemble des candidats (moins de 40%), sans parler du taux de réussite de cette sous-partie. Le jury rappelle que les modèles simples de résistance des matériaux doivent être maîtrisés par les candidats de l'option ingénierie infomatique. Les sous-parties 3.2 et 3.3 concerne la suspension de la canopée par câbles ainsi que le dimensionnement de sa motorisation. Ces deux sous-parties ont été traitées par environ 50% des candidats. Le mouflage des câbles n'a pas été pris en compte par la plupart des candidats. Afin de déterminer le couple moteur et la puissance nécessaire à l'élévation de la canopée, le jury attendait une résolution avec méthode et rigueur, l'utilisation du théorème de l'énergie cinétique a trop souvent été oublié. Le jury rappelle également qu'il attend systèmatiquement des expressions littérales et des applications numériques (avec unités) si demandées.

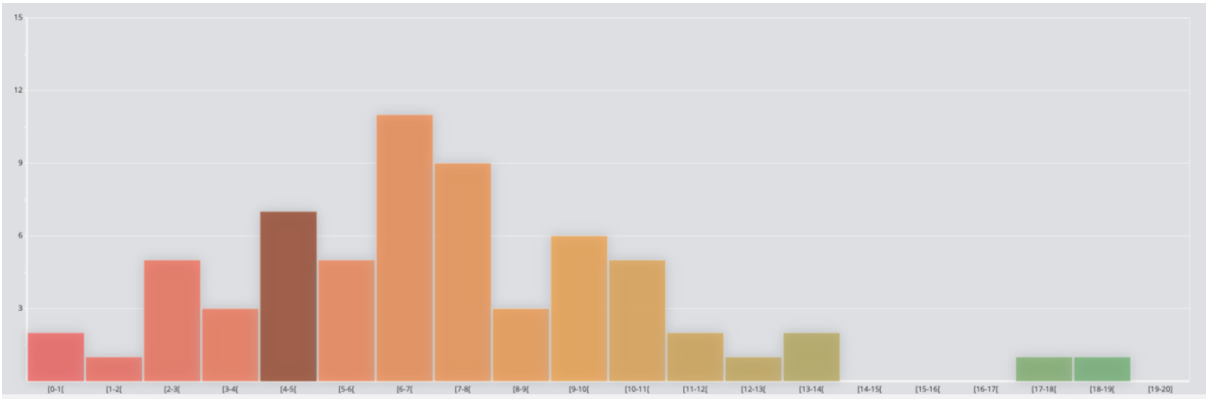
Partie 4 : la partie 4, plus difficile pour des candidats en option ingénierie informatique, traite du dimensionnement du réseau d'alimentation du sous-ensemble scénique et demande la maitrise de certaines connaissances en életrocinétique en régime sinusoïdal. Cette partie a été abordée par moins de 40% des candidats.

Partie 5 : la partie 5 de synthèse des différentes études ménées dans le sujet a été abordée par plus de 50% des candidats. Le jury a apprécié plusieurs réponses pertinentes et synthétiques aux deux dernières questions.

Les statistiques générales pour cette épreuve en option ingénierie informatique (sur un total de 64 candidats qui ont composé) sont :

note minimum : 0,60/20, note maximale : 18,60/20, moyenne : 7,06/20, écart-type : 3,60.

L'histogramme des notes pour cette épreuve est donnée ci-dessous.



4. Conseils aux futurs candidats

Les candidats ayant réussi cette épreuve sont ceux qui ont fait preuve de transversalité et qui ont fait l'effort d'aborder chacune des parties. Chaque partie était conçue avec une difficulté croissante des questions, permettant aux candidats des différentes spécialités à la fois de pouvoir aborder partiellement chaque problématique, mais également de s'affirmer dans son domaine de prédilection.

Le jury encourage ainsi fortement les candidats à traiter toutes les parties du sujet et à montrer qu'ils maîtrisent l'ensemble des domaines des sciences industrielles de l'ingénieur. Les résultats démontrent que ceux qui refusent d'évoluer vers une approche transversale et sélectionnent les questions relatives aux différentes spécialités de l'ingénierie ne réussissent pas cette épreuve, car la note finale se trouve alors fortement limitée. Par conséquent, le jury conseille aux futurs candidats de s'investir sérieusement dans toutes les parties du programme du concours et d'acquérir l'ensemble des compétences et des connaissances associées aux disciplines qui constituent les sciences industrielles de l'ingénieur.

Les candidats doivent également s'attacher à utiliser leurs connaissances universitaires afin de résoudre des problématiques techniques associées à des systèmes industriels. Les plus efficients ont su ne pas perdre de vue que les analyses, les justifications et les choix technologiques doivent être toujours menés en gardant à l'esprit les enjeux du contexte industriel spécifique à l'étude.

Le jury constate trop souvent un manque de rigueur, notamment dans l'écriture des expressions littérales, dans la manipulation des grandeurs scalaires et vectorielles, de précision dans la présentation des copies et dans la rédaction. La présentation de la copie doit être irréprochable, les notations imposées dans le sujet doivent être scrupuleusement respectées. Il convient aussi de rappeler qu'il est attendu d'un fonctionnaire de l'État qu'il

maîtrise convenablement la langue française et qu'il respecte les règles de l'orthographe et de la grammaire française afin de s'assurer que ce qu'il souhaite exprimer soit compréhensible et lisible.

Les réponses doivent être détaillées et argumentées : des résultats donnés directement, sans calcul, sans justification de principe, ne peuvent pas être pris en compte comme étant justes. Par ailleurs, les réponses montrant une maîtrise de la démarche mais n'arrivant pas jusqu'à la conclusion sont valorisées. Le jury apprécie aussi l'esprit critique face à des résultats aberrants et admet le choix délibéré de commenter ces résultats pour continuer le traitement du sujet.

Le jury souligne enfin la grande qualité d'expression constatée dans certaines copies, rédigées avec soin et un souci de clarté.

Réussir cette épreuve demande :
– de s'approprier en un temps limité un sujet technique pluridisciplinaire décrit avec les outils de modélisation de l'ingénierie système ;
– de maîtriser les modèles de connaissance des différents domaines d'étude de l'ingénierie ;
– d'analyser et d'interpréter des résultats d'étude, afin de formuler des conclusions cohérentes et pertinentes en concordance avec une problématique scientifique et technique.

5. Conclusion

Le sujet a été conçu pour permettre aux candidats d'exprimer au mieux leurs compétences dans différents champs d'application d'un système pluritechnologique correspondant au cadre de cette épreuve transversale. Le jury engage fortement les futurs candidats à se préparer conformément aux attendus de l'arrêté du 28 décembre 2009 modifié.
Les auteurs remercient les ateliers Jean Nouvel et AMG Féchoz pour l'ensemble des données communiquées.

Épreuve d'admissibilité de modélisation d'un système, d'un procédé ou d'une organisation

A. Présentation de l'épreuve

Arrêté du 28 décembre 2009 modifié

- Durée totale de l'épreuve : 6 heures
- Coefficient 1

L'épreuve est spécifique à l'option choisie.
À partir d'un dossier technique comportant les éléments nécessaires à l'étude, l'épreuve a pour objectif de vérifier que le candidat est capable de synthétiser ses connaissances pour modéliser un système technique dans le domaine de la spécialité du concours dans l'option choisie en vue de prédire ou de vérifier son comportement et ses performances.

B. Sujet

Le sujet est disponible en téléchargement sur le site du ministère à l'adresse :
https://www.devenirenseignant.gouv.fr/media/4814/download

Pilotage d'un exosquelette par une interface Cerveau-Machine

Introduction

Contexte scientifique et technique

En octobre 2019, de nombreux médias, ainsi qu'une publication scientifique majeure ont présenté une première mondiale : un homme de 30 ans, devenu paraplégique suite à un accident, équipé d'implants sur le cerveau et installé dans un exosquelette, a pu se déplacer en marchant au sein d'un laboratoire et toucher des cibles précises avec ses mains artificielles, en contrôlant le système uniquement par sa pensée, comme illustré en figure 1.

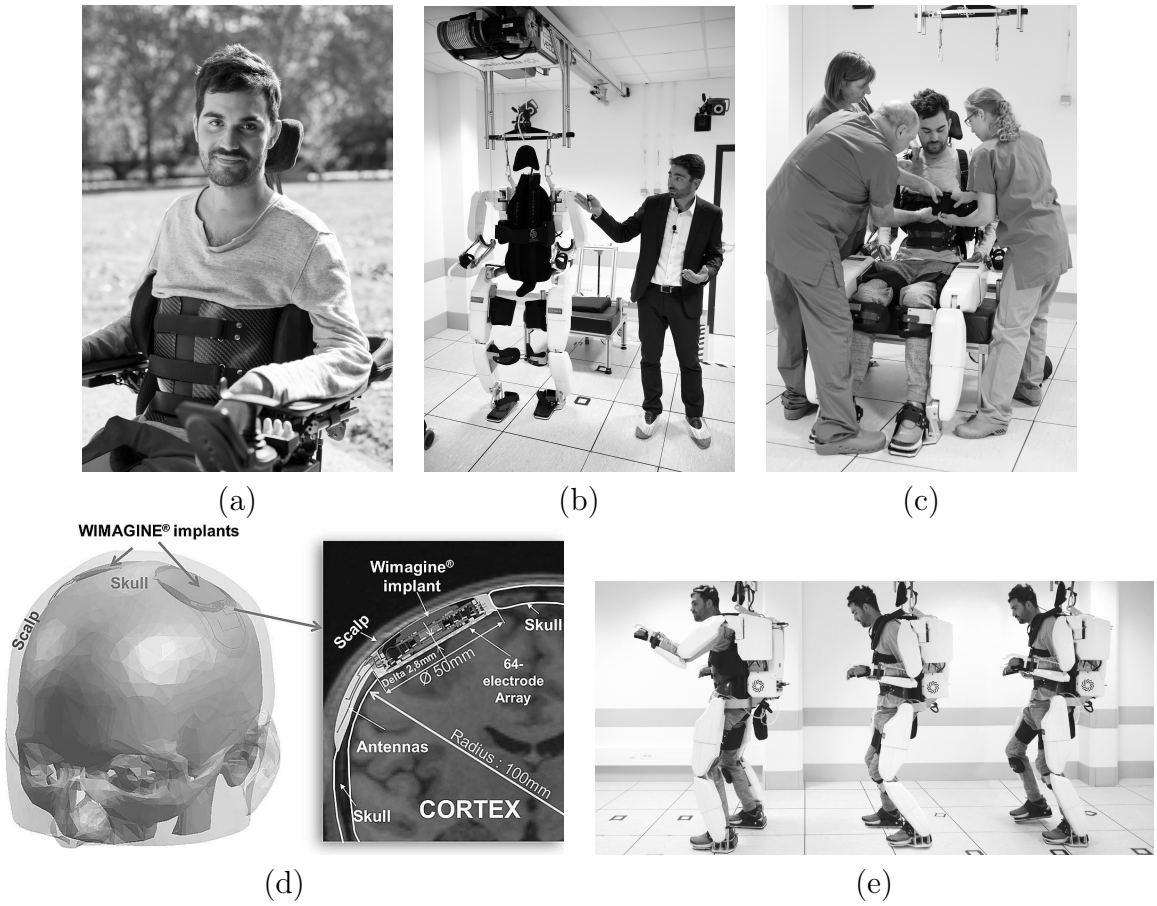


FIGURE 1 – Démonstration de mouvements autonomes contrôlés par la pensée du paraplégique : (a) patient (b) exosquelette (c) installation du patient (d) schéma et radiographie de présentation de l'implant (e) test de marche

Cette prouesse majeure, développée au sein du projet BCI de CLINATEC, est un pas important vers une utilisation ambulatoire d'exosquelettes, pour laquelle il reste encore de nombreux défis à relever.

CLINATEC

Le centre CLINATEC, basé à Grenoble, présidé par le Pr. Benabid, est un établissement de recherche issu d'une collaboration entre le CEA (Commissariat à l'énergie atomique), le CHU de Grenoble et l'UGA (Université Grenoble Alpes), dans le but de lier étroitement recherche médicale et monde industriel. Il associe sur un même site une plateforme technique, où naissent des dispositifs technologiques de pointe, et un hôpital doté des meilleurs équipements. Une particularité qui permet de réunir au chevet des malades des équipes pluridisciplinaires regroupant roboticiens, mathématiciens, physiciens, électroniciens, informaticiens, biologistes, neurologues, chirurgiens et personnels de soins. Cette organisation innovante a pour objectif d'accélérer le transfert des innovations jusqu'au patient, en évaluant rapidement le fonctionnement et la pertinence de systèmes innovants, via la réalisation d'essais cliniques dans les meilleures conditions de sécurité. CLINATEC développe des dispositifs pour des patients souffrant de pathologies neurodégénératives, de cancers cérébraux et d'handicaps moteurs d'origine lésionnelle (tétra- ou paraplégie).

Le projet BCI (Brain Computer Interface)

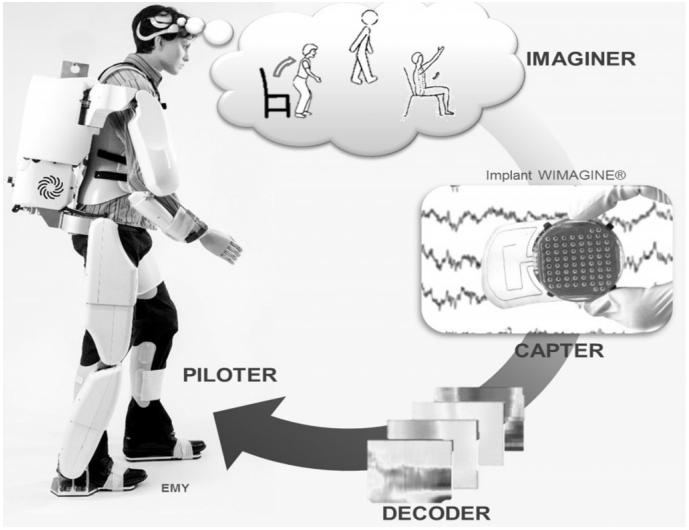
Au sein de CLINATEC, le projet BCI (Brain Computer Interface) a pour objectif de permettre aux personnes souffrant d'un handicap moteur lourd de retrouver de la mobilité grâce à un système de compensation, notamment via le pilotage d'un exosquelette à partir de signaux corticaux captés à l'aide d'un implant. Le principe du projet repose sur le fait qu'imaginer un mouvement ou l'exécuter provoque la *même activité électrique cérébrale* au niveau du cortex moteur. Deux implants permettent de capter ces signaux électriques d'électro-encéphalographie intracrânienne (*Electrocorticography* en anglais, abrégé ECoG par la suite) pour les transmettre à un système informatique, puis de les décoder, afin de piloter des objets complexes, comme bouger des systèmes mécaniques tel l'exosquelette EMY à 8 degrés de liberté.

Système étudié

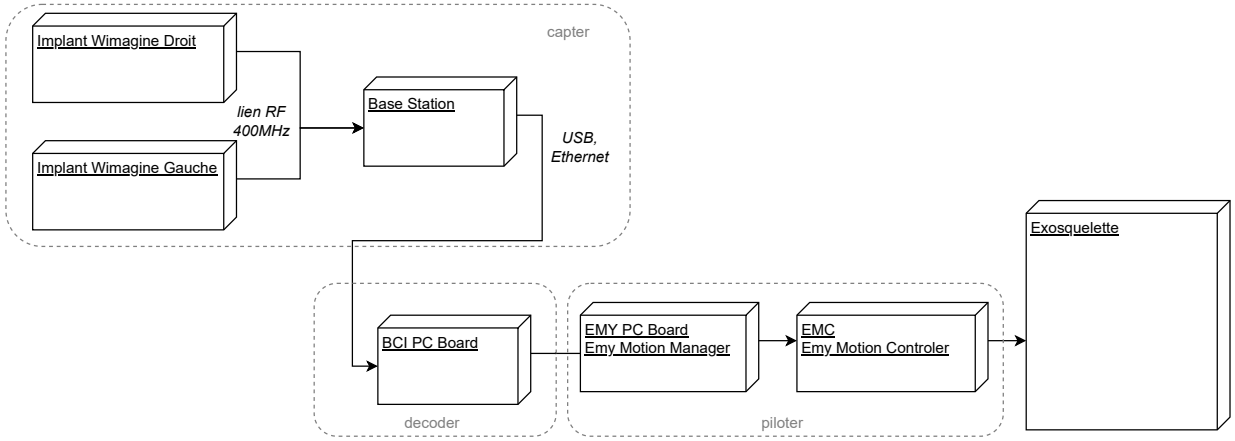
Le système étudié contient plusieurs sous-systèmes physiques permettant d'aller de l'acquisition de l'activité électrique cérébrale au pilotage des actionneurs. Une vue d'ensemble est donnée dans le diagramme de déploiement en figure 2(b). La chaîne logicielle est composée de 3 grands composants répartis sur les sous-systèmes [5] :

- la numérisation des signaux ECoG est assurée par les implants *WIMAGINE*. Lors d'une opération, deux implants sont posés à demeure dans la boîte crânienne du patient, à la place de l'os crânien et en dessous du cuir chevelu. Ils sont en contact avec le cerveau via des électrodes, pour enregistrer les hémisphères droit et gauche. Ils émettent, via des antennes, un signal reçu par un casque amovible posé sur la tête et connecté à une station de base installée dans le dos de l'exosquelette. Ce signal est synchronisé à la réception par la station de base. Les données sont découpées en lots pour le traitement de l'information en temps réel.

- Dans un seconde temps, les signaux ECoG sont traités et soumis à des modèles adaptatifs en charge d'estimer l'intention de mouvement du patient. Ce traitement est réalisé sur la cible matérielle *BCI PC Board* du diagramme de déploiement de la figure 2(b).
- Les ordres moteurs décodés sont ensuite transmis aux systèmes EMM(*EMY Motion Manager*) calculant le positionnement recherché pour l'exosquelette et EMC (*EMY Motion Controller*) gérant les commandes motrices adaptées au pilotage des actionneurs de l'exosquelette.



(a)



(b)

FIGURE 2 – (a) Présentation schématisée du projet BCI de Clinatec. Le patient souffrant d'un handicap moteur est placé dans un exosquelette et imagine le mouvement qu'il souhaite exécuter. Un implant cérébral permet de capter l'activité électrique résultante. Puis un système informatique décode ce signal et pilote l'exosquelette. (b) Diagramme de déploiement UML du système BCI.

La première partie de ce sujet se focalise sur la transmission des données d'enregistrement du signal ECoG et les possibilités de compression du signal afin de réduire le débit entre les implants et la station de base réceptionnant les données. Nous nous intéresserons ensuite au décodage des intentions dans le partie 2. Comment décoder les signaux pour prédire les mouvements ? Enfin dans la partie 3, nous étudierons l'exosquelette à 4 membres EMY, et le pilotage des axes asservis. La figure 3 présente de manière synthétique le cahier des charges

partiel en lien avec les performances étudiées.

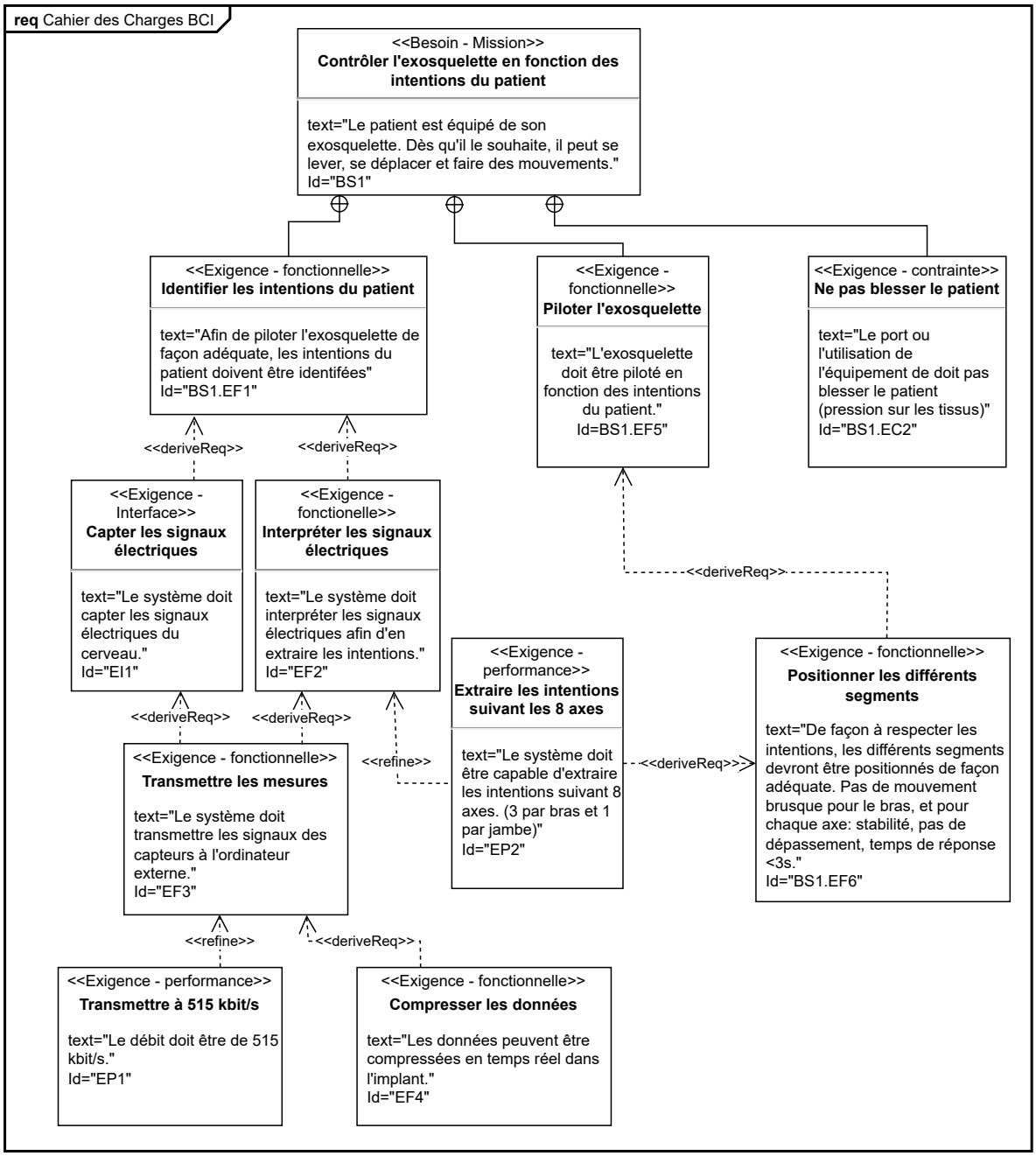


FIGURE 3 – Extrapolation du cahier des charges du système BCI développé par CLINATEC

Partie 1. Compression de l'information ECoG pour la transmission implant-station de base

L'enregistrement des signaux ECoG à la surface du cerveau est réalisé par les implants *WIMAGINE* présentés en figure 4 :

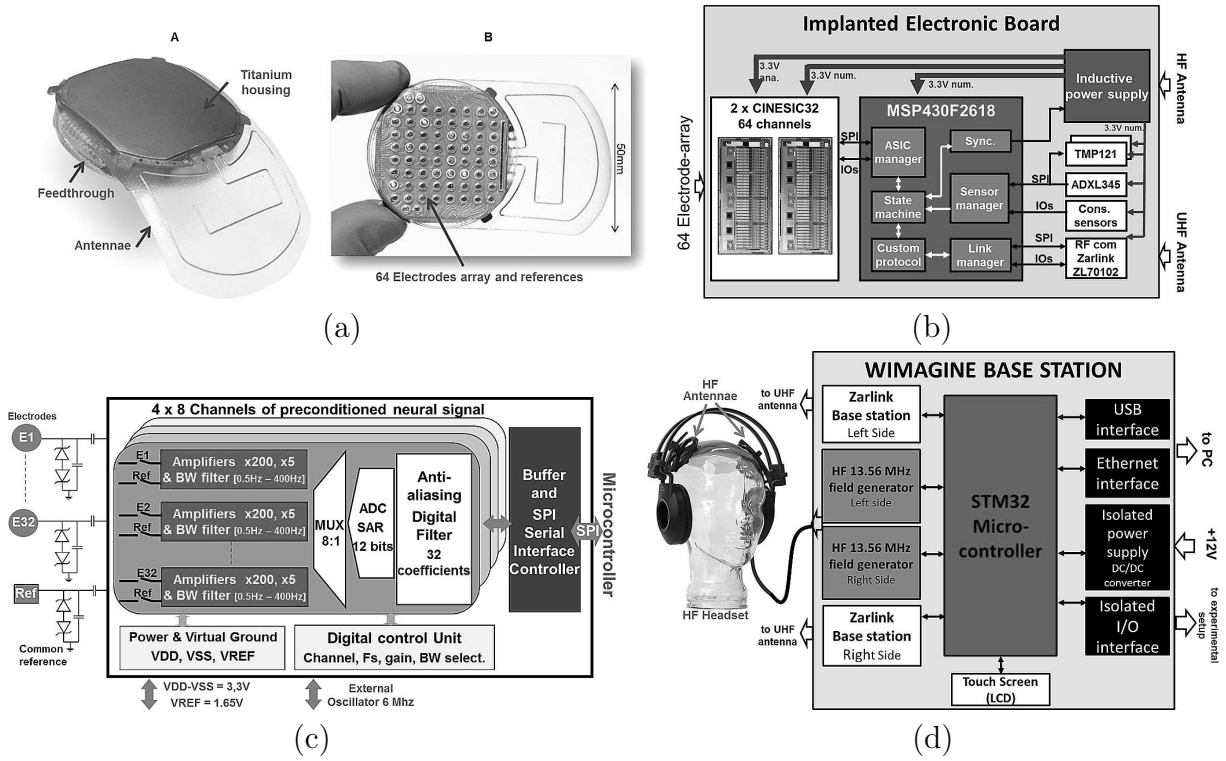


FIGURE 4 – Système implant et station de base : (a) Chaque implant est constitué d'une matrice de 64 électrodes au-dessus desquelles est positionnée l'électronique embarquée (b) Schéma synthétique de l'implant, contenant notamment un micro-contrôleur MSP420F2618 et deux ASIC CINESIC32 permettant l'enregistrement de 64 canaux en simultané (c) Schéma synthétique des ASIC CINESIC32, on peut noter la présence d'un convertisseur Analogique Numérique (SAR ADC 12 bits) (d) Schéma synthétique de la station de base : le lien UHF (400 MHz) sert à la transmission des données, le lien HF permet la transmission de puissance sans fil vers l'implant.

1.1 Contraintes à la transmission du signal ECoG

La communication entre l'implant et la station de base en charge de la réception des signaux ECoG est assurée par une liaison sans fil UHF 400 MHz conçue pour les applications médicales. Les implants utilisent un composant intégré, le *Zarlink ZL70102* dont les spécifications sont données dans le document technique DT1. Les données utilisateur transmises consistent en des paquets de 14 octets.

Question 1. L'objectif est dans un premier temps de déterminer la taille d'une mesure sur les canaux d'acquisition :

(a) Donner la taille d'un échantillon de tension ECoG pour un seul canal ?

(b) Calculer la taille de l'ensemble d'une mesure des potentiels d'ECoG ?

(c) Calculer le nombre de *Data Blocks* définis dans la documentation constructeur (document technique DT1) nécessaire pour envoyer l'ensemble de la mesure ECoG.

Solution: a) Un échantillon est codé par l'ADC sur 12 bits (Fig. 4.c)

b) 1 mesure contient 64 cannaux = $64 \times 12 = 768$bits

c) Dans un DATA BLOCK, le User Data vaut 113 bits

$$\frac{768}{113} = 6.79$$

il faut 7 Data blocks pour transmettre une mesure

Question 2. Il s'agit ici de déterminer le débit théorique et l'occupation de la bande passante par les signaux issus de l'implant. La fréquence d'échantillonnage du signal dans l'implant est de $f_e = 1\,\text{kHz}$.

(a) Déterminer le débit binaire théorique pour la transmission de l'ensemble des signaux ECoG.

Solution: $f_e = 1\,\text{kHz}$, il y a donc en tout 64×10^3 échantillons par seconde, et

$$D_{binaire} = 12 \times 64 \times 10^3 = 768\text{kbit} \cdot s^{-1}$$

(b) A partir des données constructeur en document technique DT1, quel est le débit maximum effectif que la liaison UHF peut supporter ? Conclure sur la possibilité de transmettre à priori l'ensemble des données ECoG.

Solution: comme mentionné en DT1, le débit binaire maximum en 4FSK est de $800\text{kbit} \cdot s^{-1}$, mais le débit efficace n'est que de $515\text{kbit} \cdot s^{-1}$. Sans traitement préalable des données, il n'est pas possible de transmettre les 64 cannaux échantillonnés à 1 kHz. Il faut soit réduire le nombre de canaux, soit compresser les données.

(c) A $f_e = 1\,\text{kHz}$, déterminer le nombre de canaux maximum que la liaison sans fil permettra d'envoyer.

Solution:

$$D_{binaire\,max} = f_e N_c \times 12$$

où N_c est le nombre de canaux.

$$\Rightarrow Nc = \frac{D_{binaire\,max}}{f_e \times 12} = 42.9$$

il est possible de transmettre un maximum de 42 canaux

(d) L'ensemble des canaux sont échantillonnés à $f_e = 1\,\text{kHz}$ et transmis. Calculer le nombre de bits maximum sur lesquels doit être codé un échantillon ?

Solution: de manière similaire : $D_{binaire\,max} = f_e N_c N_b$ où N_b est le nombre de bits. On obtient avec les valeurs numériques $N_b = 8.05$ bits par échantillons. Pour transmettre toute l'information il faut donc compresser les données sur 8.05 bits par échantillon

Soit $s_n(t)$ le signal ECoG du canal n, où $n \in [\![0, 63]\!]$ et t le temps. On note s_n^* le signal échantillonné par le convertisseur analogique numérique à une fréquence f_e, et

$$s_n^*(k) = s_n\left(\frac{k}{f_e}\right)$$

le k-ième échantillon du signal s_n. On note s_n^* la séquence d'échantillons du canal n correspondant à la numérisation du signal s_n. Ce signal est converti sur N_1 bits et les valeurs de $s_n^*(k)$ appartiennent donc à un alphabet A défini par :

$$A = \{a_0, \cdots a_{2^{N_1}-1}\} = \left\{-2^{N_1-1}+1, \cdots, -1, 0, 1, 2, \cdots, 2^{N_1-1}\right\}$$

Soit $p_{i,n}$ la probabilité d'apparition du caractère a_i dans la séquence s_n^*. On a donc de manière triviale :

$$0 \leq p_{i,n} \leq 1,$$

$\forall\,(i,n) \in [\![0, 2^{N_1-1}]\!] \times [\![0, 63]\!]$, et

$$\sum_{i=0}^{2^{N_1}-1} p_{i,n} = 1$$

$\forall n \in [\![0, 63]\!]$

L'entropie H de la séquence s_n^* est définie par :

$$\mathrm{H}(s_n^*) = -\sum_{i=0}^{2^{N_1}-1} p_{i,n} \log_2(p_{i,n})$$

où l'utilisation du logarithme en base 2 permet d'avoir un résultat en bits, et par définition $p_{i,n}\log_2(p_{i,n}) = 0$ si $p_{i,n} = 0$.

Question 3. Afin d'utiliser cette définition sur les signaux ECoG, les propriétés de base de H sont explorées ici :

(a) Montrer que $0 \leq \mathrm{H}(s_n^*)$.

Solution:

$$\mathrm{H}(s_n^*) = -\sum_{i=0}^{2^{N_1}-1} p_{i,n} \log_2(p_{i,n}) = \sum_{i=0}^{2^{N_1}-1} p_{i,n} \log_2\left(\frac{1}{p_{i,n}}\right)$$

or $0 \leq p_{i,n} \leq 1 \Rightarrow \frac{1}{p_{i,n}} \geq 1$

$$\Rightarrow \log_2\left(\frac{1}{p_{i,n}}\right) \geq 0$$

donc l'entropie est une somme de terme positifs, donc positive.

(b) A quelle condition sur la séquence s_n^* l'entropie $\mathrm{H}(s_n^*)$ est-elle nulle ?

Solution: si $\mathrm{H} = 0$ alors soit $p_{i,n} = 0$ soit $\log_2(p_{i,n}) = 0 \Leftrightarrow p_{i,n} = 0$ et par définition

$$\sum_{i=0}^{2^{N_1}-1} p_{i,n} = 1$$

L'unique solution est donc un signal constant, pour lequel la probabilité d'apparition de la valeur de la constante est 1, et 0 pour tous les autres caractères

(c) Dans l'hypothèse où chaque caractère de A est équiprobable dans s_n^*, que vaut l'entropie $\mathrm{H}(s_n^*)$?

Solution: Si chaque caractère est équiprobable alors $p_{i,n} = \frac{1}{2^{N_1}}\ \forall i$

$$\Rightarrow \mathrm{H} = -\sum_{i=0}^{2^{N_1}-1} \frac{1}{2^{N_1}} \log_2\left(\frac{1}{2^{N_1}}\right)$$

$$= \frac{1}{2^{N_1}} \sum_{i=0}^{2^{N_1}-1} \log_2\left(2^{N_1}\right)$$

$$= \frac{2^{N_1}}{2^{N_1}} \log_2\left(2^{N_1}\right) = N_1$$

(d) Montrer que $\mathrm{H}(s_n^*) \leq N_1$.

Solution: $\log_2(x)$ est une fonction concave, donc :

$$\frac{\log_2(x_1) + \log_2(x_2)}{2} \leq \log_2\left(\frac{x_1 + x_2}{2}\right)$$

par généralisation sur une somme à 2^{N_1} termes :

$$\frac{1}{2^{N_1}} \sum_{i=0}^{2^{N_1}-1} log_2(x_i) \leq log_2\left(\frac{1}{2^{N_1}} \sum_{i=0}^{2^{N_1}} x_i\right)$$

$$\Rightarrow \sum_{i=0}^{2^{N_1}-1} p_{i,n} \log_2\left(\frac{1}{p_{i,n}}\right) \leq log_2\left(\sum_{i=0}^{2^{N_1}} \frac{p_{i,n}}{p_{i,n}}\right)$$

$$\Rightarrow \mathrm{H} \leq \log_2\left(2^{N_1}\right) = N_1$$

(e) Conclure sur le sens physique à donner à la quantité d'entropie $\mathrm{H}\left(s_n^*\right)$ de la séquence s_n^*.

Solution: L'entropie correspond à la quantité d'information en bits du signal considéré :

- si le signal est constant il ne faut pas de bits pour le transmettre,
- si il est totalement alétoire, le signal a N_1 bits d'information

De manière générale, l'entropie donne le nombre de bits atteignable par la compression sans pertes.

Question 4. Il s'agit ici de quantifier l'entropie des signaux ECoG afin d'évaluer les possibilités de compression du signal. Les calculs sont réalisés avec le langage *Python* et avec l'API *Numpy*. Les bases d'utilisation de cette API sont rappelées en document technique DT2. Une séquence s_n^* est stockée dans une variable de type *numpy.array*, où chaque élément est un échantillon enregistré en sortie du convertisseur Analogique Numérique et stocké au format *np.int* compris entre $-2^{N_1-1}+1$ et 2^{N_1-1}.

(a) A partir du prototype donné ci-dessous, écrire le code de la fonction *compute_probabilities* qui renvoie une variable itérable contenant les probabilités d'apparition des caractères de A dans la séquence s_n^*. Il est possible d'utiliser la fonction *numpy.bincount* dont l'aide est donnée en document technique DT3.

```
def compute_probabilities(sequence, N_bits=12):
  """calcule la probabilite d'apparition de chaque
  mot binaire de N_bits

  Arguments
  ---------
    sequence : numpy array
      signal ECoG brut sur une electrode en sortie
      du convertisseur Analogique-Numerique
    N_bits : int
      nombre de bits du convertisseur Analogique
  """
```

(b) A partir du prototype donné ci-dessous, écrire le contenu de la fonction *compute_entropy* qui renvoie la valeur d'entropie de la séquence s_n^*.

```
def compute_entropy(signal, N_bits):
  """calcule l'entropie d'un signal,
  numerise sur un nombre de bits donne

  Arguments
  ---------
    signal : numpy array
      signal ECoG brut sur une electrode en sortie
```

```
        du convertisseurAnalogique-Numerique
    N_bits : int
        nombre de bits du convertisseur Analogique
    """
```

Solution:

```
import numpy as np

def compute_probabilities(sequence, N_bits=12):
    prob = np.bincount(sequence, minlength=2**N_bits)/len(sequence)
    return prob

def compute_entropy(sequence, N_bits):
    p = compute_probabilities(sequence, N_bits=N_bits)
    p_log = p
    p_log[p == 0.] = 1.
    contrib = -p*np.log2(p_log)
    return np.sum(contrib)
```

Ces fonctions de calcul de l'entropie appliquées sur une base de données de signaux d'enristrement ECoG, permettent d'obtenir des valeurs rassemblées sous la forme d'un histogramme donné en figure 5.

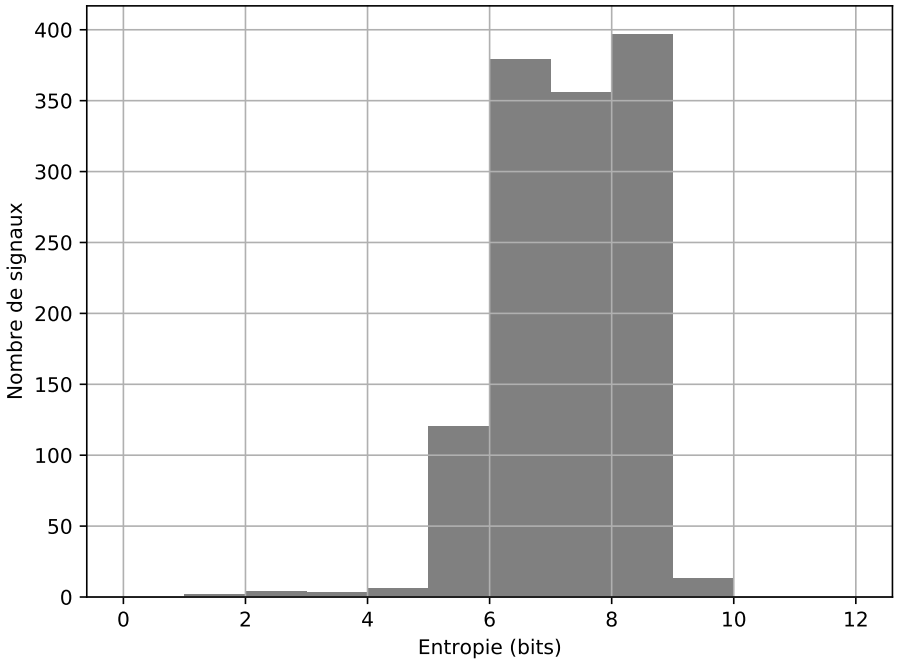


FIGURE 5 – Histogramme des entropies calculées pour une base de données de signaux ECoG

Question 5. Avec une compression sans perte, sur combien de bits est-il possible d'encoder le signal ECoG ? Ce nombre de bits peut-il permettre de transmettre l'ensemble des données ECoG sur la liaison sans fil, en respectant l'exigence EP1 du diagramme d'exigence ?

Solution: On constate que tous les signaux ont une entrpie inférieure à 10 bits. On pourrait compresser le signal sur 10 bits sans perte, ce qui ne permet pas d'envoyer l'ensemble des canaux mais

$$D_{binaire\,max} = f_e N_c \times 10$$

$$\Rightarrow N_c = \frac{515 \cdot 10^3}{10^4} = 51.5$$

soit 51 canaux sans pertes. Si on souhaite les 64 canaux échantillonnés au kHz, il faut compresser avec pertes (même si la perte ne concerne pas tous les canaux, certains contenant moins que 8 bits d'entropie).

1.2 Compression spatiale par utilisation de la Transformée en Cosinus Discrète

Pour une séquence unidimensionelle s on définit la transformée en cosinus discrete S de profondeur N, abrégée DCT-1D par la suite, par :

$$S(k) = \sqrt{\frac{2}{N}} C_k \sum_{i=0}^{N-1} \cos\left(\frac{(2i+1)\,k\pi}{2N}\right) s(i)$$

$\forall k \in [\![0, N-1]\!]$, où

$$C_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{si } k = 0 \\ 1 & \text{si } k > 0 \end{cases}$$

Il est possible d'utiliser une définition matricielle de la DCT-1D, en posant

$$\underline{S} = \begin{bmatrix} S(0) \\ \vdots \\ S(N) \end{bmatrix}, \ \underline{s} = \begin{bmatrix} s(0) \\ \vdots \\ s(N) \end{bmatrix}$$

ce qui revient à écrire

$$\underline{S} = C\underline{s}$$

où C est une matrice de taille $N \times N$ de coefficients définie par :

$$C = \begin{bmatrix} c_{0,0} & \cdots & c_{0,N-1} \\ \vdots & & \vdots \\ c_{N-1,0} & \cdots & c_{N-1,N-1} \end{bmatrix}, \ c_{k,i} = \begin{cases} \frac{1}{\sqrt{N}} & \text{si } k = 0, \forall i \in [\![0, N-1]\!] \\ \sqrt{\frac{2}{N}} \cos\left(\frac{(2i+1)\,k\pi}{2N}\right) & \text{si } k > 0, \forall i \in [\![0, N-1]\!] \end{cases}$$

Question 6. Avant d'appliquer cette transformée sur le signal ECoG, les propriétés fondamentales de la DTC-1D sont explorées ici.

(a) Compléter le document réponse DR-1 avec les valeurs numériques des coefficients de C pour $N = 8$.

(b) Calculer $S(0)$ en fonction de $< s >_{0,N-1}$, la moyenne de la séquence s sur N éléments.

Solution:

$$S(0) = \sqrt{\frac{2}{N}} C_0 \sum_{i=0}^{N-1} \cos\left(\frac{(2i+1)\,0\pi}{2N}\right) s(i)$$

$$= \sqrt{\frac{2}{N}} \frac{1}{\sqrt{2}} \sum_{i=0}^{N-1} s(i)$$

$$= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} s(i) = \sqrt{N} < s >_{0,N-1}$$

(c) Montrer que

$$CC^T = I_N$$

où I_N est la matrice identité de taille N. On pourra admettre que :

- N est pair,
- $\forall k \in [\![1, N-1]\!],\ \sum_{i=0}^{N-1} \cos\left(\frac{(2i+1)\,k\pi}{2N}\right) = 0$

Solution: posons

$$C \cdot C^T = A = \begin{bmatrix} a_{0,0} & \cdots & a_{0,N-1} \\ \vdots & & \vdots \\ a_{N-1,0} & \cdots & a_{N-1,N-1} \end{bmatrix}$$

on cherche à expliciter les termes a_{k_1,k_2}.

- calcul explicite en $k_1 = k_2 = 0$:

$$a_{0,0} = \sum_{i=0}^{N-1} \left(\frac{1}{\sqrt{N}}\right)^2 = \sum_{i=0}^{N-1} \frac{1}{N} = 1$$

- sur la diagonale :

$$a_{k,k} = \sum_{i=0}^{N-1} \left(\sqrt{\frac{2}{N}} \cos\left(\frac{(2i+1)\,k\pi}{2N}\right)\right)^2$$

$$= \frac{2}{N} \sum_{i=0}^{N-1} \cos^2\left(\frac{(2i+1)\,k\pi}{2N}\right)$$

$$= \frac{2}{N}\sum_{i=0}^{N-1}\frac{1}{2}\left(1+\cos\left(2\frac{(2i+1)\,k\pi}{2N}\right)\right)$$

$$= 1+\frac{1}{N}\sum_{i=0}^{N-1}\cos\left(\frac{(2i+1)\,k\pi}{N}\right)$$

$$= 1+\frac{1}{N}\left[\sum_{i=0}^{-\frac{N}{2}-1}\cos\left(\frac{(2i+1)\,k\pi}{N}\right)+\sum_{i=\frac{N}{2}}^{-N-1}\cos\left(\frac{(2i+1)\,k\pi}{N}\right)\right]$$

$$= 1+\frac{1}{N}\left[\sum_{i=0}^{-\frac{N}{2}-1}\cos\left(\frac{(2i+1)\,k\pi}{N}\right)-\sum_{j=0}^{-\frac{N}{2}-1}\cos\left(\frac{(2j+1)\,k\pi}{N}\right)\right]$$

$$= 1$$

- sur les bords de la matrice, de manière évidente $a_{0,k} = a_{k,0}$ et

$$a_{0,k} = \sum_{i=0}^{N-1}\frac{1}{\sqrt{N}}\sqrt{\frac{2}{N}}\cos\left(\frac{(2i+1)\,k\pi}{N}\right)$$

$$= \frac{\sqrt{2}}{N}\sum_{i=0}^{N-1} cos\left(\frac{(2i+1)\,k\pi}{N}\right) = 0$$

  avec l'indication donnée en consigne

- en dehors des diagonales ou bords

$$a_{k_1,k_2} = \sum_{i=0}^{N-1}\left(\sqrt{\frac{2}{N}}\right)^2\cos\left(\frac{(2i+1)\,k_1\pi}{2N}\right)\cos\left(\frac{(2i+1)\,k_2\pi}{2N}\right)$$

$$= \frac{1}{N}\sum_{i=0}^{N-1}\left[\cos\left(\frac{(2i+1)\,(k_1-k_2)\,\pi}{2N}\right)+\cos\left(\frac{(2i+1)\,(k_1+k_2)\,\pi}{2N}\right)\right]$$

$$= \frac{1}{N}\left[\sum_{i=0}^{N-1}\cos\left(\frac{(2i+1)\,(k_1-k_2)\,\pi}{2N}\right)+\sum_{i=0}^{N-1}\cos\left(\frac{(2i+1)\,(k_1+k_2)\,\pi}{2N}\right)\right]$$

$$= 0$$

  car chacune des sommes vaut 0 en utilisant l'indication donnée en consigne.

on a donc

$$CC^T = I_N$$

(d) En déduire que C est orthogonale, c'est à dire que $C^{-1} = C^T$.

Solution:

$$CC^{-1} = I_N = CC^T$$
$$\Leftrightarrow C^{-1}CC^{-1} = C^{-1}CC^T$$
$$\Leftrightarrow C^{-1} = C^T$$

(e) Retrouver la DCT-1D inverse, c'est-à-dire l'expression de $s(k)$ en fonction de S.

Solution: on sait que $\underline{s} = C^T \underline{S}$

$$\Rightarrow s(i) = \sqrt{\frac{2}{N}} \sum_{i=0}^{N-1} C_k \cos\left(\frac{(2i+1)\, k\pi}{2N}\right) S(k)$$

(f) L'énergie d'une séquence est définie par

$$E_S = \| S \|^2 \triangleq \sum_{k=0}^{N-1} S(k)^2 = \underline{S}^t \underline{S}$$

montrer que la DCT-1D conserve l'énergie, c'est-à-dire que $E_S = E_s$.

Solution:

$$\| S \|^2 = \underline{S}^T \underline{S} = (C\underline{s})^T (C\underline{s})$$
$$\Rightarrow E_S = \underline{s}^T C^T C \underline{s} = \underline{s}^T \left(CC^T\right)^T \underline{s} = \underline{s}^T I_N^T \underline{s} = \underline{s}^T I_N \underline{s} = \| s \|^2$$

on a donc $E_S = E_s$

Dans la suite du sujet, les données issues de la mesure ECoG sont représentées sous la forme d'un tenseur **S** à 3 dimensions comme illustré en figure 6.

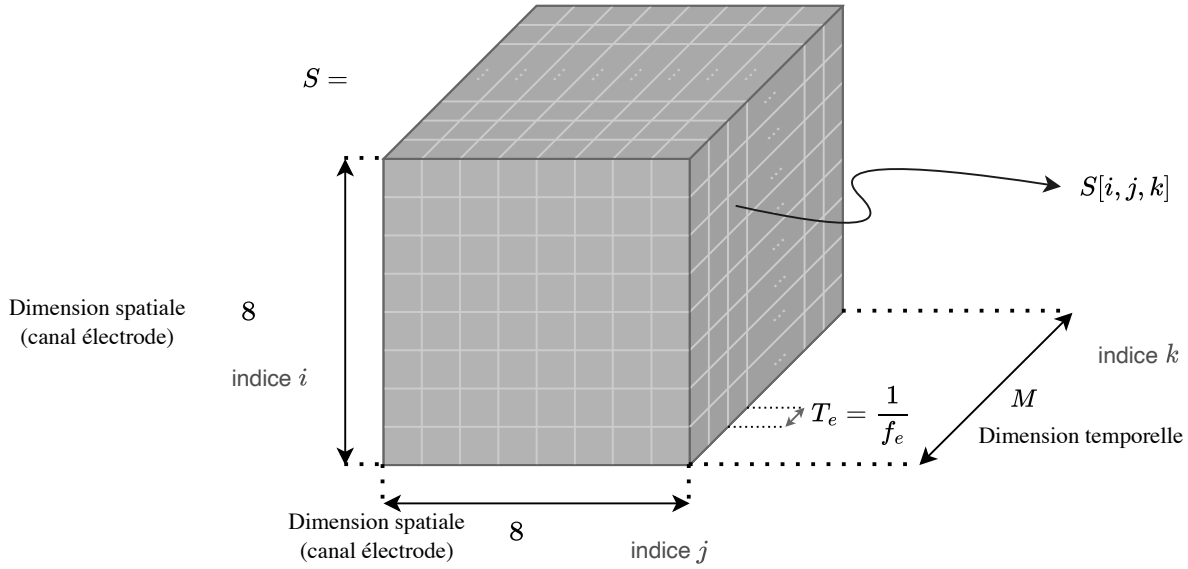


FIGURE 6 – Représentation sous forme de tenseur de dimension 3 des données ECoG, la matrice d'électrodes d'enregistrement est représentée sous forme d'une matrice carrée dont les deux premières dimensions sont spatiales, la troisième dimension représente l'évolution dans le temps

A un numéro d'échantillon temporel donné k, le signal s est donc une matrice 8×8 ($= 64$ canaux) qu'il est possible de traiter avec des méthodes issues du traitement de l'image. En particulier, la corrélation spatiale entre les électrodes étant forte, il est intéressant d'appliquer une Transformée en Cosinus Dicrète (*Discrete Cosine Transform (DCT)*) en 2 dimensions (DCT-2D) pour encoder le signal de manière spatiale et non temporelle. Dans le cas du signal ECoG présent, on transforme un signal contenu dans un tenseur **s**, de dimension $8 \times 8 \times M$, en un tenseur transformé par DCT-2D **S**, de dimension $8 \times 8 \times M$ avec la relation suivante :

$$\begin{aligned}\mathbf{S}\left[i,j,k\right] &= 4C_iC_j \sum_{l=0}^{7}\sum_{m=0}^{7} \cos\left(\frac{\left(2l+1\right)i\pi}{16}\right)\cos\left(\frac{\left(2m+1\right)j\pi}{16}\right)\mathbf{s}\left[l,m,k\right] \\ &= 4C_iC_j \sum_{l=0}^{7}\sum_{m=0}^{7} B_{i,j}\left[l,m\right]\mathbf{s}\left[l,m,k\right]\end{aligned}$$

où les $8 \times 8 = 64$ (indices i et j) matrices B de taille 8×8 (indices l et m) sont des fonctions de base.

La transformée inverse est donnée par la relation suivante :

$$\begin{aligned}\mathbf{s}\left[l,m,k\right] &= \frac{1}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} C_iC_j \cos\left(\frac{\left(2l+1\right)i\pi}{16}\right)\cos\left(\frac{\left(2m+1\right)j\pi}{16}\right)\mathbf{S}\left[i,j,k\right] \\ &= \frac{1}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} C_iC_jB_{i,j}\left[l,m\right]\mathbf{S}\left[i,j,k\right]\end{aligned}$$

Question 7. Afin d'évaluer la sortie de la DCT-2D sur le signal ECoG, il est possible d'implémenter cette transformée en Python.

(a) Implémenter la fonction *DCT2D_matrix* prenant en argument les indices i et j et renvoyant la matrice $B_{i,j}$ dont l'entête est donné ci-dessous :

```
import numpy as np

def DCT2D_matrix(i, j, N=8):
    '''
    Construit la matrice B i,j

    Arguments:
    ----------
    i, j : int
        indices du signal temporel

    Returns:
    --------
    B : numpy.array
    '''
```

(b) Implémenter la fonction *DCT2D_all_matrix* renvoyant l'ensemble des matrices B sous la forme d'une liste de listes (lignes puis colonnes) comme indiqué dans l'entête ci-dessous :

```
import numpy as np

def DCT2D_all_matrix(N):
    '''
    Construit la liste 2D des matrices B i,j pour une profondeur
    de DCT de N

    Returns:
    --------
    mat_listoflist : list of lists
        liste (lignes) de listes (colonnes) des matrices B
        pour une profondeur de DCT de N
    '''
    mat_listoflist = []
```

(c) Implémenter la fonction *DCT2D* renvoyant une matrice correspondant à la DCT2D pour le tenseur **s** à l'échantillon k comme indiqué dans l'entête suivant :

```
import numpy as np

def DCT2D(s,k,N=8):
    '''
    Calcule la DCT2D sur l'echantillon temporel
    k du tenseur s

    Arguments:
    ----------
    s : numpy array
        tenseur ECoG en domaine temporel
    k : int
        indice en temps de l echantillon
```

```
    N : int
      profondeur de DCT, par defaut a 8

    Returns:
    ----------
    S : numpy array
      image de s[:,:,k] par la DCT2D
    '''
```

Solution:

```
import numpy as np

def DCT2D_matrix( i , j , N=8):
    B = np.zeros(N,N)
    for l in range(N):
        for m in range(N):
            B[l,m] = np.cos((2*l+1)*i*np.pi/(2*N)) * np.cos((2*m+1)*j*
    np.pi/(2*N))
    return B

def DCT2D_all_matrix(N):
    mat_listoflist = []
    for i in range(N):
        mat_list = []
        for j in range(N):
            mat_list.append(DCT2D_matrix(i, j, N))
        mat_listoflist.append(mat_list)
    return mat_listoflist

def DCT2D(s,k,N=8):
    mat_listoflis = DCT2D_all_matrix(N)
    slice = s[k,:,:]
    S = np.zeros(N,N)
    for i in range(N):
        Ci = 1 if i>0 else 1/np.sqrt(2)
        for j in range(N):
            Cj = 1 if j>0 else 1/np.sqrt(2)
            S[i, j] = Ci*Cj*np.sum(np.multiply(mat_listoflist[i][j],
    slice))
    return S
```

Question 8. La DCT est implémentée avant l'envoi des données, donc sur le micro-contrôleur embarqué. Cette fonction n'est pas implémentée dans les faits avec le code python, cependant il est possible de se baser sur ce code pour évaluer le coût de calcul.

(a) En considérant qu'une multiplication coûte 3 périodes d'horloge, calculer le nombre de périodes d'horloge nécessaires pour le calcul de la transformée d'une image spaciale de **S**.

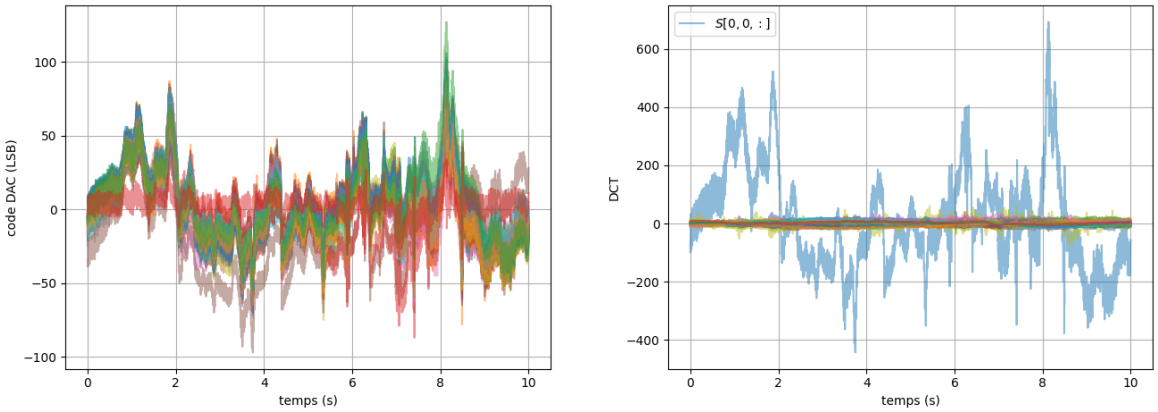
Solution: Pour une image, et à partir du code de la question précédente : il y a 8×8 multiplications matricielles 8×8 par 8×8 soit 64×64 fois 64 additions ou 64 multiplications

$$\Rightarrow N_{coup\,horloge} = 64 \times 64 \times (64 \times (1 + 3)) = 1048567$$

(la DCT est en $\mathcal{O}(N^3)$)

(b) Le signal ECoG étant toujours échantillonné à $f_e = 1\,\text{kHz}$, calculer la fréquence d'horloge minimale pour le micro-contrôleur en ne considérant que le calcul de la DCT.

Solution: à $f_e = 1\,\text{kHz}$: $f_{clk} = f_e \times N_{coup\,horloge} = 1.048\,\text{GHz}$ (l'ordre de grandeur sera rediscuté à la question suivante)



(a) (b)

FIGURE 7 – Exemple de tenseur du signal ECoG brut et après application de la DCT-2D. (a) tracé de 10 secondes des canaux ECoG (un canal par trace) brut en aval de la conversion analogique numérique (b) même signal, après application de la DCT. Le canal se démarquant correspond au tracé à travers le temps pour les indices $(i, j) = (0, 0)$, l'ensemble des autres indices $(i, j) \neq (0, 0)$ apparaissent groupés et de plus faibles amplitudes. Pour ce signal et sur la base de données considérée, les valeurs de DCT pour les indices $(i, j) = (0, 0)$ sont bornées entre -4095 et 4096. Les valeurs de DCT pour les indices $(i, j) \neq (0, 0)$ sont bornées entre -127 et 128.

La figure 7 présente un exemple de tenseur ECoG $\mathbf{s}$ et le résultat $\mathbf{S}$ de l'application de la DCT-2D. En répétant l'acquisition, il est à remarquer que les valeurs de DCT-2D pour les indices $(i, j) = (0, 0)$ sont bornées entre -4095 et 4096. Les valeurs de DCT-2D pour les indices $(i, j) \neq (0, 0)$ sont bornées entre -127 et 128.

Question 9. La DCT-2D permet de transformer l'information, les données dans le micro-contrôleur sont converties en *int* (la partie décimale n'est pas conservée) pour l'ensemble des valeurs issues du calcul de la DCT.

(a) Sur combien de bits devrait être codé le canal $(i, j) = (0, 0)$, après l'application de la DCT-2D ?

> **Solution:** le signal est compris entre -4095 et 4096, donc sur 8192 valeurs et peut être côté sur $\log_2(8192) = 13$bits

(b) Sur combien de bits devraient être codés les canaux $(i, j) \neq (0, 0)$ après l'application de la DCT-2D ?

> **Solution:** pour les autres cas, le signal est sur des valeurs entre -127 et 128, soit 256 valeurs et peut être codé sur 8bits

(c) Calculer le nombre de bits moyens par échantillon.

> **Solution:** En moyenne, on obtient :
>
> $$N_{bits} = \frac{1}{64} \times 13 + \frac{63}{64} 8 = 8.08 \text{ bits}$$

(d) Conclure sur la possibilité du système de transmettre l'ensemble des informations ECoG enregistrables dans l'implant (exigence EF4 du diagramme d'exigence), et proposer éventuellement des solutions permettant d'améliorer l'embarquabilité du calcul de la DCT-2D dans l'implant.

> **Solution:** Cette valeur permet d'être proche du nombre de bits maximum par canal. La DCT utilisée permet d'envisager la compression des signaux afin de transmettre l'ensemble des canaux pour $f_e = 1\,\text{kHz}$. Cependant, le calcul de la DCT sans optimisation supplémentaire impose une fréquence d'horloge du micro-controleur dans le GHz, ce qui est irréaliste est serait plutôt adapté à une cible de calcul de type microprocesseur. La DCT peut être envisagée au prix d'une consommation forte avec des algorithme des type *prunning*.

Partie 2. Décodage des trajectoires à partir des enregistrements ECoG

L'objectif de cette partie est de vérifier l'extraction des intentions de mouvement (EP2 dans le diagramme d'exigence). Les données ont été envoyées par les implants *WIMAGINE* à la station de base, qui restitue les données au '*BCI PC Board*' (cf Fig. 2 (b)). Ce dispositif contient un système appelé *Online Cerebral Decoder* en charge de décoder le signal ECoG et de générer les commandes des mouvements pour le pilotage de l'exosquelette.

La notation sous forme de tenseur utilisée dans la partie précédente sera toujours utilisée. Les données ECoG sont mises sous la forme d'un tenseur non compressé et converties en μV et non plus en codes issus des convertisseurs Analogiques Numériques, il n'est pas nécessaire de considérer les unités des canaux ECoG dans cette partie. Les données $\mathbf{s}$ d'enregistrement ECoG constituent le signal d'entrée du décodage. Dans la suite du sujet, nous nous limiterons à l'enregistrement de l'activité de l'hémisphère gauche par l'implant.

On note $\mathbf{y} \in \mathbb{R}^M$ le mouvement à générer. Sur le système étudié, une première phase de *pre-processing* et d'extraction de caractéristiques (ou *features*) est appliquée, générant un vecteur noté $\mathbf{x} \in \mathbb{R}^N$. Le décodeur est déduit de données ECoG et de mouvements enregistrés sur des sujets valides [6] lors d'une première phase d'apprentissage comme illustré en Figure 8. Dans une seconde phase, les données arrivant en temps réel du tenseur ECoG passent par l'étape de *pre-processing* et d'extraction de *features*, puis sont appliquées au modèle identifié et servent de commande pour l'exosquelette après une étape de *post-processing*.

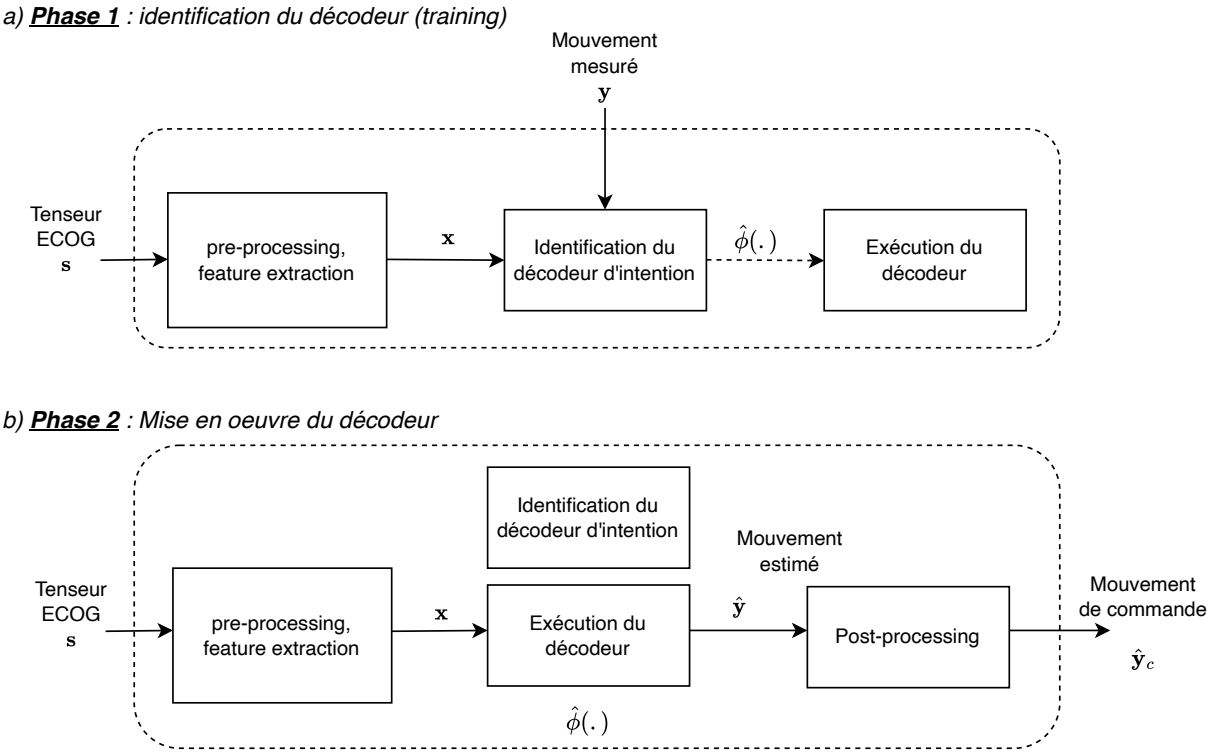


FIGURE 8 – (a) Principe de l'identification d'un décodeur à partir d'enregistrements de signaux ECoG et de mouvements mesurés (b) application du décodeur pour la prédiction d'intention de mouvement.

2.1 Identification des *features*

Pour former un tenseur de caractéristiques, chaque signal ECoG est cartographié dans l'espace temps-fréquences par transformée en ondelettes continue (CWT). Si on considère un signal $s_n(t)$ d'une électrode n, la CWT est définie par :

$$S_{n,w}(s,\tau) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} s(t)\, \overline{\psi}\left(\frac{t-\tau}{s}\right) dt$$

où ψ est une fonction de $\mathbb{R}$ dans $\mathbb{R}$ ou $\mathbb{C}$ appelée ondelette mère ; s et τ sont respectivement les échelle et translation de l'ondelette fille. L'opération $\bar{\cdot}$ désigne la conjugaison complexe. Nous utiliserons dans la suite l'ondelette de Morlet complexe définie par :

$$\psi(t) = \frac{1}{\sqrt[4]{\pi}} e^{\jmath\omega t} e^{-\frac{t^2}{2}}$$

où $\jmath^2 = -1$, et ω est la pulsation en $\mathrm{rad} \cdot \mathrm{s}^{-1}$. La notation $*$ pourra désigner le produit de corrélation.

Question 10. Écrire une fonction *Python* nommée *morlet*, prenant en argument un vecteur t et ω par défaut à la valeur 5, et renvoyant le calcul de l'ondelette mère utilisée.

Solution:

```
def morlet(t, w=5):
  return np.exp(1j * w * t) * np.exp(-0.5 * t**2) * np.pi**(-0.25)
```

Question 11. Dans un premier temps, les propriétés nécessaires à l'utilisation de l'ondelette de Morlet sont étudiées :

(a) Calculer $\Psi(f)$ la transformée de Fourier de $\psi(t)$.

Solution:

$$\begin{aligned}
\Psi(f) &= \mathcal{F}(\psi(t)) \\
&= \frac{1}{\sqrt[4]{\pi}} \frac{1}{2\pi} \left(\mathcal{F}\left(e^{\jmath 2\pi f_0 t}\right) * \mathcal{F}\left(e^{-\frac{t^2}{2}}\right) \right) \\
&= \frac{1}{\sqrt[4]{\pi}} \frac{1}{2\pi} \left(\delta(f - f_0) * e^{-(2\pi f)^2} \right) \\
&= \frac{e^{-4\pi^2 (f-f_0)^2}}{\sqrt{2}\pi^{\frac{5}{4}}}
\end{aligned}$$

avec $\omega = 2\pi f_0$

(b) Identifier la fréquence centrale de l'ondelette mère.

Solution: la fréquence centrale est $f_0 = \frac{w}{2\pi}$

(c) Pour un facteur d'échelle s, quelle sera la fréquence centrale de l'ondelette fille centrée en 0 ($\tau = 0$) : $\tilde{\psi}\left(\frac{t-\tau}{s}\right) = \frac{1}{s}\psi\left(\frac{t}{s}\right)$?

Solution:

$$\tilde{\Psi}(f) = \frac{1}{s}\left(\frac{1}{s}\Psi\left(\frac{f}{s}\right)\right)$$

la fréquence centrale de l'ondelette fille est $sf_0 = \frac{s\omega}{2\pi}$ le facteur d'échelle est directement appliqué sur le spectre de l'ondelette fille.

La CWT introduite ci-dessus est dans le cas présent appliquée à un signal échantillonné à la période $T_e = \frac{1}{f_e}$. La formule intégrale précédente est couramment discrétisée sous la forme :

$$S_{n,w}[s,\tau] = \frac{1}{\sqrt{|s|}}\sum_{n=0}^{N-1} s_n[n]\,\overline{\tilde{\psi}}\left[\frac{\tau-n}{s}\right]$$

soit sous la forme d'un produit de convolution discret. Cette opération peut être réalisée avec la fonction *numpy.convolve* dont la documentation est donnée dans le document technique DT4.

Question 12. Calculer la complexité algorithmique en temps de la CWT discrétisée.

Solution: Pour chaque point on a une some/produit de N termes, réalisée sur N points au total. La complexité est donc en $\mathcal{O}(N^2)$

Question 13. Compléter le code de la fonction de la transformée en ondelette *cwt_ convolve* du document réponse DR2 :

- cette fonction prend en entrée un signal échantillonné *sig*, la période d'échantillonnage T_e et un vecteur (*array_ like*) contenant les fréquences où l'on cherche à identifier les *features*,
- le nombre de valeurs et les valeurs de translations τ sont identiques au nombre d'échantillons et valeurs de temps des échantillons du signal d'entrée,
- il est possible d'utiliser la fonction *morlet*, définie précédemment.

Solution:

```python
def wavelet2(M, s, w=5):
    x = np.arange(0, M) - (M - 1.0) / 2
    x = x / s
    wave_dat = morlet(x, w=w)
    return (1/np.sqrt(s))*wave_dat

def cwt_convolve(sig, freqs, Ts):
    sigma0 = 5
    tlen = len(sig)
    t = np.linspace(0, (tlen-1)*Ts, num=tlen)
    N = len(sig)
    output = np.zeros((len(freqs), tlen), dtype=np.complex128)
    for i, freq in enumerate(freqs):
        s = freq/freqs[0]
        # generate wavelet
        width = 5 / (2*freq*np.pi*Ts)
        x = np.arange(0, N) - (N - 1.0) / (2*width)
        wavelet_data2 = np.conjugate(wavelet2(N, width))
        # convolution
        output[i, :] = np.convolve(sig, wavelet_data2, mode='same')
    return output
```

Question 14. Soit N le nombre d'échantillons et L le nombre de fréquences sur lesquelles sont extraites les *features*. Exprimer la complexité algorithmique en temps de la fonction précédente.

Solution: L'opération de convolution est appelée pour chaque fréquence, soit L fois. La complexité est donc en $\mathcal{O}(LN^2)$.

Les fonctions précédentes sont enregistrées dans une bibliothèque nommée *WaveletTransform*. Une bibliothèque *ecog* permet de charger un signal ECoG dans un objet. Le code suivant est exécuté :

```
import WaveletTransform as wt
import ecog
import numpy as np

# ouverture d un enregistrement ECoG de la base de donnees
# d identification du decodeur (Patient numero 1)
enregistrement = ecog.raw_ECoG('./DB/Patient_1')
T_sample = 1e-3

print(enregistrement.ecog_tensor.shape)
print(type(enregistrement.ecog_tensor[0,0,0]))

# parametres
nsec = 51            # temps de depart
tensor_width = enregistrement.ecog_tensor.shape[0]
start = int((nsec)/T_sample)
stop = int((nsec+1)/T_sample)
freqs = np.linspace(10,150,15)  # frequences pour la CWT

# extraction des features entre les secondes 51 et 52
feature_vector = np.zeros((tensor_width**2, stop-start, len(freqs)))
for i in range(enregistrement.tensor_width**2):
  sig = enregistrement.get_electrode_signal(i)[start:stop]
  cwt = np.log10(np.abs(wt.cwt_convolve(sig, freqs, T_sample)))
  feature_vector[i,:,:] = np.transpose(cwt)
```

L'exécution de ce code renvoie la sortie de console suivante :

```
(8, 8, 1045828)
<class 'numpy.float64'>
```

Question 15. Quelles sont les tailles en nombre d'éléments, puis en octets :

(a) de la variable *sig*,

> **Solution:** *sig* est un tableau de 1045828 points en *float64* soit 8 octets par point. Le poids est donc de 8.39 MOctets (8388508 Octets)

(b) de la variable *cwt*,

> **Solution:** Il y a 15 fréquences dans le vecteur *freqs* donc *cwt* est un tableau de 15×1045828 points en *float64* soit un total de 125.8 MOctets.

(c) de la variable *feature_vector*

> **Solution:** la largeur du tenseur est de 8 (64 électrodes), soit $64 \times 15 \times 1045828$ points en *float64*, soit un poids de 8.05 GOctets

La fonction de convolution utilisée est en majeure partie responsable du temps d'exécution de la transformée en ondelette. Les prochaines questions traitent d'une optimisation possible de ce calcul.

Soit f et g deux signaux échantillonnés. f_N désigne le signal f périodisé à période $N \in \mathbb{N}$ tel que :

$$f_N[kN+n] = f[n], k \in \mathbb{Z} \text{ et } n \in [\![0, N-1]\!]$$

La convolution cyclique est définie par :

$$(f *_N g)[n] = \sum_{m=0}^{N-1} f_N[m]\, g_N[n-m]$$

L'implémentation suivante permet la convolution cyclique en python :

```python
import numpy as np

def cyclic_convolution(f,g):
  N = len(f)
  conv = np.zeros(N)

  for n in range(N):
    for m in range(N):
      if n-m >= 0:
        conv[n] += f[m]*g[n-m]
      else:
        conv[n] += f[m]*g[N+(n-m)]
  return conv
```

Question 16. Dans un premier temps, cette fonction est testée :

(a) Calculer la sortie du code suivant :

```python
import numpy as np

f = np.array([1,3,2])
g = np.array([3,6,4])

print(cyclic_convolution(f,g))
```

> **Solution:** $[27, 23, 28]$

(b) Quelle est la complexité algorithmique en temps du calcul de convolution cyclique ?

> **Solution:** la fonction utilisée contient deux boucles imbriquées de taille N, la complexité est en $\mathcal{O}(N^2)$

La Transformée de Fourier Discrète (abrégée DFT) est définie pour un signal échantillonné f par :

$$DFT\left(f\right)\left[k\right] = \sum_{n=0}^{N-1} f\left[n\right] e^{\frac{-2\jmath\pi}{N}kn}$$

Cette opération est en particulier réalisée par la FFT (*Fast Fourier Transform*) avec une complexité algorithmique en temps de $\mathcal{O}\left(N \log N\right)$. La transformée inverse, ou IDFT, est réalisée par la IFFT (*Inverse FFT*) avec la même complexité.

Question 17. Cette question vise à évaluer l'utilisation de la FFT pour calculer la convolution cyclique :

(a) Montrer que $\forall m \in [\![0, N-1]\!]$:

$$DFT\left(f_N\right)\left[k\right] = \sum_{n=0}^{N-1} f_N\left[n-m\right] e^{\frac{-2\jmath\pi}{N}k(n-m)}$$

Solution:

$$DFT\left(f_N\right)\left[k\right] = \sum_{n=0}^{N-1} f_N\left[n\right] e^{\frac{-2\jmath\pi}{N}kn}$$

soit $l \in [\![0, N-1]\!]$

$$DFT\left(f_N\right)\left[k\right] = \sum_{n=0}^{N-1-l} f_N\left[n\right] e^{\frac{-2\jmath\pi}{N}kn} + \sum_{n=N-l}^{N-1} f_N\left[n\right] e^{\frac{-2\jmath\pi}{N}kn}$$

$$= \sum_{n=0}^{N-1-l} f_N\left[n\right] e^{\frac{-2\jmath\pi}{N}kn} + \sum_{n=l-N}^{-1} f_N\left[n\right] e^{\frac{-2\jmath\pi}{N}kn}$$

en utilisant la N-périodicité de f_N et par $\frac{2\pi}{N}$-périodicité de l'exponentielle complexe,

$$\Rightarrow DFT\left(f_N\right)\left[k\right] = \sum_{n=l-N}^{N-1-l} f_N\left[n\right] e^{\frac{-2\jmath\pi}{N}kn} = \sum_{n=0}^{N-1} f_N\left[n-m\right] e^{\frac{-2\jmath\pi}{N}k(n-m)}$$

en posant $n - m = l - N$

(b) Montrer que

$$DFT\left(f *_N g\right)\left[k\right] = DFT\left(f_N\right)\left[k\right] \cdot DFT\left(g_N\right)\left[k\right]$$

Solution:

$$DFT\left(f *_N g_N\right)[k] = \sum_{n=0}^{N-1}\left(\sum_{m=0}^{N-1} f_N(m)\, g_N(n-m)\right) e^{\frac{-2\jmath\pi}{N}kn}$$

$$= \sum_{m=0}^{N-1}\left(\sum_{n=0}^{N-1} f_N(m)\, g_N(n-m)\right) e^{\frac{-2\jmath\pi}{N}kn}$$

$$= \sum_{m=0}^{N-1} f_N(m)\left(\sum_{n=0}^{N-1} g_N(n-m)\, e^{\frac{-2\jmath\pi}{N}kn}\right)$$

$$= \left(\sum_{m=0}^{N-1} f_N(m)\, e^{\frac{-2\jmath\pi}{N}km}\right)\left(e^{\frac{2\jmath\pi}{N}km}\sum_{n=0}^{N-1} g_N(n-m)\, e^{\frac{-2\jmath\pi}{N}kn}\right)$$

$$= \left(\sum_{m=0}^{N-1} f_N(m)\, e^{\frac{-2\jmath\pi}{N}km}\right)\left(\sum_{n=0}^{N-1} g_N(n-m)\, e^{\frac{-2\jmath\pi}{N}k(n-m)}\right)$$

$$= DFT\left(f_N\right)[k] \cdot DFT\left(g_N\right)[k]$$

(c) En déduire une méthode de calcul de $(f *_N g)$.

Solution: On en déduit que la convolution cyclique peut être calculée en réalisant l'opération $IFFT\left(FFT\left(f_N\right) \cdot FFT\left(g_N\right)\right)$

(d) Quelle sera la complexité de cette méthode de calcul ?

Solution: avec cette méthode, on a :

- 3 appels de fonction de complexité $\mathcal{O}(N \log N)$
- N produits,

au global, la complexité de cette méthode est en $\mathcal{O}(N \log N)$

Question 18. Il est donc possible d'implémenter une autre méthode de calcul de la convolution cyclique :

(a) Proposer une implémentation du calcul de convolution cyclique à partir de l'entête de code suivante :

```
1 import numpy.fft.fft as fft
2 import numpy.fft.ifft as ifft
3
4 def FFT_cyclic_convolution(f,g):
```

Solution:

```
def Fast_Cyclic_Conv(f,g):
  return np.real(np.fft.ifft(np.fft.fft(f)*np.fft.fft(g)))
```

(b) Calculer explicitement la sortie du code suivant :

```
import numpy as np

f = np.array([1,3,2])
g = np.array([3,6,4])

print(FFT_cyclic_convolution(f,g))
```

Solution: $FFT([1,3,2]) = [6, -1.5 - 0.86\jmath, -1.5 + 0.86\jmath]$
$FFT([3,6,4]) = [13, -2 - 1.7\jmath, -2 + 1.7\jmath]$
$FFT([1,3,2]) \times FFT([3,6,4]) = [78, 1.5 + 4.33\jmath, 1.5 - 4.33\jmath]$
$IFFT = [27, 23, 28]$

Comme rappelé dans la documentation *Numpy*, la définition première de la convolution discrète impose d'avoir des signaux définis pour $n \in \mathbb{Z}$:

$$(f * g)[n] = \sum_{m=-\infty}^{+\infty} f[m]\, g[n-m]$$

Dans le problème traité, les signaux f ont des échantillons pour $n \in [\![0, N-1]\!]$. Soit f_P le signal défini par *zero padding* :

$$f^P[n] = \begin{cases} f[n] & \text{si } 0 \leq n < N-1 \\ 0 & \text{si } N \leq n < P-1 \end{cases}$$

avec $P \in \mathbb{N}$ et $n \in [\![0, P-1]\!]$

Question 19. Calculer la valeur minimum de P telle que

$$\forall n \in [\![0, P-1]\!], \left(f^P * g^P\right)[n] = \left(f^P *_P g^P\right)[n]$$

Solution: pour éviter un effet de bord la séquence doit pouvoir translater de $n-m$ jusqu'à un décalage complet (par P-périodicité), soit

$$P = 2N - 1$$

Question 20. L'utilisation de ce résultat permet de remplacer l'appel coûteux en temps à la fonction *numpy.convolve* :

(a) Écrire le code de la fonction *fast_convolution* dont l'entête est donné ci-dessous, en réutilisant la fonction de convolution cyclique avec FFT :

```
import numpy as np
import numpy.fft.fft as fft
import numpy.fft.ifft as ifft

def fast_convolution(f,g):
```

Solution:

```
def Fast_Convolution(f,g):
  N = len(f)
  M = len(g)
  g_pad = np.concatenate((g, np.zeros(N-1)))
  f_pad = np.concatenate((f, np.zeros(M-1)))

  return Fast_Cyclic_Conv(f_pad,g_pad)
```

(b) Calculer la complexité algorithmique en temps de cette fonction.

Solution: La complexité est celle du calcul de la convolution cyclique sur P éléments, soit :

$$\mathcal{O}(P \log P) = \mathcal{O}(2N \log 2N) = \mathcal{O}(N \log N)$$

(c) Le code suivant est implémenté et exécuté :

```
1 import numpy as np
2
3 f = np.array([1,3,2])
4 g = np.array([3,6,4])
5
6 print(np.convolve(f,g,'same'))
7 print(np.convolve(f,g,'full'))
8 print(np.fast_convolution(f,g))
```

Les sorties correspondant aux lignes 6 et 7 sont :

```
[3  15  28  24   8]
[15  28  24]
```

Calculer la sortie de la ligne 8

Solution:

```
[3  15  28  24   8]
```

(d) En fonction de N la taille d'origine de f et g comment indexer le résultat de *fast_convolution* pour obtenir un vecteur de taille N ?

Solution:

```
resultat = np.fastconvolution(f, g)[N//2, N+N//2]
```

Question 21. Compléter le code de calcul de la transformée en ondelette du document réponse DR3.

Question 22. Toujours avec N le nombre d'échantillons et L le nombre de fréquences sur lesquelles sont extraites les *features*., exprimer la complexité algorithmique en temps de la fonction précédente. Comparer cette complexité à la première implémentation de la CWT et conclure sur le gain apporté par le changement d'implémentation au regard du volume des données traitées.

Solution: On a simplement remplacé l'appel en $\mathcal{O}(N^2)$ par $\mathcal{O}(N \log N)$, la complexité globale est en $\mathcal{O}(LN \log N)$. Les données étant de taille particulièrement grande, le gain est important.

2.2 Décodage des intentions de mouvement

Comme décrit en figure 8, le décodage des intentions de mouvement consiste à identifier dans une phase de *training* une fonction $\hat{\phi}$ capable de reconstruire au mieux un mouvement mesuré $\mathbf{y} \in \mathbb{R}^M$ (vecteur d'observation) à partir des *features* extraites $\mathbf{x} \in \mathbb{R}^N$. On peut écrire :

$$\hat{\mathbf{y}} = \hat{\phi}(\mathbf{x})$$

où $\hat{\mathbf{y}} \in \mathbb{R}^M$ est le mouvement estimé, et $\hat{\phi}$ l'estimation de la fonction ϕ. Pour estimer la fonction ϕ lors de la phase d'entrainement, les mesures permettent de disposer de n vecteurs $\mathbf{x}$ et $\mathbf{y}$ mis sous forme de matrices notées respectivement $\mathbf{X} \in \mathbb{R}^{n \times N}$ et $\mathbf{Y} \in \mathbb{R}^{n \times M}$.

Le vecteur de *features* $\mathbf{x}$, correspond à une durée (ou *epoch*) de signal ECoG de 1 seconde (toujours échantillonné à 1 kHz). Ces données sont ensuite passées en entrée de la CWT étudiée à la partie précédente avec pour bande de fréquence 10 Hz à 150 Hz et avec un pas de fréquence de $\delta f = 10$ Hz. Le résultat de la CWT étant un nombre complexe, la valeur du module est prise pour résultat. Les données fréquentielles sont ensuite sous-échantillonnées d'un facteur 100 par moyennage. L'étape de filtrage des artefacts de mouvement visible dans la figure 9 n'a pas de conséquence sur la taille des données en entrée et ne sera pas considérée dans ce sujet.

Les n vecteurs $\mathbf{x}$ sont générés sur une période de *training* de 1400 s en déplaçant successivement la fenêtre de l'*epoch* d'un pas $\delta t = 0.1$ s.

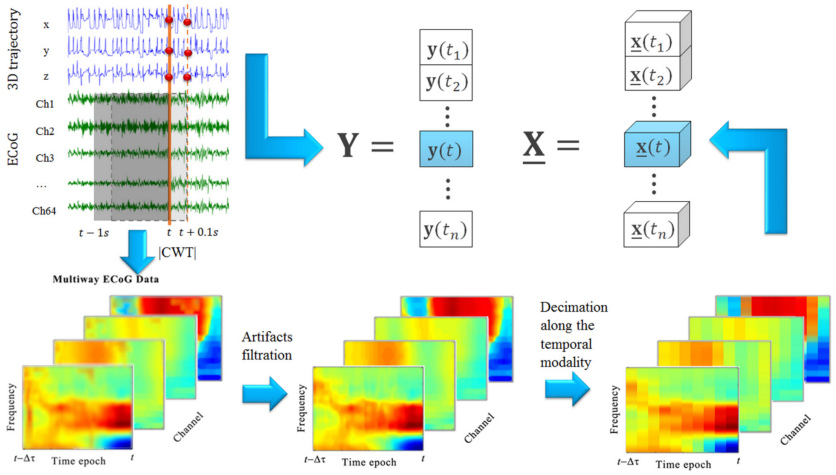


FIGURE 9 – *Mapping* spatial, fréquentiel et temporel des données d'enregistrement ECoG pour le décodage des intentions de mouvement [7].

Question 23. L'objectif de cette question est d'estimer les tailles des vecteurs d'entrée lors de la phase de *training*.

(a) Calculer la taille d'un vecteur de *features* $\mathbf{x}$.

> **Solution:** $\mathbf{x}$ est une partie du tenseur d'entrée sur 1 seconde, donc un tableau de $8 \times 8 \times 10^3 = 64 \cdot 10^3$ éléments. Puis on applique la CWT sur 15 points, soit

$(64 \times 15)\, 10^3 = 960 \cdot 10^3$ éléments. Enfin on sous échantillonne par 100 soit au final 9600 éléments

(b) A partir de la taille d'une *epoch* et le pas δt, calculer le nombre n de vecteurs de *features*.

Solution: on prend $\frac{1}{\delta t} = 10$ features par seconde, sur 1400 secondes, soit

$$n = \frac{1}{\delta t}\,(1400 - 1) = 13990$$

(c) En déduire la taille puis le nombre d'éléments de la matrice $\mathbf{X}$ pour la phase de *training*.

Solution: $n = 13990$ et $N = 9600$ soit $134.304 \cdot 10^6$ éléments

(d) Les données sont constituées de *float64*. Calculer le poids mémoire de la matrice $\mathbf{X}$ lors de la phase de *training*.

Solution: un élément est sur 64 bits soit 8 octets donc $\mathbf{X}$ es sur $\approx$ 1.072 GOctets

Question 24. Le tenseur du signal brut ECoG est mis comme attribut d'un objet de type *ecog* défini en *Python*. Un diagramme de la classe ainsi que les prototypes des méthodes disponibles sont donnés en document technique DT5.

(a) Écrire une méthode *get_ecog_epoch* qui renvoie un tenseur (tableau à 3 dimensions) pour l'*epoch* commençant au temps $'t'$.

(b) Écrire une méthode *compute_features* qui pour un temps $'t'$ calcule et renvoie le vecteur de *feature* $\mathbf{x}$. Il est possible d'utiliser la fonction *Numpy.ravel* comme rappelé dans le document technique DT2 pour transformer le tenseur en vecteur.

Solution:

```
1
2  def get_epoch(self, t):
3      start = int((t)/self.T_sample)
4      stop = int ((t+1)/self.T_sample)
5      epoch = self.ecog_tensor[]
6      return epoch
7
8  def compute_features(self, t):
9      raw_epoch = self.get-epoch(t)
10     # calcul de la CWT
11     freqs = np.linspace(10,150,15)
```

```
        feature_vector = np.zeros((**tensor_width2, -stopstart , len(freqs)
    ))
        for i in range(enregistrement.tensor_width**2):
            sig = raw_epoch[i,:,:]
            cwt_epoch = np.log10(np.abs(wt.cwt_convolve(sig , freqs , self.
    T_sample)))
            feature_vector[i ,: ,:] = np.transpose(cwt)
        return feature_vector
```

Les modèles les plus simples permettant une régression des données sont linéaires. Dans ce cas, la fonction ϕ peut s'écrire :

$$\phi(\mathbf{X}) = \mathbf{BX} = \sum_{i=1}^{n} \beta_i \mathbf{x_i}$$

avec $\mathbf{B} = (\beta_1, \cdots, \beta_n)^T$ respectivement une matrice ou des vecteurs de poids, et $\mathbf{x_i}$ un vecteur de *feature* indexé i parmi les n constituant $\mathbf{X}$.

Identifier le modèle revient à trouver la matrice $\mathbf{B}$ dans ce cas. Une première méthode par les moindres carrés ordinaires, ou *Ordinary Least Square* (noté OLS dans la suite du sujet) permet l'évaluation de cette matrice par minimisation de la quantité $\mathbf{J}$ définie par :

$$\mathbf{J} = \|\mathbf{Y} - \phi(\mathbf{X})\|_2 = (\mathbf{Y} - \mathbf{BX})^T (\mathbf{Y} - \mathbf{BX})$$

Question 25. Pour la méthode OLS :

(a) Montrer que

$$\frac{\partial \mathbf{J}}{\partial \mathbf{B}} = 2(\mathbf{Y} - \mathbf{XB})^T (-\mathbf{X})$$

Solution:

$$\mathbf{J} = (\mathbf{Y} - \mathbf{BX})^T (\mathbf{Y} - \mathbf{BX})$$

$$\Rightarrow \frac{\partial \mathbf{J}}{\partial \mathbf{B}} = \left(\frac{\partial}{\partial \mathbf{B}}(\mathbf{Y} - \mathbf{BX})\right)^T (\mathbf{Y} - \mathbf{BX}) + (\mathbf{Y} - \mathbf{BX})^T \frac{\partial}{\partial \mathbf{B}}(\mathbf{Y} - \mathbf{XB})$$

$$= 2(\mathbf{Y} - \mathbf{XB})^T (-\mathbf{X})$$

(b) En déduire une solution analytique $\hat{\mathbf{B}}$ permettant d'évaluer le modèle au sens des moindres carrés.

Solution: On cherche $\hat{\mathbf{B}}$ correspondant à

$$\frac{\partial \mathbf{J}}{\partial \mathbf{B}} = 0 \Leftrightarrow 2(\mathbf{Y} - \mathbf{XB})^T (-\mathbf{X}) = 0$$

$$
\begin{aligned}
&\Leftrightarrow \mathbf{X}^T(\mathbf{Y} - \mathbf{XB}) = 0 \\
&\mathbf{X}^T\mathbf{Y} = \mathbf{X}^T\mathbf{XB} \\
&\Leftrightarrow \mathbf{B} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}
\end{aligned}
$$

La méthode OLS pose cependant deux problèmes majeurs : les données d'observation sont particulièrement grandes et également fortement corrélées entre elles. La régression sur un modèle linéaire se fait dans ce cas en utilisant des moindres carrés partiels ou *Partial Least Squares* (notés PLS dans la suite du document). Il est possibe de décomposer les matrices $\mathbf{X}$ et $\mathbf{Y}$:

$$
\begin{cases}
\mathbf{X} = & \sum_{f=1}^{F} \mathbf{t}_f \mathbf{p}_f^T + \mathbf{E} = \mathbf{TP}^T + \mathbf{E} \\
\mathbf{Y} = & \sum_{f=1}^{F} \mathbf{u}_f \mathbf{q}_f^T + \mathbf{F} = \mathbf{UQ}^T + \mathbf{F}
\end{cases}
$$

où les F vecteurs $\mathbf{p}$ et $\mathbf{q}$ sont les F composantes principales de $\mathbf{X}$ et $\mathbf{Y}$, et les vecteurs $\mathbf{t}$ et $\mathbf{u}$ sont les F projections de $\mathbf{X}$ et $\mathbf{Y}$ sur ces composantes. Cette projection s'écrit de manière matricielle avec $\mathbf{T}$ et $\mathbf{U}$ les matrices de score, $\mathbf{P}$ et $\mathbf{Q}$ les matrices de charge et $\mathbf{E}$ et $\mathbf{F}$ les matrices de résidu de respectivement $\mathbf{X}$ et $\mathbf{Y}$. Cette décomposition est illustrée en figure 10. Les matrices $\mathbf{T}$ et $\mathbf{U}$ sont choisies afin de maximiser leur covariance et ainsi de projeter les *features* et observations dans une base permettant de faciliter la régression.

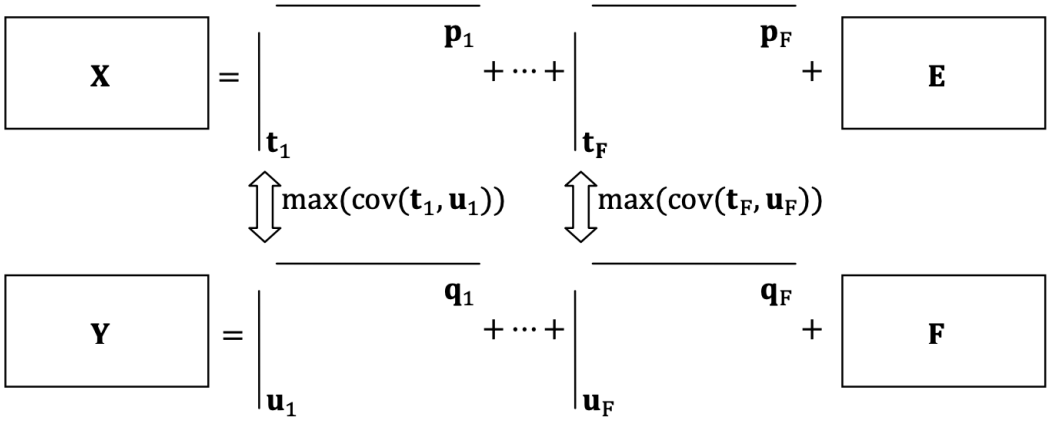


FIGURE 10 – Représentation schématisée des matrices $\mathbf{X}$ et $\mathbf{Y}$ comme étant la somme de scores ($\mathbf{t}$ et $\mathbf{u}$) sur des vecteurs de composantes principales ($\mathbf{p}$ et $\mathbf{q}$), ici au nombre de F et de matrices de résidus $\mathbf{E}$ et $\mathbf{F}$. La décomposition pour les PLS, contrairement à une décomposition par composantes principales simple a pour but de maximiser la corrélation entre les scores $\mathbf{t}$ et $\mathbf{u}$ pour dégager des composantes explicatives du modèle ϕ. Cette figure et les notations sont tirées de [7], attention la notation F désigne un entier, et les vecteurs $\mathbf{q}$ sont indicés de 1 à F, $\mathbf{F}$ désigne une matrice.

Le modèle peut alors s'écrire :

$$
\mathbf{Y} = \phi(\mathbf{X}) = \mathbf{TBQ}^T + \mathbf{F}
$$

Question 26. Si les matrices $\mathbf{T}$, $\mathbf{U}$, $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{E}$ et $\mathbf{F}$ sont identifiées, exprimer la solution analytique permettant d'identifier $\mathbf{B}$ au sens des moindres carrés.

Solution:

$$\begin{cases} \mathbf{Y} &= \mathbf{TBQ}^T + \mathbf{F} \\ \mathbf{Y} &= \mathbf{UQ}^T + F \end{cases} \Leftrightarrow \mathbf{U} = \mathbf{TB}$$

La décomposition est réalisée avec un algorithme itératif développé pour l'extraction de composantes principales nommé NIPALS (*Non linear Iterative PArtial Least Square*). Cet algorithme est synthétisé en figure 11. Afin de simplifier l'écriture, on notera $\mathbf{Y}_k$ la k-ième colonne de la matrice $\mathbf{Y}$ et de manière similaire $\mathbf{X}_k$ la k-ième colonne de la matrice $\mathbf{X}$ avec $k \in [\![1, n]\!]$.

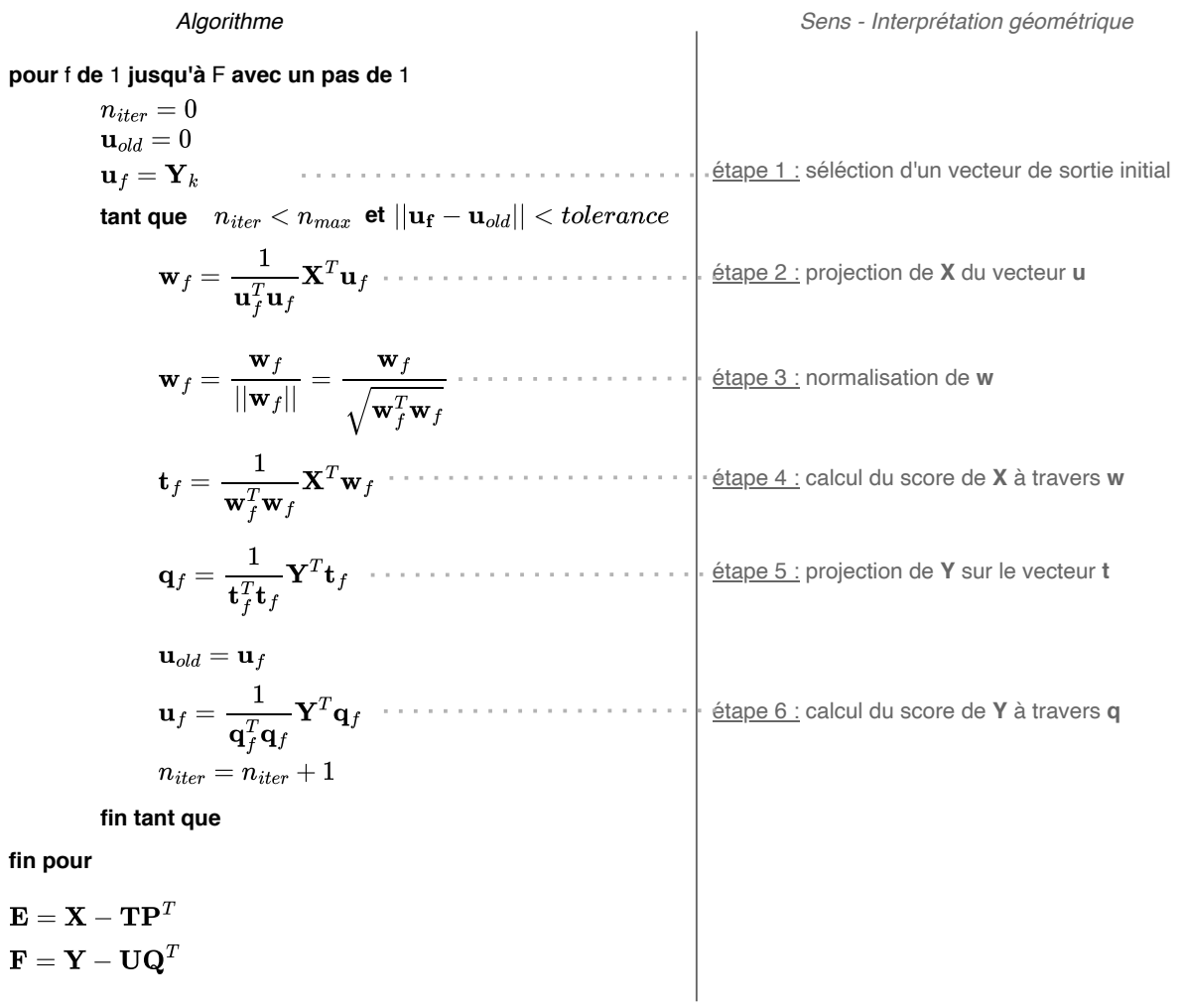


FIGURE 11 – Pseudo-code et éléments d'interprétation géométrique de l'algorithme NIPALS

Question 27. Compléter le tableau correspondant au diagramme du document réponse DR4 à partir de l'algorithme NIPALS. Les numéros d'étapes permettant de répondre sont notés en figure 11.

Solution:

endroit	(a)	(b)	(c)	(d)	(e)
étape	5	2	4	6	3

Question 28. Une classe *PLS* destinée à faire la décomposition et trouver la matrice **B** est implémentée. Cette classe possède une méthode *fit* qui intègre l'algorithme NIPALS et identifie les matrices **T**, **U**, **P**, **Q**, **E**, **F** et **B**. Le code, incomplet est donné en document technique DT6. Les parties à compléter sont écrites dans le document réponse DR5.

(a) Compléter les calculs des variables dans la boucle d'itération à partir de l'algorithme et du diagramme vu en question précédente. (lignes 15, 17, 21, 24, 27 du DR5)

(b) Compléter le calcul de la matrice **B** à la ligne 48 du DR5.

Solution:

```
for comp in range(ncomp):
    train_x_mat = self.x_mat
    train_y_mat = self.y_mat
    nrt, x_nct = train_x_mat.shape
    y_nct = train_y_mat.shape[1]
    train_x_miss = np.isnan(train_x_mat)
    train_y_miss = np.isnan(train_y_mat)
    # Set u to some column of Y
    uh = train_y_mat[:, startcol]
    th = uh

    it = 0
    while True:
        # X-block weights
        wh = train_x_mat.T.dot(uh) / sum(uh * uh)
        # Normalize
        wh = wh / math.sqrt(np.nansum(wh * wh))

        # X-block Scores
        th_old = th
        th = train_x_mat.dot(wh) / sum(wh * wh)

        # Y-block weights
        qh = train_y_mat.T.dot(th) / sum(th * th)

        # Y-block Scores
        uh = train_y_mat.dot(qh) / sum(qh * qh)

        # Check convergence
        if np.nansum((th - th_old) ** 2) < tol:
            break
        it += 1
        if it >= maxiter:
            raise RuntimeError(
                "Convergence was not reached in {} iterations for
   component {}".format(
```

```
                    maxiter, comp
                )
            )

    # Calculate X loadings and rescale the scores and weights
    ph = train_x_mat.T.dot(th) / sum(th * th)

    loadings[:, comp] = ph
    scores[:, comp] = th
    u[:, comp] = uh
    q[:, comp] = qh
    weights[:, comp] = wh
    bh = sum(uh * th) / sum(th**2)
    b[comp] = bh

    self.x_mat = self.x_mat - np.outer(th, ph)
    self.y_mat = self.y_mat - bh * np.outer(th, qh)
```

Question 29. Proposer le code *Python* d'une méthode *eval* qui pour un vecteur de *features* $\mathbf{x}$ calcule le mouvement estimé $\hat{\mathbf{y}}$. Cette méthode sera utilisée dans la phase de mise en oeuvre du décodeur d'intention de mouvement.

Le modèle PLS et d'autres modèles plus complexes, non étudiés en détail ici, ont été mis en oeuvre dans le cadre du développement de l'exosquellette. Le document technique DT7 montre des résultats obtenus par CLINATEC pour l'estimation des coordonnée de mouvement du poignet. Les modèles utilisés sont :

- un filtre de Kalman,
- PLS,
- *Multiway (N-way) PLS* : NPLS,
- *Penalized regression* : SNPLS,
- *Polynomial Penalized PLS* : PNPLS.

Question 30. À partir des résultats quantitatifs présentés en document technique DT7 et d'élements qualitatifs (bruit, rapidité), quels modèles apparaissent comme performants et quelles sont les limites de l'algorithme PLS étudié en détail ?

Solution: Le modèle PLS parvient à prédire le mouvement observé, mais un bruit est présent sur la sortie de l'algorithme. Parmi les algorithmes proposés, le filtre de Kalman (prédicteur) semble inadapté et montre un problème de dérive. Les modèles NPLS, SNPLS semblent plus précis et moins bruités. Le PNPLS est peu bruité, mais ces modèles sont cependant moins rapides.

Les données prédites par le modèle sont ensuite soumises à une étape de *post-processing* avant de servir de signal de contrôle de l'exosquelette, comme étudié dans la partie suivante.

Partie 3. Commande de l'exosquelette

L'objectif de cette partie est de positionner adéquatement les segments de l'exosquelette (objectif DB1.EF6 du diagramme d'exigences). Pour le pilotage de l'exosquelette, deux systèmes sont utilisés : l'EMM (*EMY Motion Manager*) et l'EMC (*EMY Motion Controller*). L'EMM s'occupe de la gestion haut-niveau de l'exosquelette. Ceci est nécessaire car l'exosquelette doit marcher seul, une fois qu'il a reçu l'ordre du patient d'avancer ou de s'arrêter. L'EMC s'occupe quant à lui de l'asservissement de chaque axe. La consigne angulaire est calculée par l'EMM, et devient une entrée de l'EMY, qui gère l'asservissement des axes.

3.1 Motion manager

Les jambes et les bras du robot constituent une structure série (tant que le pied ou la main ne touche rien) avec des liaisons rotoïdes. Le paramétrage de chaque liaison, ou articulation, introduit des coordonnées articulaires. La connaissance de ces coordonnées articulaires permet de trouver la position des effecteurs de bout de chaîne (pied ou main). On parle de l'attitude (*pose*, en anglais) des différents corps les uns par rapport aux autres. Elle est décrite par un vecteur position entre un point du corps et le référentiel (3 distances), ainsi que par l'orientation spatiale (3 angles) pour obtenir les 6 degrés de liberté. Le passage des coordonnées articulaires aux coordonnées opérationnelles s'appelle le modèle géométrique direct et se calcule avec les lois de la cinématique. Des difficultés apparaissent lorsque l'on veut calculer les coordonnées articulaires nécessaires pour suivre une certaine trajectoire du pied (ou de la main) : il faut établir la fonction inverse, appelée modèle géométrique indirect.

Une généralisation des matrices de rotation, ou matrices homogènes, est classiquement utilisée, permettant une intégration de la translation à la matrice de rotation. Ceci permet de transformer l'addition vectorielle nécessaire pour décrire la translation en une multiplication combinée avec les rotations, au prix d'une augmentation de l'ordre de la matrice. Dans ce sujet nous ne travaillerons pas sur les matrices homogènes et nous nous limiterons dans notre étude uniquement à l'orientation, et plus particulièrement au paramétrage des orientations dans l'espace et à l'intérêt des quaternions par rapport aux angles d'Euler.

3.1.1 Angles d'Euler et problème de singularité.

Dans la littérature, les angles proposés par Euler en 1770 pour représenter l'orientation des solides rigides dans l'espace ne sont pas définis toujours de la même façon. Il existe les formulations *roll-yaw-roll*, *roll-pitch-roll* et *roll-pitch-yaw* (roulis-tangage-lacet). C'est cette dernière qui est plutôt utilisée en robotique, et on parle alors parfois d'angles de Cardan. On peut alors décrire la rotation totale $\mathbf{R}(\varphi, \theta, \psi)$ en utilisant des matrices de rotation de $\mathbb{R}^3$ représentant la rotation de chaque angle (aussi appelées matrices aux cosinus directeurs), par une multiplication :

$$\mathbf{R}(\varphi, \theta, \psi) = \underbrace{e^{\psi \mathbf{k}\wedge}}_{\mathbf{R}_\psi} \cdot \underbrace{e^{\theta \mathbf{j}\wedge}}_{\mathbf{R}_\theta} \cdot \underbrace{e^{\varphi \mathbf{i}\wedge}}_{\mathbf{R}_\varphi}$$

où $\mathbf{i} = (1,0,0)^{\mathrm{T}}, \mathbf{j} = (0,1,0)^{\mathrm{T}}, \mathbf{k} = (0,0,1)^{\mathrm{T}}$.

Après développement on obtient :

$$\begin{pmatrix} \cos\theta\cos\psi & -\cos\varphi\sin\psi+\sin\theta\cos\psi\sin\varphi & \sin\psi\sin\varphi+\sin\theta\cos\psi\cos\varphi \\ \cos\theta\sin\psi & \cos\psi\cos\varphi+\sin\theta\sin\psi\sin\varphi & -\cos\psi\sin\varphi+\sin\theta\cos\varphi\sin\psi \\ -\sin\theta & \cos\theta\sin\varphi & \cos\theta\cos\varphi \end{pmatrix}$$

où les colonnes correspondent respectivement à $\mathbf{i}_1|_{\mathbf{R}_0}, \mathbf{j}_1|_{\mathbf{R}_0}, \mathbf{k}_1|_{\mathbf{R}_0}$ et où les angles φ, θ, ψ sont les angles d'Euler.

Question 31. Montrer que lorsque $\cos\theta = 0$ nous avons

$$\frac{d\mathbf{R}}{d\psi} = -\frac{d\mathbf{R}}{d\varphi}.$$

Ceci correspond à une singularité, parfois appelée blocage de Cardan, indiquant que nous perdons un degré de liberté et que nous ne pouvons plus bouger dans toutes les directions de l'espace.

Solution:

$$\mathbf{R}(\varphi,\theta,\psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \underbrace{\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix}}$$

$$= \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \overbrace{\begin{pmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}}$$

$$= \begin{pmatrix} \cos(\psi-\varphi) & -\sin(\psi-\varphi) & 0 \\ \sin(\psi-\varphi) & \cos(\psi-\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

Plutôt que d'utiliser des matrices de rotation, il est possible d'exprimer l'orientation spatiale d'un solide dans l'espace comme la rotation d'un certain angle autour d'un vecteur de l'espace. Si l'axe de rotation est donné par un vecteur (trois paramètres), la longueur de ce vecteur est redondante. Nous avons donc quatre paramètres, le vecteur de l'axe et l'angle de rotation, et une contrainte, limitant par exemple la longueur du vecteur à 1. Nous retrouvons ainsi les trois paramètres libres de l'orientation spatiale.

Des formules pour passer de la représentation (axe/angle) à la matrice de cosinus directeurs et vice-versa sont données dans la littérature [1]. Pour une rotation d'un angle ϑ autour de l'axe $[x, y, z]^{\mathrm{T}}$ avec $\left\|[x, y, z]^{\mathrm{T}}\right\| = 1$, on obtient la matrice :

$$(1 - \cos\vartheta) \begin{bmatrix} xx & xy & xz \\ xy & yy & yz \\ xz & yz & zz \end{bmatrix} + \cos\vartheta \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \sin\vartheta \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$$

La formule suivante nous permet l'inverse, donc de trouver l'axe de rotation et l'angle à partir d'une matrice $\mathbf{R}$, et ce plus directement que par le vecteur propre :

$$\text{avec } \mathbf{R} = \begin{bmatrix} \mathrm{a} & \mathrm{d} & \mathrm{g} \\ \mathrm{b} & \mathrm{e} & \mathrm{h} \\ \mathrm{c} & \mathrm{f} & \mathrm{i} \end{bmatrix} \text{ on obtient l'axe } \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \\ \mathrm{z} \end{bmatrix} = \frac{1}{2\sin(\vartheta)} \begin{bmatrix} \mathrm{f} - \mathrm{h} \\ \mathrm{g} - \mathrm{c} \\ \mathrm{b} - \mathrm{d} \end{bmatrix} \text{ et l'angle } \vartheta$$

$$\cos(\vartheta) = \frac{1}{2}(\mathrm{tr}(R) - 1)$$

$$\sin(\vartheta) = \frac{1}{2}\sqrt{(f-h)^2 + (g-c)^2 + (b-d)^2}$$

Question 32. Justifier pourquoi l'expression de $\sin(\vartheta)$ n'est pas unique. Que se passe-t-il pour l'expression de l'axe si $\vartheta = 0$?

Solution: Ces expressions sont non-uniques : Le signe de la racine peut être positif ou négatif. Pire, pour $\vartheta = 0$ l'axe n'est pas défini. Cela pose des problèmes numériques dans la réalisation du contrôle.

Ces deux inconvénients disparaissent de façon élégante en employant des quaternions.

3.1.2 Quaternions

Les quaternions sont une généralisation des nombres complexes à des nombres à quatre dimensions. Ces nombres hypercomplexes contiennent une partie réelle scalaire λ_0 et trois parties imaginaires $[\lambda_1, \lambda_2, \lambda_3]^{\mathrm{T}}$ qui sont interprétées comme partie vectorielle $\underline{\lambda}$. Le quaternion Q est donc le quadruple :

$$Q = \{\lambda_0, \lambda_1, \lambda_2, \lambda_3\} = \{\lambda_0, \underline{\lambda}\}$$

et peut également s'écrire :

$$Q = \lambda_0 + \imath\lambda_1 + \jmath\lambda_2 + k\lambda_3$$

avec $\imath^2 = \jmath^2 = k^2 = \imath\jmath k = -1$ les imaginaires purs. Les produits de ces imaginaires purs entre eux décrivent des rotations directes sur la partie vectorielle :

$$\begin{cases} \imath\jmath & = k \\ \jmath k & = \imath \\ k\imath & = \jmath \end{cases}$$

et des rotations indirectes :

$$\begin{cases} \jmath\imath & = -k \\ k\jmath & = -\imath \\ \imath k & = -\jmath \end{cases}$$

La direction de l'axe de rotation $[\mathrm{x}, \mathrm{y}, \mathrm{z}]^{\mathrm{T}}$ peut donc être donnée par le vecteur $\underline{\lambda} = [\lambda_1, \lambda_2, \lambda_3]^{\mathrm{T}}$.

L'angle de rotation ϑ est introduit de la façon suivante dans le quaternion Q :

$$\lambda_0 = \cos(\vartheta/2) \quad \text{et} \quad \underline{\lambda} = \sin(\vartheta/2)[x, y, z]^{\mathrm{T}}, \|\mathrm{x}, \mathrm{y}, \mathrm{z}\| = 1$$

Les rotations sont donc représentées par des quaternions unitaires :

$$\lambda_0{}^2 + \lambda_1{}^2 + \lambda_2{}^2 + \lambda_3^2 = 1$$

Une propriété intéressante des quaternions unitaires est que :

$$Q^{-1} = \bar{Q} = \lambda_0 - \imath\lambda_1 - \jmath\lambda_2 - k\lambda_3$$

Il est possible de calculer le résultat de la rotation d'angle ϑ autour d'un axe $[\mathrm{x},\mathrm{y},\mathrm{z}]^{\mathrm{T}}$ d'un vecteur $[\mathrm{a},\mathrm{b},\mathrm{c}]^{\mathrm{T}}$ vers $[\mathrm{a}',\mathrm{b}',\mathrm{c}']^{\mathrm{T}}$ en posant les quaternions :

$$\begin{cases} P &= \imath\mathrm{a} + \jmath\mathrm{b} + k\mathrm{c} \\ P' &= \imath\mathrm{a}' + \jmath\mathrm{b}' + k\mathrm{c}' \end{cases}$$

au moyen de la relation :

$$P' = Q^{-1}PQ$$

Question 33. Une classe *Quaternion* est écrite en *Python*, on souhaite y ajouter la possibilité de multiplication.

(a) Soient les quaternions

$$\begin{cases} Q_A = 1 + \jmath + k \\ Q_B = 2 + 2\imath + 3\jmath + 2k \end{cases}$$

Calculer $Q_A \cdot Q_B$ puis $Q_B \cdot Q_A$

(b) De manière plus générale, soient :

$$\begin{cases} Q_A = \lambda_{0A} + \imath\lambda_{1A} + \jmath\lambda_{2A} + k\lambda_{3A} \\ Q_B = \lambda_{0B} + \imath\lambda_{1B} + \jmath\lambda_{2B} + k\lambda_{3B} \end{cases}$$

Calculer explicitement le produit $Q_A \cdot Q_B$ en fonction des λ_{0A}, λ_{0B}, λ_{1A}, λ_{1B}, λ_{2A}, λ_{2B}, λ_{3A} et λ_{3B}.

(c) Il est possible d'ajouter en python les opérations algébriques pour des objets. Ecrire le code de la méthode $\_\_mul\_\_$ qui a pour argument l'objet (*self*) et un autre nombre (réel, complexe ou quaternion) nommé *other*, qui sera appelée lors de l'exécution du code *self*other*.

Solution: Q33 a

$$Q_A * Q_B = -3 + i + 7j + 2k$$
$$Q_B * Q_A = -3 + 3i + 3j + 6k$$

Q33 b

$$(\lambda_{0A} + \lambda_{1A} * i + \lambda_{2A} * j + \lambda_{3A} * k) * (\lambda_{0B} + \lambda_{1B} * i + \lambda_{2B} * j + \lambda_{3B} * k)$$

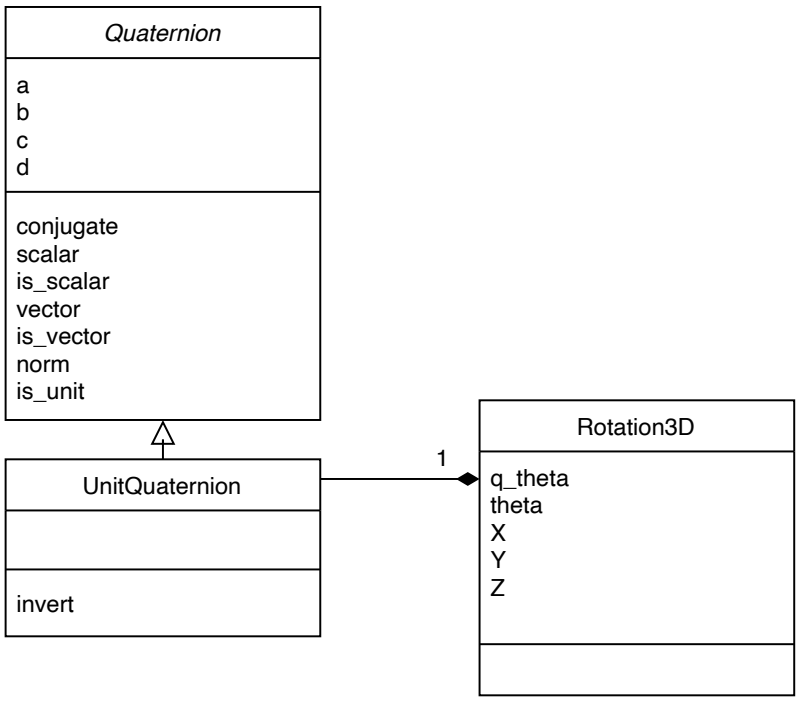
$$\begin{aligned} &= \lambda_{0A} * (\lambda_{0B} + \lambda_{1B} * i + \lambda_{2B} * j + \lambda_{3B} * k) \\ &+\lambda_{1A} * i * (\lambda_{0B} + \lambda_{1B} * i + \lambda_{2B} * j + \lambda_{3B} * k) \\ &+\lambda_{2A} * j * (\lambda_{0B} + \lambda_{1B} * i + \lambda_{2B} * j + \lambda_{3B} * k) \\ &+\lambda_{3A} * k * (\lambda_{0B} + \lambda_{1B} * i + \lambda_{2B} * j + \lambda_{3B} * k) \end{aligned}$$

Q33 c

```python
def __mul__(self, other):
    if isinstance(other, Quaternion):
        # Hamiltonian product
        new_a = self.a*other.a - self.b*other.b - self.c*other.c - self.d*other.d
        new_b = self.a*other.b + self.b*other.a + self.c*other.d - self.d*other.c
        new_c = self.a*other.c - self.b*other.d + self.c*other.a + self.d*other.b
        new_d = self.a*other.d + self.b*other.c - self.c*other.b + self.d*other.a
        return Quaternion(new_a, new_b, new_c, new_d)
    elif isinstance(other, int) or isinstance(other, float):
        return Quaternion(self.a*other, self.b*other, self.c*other, self.d*other)
    elif isinstance(other, complex):
        new_a = self.a*other.real - self.b*other.imag
        new_b = self.a*other.imag + self.b*other.a
        new_c = self.c*other.real + self.d*other.imag
        new_d = - self.c*other.imag + self.d*other.real
    else:
        raise TypeError("Operation not allowed, multiplication can be with Quaternion, int, float or complex, not"+str(type(other)))
```

Question 34. Une bibliothèque de code (incomplet) permettant de réaliser les rotations est proposée en document technique DT8. Compléter le diagramme UML des classes en document réponse DR6.

Solution:



Question 35. Compléter le code de la méthode d'initialisation de la classe *Rotation3D* en document réponse DR7.

Solution: Q35

```
class Rotation3D:
  def __init__(self, theta, X, Y, Z, deg=True):
    if deg:
      self.theta_deg = theta
      self.theta = np.radians(theta)
    else:
      self.theta = theta
      self.theta_deg = np.degrees(theta)

    norm = np.sqrt(X**2+Y**2+Z**2)
    self.X = X/norm
    self.Y = Y/norm
    self.Z = Z/norm
    Rq = np.cos(self.theta/2)
    Iq = np.sin(self.theta/2)
    self.q_theta = UnitQuaternion(Rq, self.X*Iq, self.Y*Iq, self.Z*Iq)
    self.q_theta_inv = self.q_theta.invert()

```

Question 36. Ajouter une méthode *__call__* à la classe *Rotation3D* prenant en argument les coordonnées *a, b, c* d'un vecteur et renvoyant les coordonnées résultant de sa rotation par l'angle et l'axe définis dans l'objet de type *Rotation3D*.

Solution: Q36

```
  def __call__(self, x, y, z):
    p = Quaternion(0, x, y, z)
    p_prime = self.q_theta_inv*p*self.q_theta
    return p_prime.b, p_prime.c, p_prime.d

```

3.2 Motion Controller

Une étude dynamique (équations du mouvement) a été établie pour le dimensionnement de la mécanique, la conception de l'entraînement, et finalement le contrôle du robot. Ce contrôle nécessite de mettre en place un asservissement au niveau de chaque axe, dont la consigne a été établie par le Motion Manager : nous allons étudier la modélisation de l'un de ces asservissements.

3.2.1 Modélisation du comportement fréquentiel

Nous commencons par étudier la stabilité du système, en s'intéressant au comportement fréquentiel de sa Fonction de Transfert en Boucle Ouverte (notée FTBO par la suite). Tout d'abord nous nous s'intéresserons au calcul de sa phase, pour ensuite tracer son diagramme de Black, et établir un calcul de la marge de stabilité du système. Le code permettant le tracé du diagramme de Bode est fourni dans le document technique DT9.

On remarque que la fonction angle renvoie une valeur entre -180° et +180° (ceci est dû à l'utilisation de la fonction arctangente), ce qui crée des discontinuités dans la courbe représentant la phase, comme illustré dans le graphique de la figure 12 représentant le gain et la phase brute. Un traitement permettant d'éviter les discontinuités de la phase a permis d'obtenir le 3ème graphique représentant la phase corrigée.

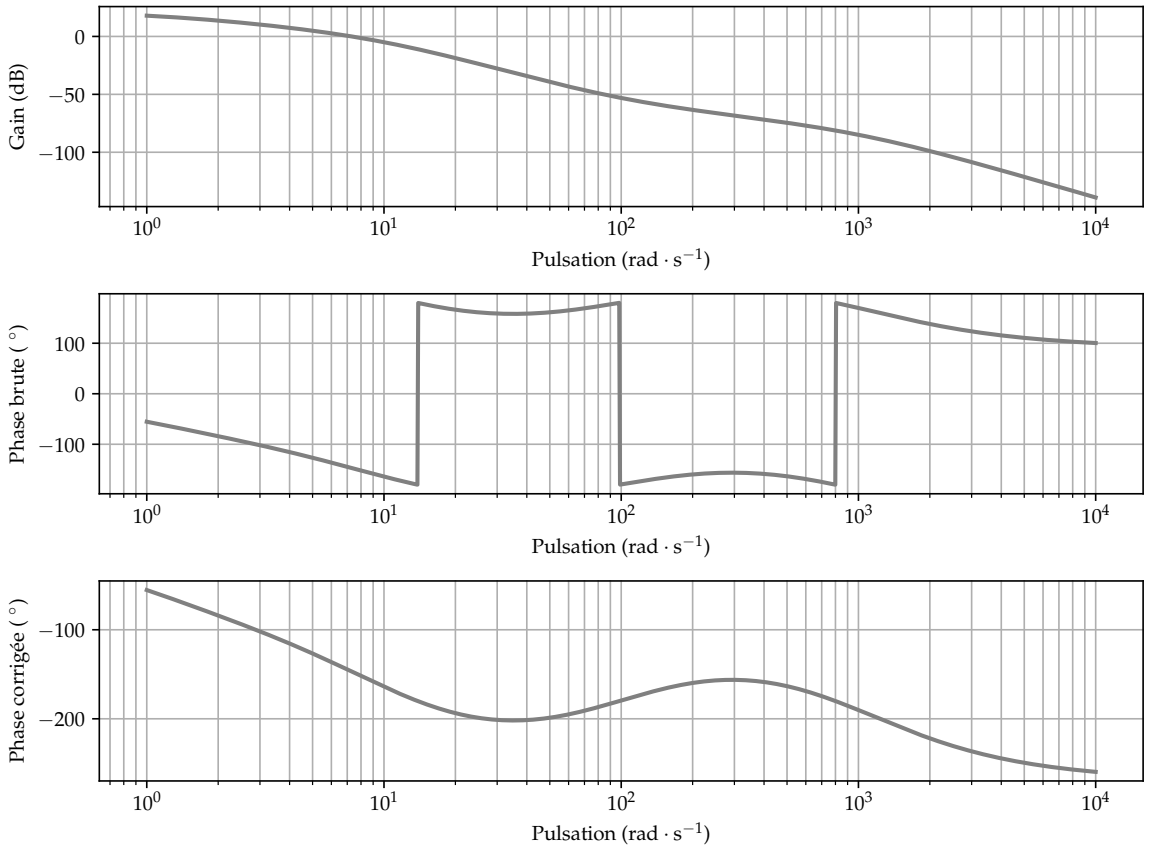


Figure 12 – Exemple de diagramme de Bode d'un système asservi présentant une discontinuité au calcul de la phase.

Question 37. Ce problème est traité par la librairie *python-control* qui implémente des fonctions d'analyse et d'aide à la conception d'asservissement, avec la fonction *unwrap* qui a été partiellement recopiée aux lignes 14 et 19 dans le code fourni dans le document technique DT9.

(a) A partir du code donné en document technique DT9, compléter les valeurs des variables demandées en document réponse DR8.

> **Solution:** correction [0 0 -360 0 0 -360 -360 -360] Phi [-55 -104 -164 -202 -179 -156 -190 -238 -260]

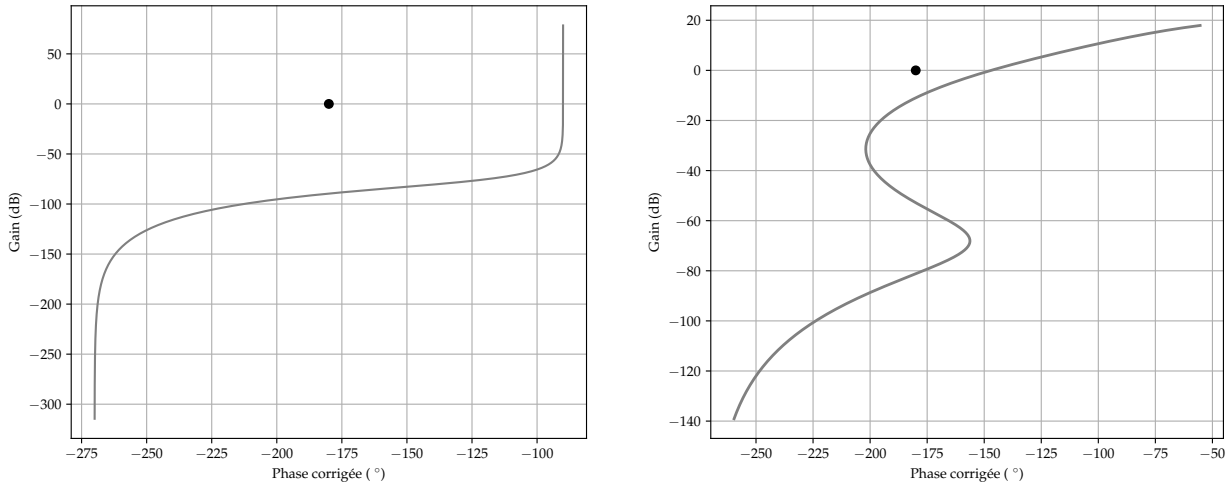
(b) A partir du code donné en document technique DT9, écrire le code de la ligne 19 permettant d'obtenir la phase sans discontinuité.

> **Solution:** Phi2[1 :] += correction

Question 38. Proposer une suite au code présenté en DT9 traçant le diagramme de Black (Gain en fonction de la phase) de la FTBO comme illustré en figure 13(a).

Solution:

```
1  def Black(G, Phi, l=""):
2      """tracé du diagramme de Black Nichols"""
3      plt.legend(loc='best')
4      #plt.title("Diagramme de Black", l)
5      plt.xlabel("Phase en rad")
6      plt.ylabel("Gain en dB")
7      plt.minorticks_on()
8      plt.grid(which='both')
9      plt.tight_layout()
10     plt.plot(Phi, G, label=l)
11     plt.show()
12
13 plt.figure()
14 Black(Gain, Phase2, "pas corrigé")
```



(a) (b)

FIGURE 13 – (a) Exemple de diagramme de Black d'un système asservi. (b) Diagramme de Black du système étudié. Le point tracé correspond au lieu d'instabilité.

Question 39. Évaluation des marges de stabilité.

(a) Écrire une fonction *dicho_tableau(F, v)* retournant les indices qui encadrent la valeur v dans un tableau trié F par dichotomie.

Solution:

```
def dicho_tableau(F, v):
    """Retourne les indices encadrant la valeur v dans un tableau trié F """
    a, b = 0, len(F)-1  # attention au -1 car on veut l'indice, pas la position
    while b-a > 1:          # tant que l'intervalle des indices est supérieur á 1
        m = (a+b)//2          # div euclidienne car les indices sont des entiers
        if (F[a]-v)*(F[m]-v) <= 0:        # signe opposés de F, c'est de ce côté
            b = m               # faire le -v dans le if donne moins d'opérations
        else:             # que si on l'avait soustrait á toutes les valeurs de F
            a = m
    return a, b
```

(b) Donner le code d'une ligne de commande permettant de trouver la marge de phase $M_\phi = Arg\,(FTBO\,(\jmath\omega_{0dB})) + 180°$ pour le système de la figure 13 (a).

Solution:

```
a, b = dicho_tableau(Gain, 0)
Mϕ = (Phase2[a]+Phase2[b])/2 + 180
print("Marge de phase", Mϕ, "degrés")    #  72deg au début ?
```

```
4
```

(c) Faire de même pour la marge de gain $M_G = -20\log|FTBO\left(\jmath\omega_{-180^\circ}\right)|$.

Solution:

```
c, d = dicho_tableau(Phase2, -180)
Mϕ = -(Gain[a]+Gain[b])/2
```

Question 40. Le tracé de Black du système étudié par la suite est représenté figure 13 (b).

(a) Justifier qu'une méthode similaire n'est pas utilisable pour trouver une des marges.

Solution: La fonction n'est plus monotone, on ne peut pas utiliser la dichotomie.

(b) Enfin, qu'en est-il du cas où il y aurait une résonance (le gain augmente avant de redescendre) ?

Solution: De la même façon, la fonction n'est plus monotone.

3.2.2 Simulation de la réponse temporelle et des performances dynamiques du système

On s'intéresse au réglage optimal de l'asservissement de l'articulation de l'exosquelette. La première étape est la mesure des performances. Pour dépasser les blocages précédents, on va construire une nouvelle modélisation de la commande, qui nous permettra d'évaluer la stabilité, puis nous mettrons en place des outils d'évaluation des performances et enfin des outils de réglage du correcteur.

La commande en position angulaire de l'articulation est asservie pour compenser les perturbations (effets de la gravité, chocs externes, frottements internes). Soit θ_C l'angle de consigne élaboré par le motion manager, et θ_S l'angle réellement obtenu, tous les deux en radians. C_r en N.m représente le couple résistant exercé par l'extérieur sur l'axe.

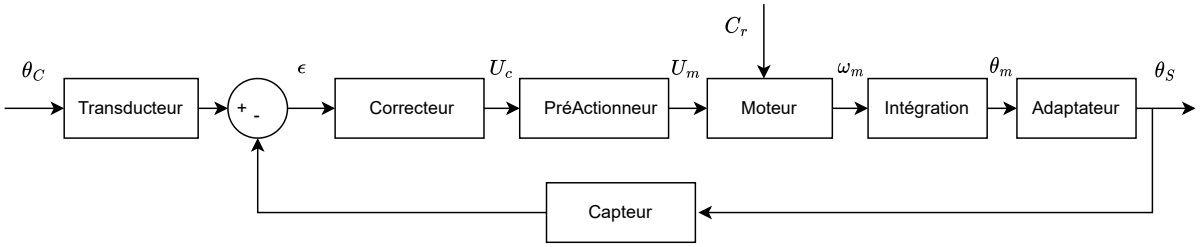


FIGURE 14 – Schéma bloc de l'asservissement

L'image de la consigne de position θ_C est comparée à l'image de la position réelle du bras mesurée par un capteur de position absolu. L'écart généré ϵ est adapté via un correcteur de fonction de transfert $C(p)$ pour donner une tension U_c :

$$C\left(p\right) = K_d \left(1 + \frac{1}{\tau_i \cdot p} + \frac{\tau_d \cdot p}{1 + \frac{\tau_d}{N} p}\right)$$

où l'action dérivée est filtrée avec un filtre passe-bas du premier ordre de constante de temps τ_d/N de façon à éviter une trop grande sensibilité aux bruits de mesure et aux fronts de la consigne, et permet d'être synthétisable.

Néanmoins, afin de simplifier le problème et parce qu'en pratique le filtre anti-repliement après le convertisseur numérique-analogique crée un effet similaire, nous nous contenterons d'un modèle théorique simplifié :

$$C(p) = K_p \left(1 + \frac{1}{T_i.p} + T_d.p\right)$$

Pour commencer, on suppose que le gain du correcteur est de 1 et on traitera son réglage par la suite.

La tension créée par le correcteur rentre dans le pré-actionneur puis le moteur qui fournit θ_S. Il inclut un moteur électrique *brushless* avec son pré-actionneur, et une transmission constituée d'engrenages et d'un système à cabestan permettant de mettre en rotation l'axe que l'on cherche à asservir. On désignera par "moto-réducteur" ce système dans la suite de cette partie. Il reçoit le couple résistant venant de l'extérieur, noté $C_r(p)$. Le schéma bloc correspondant se trouve figure 15, où l'algèbre des schémas blocs a été utilisée pour faire apparaître un retour unitaire.

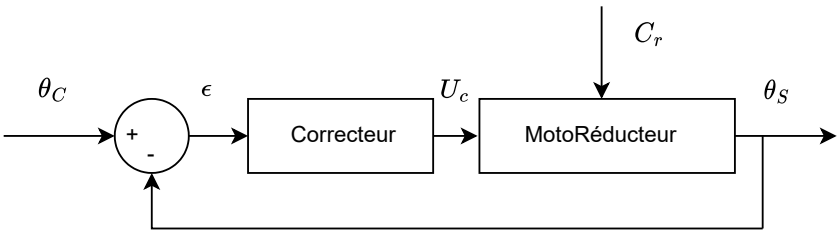


FIGURE 15 – Schéma bloc de l'asservissement

Nous allons établir une simulation numérique de ce système, pour étudier son comportement vis-à-vis de la consigne. En utilisant le théorème de superposition, on supposera donc la perturbation nulle pour la suite du sujet. La première étape est de vérifier la stabilité, en vérifiant que la partie réelle des pôles de la Fonction de Transfert en Boucle Fermée (notée FTBF par la suite) est strictement négative. Pour cela nous allons rentrer la fonction étudiée sous la forme de deux listes correspondant aux coefficients des polynômes constituant son numérateur et son dénominateur. Pour garder une écriture selon la convention de *MATLAB* ou du module *python-control*, dans toute la suite les coefficients seront rangés selon les puissances décroissantes.

Question 41. Écrire une fonction *somme_poly(P, Q)* prenant en arguments deux listes représentant les coefficients de polynômes et renvoyant une liste représentant le polynôme somme.

Solution:

```
def somme_poly(P, Q):
    """somme de polynomes avec puiss décroissantes"""
    temp = np.zeros(max([len(P), len(Q)]))
    if len(P) < len(Q):
        temp[len(Q)-len(P):] = P
        S = temp+Q
    else:
        temp[len(P)-len(Q):] = Q
        S = temp+P
    return(S)
```

Question 42. Écrire une fonction *multi_poly(P, Q)* prenant en arguments deux listes représentant les coefficients de polynômes et renvoyant une liste représentant le polynôme produit.

Solution:

```
def multi_poly(P, Q):
    """produit de polynomes avec puiss décroissantes"""
    deg_max = (len(P)-1)+(len(Q)-1)
    P1 = np.zeros(deg_max+1)
    Q1 = np.zeros(deg_max+1)
    for j in range(len(P)):
```

```
    P1[j] = P[j]
  for j in range(len(Q)):
    Q1[j] = Q[j]
  R = np.zeros(deg_max+1)
  for k in range(deg_max+1):
    for i in range(k+1):
      R[k] = R[k]+P1[i]*Q1[k-i]
  return R
```

Des fonctions sont fournies dans le DT10. La fonction *multi_FT(num1, den1, num2, den2)* permet de calculer le numérateur et le dénominateur représentant la simplification de deux fonctions de transfert en série. Elle sera utile pour regrouper le correcteur et le motoréducteur en une seule fonction de transfert appelée FTBO. La fonction *FTBF(num, den)* permet de calculer la fonction de transfert du système bouclé à retour unitaire à partir du numérateur et du dénominateur de la FTBO.

Question 43. Un système est stable si les parties réelles des racines du dénominateur de sa FTBF sont négatives.

(a) Écrire une fonction *Est_stable(P)* qui prend en argument une liste représentant un polynôme (ordre des puissances décroissantes), et renvoie *True* si les parties réelles des racines du dénominateur de sa FTBF sont négatives et *False* sinon. On pourra utiliser la fonction *np.roots* décrite à la fin du DT2.

Solution:

```
def stabilite(P):
  """prend un dénominateur de ftbf et dit si elle est stable"""
  rep = True
  racines = np.roots(P)
  for k in range(len(racines)):
    if np.real(racines[k]) > 0:
      rep = False
      break
  return(rep)
```

(b) Donner le code permettant d'appeler cette fonction pour valider la stabilité du système corrigé, en prenant un correcteur de gain unitaire.

Solution:

```
A = multi_FT([1], [1], numG, denG)
B = FTBF(*A)
S_0 = stabilite(B[1])
```

3.2.3 Evaluation des performances

La réponse indicielle du système, obtenue avec le code fourni, suit la courbe suivante :

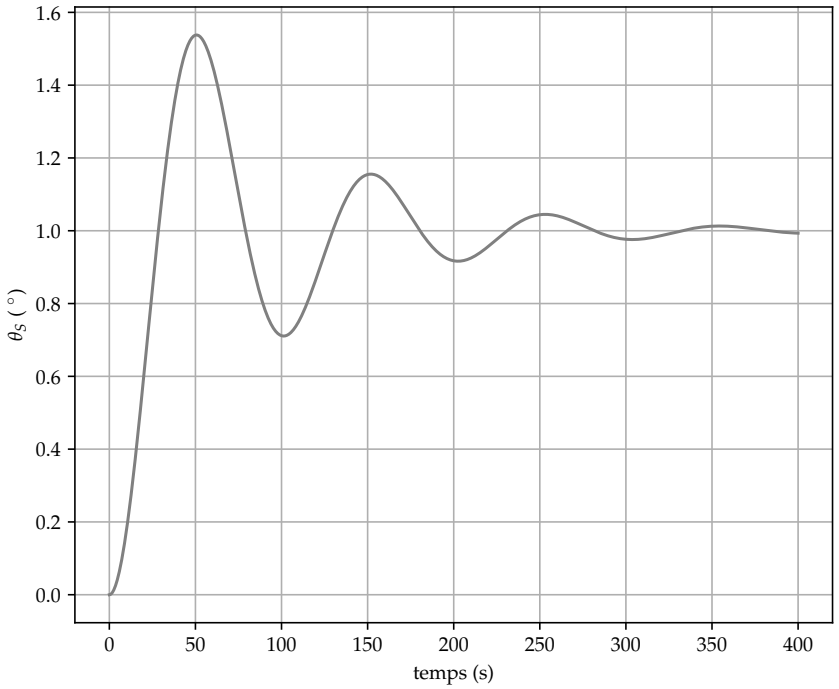


FIGURE 16 – Réponse indicielle du système non corrigé

Pour régler au mieux le système, nous allons étudier sa réponse indicielle. Pour étudier sa rapidité, nous allons calculer son temps de réponse à 5%. On suppose le système stable. On suppose que la simulation a été faite sur un temps suffisamment long pour que la sortie atteigne un régime établi quasi constant.

Question 44. Écrire une fonction *TR5(S, T)* prenant en argument les valeurs de la sortie S, et le vecteur temps correspondant T, et retournant une valeur approchée majorant le temps de réponse à 5%, c'est à dire le temps à partir duquel la sortie est comprise, et reste comprise entre + ou - 5% de sa valeur finale.

Solution:

```
def TR52(s, t):
    s_fin = s[-1]
    j = len(t)-1
    while (s[j] > s_fin*.95 and s[j] < s_fin*1.05):
        j = j-1
    return(t[j+1])
```

Sur le document technique DT10 est fournie la fonction *valeur_ depassement(S, T)* qui renvoie le 1er dépassement relatif, c'est-à-dire

$$D1 = \frac{s_{max} - s_\infty}{s_\infty}$$

La fonction *precision(S)* renvoie, quant à elle, l'erreur statique du système, c'est-à-dire la différence entre la valeur de consigne et s_∞. Pour améliorer la réponse en régime transitoire, on va aussi prendre en compte l'erreur dynamique, en calculant l'intégrale de la valeur absolue de l'erreur $A = \int_0^\infty |e(t) - s(t)|dt$. En pratique nous ferons le calcul sur un intervalle de temps fini.

Question 45. Écrire une fonction *Aire(S, T)*, qui renvoie l'intégrale de la valeur absolue de l'erreur en utilisant la méthode des trapèzes. S est le tableau des valeurs de la sortie correspondant au vecteur T représentant le temps.

Solution:

```
def critere_Aire_trap(s, t):
  return np.sum(np.abs((s[1:]+s[:-1]-2)/2 * np.diff(t)))
```

Afin de chercher une solution optimale, il faut combiner les différents critères de performance dans un indicateur global. On utilise pour cela la fonction *ponderation_ cout(S, T)* fournie dans le DT10 qui fait appel aux fonctions précédentes et renvoie un indicateur de performance globale (plus il est petit, meilleur c'est).

3.2.4 Détermination du meilleur correcteur possible

Approche brute-force

Dans cette partie nous cherchons un réglage optimal du correcteur en comparant 4 méthodes de réglage du correcteur PID : force brute, la méthode de Ziegler-Nichols temporelle, une approche par algorithme génétique, une approche par essaim particulaire.

On cherche donc le triplet K_p, T_i, T_d minimisant la fonction *ponderation_ cout*, en prenant $K_p \in [0.01, 100]$ car la tension d'alimentation du moteur ne peut pas être trop grande, et $T_i \in [0.01, 100]$ et $T_d \in [0, 50]$. Une première approche, par force brute, consiste à essayer le maximum de valeurs différentes.

Question 46. Évaluer la complexité de l'appel à la fonction *Indicateur*. Si cet appel prend 0.01 s, combien de temps faudrait-il pour tester tous les cas avec 2^{10} valeurs pour chaque coefficient ?

Solution: Y a plusieurs appels à des boucles simples, ça reste du O(n). 0.01*(2**10)**3 = 10737418s environ 3000 heures = 124 jours

Méthode de Ziegler-Nichols

La méthode de Ziegler-Nichols temporelle utilise la réponse indicielle du processus seul (sans correcteur). Il faut déterminer le point d'inflexion de la courbe (celui correspondant au temps le plus faible). On mesure ensuite la pente p en ce point, et le retard apparent L correspondant au point d'intersection de la tangente avec l'axe des abscisses. On prend ensuite comme coefficients $K_p = 1,2/(pL)$, $T_i = 2L$ et $T_d = 0,5L$.

Question 47. Écrire le code permettant de déterminer les coefficients du correcteur avec cette méthode.

Solution:

```
E, F = FTBF(numG, denG)
ftbfCalc = ct.tf(E, F)
t, sCalc = ct.step_response(ftbfCalc, T)
plt.figure()
plt.title("réponse indicielle du systéme non corrigé")
plt.plot(T, sCalc)

DsCalc = np.diff(sCalc)
#plt.figure()
#plt.plot(DsCalc)
Ds2Calc = np.diff(DsCalc)
# plt.figure()
# plt.title("dérivéee seconde")
# plt.plot(Ds2Calc)
for i in range(len(Ds2Calc)):
  if Ds2Calc[i] < 0:
    indice = i
    break
t_inf = T[indice]
#print(t_inf)

# On mesure ensuite sa pente p et le retard apparent L correspondant au
    point 'dintersection de la tangente avec 'labscisse
pe = DsCalc[indice]
L = t_inf - DsCalc[indice]/pe

# PID Kp 1.2 / pL Ti 2L Td 0.5L
Kp = 1.2 / (pe*L)
Ti = 2*L
Td = 0.5*L
print("réglage ZN tempo :", Kp, Ti, Td)   # 14, 146, 36
```

Le système non corrigé a un score de 400, qui tombe à 43 avec cette méthode. Nous allons voir par la suite s'il est encore possible d'optimiser ce score.

Optimisation par algorithme génétique

Cet algorithme repose sur un processus d'optimisation itératif évolutionniste, reproduisant les mécanismes de la sélection naturelle. Le principe consiste à initialiser un groupe de candidats possibles dont on va sélectionner les meilleurs éléments, qui seront conservés et croisés pour obtenir de nouveaux candidats. A chaque génération, la population se conserve et les caractéristiques du groupe s'améliorent jusqu'à converger vers un optimum.

- La population est l'ensemble des solutions envisageables. C'est l'ensemble des triplets (K_p, T_i, T_d). On en considère 100.
- L'individu représente une solution. C'est un triplet (K_p, T_i, T_d).
- Le chromosome est une composante de la solution. K_p, T_i, T_d sont 3 chromosomes.
- Le gène est une caractéristique, une particularité. Ce sera les bits dans le codage en binaire de la valeur numérique d'un chromosome.

Il y a trois opérateurs d'évolution dans les algorithmes génétiques :

- La sélection : choix des individus les mieux adaptés. On conservera les 20 meilleurs.
- Le croisement : mélange par la reproduction des particularités des individus choisis. On considère qu'un enfant hérite d'un chromosome d'un parent et de deux de l'autre, aléatoirement. On créera des reproductions entre les 20 meilleurs.
- La mutation : altération aléatoire des particularités d'un individu pour éviter les minima locaux. À chaque reproduction 1 bit sur un 1 chromosome est éventuellement modifié.

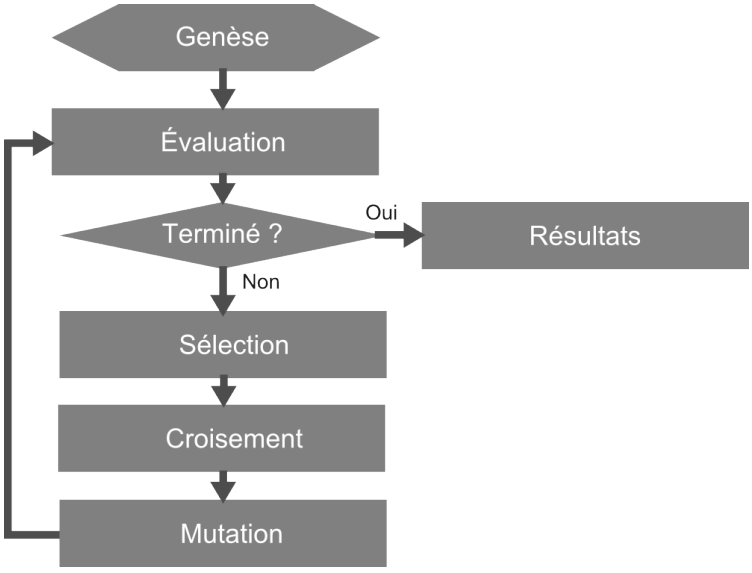


Figure 17 – Principe de l'algorithme génétique

Pour mettre en place la mutation, on a besoin de normaliser la façon dont on va représenter les valeurs numériques de K_p, T_i, T_d. On crée une bijection entre les valeurs et leur codage en binaire sur 10 bits, N_p, N_i, N_d. Pour cela nous utiliserons :

$$K_p = K_{pmin} + (K_{pmax} - K_{pmin})\frac{N_p}{2^{10}-1}$$

$$T_i = T_{imin} + (T_{imax} - T_{imin})\frac{N_i}{2^{10}-1}$$

$$T_d = T_{dmin} + (T_{dmax} - T_{dmin})\frac{N_d}{2^{10}-1}$$

Pour calculer le coût, il faut pour un jeu de valeurs N_p, N_i, N_d calculer les coefficients du PID, puis vérifier que le système est stable, et si c'est le cas calculer le coût, et sinon renvoyer 10000.

Question 48. Écrire une fonction *calcul_ cout(Np, Ni, Nd)* déterminant le coût d'une solution pour un jeu de valeurs de N_p, N_i, N_d.

Solution:

```
def calcul_cout(Np, Ni, Nd):
  Kp, Ti, Td = calcul_coef_correcteur(Np, Ni, Nd)
  numC, denC = correcteur(Kp, Ti, Td)
  num_BO, den_BO = multi_FT(numG, denG, numC, denC)
  numBF, den_BF = FTBF(num_BO, den_BO)
  stable = stabilite(den_BF)
  if stable:
    temps_BF, sol_BF = rep_indicielle(Kp, Ti, Td)
    T5 = TR5(sol_BF, temps_BF)
    D1 = depassement(sol_BF, temps_BF)
    P1 = precision(sol_BF, temps_BF)
    A1 = critere_Aire_trap(sol_BF, temps_BF)
    cout = ponderation_cout(T5, D1, P1, A1)
  else:
    cout = 10000
  return(cout)
```

Question 49. Écrire une fonction *generer_ chromosome(n)* créant une chaîne aléatoire de '0' ou '1' de longueur n.

Solution:

```
def generer_chromosome(n=10):
  b2 = ''
  for i in range(c):
    b2 = b2+str(int(random()*2))
  return(b2)
```

Question 50. Écrire une fonction *generer_population_initiale(p, n)* créant une population de p individus aléatoirement. On rappelle que chaque individu a 3 chromosomes, constitués de n gènes. Ainsi par exemple individu = ["1000101011", "0101010101", "0011001010"]. On utilise une convention gros-boutiste (poids le plus fort à gauche, *big endian*).

Solution:

```
def generer_population_initiale(p, n):
  population = []
  for i in range(p):
    individu = [generer_chromosome(n), generer_chromosome(n),
   generer_chromosome(n)]
    population.append(individu)
  return population
```

Question 51. Écrire une fonction *decodage(b)* prenant une chaîne de caractères représentant un nombre binaire et renvoyant le nombre décimal correspondant.

Solution:

```
def decodage(b):
  return int(b, 2)
  # b10 = 0       # autre solution
  # for k in range(len(b)):
  #     b10 += int(b[len(b)-1-k])*2**k
  # return b10
```

Question 52. Écrire une fonction *tri(L)* la plus rapide possible, triant les individus en fonction de leur performance (le coût le plus faible en premier). Donner le nom de votre tri ainsi que sa complexité dans le pire et dans le meilleur des cas.

Solution:

```
def perf(Li):
  Ip, Ii, Id = decodage(Li[0]), decodage(Li[1]), decodage(Li[2])
  return calcul_cout(Ip, Ii, Id)

def tri(L):
  print("tri", end=" ")
  for i in range(1, len(L)):
    print(i, end=" ")
    if perf(L[i]) < perf(L[i-1]):
      p = i
      while p > 0 and perf(L[i]) < perf(L[p-1]):
        p = p-1
      X = L.pop(i)
```

```
      L.insert(p, X)
  return L # pas vraiment utile...
#Tri par insertion; complexité en o(n) dans le meilleur des cas, et en o(n^2) dans le pire des cas

#Version bien plus rapide qui évite de faire plusieurs fois les mêmes calculs
def tri(L):
  LL = []
  for i, v in enumerate(L):
    v.append(perf(v))
    #print(v)
    LL.append(v)
  # print(LL)
  # print("tri", end=" ")
  for i in range(1, len(L)):
    #print(i, end=" ")
    if LL[i][3] < LL[i-1][3]:
      p = i
      while p > 0 and LL[i][3] < LL[p-1][3]:
        p = p-1
      X = LL.pop(i)
      LL.insert(p, X)
  return LL # pas vraiment utile...
```

Le résultat de ce tri sera utilisé pour sélectionner les 20 meilleurs individus que l'on conservera à chaque génération.

Question 53. Écrire une fonction *croisement(P1, P2)* qui permet de générer deux nouveaux individus à partir de deux parents P1 et P2. L'enfant hérite aléatoirement d'un chromosome d'un parent et de deux de l'autre.

Solution:

```
def croisement(P1, P2):
  LR = (randint(0, 1), randint(0, 1), randint(0, 1))
  E = []
  for i, v in enumerate(LR):
    if v:
      E.append(P1[i])
    else:
      E.append(P2[i])
  return E
```

De plus il subit une mutation d'un de ses gènes : un bit d'un de ses chromosomes change aléatoirement.

Question 54. Écrire une fonction *mutation(E)* changeant aléatoirement un bit d'un des chromosomes d'un individu E.

Solution:

```
def mutation(E):
    a = randint(0, 2)
    b = randint(0, 9)
    c = randint(0,1)
    chromosome = E[a]
    chromosome_mute = chromosome[:b] + str(c) + chromosome[b+1:]
    E[a] = chromosome_mute
    return(E)
```

Pour établir une nouvelle génération :

- sélection : on garde les 20 meilleurs candidats,
- croisement : on fait se reproduire le 1er avec les 19 suivants, puis le deuxième avec les 18 suivants, puis le 3ème avec les 17 suivants, le 4ème avec les 16 suivants, et on complète avec un tirage aléatoire de 10 individus pour arriver à 100.

Question 55. Écrire une fonction *nouvGeneration(L)* traduisant l'algorithme ci-dessus.

Solution:

```
def nouv_generation(L):
    L_new = L.copy()  # Attention PB Enoncé sinon population_triée modifié á la fin de la fonction
    c = 0
    for i in range(4):
        for j in range(i+1, 20):
            L_new[c+20] = mutation(croisement(L[i], L[j]))
            c += 1
    return L_new
```

Question 56. Écrire la fonction *doublons(L)* qui prend en argument une population (sous forme de liste) et renvoie cette population débarrassée de ses clones, remplacés par des individus tirés aléatoirement.

Solution:

```
def doublons(L):
    for k in range(len(L)-1):
        for j in range(k+1, len(L)):
            if L[j] == L[k]:
                L[j] = [generer_chromosome(10), generer_chromosome(10), generer_chromosome(10)]
    return L

```

Question 57. Écrire le code permettant de trouver les valeurs optimales du correcteur après 30 générations.

Solution:

```
print("=== Par algo génétique ===")
cycles = 30
population = generer_population_initiale(100, 10)
enregistrement = []
gest = []
for i in range(cycles):
  print("Algo génétique, début cycle", i)
  population_triée = tri(population)
  print("    top réglage", calcul_cout(decodage(population[0][0]),
   decodage(population[0][1]), decodage(population[0][2])))
  population = nouv_generation(population_triée)
  population = doublons(population)
solution = population[0]
KpGéné, TiGéné, TdGéné = calcul_coef_correcteur(
  decodage(solution[0]), decodage(solution[1]), decodage(solution[2]))
print("réglage optimal algo géné :", KpGéné, TiGéné, TdGéné)
print("perf algo géné :", perf(solution))
```

Optimisation par essaim de particules

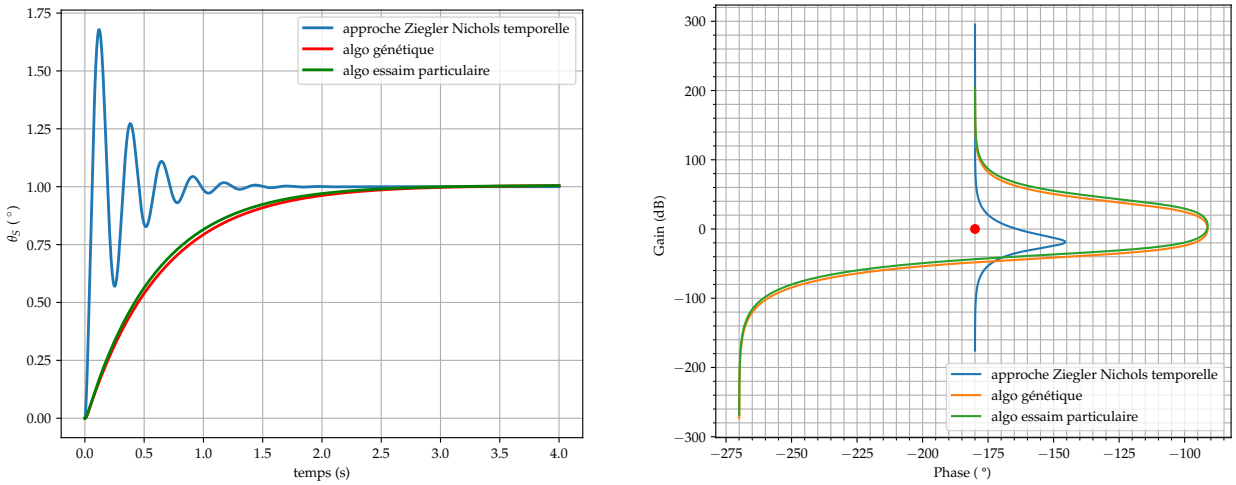
L'optimisation par essaim de particules (*Particle Swarm Optimization*, abrégée ici PSO) est une méthode d'optimisation stochastique, pour les fonctions non-linéaires, basée sur la reproduction d'un comportement social. L'origine de cette méthode vient des observations faites lors des simulations informatiques de vols groupés d'oiseaux et de bancs de poissons. Ces simulations ont mis en valeur la capacité des individus d'un groupe en mouvement à conserver une distance optimale entre eux et à suivre un mouvement global par rapport aux mouvements locaux de leur voisinage.

D'autre part, ces simulations ont également révélé l'importance du mimétisme dans la compétition qui oppose les individus à la recherche de la nourriture. En effet, les individus sont à la recherche de sources de nourriture qui sont dispersées de façon aléatoire dans un espace de recherche, et dès lors qu'un individu localise une source de nourriture, les autres individus vont chercher à reproduire son comportement. Ce comportement social basé sur l'analyse de l'environnement et du voisinage constitue alors une méthode de recherche d'optimum par l'observation des tendances des individus voisins. Chaque individu cherche à optimiser ses chances en suivant une tendance qu'il modère par son propre vécu.

Question 58. Le document technique DT11 donne un exemple d'utilisation de la fonction *pso* du module *pyswarm*. Écrire le code permettant d'utiliser l'optimisation par essaim particulaire pour trouver les valeurs optimales de K_d, T_i et T_p.

Solution:

```python
print("--- Par essaim particulaire ---")
lb = [0, 0, 0]      # borne inférieur des variables
ub = [2**10-1, 2**10-1, 2**10-1]      # borne supérieure des variables

def calcul_cout2(I):
  Kp, Ti, Td = calcul_coef_correcteur(I[0], I[1], I[2])
  numC, denC = correcteur(Kp, Ti, Td)
  num_BO, den_BO = multi_FT(numG, denG, numC, denC)
  numBF, den_BF = FTBF(num_BO, den_BO)
  stable = stabilite(den_BF)
  if stable:
    temps_BF, sol_BF = rep_indicielle(Kp, Ti, Td)
    T5 = TR52(sol_BF, temps_BF)
    D1 = depassement(sol_BF, temps_BF)
    P1 = precision(sol_BF, temps_BF)
    A1 = critere_Aire_trap(sol_BF, temps_BF)
    cout = ponderation_cout(T5, D1, P1, A1)
  else:
    cout = 10000
  return(cout)

xopt, fopt = pso(calcul_cout2, lb, ub)
KpEssaim, TiEssaim, TdEssaim = calcul_coef_correcteur(*xopt)
print("réglage optimal essaim particulaire :", KpEssaim, TiEssaim,
    TdEssaim)
print("perf essaim particulaire :", calcul_cout(*xopt), calcul_cout2(
    xopt))
```



(a) (b)

FIGURE 18 – (a) Réponse indicielle du système pour différents réglages du correcteur (b) Diagramme de Black pour différents réglages du correcteur

La méthode de l'algorithme génétique avec 30 générations prend approximativement autant de temps que celle utilisant des essaims particulaires.

Question 59. Conclure quant à l'adéquation et l'efficacité de ces méthodes dans le cadre

du réglage du correcteur adapté à l'exosquelette, et proposer des pistes de solutions pour améliorer les performances de l'asservissement.

Solution: ZN a trop de dépassement pour être utilisable dans notre cas. L'essaim particulaire et l'algo génétique semble proches en solution, et correspondent au CdCf. Il faudrait regarder la vitesse de convergence de l'algo génétique pour l'affiner et l'optimiser.

Avertissement

Ce sujet est basé sur des discussions avec des membres de l'équipe de CLINATEC, que les auteurs remercient pour leur disponibilité, ainsi que sur des informations disponibles dans la littérature scientifique et publiées entre autre par CLINATEC. Néanmoins il n'engage pas et ne représente nullement le travail de CLINATEC. Etant donné le caractère confidentiel d'un sujet de concours, il n'a pas été lu par CLINATEC avant parution, et est basé sur des extrapolations libre effectuées par les auteurs à partir de leur compréhension du projet, notamment pour les parties 1 et 3.

Bibliographie

Liste non-exhaustive des ouvrages et articles de recherche ayant permis de concevoir ce sujet :

Références

[1] M Dombre and W Khalil. Modelisation et commande des robots, hermes, 1988.

[2] Andrey Eliseyev and Tatiana Aksenova. Stable and artifact-resistant decoding of 3d hand trajectories from ecog signals using the generalized additive model. *Journal of neural engineering*, 11(6) :066005, 2014.

[3] Andrey Eliseyev and Tetiana Aksenova. Penalized multi-way partial least squares for smooth trajectory decoding from electrocorticographic (ecog) recording. *PloS one*, 11(5) :e0154878, 2016.

[4] Corinne S Mestais, Guillaume Charvet, Fabien Sauter-Starace, Michael Foerster, David Ratel, and Alim Louis Benabid. Wimagine : wireless 64-channel ecog recording implant for long term clinical applications. *IEEE transactions on neural systems and rehabilitation engineering*, 23(1) :10–21, 2014.

[5] Alexandre Moly. *Innovative decoding algorithms for Chronic ECoG-based Brain Computer Interface (BCI) for motor disabled subjects in laboratory and at home*. PhD thesis, Université Grenoble Alpes [2020-....], 2020.

[6] Kentaro Shimoda, Yasuo Nagasaka, Zenas C Chao, and Naotaka Fujii. Decoding continuous three-dimensional hand trajectories from epidural electrocorticographic signals in japanese macaques. *Journal of neural engineering*, 9(3) :036015, 2012.

[7] Andriy Yelisyeyev. *Interface cerveau-machine à partir d'enregistrement électrique cortical*. PhD thesis, Grenoble, 2011.

Documents Techniques

Document technique DT1 : Extraits de la documentation technique du ZL70102

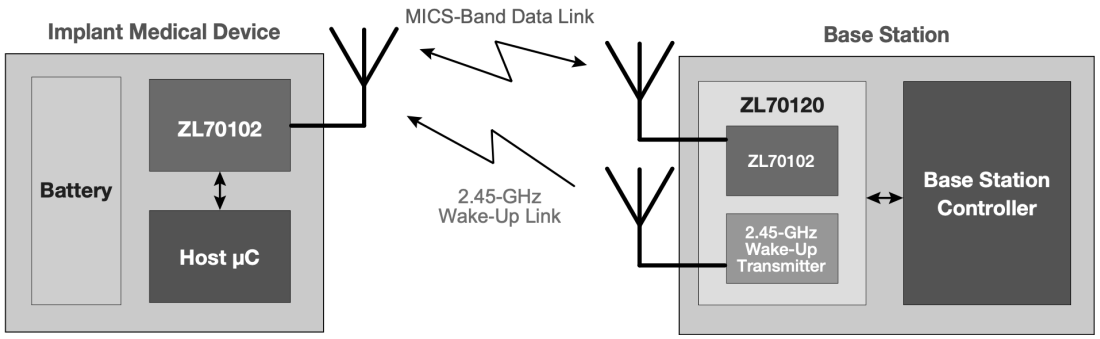


Figure 1-1 • **Application Example**

400-MHz Packet Definition

The packet definition is chosen to enable a high effective data rate. The packet header should be kept as small as possible and the payload should be as large as possible. The same packet definition is used in both the uplink and downlink. The basis for the packet definition and the link protocol is fully described in the ZL70102 Design Manual.

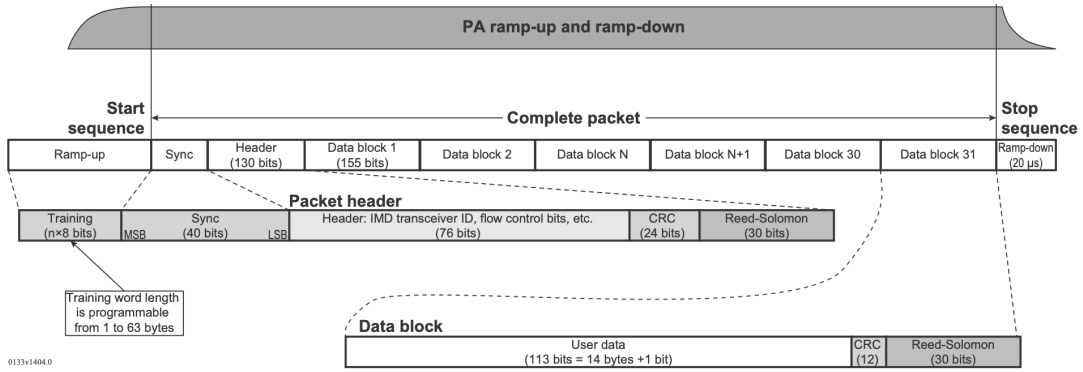


Figure 3-9 • **Packet Definition (first in time on the left side)**

Table 3-3 • **Options for Modulation Modes, Data Rates, and Receiver Sensitivity**

Modulation Mode	Maximum Raw Radio Data Rate (kbit/s)	Maximum Effective Data Rate (kbit/s)	Typical Receiver Sensitivity (Note 1)
2FSK-fallback	200	134	−98dBm
2FSK	400	265	−91dBm
4FSK	800	515 (Note 2)	−79dBm

Notes:

1. *The sensitivity is based on the application circuit in Figure 10-1 on page 10-1, at the reference point of the dual-band antenna (50ohm). This value represents a packet error rate of 10%.*
2. *Requires calibration of the RX ADC. Refer to the ZL70102 Design Manual for the calibration procedure.*

Python For Data Science *Cheat Sheet*

NumPy Basics

Learn Python for Data Science Interactively at www.DataCamp.com

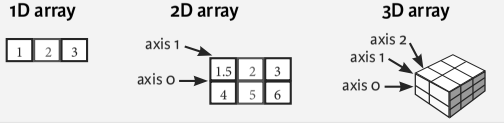
NumPy

The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays



Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                 dtype = float)
```

Initial Placeholders

```
>>> np.zeros((3,4))                    Create an array of zeros
>>> np.ones((2,3,4),dtype=np.int16)    Create an array of ones
>>> d = np.arange(10,25,5)             Create an array of evenly
                                       spaced values (step value)
>>> np.linspace(0,2,9)                 Create an array of evenly
                                       spaced values (number of samples)
>>> e = np.full((2,2),7)               Create a constant array
>>> f = np.eye(2)                      Create a 2X2 identity matrix
>>> np.random.random((2,2))            Create an array with random values
>>> np.empty((3,2))                    Create an empty array
```

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Data Types

```
>>> np.int64          Signed 64-bit integer types
>>> np.float32        Standard double-precision floating point
>>> np.complex        Complex numbers represented by 128 floats
>>> np.bool           Boolean type storing TRUE and FALSE values
>>> np.object         Python object type
>>> np.string_        Fixed-length string type
>>> np.unicode_       Fixed-length unicode type
```

Inspecting Your Array

```
>>> a.shape           Array dimensions
>>> len(a)            Length of array
>>> b.ndim            Number of array dimensions
>>> e.size            Number of array elements
>>> b.dtype           Data type of array elements
>>> b.dtype.name      Name of data type
>>> b.astype(int)     Convert an array to a different type
```

Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

Array Mathematics

Arithmetic Operations

```
>>> g = a - b                          Subtraction
  array([[-0.5,  0. ,  0. ],
         [-3. , -3. , -3. ]])
>>> np.subtract(a,b)                   Subtraction
>>> b + a                              Addition
  array([[ 2.5,  4. ,  6. ],
         [ 5. ,  7. ,  9. ]])
>>> np.add(b,a)                        Addition
>>> a / b                              Division
  array([[ 0.66666667,  1.        ,  1.        ],
         [ 0.25      ,  0.4       ,  0.5       ]])
>>> np.divide(a,b)                     Division
>>> a * b                              Multiplication
  array([[  1.5,   4. ,   9. ],
         [  4. ,  10. ,  18. ]])
>>> np.multiply(a,b)                   Multiplication
>>> np.exp(b)                          Exponentiation
>>> np.sqrt(b)                         Square root
>>> np.sin(a)                          Print sines of an array
>>> np.cos(b)                          Element-wise cosine
>>> np.log(a)                          Element-wise natural logarithm
>>> e.dot(f)                           Dot product
  array([[ 7.,  7.],
         [ 7.,  7.]])
```

Comparison

```
>>> a == b                             Element-wise comparison
  array([[False,  True,  True],
         [False, False, False]], dtype=bool)
>>> a < 2                              Element-wise comparison
  array([True, False, False], dtype=bool)
>>> np.array_equal(a, b)               Array-wise comparison
```

Aggregate Functions

```
>>> a.sum()                            Array-wise sum
>>> a.min()                            Array-wise minimum value
>>> b.max(axis=0)                      Maximum value of an array row
>>> b.cumsum(axis=1)                   Cumulative sum of the elements
>>> a.mean()                           Mean
>>> b.median()                         Median
>>> a.corrcoef()                       Correlation coefficient
>>> np.std(b)                          Standard deviation
```

Copying Arrays

```
>>> h = a.view()          Create a view of the array with the same data
>>> np.copy(a)            Create a copy of the array
>>> h = a.copy()          Create a deep copy of the array
```

Sorting Arrays

```
>>> a.sort()              Sort an array
>>> c.sort(axis=0)        Sort the elements of an array's axis
```

Subsetting, Slicing, Indexing

Also see Lists

Subsetting

```
>>> a[2]                               Select the element at the 2nd index
  3
>>> b[1,2]                             Select the element at row 1 column 2
  6.0                                  (equivalent to b[1][2])
```

Slicing

```
>>> a[0:2]                             Select items at index 0 and 1
  array([1, 2])
>>> b[0:2,1]                           Select items at rows 0 and 1 in column 1
  array([ 2.,  5.])
>>> b[:1]                              Select all items at row 0
  array([[1.5, 2., 3.]])               (equivalent to b[0:1, :])
>>> c[1,...]                           Same as [1,:,:]
  array([[[ 3.,  2.,  1.],
          [ 4.,  5.,  6.]]])
>>> a[ : :-1]                          Reversed array a
  array([3, 2, 1])
```

Boolean Indexing

```
>>> a[a<2]                             Select elements from a less than 2
  array([1])
```

Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]      Select elements (1,0),(0,1),(1,2) and (0,0)
  array([ 4. , 2. , 6. , 1.5])
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]      Select a subset of the matrix's rows
  array([[ 4. ,5. , 6. , 4. ],         and columns
         [ 1.5, 2. , 3. , 1.5],
         [ 4. , 5. , 6. , 4. ],
         [ 1.5, 2. , 3. , 1.5]])
```

Array Manipulation

Transposing Array

```
>>> i = np.transpose(b)                Permute array dimensions
>>> i.T                                Permute array dimensions
```

Changing Array Shape

```
>>> b.ravel()                          Flatten the array
>>> g.reshape(3,-2)                    Reshape, but don't change data
```

Adding/Removing Elements

```
>>> h.resize((2,6))                    Return a new array with shape (2,6)
>>> np.append(h,g)                     Append items to an array
>>> np.insert(a, 1, 5)                 Insert items in an array
>>> np.delete(a,[1])                   Delete items from an array
```

Combining Arrays

```
>>> np.concatenate((a,d),axis=0)       Concatenate arrays
  array([ 1,  2,  3, 10, 15, 20])
>>> np.vstack((a,b))                   Stack arrays vertically (row-wise)
  array([[ 1. ,  2. ,  3. ],
         [ 1.5,  2. ,  3. ],
         [ 4. ,  5. ,  6. ]])
>>> np.r_[e,f]                         Stack arrays vertically (row-wise)
>>> np.hstack((e,f))                   Stack arrays horizontally (column-wise)
  array([[ 7.,  7.,  1.,  0.],
         [ 7.,  7.,  0.,  1.]])
>>> np.column_stack((a,d))             Create stacked column-wise arrays
  array([[ 1, 10],
         [ 2, 15],
         [ 3, 20]])
>>> np.c_[a,d]                         Create stacked column-wise arrays
```

Splitting Arrays

```
>>> np.hsplit(a,3)                     Split the array horizontally at the 3rd
  [array([1]),array([2]),array([3])]   index
>>> np.vsplit(c,2)                     Split the array vertically at the 2nd index
[array([[[ 1.5,  2. ,  1. ],
         [ 4. ,  5. ,  6. ]]]),
 array([[[ 3.,  2.,  3.],
         [ 4.,  5.,  6.]]])]
```

3.2.5 Trigonometric functions

sin(x, /[, out, where, casting, order, ...])	Trigonometric sine, element-wise.
cos(x, /[, out, where, casting, order, ...])	Cosine element-wise.
tan(x, /[, out, where, casting, order, ...])	Compute tangent element-wise.
arcsin(x, /[, out, where, casting, order, ...])	Inverse sine element-wise.
arccos(x, /[, out, where, casting, order, ...])	Inverse cosine, element-wise.
arctan(x, /[, out, where, casting, order, ...])	Trigonometric inverse tangent, element-wise.
hypot($x1, x2$, /[, out, where, casting, ...])	Given the "legs" of a right triangle, return its hypotenuse.
arctan2($x1, x2$, /[, out, where, casting, ...])	Element-wise arc tangent of x1/x2 choosing the quadrant correctly.
degrees(x, /[, out, where, casting, order, ...])	Convert angles from radians to degrees.
radians(x, /[, out, where, casting, order, ...])	Convert angles from degrees to radians.
unwrap(p[, discont, axis, period])	Unwrap by taking the complement of large deltas with respect to the period.
deg2rad(x, /[, out, where, casting, order, ...])	Convert angles from degrees to radians.
rad2deg(x, /[, out, where, casting, order, ...])	Convert angles from radians to degrees.

Hyperbolic functions

sinh(x, /[, out, where, casting, order, ...])	Hyperbolic sine, element-wise.
cosh(x, /[, out, where, casting, order, ...])	Hyperbolic cosine, element-wise.
tanh(x, /[, out, where, casting, order, ...])	Compute hyperbolic tangent element-wise.
arcsinh(x, /[, out, where, casting, order, ...])	Inverse hyperbolic sine element-wise.
arccosh(x, /[, out, where, casting, order, ...])	Inverse hyperbolic cosine, element-wise.
arctanh(x, /[, out, where, casting, order, ...])	Inverse hyperbolic tangent element-wise.

Rounding

around(a[, decimals, out])	Evenly round to the given number of decimals.
round_(a[, decimals, out])	Round an array to the given number of decimals.
rint(x, /[, out, where, casting, order, ...])	Round elements of the array to the nearest integer.
fix(x[, out])	Round to nearest integer towards zero.
floor (x, /[, out, where, casting, order, ...])	Return the floor of the input, element-wise.
ceil(x, /[, out, where, casting, order, ...])	Return the ceiling of the input, element-wise.
trunc(x, /[, out, where, casting, order, ...])	Return the truncated value of the input, element-wise.

Sums, products, differences

prod(a[, axis, dtype, out, keepdims, ...])	Return the product of array elements over a given axis.
sum(a[, axis, dtype, out, keepdims,..])	Sum of array elements over a given axis.
nanprod(a[, axis, dtype, out, keepdims])	Return the product of array elements over a given axis treating Not a Numbers (NaNs) as ones.
nansum(a[, axis, dtype, out, keepdims])	Return the sum of array elements over a given axis treating Not a Numbers (NaNs) as zero.
cumprod(a[, axis, dtype, out])	Return the cumulative product of elements along a given axis.
cumsum(a[, axis, dtype, out])	Return the cumulative sum of the elements along a given axis.
nancumprod(a[, axis, dtype, out])	Return the cumulative product of array elements over a given axis treating Not a Numbers (NaNs) as one.
nancumsum(a[, axis, dtype, out])	Return the cumulative sum of array elements over a given axis treating Not a Numbers (NaNs) as zero.
diff(a[, n, axis, prepend, append])	Calculate the n-th discrete difference along the given axis.
ediff1d(ary[, to_end, to_begin])	The differences between consecutive elements of an array.
gradient(f, *varargs[, axis, edge_order])	Return the gradient of an N-dimensional array.
cross(a, b[, axisa, axisb, axisc, axis])	Return the cross product of two (arrays of) vectors.
trapz(y[, x, dx, axis])	Integrate along the given axis using the composite trapezoidal rule.

Exponents and logarithms

exp(x, /[, out, where, casting, order, ...])	Calculate the exponential of all elements in the input array.
expm 1(x, /[, out, where, casting, order, ..])	Calculate exp (x) − 1 for all elements in the array.
exp 2(x, /[, out, where, casting, order, ...])	Calculate $2 * *p$ for all p in the input array.
log(x, /[, out, where, casting, order, ...])	Natural logarithm, element-wise.
log 10(x, /[, out, where, casting, order, ...])	Return the base 10 logarithm of the input array, elementwise.
log 2(x, /[, out, where, casting, order, ...])	Base-2 logarithm of x.
log 1 p(x, /[, out, where, casting, order,..])	Return the natural logarithm of one plus the input array, element-wise.
logaddexp(x1, x2, /[, out, where, casting,,...])	Logarithm of the sum of exponentiations of the inputs.
logaddexp2(x1, x2, /[, out, where, casting,,..])	Logarithm of the sum of exponentiations of the inputs in base- 2.

Polynomials

numpy.roots()

return the roots of a polynomial with coefficients given in p. The values in the rank-1 array p are coefficients of a polynomial. If the length of p is n+1 then the polynomial is described by : $p[0] * x * *n + p[1] * x * *(n-1) + \ldots + p[n-1] * x + p[n]$

Syntax : numpy.roots(p)

Parameters : p : [array_like] Rank-1 array of polynomial coefficients.

Return : [ndarray] An array containing the roots of the polynomial.

Document technique DT3 : Extrait de la documentation *numpy.bincount*

numpy.bincount (x, /, weights=None, minlength=0)

Count number of occurrences of each in array of non-negative ints.

The number of bins (of size 1) is one larger than the largest value in x. If *minlength* is specified, there will be at least this number of bins in the output array (though it will be longer if necessary, depending on the contents of x). Each bin gives the number of occurrences of its index value in x. If *weights* is specified the input array is weighted by it, i.e. if a value n is found at position i, out[n] += weight[i] instead of out[n] += 1.

Parameters :	x : array_like, 1 dimension, nonnegative ints Input array.
	weights : array_like, optional Weights, array of the same shape as x.
	minlength : int, optional A minimum number of bins for the output array.
Returns :	out : ndarray of ints The result of binning the input array. The length of out is equal to np.amax(x) + 1.
Raises :	ValueError If the input is not 1-dimensional, or contains elements with negative values, or if *minlength* is negative.
	TypeError If the type of the input is float or complex.

Examples

```
>>> np.bincount(np.arange(5))
array([1, 1, 1, 1, 1])
>>> np.bincount(np.array([0, 1, 1, 3, 2, 1, 7]))
array([1, 3, 1, 1, 0, 0, 0, 1])

>>> x = np.array([0, 1, 1, 3, 2, 1, 7, 23])
>>> np.bincount(x).size == np.amax(x)+1
True
```

Document technique DT4 : Extrait de la documentation *numpy.convolve*

numpy.convolve(a, v, mode='full')

Returns the discrete, linear convolution of two one-dimensional sequences.

The convolution operator is often seen in signal processing, where it models the effect of a linear timeinvariant system on a signal [1]. In probability theory, the sum of two independent random variables is distributed according to the convolution of their individual distributions.

If v is longer than a, the arrays are swapped before computation.

Parameters :	a : (N,) array_like First one-dimensional input array.
	v : (M,)array_like Second one-dimensional input array.
	mode : {'full', 'valid', 'same'}, optional 'full' : By default, mode is 'full'. This returns the convolution at each point of overlap, with an output shape of (N + M − 1,). At the end-points of the convolution, the signals do not overlap completely, and boundary effects may be seen. 'same' : Mode 'same' returns output of length max(M, N). Boundary effects are still visible. 'valid' : Mode 'valid' returns output of length max(M, N) − min(M, N) + 1. The convolution product is only given for points where the signals overlap completely. Values outside the signal boundary have no effect.
Returns :	out : ndarray Discrete, linear convolution of a and v.

Notes

The discrete convolution operation is defined as

$$(a * v)_n = \sum_{m=-\infty}^{\infty} a_m v_{n-m}$$

Note how the convolution operator flips the second array before "sliding" the two across one another :

```
>>>np.convolve([1, 2, 3], [0, 1, 0.5])
array([0., 1., 2.5, 4., 1.5])
```

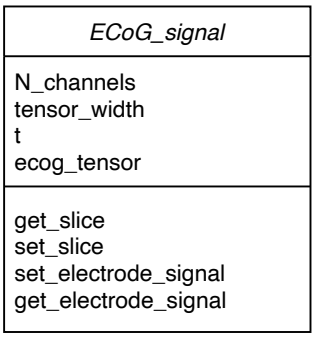
Only return the middle values of the convolution. Contains boundary effects, where zeros are taken into account :

```
np.convolve([1, 2, 3], [0, 1, 0.5], 'same')
array([1., 2.5, 4.])
```

The two arrays are of the same length, so there is only one position where they completely overlap :

```
np.convolve([1, 2, 3], [0, 1, 0.5], 'valid')
array([2.5])
```

Document technique DT5 : Classe ECoG de stockage d'un signal sous forme de tenseur




```python
import numpy as np
import scipy.io
from scipy import signal
import copy

#################################
## miscalleneous functions ##
#################################
def open_ECoG_recording(foldername, N_channels=64, T_sample=1e-3, unit=1e-6):
    ''' opens files from Shimoda 2012 ECoG Dataset '''
    ECoG_channels = []
    for k in range(1, N_channels+1):
        current_ECoG_sig = scipy.io.loadmat(foldername+'/ECoG_ch'+str(k)+'.mat')
        ECoG_channels.append(current_ECoG_sig['ECoGData_ch'+str(k)][0])#*unit)
    N_samples = len(ECoG_channels[0])
    t = np.linspace(0, (N_samples-1)*T_sample, num=N_samples)
    return t, ECoG_channels

#############################################
## class for ECoG recording as a tensor ##
#############################################
class ECoG_signal:
    '''
    Basic ECoG class containing tensor definition and operations on tensors
    '''
    def __init__(self, N_channels=64):
        '''
        Init the ECoG signal

        Parameters:
        ----------
        N_channels : int
            number of electrode on the recording matrix,
            should be a power of 2 (square matrix)
            by default set to 64
        '''
        if (N_channels & (N_channels-1) == 0) and N_channels != 0:
            self.N_channels = int(N_channels)
            self.tensor_width = int(np.sqrt(N_channels))
            self.T_sample = None
        else:
```

```
42          raise ValueError('Work with a power of 2 number of channels only')
43
44      def create_tensor(self, length_t_vector, T_sample=1e-3):
45          '''
46          create the 3D tensor (N_channels*Nchannels*length_t_vector)
47
48          Parameters:
49          ----------
50          length_t_vector: int
51              length of the time vector
52          T_sample: float
53              blablabla
54          '''
55          self.ecog_tensor = np.zeros((self.tensor_width, self.tensor_width,
            length_t_vector))
56          self.t = np.linspace(0,(length_t_vector-1)*T_sample, num= length_t_vector
            )
57          self.T_sample = T_sample
58
59
60      def set_slice(self, m, t):
61          '''
62          Set a slice of ECoG (all channels value) as a matrix for a given time
            index
63
64          Parameters:
65          ----------
66          m: array or np.array
67              ECoG value at a given time index
68          t: int
69              time index
70          '''
71          if m.shape[0] == m.shape[1] and m.shape[0] == self.tensor_width:
72              self.ecog_tensor[:,:,t] = m
73          else:
74              raise IndexError('specified slice cannot fit the tensor')
75
76      def get_slice(self, t):
77          '''
78          Get a slice of ECoG (all channels value) as a matrix for a given time
            index
79
80          Parameters:
81          ----------
82          t: int
83              time index
84
85          Returns:
86          -------
87          np.array: all ECoG channels value at the given time index
88          '''
89          return self.ecog_tensor[:,:,t]
90
91      def set_electrode_signal(self, N_elec, signal):
92          '''
93          Set the signal of one electrode along the time dimension
```

```
94
95      Parameters:
96      ----------
97      N_elec: int
98        number of the channel to set
99      signal: array or np.array
100       signal of the electrode
101     '''
102     if N_elec<self.N_channels: # test if the electrode number is in the
        electrde matrix
103       # get electrode position in the tensor
104       row = N_elec // self.tensor_width
105       column = N_elec % self.tensor_width
106       if  len(signal) == self.ecog_tensor.shape[2]:
107         self.ecog_tensor[row, column, :] = signal
108       else:
109         raise IndexError('Given signal is not of the time length of the
        tensor')
110     else:
111       raise ValueError('Specified electrode is out of electrode set')
112
113   def get_electrode_signal(self, N_elec):
114     '''
115     Get the signal of one electrode along the time dimension
116
117     Parameters:
118     ----------
119     N_elec: int
120       number of the chanel to get
121
122     Returns:
123     --------
124     np.array: signal of the specified channel number
125     '''
126     if N_elec<self.N_channels: # test if the electrode number is in the
        electrde matrix
127       # get electrode position in the tensor
128       row = N_elec // self.tensor_width
129       column = N_elec % self.tensor_width
130       return self.ecog_tensor[row, column, :]
131     else:
132       raise ValueError('Specified electrode is out of electrode set')
```

Document technique DT6 : classe PLS de description des *Partial Least Squares* utilisant l'algorithm NIPALS

Le code suivant comporte des trous faisant référence au Document Réponse DR5 :

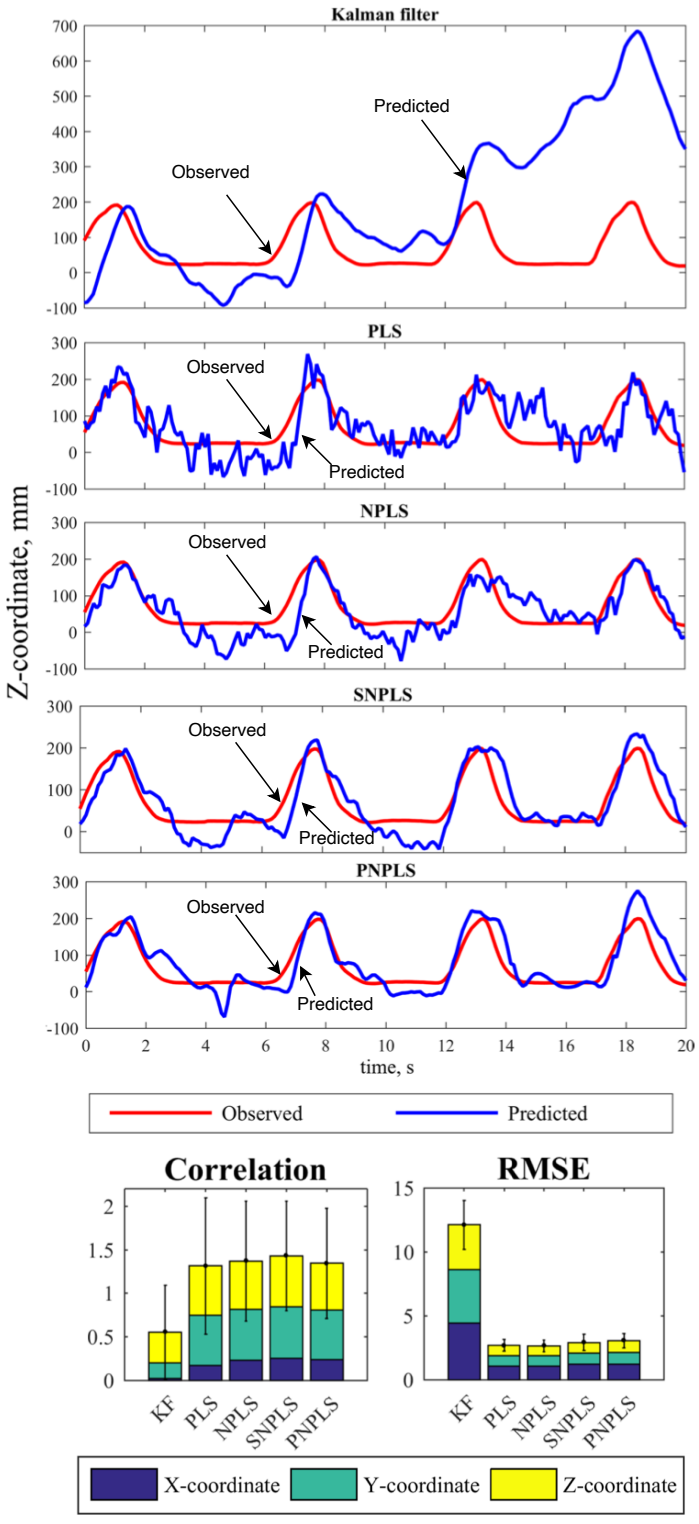
```
class PLS(object):
    """A class for PLS calculated by the NIPALS algorithm.
    Initialize with a Pandas DataFrame or an object that can be turned into a
    DataFrame
    (e.g. an array or a dict of lists)"""

    def __init__(self, x_df, y_df):
        super(PLS, self).__init__()
        if type(x_df) != pd.core.frame.DataFrame:
            x_df = pd.DataFrame(x_df)
        if type(y_df) != pd.core.frame.DataFrame:
            y_df = pd.DataFrame(y_df)
        # Make sure data is numeric
        self.x_df = x_df.astype("float")
        self.y_df = y_df.astype("float")
        # Check for and remove infs
        if np.isinf(self.x_df).any().any():
            logging.warning(
                "X data contained infinite values, converting to missing
    values"
            )
            self.x_df.replace([np.inf, -np.inf], np.nan, inplace=True)
        if np.isinf(self.y_df).any().any():
            logging.warning(
                "Y data contained infinite values, converting to missing
    values"
            )
            self.y_df.replace([np.inf, -np.inf], np.nan, inplace=True)

    def fit(
        self,
        ncomp,
        startcol
        center=True,
        scale=True,
        tol=0.000001,
        maxiter=500,
        dropzerovar=False,
    ):
        """The Fit method, will fit a PLS to the X and Y data"""

        # Convert to np array
        self.x_mat = self.x_df.values
        self.y_mat = self.y_df.values
        self.center = center
        self.scale = scale
        self.x_mean = np.nanmean(self.x_mat, axis=0)
        self.y_mean = np.nanmean(self.y_mat, axis=0)
        self.x_std = np.nanstd(self.x_mat, axis=0, ddof=1)
        self.y_std = np.nanstd(self.y_mat, axis=0, ddof=1)

        if center:
```

```
            self.x_mat = self.x_mat - self.x_mean
            self.y_mat = self.y_mat - self.y_mean
        if scale:
            self.x_mat = self.x_mat / self.x_std
            self.y_mat = self.y_mat / self.y_std

        # initialize outputs
        loadings = np.empty((x_nc, ncomp))
        scores = np.empty((nr, ncomp))
        u = np.empty((nr, ncomp))
        weights = np.empty((x_nc, ncomp))
        q = np.empty((y_nc, ncomp))
        b = np.empty((ncomp,))

        for comp in range(ncomp):
            train_x_mat = self.x_mat
            train_y_mat = self.y_mat
            # Set u to some column of Y
            uh = train_y_mat[:, startcol]
            th = uh

            it = 0
            while True:
                ## Voir Document Reponse DR5 ##

                # Check convergence
                if np.nansum((th - th_old) ** 2) < tol:
                    break
                it += 1
                if it >= maxiter:
                    raise RuntimeError(
                        "Convergence was not reached in {} iterations for component {}".format(
                            maxiter, comp
                        )
                    )

            # Calculate X loadings and rescale the scores and weights
            ph = train_x_mat.T.dot(th) / sum(th * th)

            loadings[:, comp] = ph
            scores[:, comp] = th
            u[:, comp] = uh
            q[:, comp] = qh
            weights[:, comp] = wh
            bh = ## voir Document Reponse DR5 ##
            b[comp] = bh

            self.x_mat = self.x_mat - np.outer(th, ph)
            self.y_mat = self.y_mat - bh * np.outer(th, qh)

        # Convert results to DataFrames
        self.scores = pd.DataFrame(
            scores, index=self.x_df.index, columns=["PC{}".format(i + 1) for i in range(ncomp)],
        )
```

```
104         self.loadings = pd.DataFrame(
105             loadings, index=self.x_df.columns, columns=["PC{}".format(i + 1)
    for i in range(ncomp)],
106         )
107         self.u = pd.DataFrame(
108             u, index=self.x_df.index, columns=["PC{}".format(i + 1) for i in
    range(ncomp)],
109         )
110         self.q = pd.DataFrame(
111             q, index=self.y_df.columns, columns=["PC{}".format(i + 1) for i
    in range(ncomp)],
112         )
113         self.weights = pd.DataFrame(
114             weights, index=self.x_df.columns, columns=["PC{}".format(i + 1)
    for i in range(ncomp)],
115         )
116         self.b = pd.Series(b, index=["PC{}".format(i + 1) for i in range(
    ncomp)])
117         return True
```

Document technique DT7 : Résultats de test de différentes méthodes et modèles de décodage des intentions de mouvement

Les figures suivantes sont extraites de [3].



La racine de l'erreur quadratique moyenne est définie par $RMSE = ||\mathbf{y} - \hat{\mathbf{y}}||_2 / ||\mathbf{y} - \bar{\mathbf{y}}||_2$.

Document technique DT8 : Code des classes concernant les quaternions et rotations

```
import math
import numpy as np

#### OBJECT
class Quaternion:
  '''Quaternion
  object to represent and perform operations on quanternions

  '''
  def __init__(self, a, b, c, d):
    self.a = float(a)
    self.b = float(b)
    self.c = float(c)
    self.d = float(d)

  ## Operator overloading
  def __str__(self):
    sign_i = '+' if self.b >=0 else '-'
    sign_j = '+' if self.c >=0 else '-'
    sign_k = '+' if self.d >=0 else '-'
    return str(self.a)+sign_i+str(abs(self.b))+".i"+sign_j+str(abs(self.c))+".j"+sign_k+str(abs(self.d))+".k"
  def __eq__(self, other):
    if isinstance(other, Quaternion):
      return self.a==other.a and self.b==other.b and self.c==other.c and self.d==other.d
    else:
      return False

  def __ne__(self, other):
    return not self==other

  def __add__(self, other):
    if isinstance(other, Quaternion):
      return Quaternion(self.a+other.a, self.b+other.b, self.c+other.c, self.d+other.d)
    elif isinstance(other, int) or isinstance(other, float):
      return Quaternion(self.a+other, self.b, self.c, self.d)
    elif isinstance(other, complex):
      return Quaternion(self.a+other.real, self.b+other.imag, self.c, self.d)
    else:
      raise TypeError("Operation not allowed, addition can be with Quaternion, int, float or complex, not"+str(type(other)))

  __radd__ = __add__

  def __sub__(self, other):
    if isinstance(other, Quaternion):
      return Quaternion(self.a-other.a, self.b-other.b, self.c-other.c, self.d-other.d)
    elif isinstance(other, int) or isinstance(other, float):
      return Quaternion(self.a-other, self.b, self.c, self.d)
    elif isinstance(other, complex):
```

```
49        return Quaternion(self.a-other.real, self.b-other.imag, self.c, self.d)
50      else:
51        raise TypeError("Operation not allowed, substraction can be with
      Quaternion, int, float or complex, not"+str(type(other)))
52
53    def __truediv__(self, other):
54      if isinstance(other, Quaternion):
55        # 1/other computation
56        inv_other = (1/(other.a**2 + other.b**2 + other.c**2 + other.d**2))*
      Quaternion(other.a, -other.b, -other.c, -other.d)
57        return self.__mul__(inv_other)
58      elif isinstance(other, int) or isinstance(other, float):
59        return Quaternion(self.a/other, self.b/other, self.c/other, self.d/
      other)
60      elif isinstance(other, complex):
61        inv_other = to_Quaternion(1/other)
62        return self.__mul__(inv_other)
63      else:
64        raise TypeError("Operation not allowed, division can be with Quaternion
      , int, float or complex, not"+str(type(other)))
65
66    def __rtruediv__(self, other):
67      if isinstance(other, int) or isinstance(other, float):
68        inv = (other/(self.a**2 + self.b**2 + self.c**2 + self.d**2))*
      Quaternion(self.a, -self.b, -self.c, -self.d)
69        return inv
70      elif isinstance(other, complex):
71        q = to_Quaternion(other)
72        return q.__truediv__(self)
73      else:
74        raise TypeError("Operation not allowed, division can be with Quaternion
      , int, float or complex, not"+str(type(other)))
75
76    def __pow__(self, power, modulo=None):
77      if isinstance(power, int):
78        powered_quat = Quaternion(self.a, self.b, self.c, self.d)
79        for k in range(1, abs(power)):
80          powered_quat = powered_quat*self
81        if power < 0:
82          powered_quat = 1/powered_quat
83        return powered_quat
84      else:
85        raise TypeError("Operation not allowed, power of Quaternion only
      computed for int, not"+str(type(other)))
86
87    ## usefull methods
88    def conjugate(self):
89      return Quaternion(self.a, -self.b, -self.c, -self.d)
90
91    def scalar(self, type=None):
92      if type is 'float':
93        return self.a
94      else:
95        return Quaternion(self.a, 0, 0, 0)
96
97    def is_scalar(self):
```

```
    if self.b==0 and self.c==0 and self.d==0:
      return True
    else:
      return False

  def vector(self):
    return Quaternion(0, self.b, self.c, self.d)

  def is_vector(self):
    if self.a == 0:
      return True
    else:
      return False

  def norm(self):
    return math.sqrt(self.a**2 + self.b**2 + self.c**2 + self.d**2)

  def is_unit(self, tol=1e-7):
    return np.abs(self.norm() - 1.) < tol

class UnitQuaternion(Quaternion):
  def __init__(self, a, b, c, d):
    assert is_unit(Quaternion(a,b,c,d))
    super().__init__(a,b,c,d)

  def invert(self):
    return self.conjugate()

class Rotation3D:
  def __init__(self, theta, X, Y, Z, deg=True):
    pass
    ## Question 36

  def __call__(self, a, b, c):
    pass
    ## Question 36

### FUNCTIONS
def is_unit(q):
  if not isinstance(q,Quaternion):
    q = to_Quaternion(q)
  return q.is_unit()
```

Document technique DT9 : Code du tracé de la réponse fréquentielle de la FTBO

```
import numpy as np
import matplotlib.pyplot as plt

# definition de la FTBO
w = np.logspace(0, 4, 1200)
p = 1j*w
FTBO = (5/(1+p))*(2/(1+0.1*p))**2*(4/(1+0.01*p))**-2*(3/(1+0.001*p))**2

```

```
# calcul du gain 'G' et de la phase 'Phi'
G = 20 * np.log10(abs(FTBO))
Phi = np.angle(FTBO) * 180 / np.pi

# correction de la phase pour enlever les discontinuites
Phi2 = np.copy(Phi) # Phi2 contiendra la phase corrigee
period = 360
dangle = np.diff(Phi)
dangle_desired = (dangle + period/2.) % period - period/2.
correction = np.cumsum(dangle_desired - dangle)
Phi2[1:] _____________________________

plt.figure()
plt.subplot(311)
plt.semilogx (w, G, color="blue", linewidth="1")
plt.xlabel ("Pulsation")
plt.ylabel ("Gain")
plt.minorticks_on()
plt.grid(which='both')

plt.subplot(312)
plt.semilogx (w, Phi, color="red", linewidth="1")
plt.xlabel("Pulsation")
plt.ylabel("Phase brute")
plt.minorticks_on()
plt.grid(which='both')

plt.subplot(313)
plt.semilogx (w, Phi2, color="red", linewidth="1")
plt.xlabel("Pulsation")
plt.ylabel("Phase corrigee")
plt.minorticks_on()
plt.grid(which='both')
```

Document technique DT10 : Code mesurant les performances et réglant le correcteur de l'asservissement

```
import numpy as np
import matplotlib.pyplot as plt
import control as ct
from scipy.signal import tf2ss
from random import random, randint
from pyswarm import pso

Ki, Kc, Kr, L, R, f = 0.6, 0.1, 1/20, 4E-3, 1, 4E-3, 1.5E-3
numG = [Ki*Kc*Kr]
denG = [L*J, (R*J+L*f), Ki**2+R*f, 0]
# FTBO retour unitaire non corrigee
G = ct.tf(numG, denG)

# Calcul de la FTBF corrigee
def correcteur(Kp, Ti, Td):
    """definition classique du PID, renvoie les polynomes"""
    numC = [Kp*Td*Ti, Kp*Ti, Kp]
```

```
18      denC = [Ti, 0]
19      return numC, denC
20
21  def multi_FT(num1, den1, num2, den2):
22      """multiplication de fonctions de transferts, polynomes avec puiss
        decroissantes"""
23      num = multi_poly(num1, num2)
24      den = multi_poly(den1, den2)
25      return num, den
26
27  def FTBF(num, den):
28      """calcul de la ftbf a retour unitaire pour une ftbo donnee avec 2 polys
        decroissants"""
29      num_bf = num
30      den_bf = somme_poly(num, den)
31      return num_bf, den_bf
32
33  T = np.linspace(0, 4, 2000)
34
35  A = multi_FT([1], [1], numG, denG)
36  B = FTBF(*A)
37  ftbfCorCalc = ct.tf(*B)
38  t, sCorCalc = ct.step_response(ftbfCorCalc, T)
39  plt.figure()
40  plt.grid()
41  plt.plot(t, sCorCalc, label='Reponse avec FTBF manuelle (correcteur unitaire)
        ')
42  plt.legend()
43  plt.show()
44
45  # PERFORMANCES
46
47  def depassement(s, t):
48      s_fin = s[-1]
49      return((max(s)-s_fin)/s_fin)
50
51  def precision(s, t):
52      return abs(s[-1] - 1)
53
54  def ponderation_cout(T5, D1, P1, A1):
55      duree = T[-1]-T[0]
56      #print(duree)
57      a, b, c, d  = duree, duree, 20, 1   # **2 pour b
58      cout = a*T5 + b*D1*100 + c*P1 + d*A1
59      return cout
60
61  def rep_indicielle(Kp, Ti, Td):
62      numC, denC = correcteur(Kp, Ti, Td)
63      num_BO, den_BO = multi_FT(numG, denG, numC, denC)
64      num_BF, den_BF = FTBF(num_BO, den_BO)
65      BF = ct.tf(num_BF, den_BF)
66      temps, sortie = ct.step_response(BF, T)
67      return(temps, sortie)
68
69  #    Mise en place des fonctions pour optimisation
70  def calcul_coef_correcteur(Np, Ni, Nd):
```

```
    Kpmin, Kpmax = 0.01, 200
    Timin, Timax = .01, 100
    Tdmin, Tdmax = 0, 50
    Kp = (Kpmax-Kpmin)*Np/(2**10-1)+Kpmin
    Ti = (Timax-Timin)*Ni/(2**10-1)+Timin
    Td = (Tdmax-Tdmin)*Nd/(2**10-1)+Tdmin
    return(Kp, Ti, Td)

#   Optimisation par algorithme genetique
def perf(Li):
    Ip, Ii, Id = decodage(Li[0]), decodage(Li[1]), decodage(Li[2])
    return calcul_cout(Ip, Ii, Id)

cycles = 50
population = generer_population_initiale(100, 10)
for i in range(cycles):
    population_triee = tri(population)
    population = nouv_generation(population_triee)
    population = doublons(population)
solution = population[0]
KpGene, TiGene, TdGene = calcul_coef_correcteur(
    decodage(solution[0]), decodage(solution[1]), decodage(solution[2]))

print("reglage optimal algo gene :", KpGene, TiGene, TdGene)
print("perf algo gene :", perf(solution))
```

Document technique DT11 : Exemple d'utilisation de *pyswarm*

```
from pyswarm import pso
def cout(x):
    x1 = x[0]
    x2 = x[1]
    return x1**4 - 2*x2*x1**2 + x2**2 + x1**2 - 2*x1 + 5
lb = [-3, -1]
ub = [2, 6]
xopt, fopt = pso(cout, lb, ub)
```

le vecteur *xopt* est la solution optimale de la fonction *cout*. *lb* et *ub* représentent la borne inférieure et la borne supérieure que peuvent prendre les valeurs de *xopt*.

Document réponse DR1 : Question 6 (a)

Valeurs de coefficients de la matrice C pour une profondeur de 8 :

0.49	0.416	0.278	0.098				
0.462	0.191	-0.191	-0.462				
0.416	-0.098	-0.49	-0.278	0.278	0.49	0.098	-0.416
0.278	-0.49	0.098	0.416	-0.416	-0.098	0.49	-0.278
0.098	-0.278	0.416	-0.49	0.49	-0.416	0.278	-0.098

Document réponse DR2 : Question 13

```
import numpy as np

def wavelet2(M, s, w=5):
  x = np.arange(0, M) - (M - 1.0) / 2
  x = x / s
  wave_dat = morlet(x, w=w)
  return (1/np.sqrt(s))*wave_dat

def cwt_convolve(sig, freqs, Ts):
  '''
  Calcul de la CWT en utilisant la fonction numpy.convolve

  Arguments
  _________
  sig : array_like
    signal a tranformer par ondelette
  freqs : array_like
    frequences centrales des ondelettes filles
  Ts : float
    periode d echantillonage du signal 'sig'

  Returns
  _______
  output : array_like
    transformee en ondelette de sig, pour un vecteur tau de
    meme longueur que sig, aux frequences donnees par freqs
  '''
  w0 = 5
  tlen = len(sig)
  t = np.linspace(0, (tlen-1)*Ts, num=tlen)
  output = np.zeros((_____________________), dtype=np.complex128)
  for i, freq in enumerate(freqs):
    width = _____________________
    tau = np.arange(0, tlen) - (tlen - 1.0) / (2*width)
    wavelet_data = _____________________________
    # convolve
    output[i, :] = np.convolve(sig, wavelet_data2, mode=______)
  return output
```

Document réponse DR3 : Question 21

```
import numpy as np

def wavelet2(M, s, w=5):
  x = np.arange(0, M) - (M - 1.0) / 2
  x = x / s
  wave_dat = morlet(x, w=w)
  return (1/np.sqrt(s))*wave_dat

def cwt_convolve(sig, freqs, Ts):
  '''
  Calcul de la CWT en utilisant la fonction numpy.convolve

  Arguments
  _________
  sig : array_like
    signal a tranformer par ondelette
  freqs : array_like
    frequences centrales des ondelettes filles
  Ts : float
    periode d echantillonage du signal 'sig'

  Returns
  _______
  output : array_like
    transformee en ondelette de sig, pour un vecteur tau de
    meme longueur que sig, aux frequences donnees par freqs
  '''
  w0 = 5
  tlen = len(sig)
  t = np.linspace(0, (tlen-1)*Ts, num=tlen)
  output = np.zeros((_____________________), dtype=np.complex128)
  for i, freq in enumerate(freqs):
    width = ____________________________________
    tau = ____________________________________
    wavelet_data = ____________________________________
    # convolve
    output[i, :] = ____________________________________
  return output
```

Document réponse DR4 : Question 27

Diagramme schématique des matrices mises en jeu dans l'algorithme NIPALS :

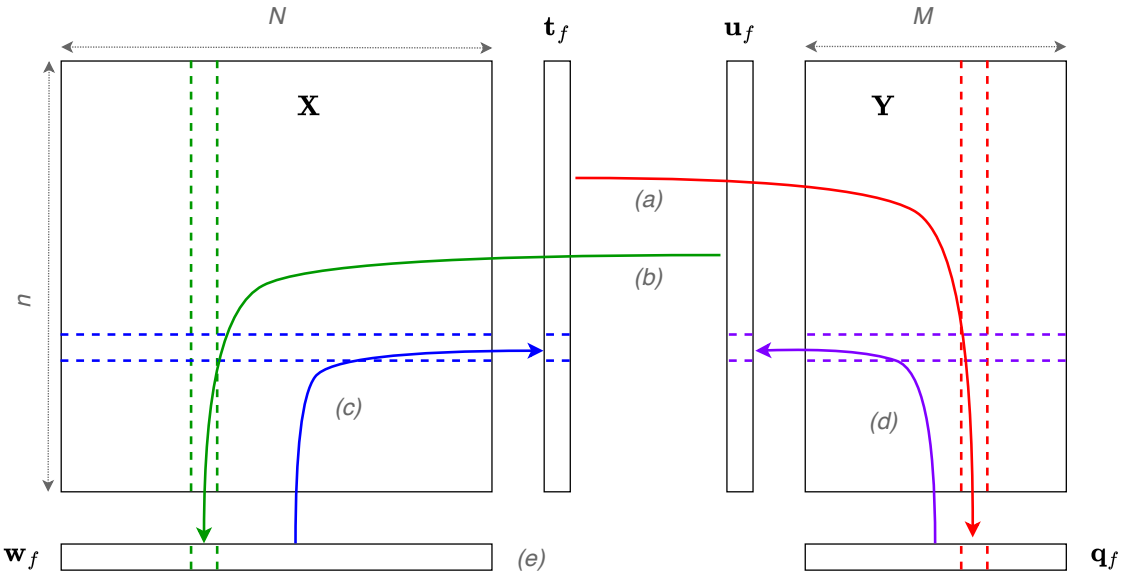


Tableau de correspondance endroit du diagramme/étape de l'algorithme :

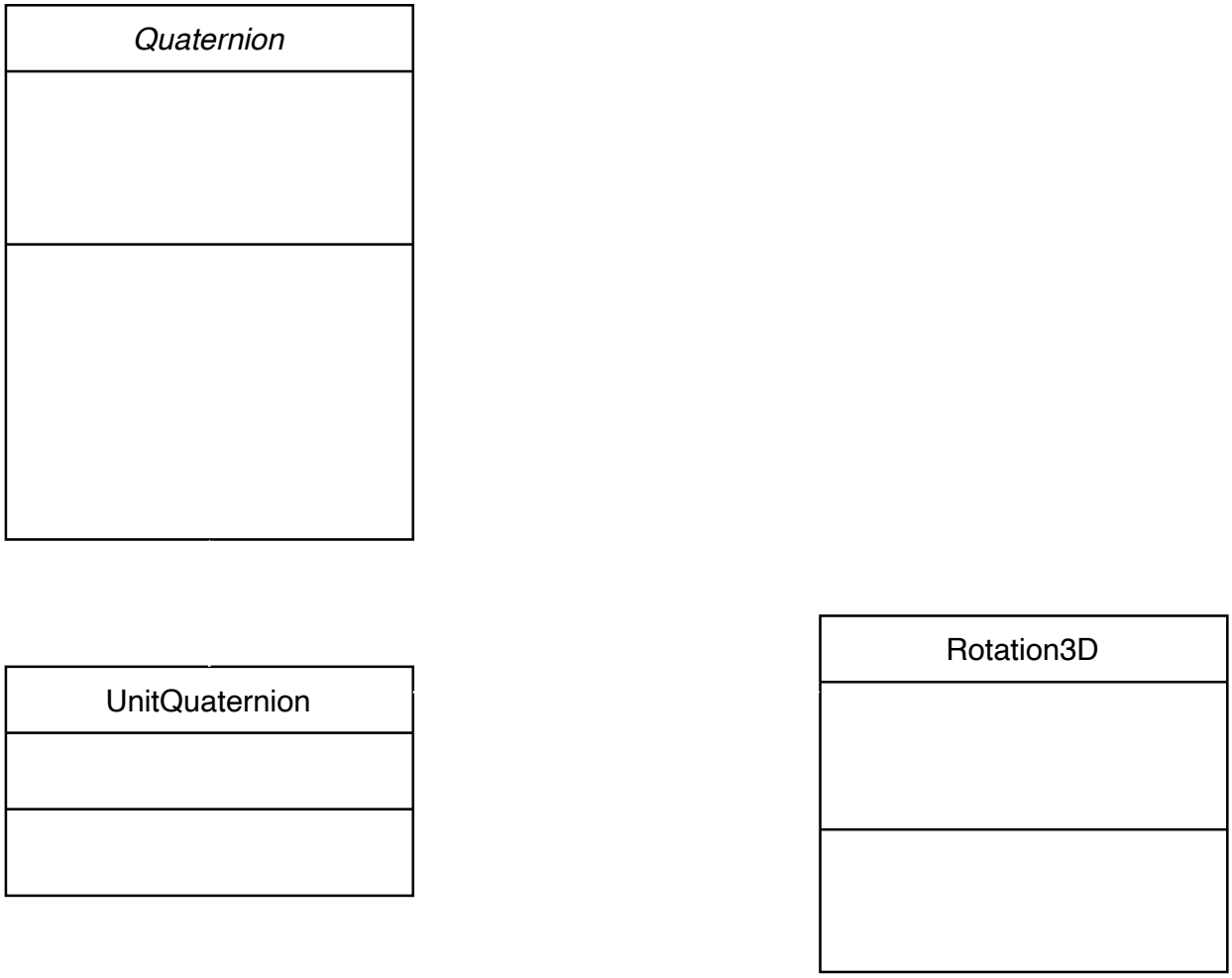
endroit	(a)	(b)	(c)	(d)	(e)
étape (cf figure 11)					

Document réponse DR5 : Question 28

```
for comp in range(ncomp):
    train_x_mat = self.x_mat
    train_y_mat = self.y_mat
    nrt, x_nct = train_x_mat.shape
    y_nct = train_y_mat.shape[1]
    train_x_miss = np.isnan(train_x_mat)
    train_y_miss = np.isnan(train_y_mat)
    # Set u to some column of Y
    uh = train_y_mat[:, startcol]
    th = uh

    it = 0
    while True:
        # X-block weights
        wh = _____________________________
        # Normalize
        wh = _____________________________

        # X-block Scores
        th_old = th
        th = _____________________________

        # Y-block weights
        qh = _____________________________

        # Y-block Scores
        uh = _____________________________

        # Check convergence
        if np.nansum((th - th_old) ** 2) < tol:
            break
        it += 1
        if it >= maxiter:
            raise RuntimeError(
                "Convergence was not reached in {} iterations for
   component {}".format(
                    maxiter, comp
                )
            )

```

```
40      # Calculate X loadings and rescale the scores and weights
41      ph = train_x_mat.T.dot(th) / sum(th * th)
42
43      loadings[:, comp] = ph
44      scores[:, comp] = th
45      u[:, comp] = uh
46      q[:, comp] = qh
47      weights[:, comp] = wh
48      bh = _____________________________
49      b[comp] = bh
50
51      self.x_mat = self.x_mat - np.outer(th, ph)
52      self.y_mat = self.y_mat - bh * np.outer(th, qh)
```

Document réponse DR6 : Question 34



Document réponse DR7 : Question 35

```
class Rotation3D:
  def __init__(self, theta, X, Y, Z, deg=True):
    if deg:
      self.theta_deg = theta
      self.theta = np.radians(theta)
    else:
      self.theta = theta
      self.theta_deg = np.degrees(theta)
    norm = np.sqrt(X**2+Y**2+Z**2)
    self.X = X/norm
    self.Y = Y/norm
    self.Z = Z/norm
    Rq = ____________________________________
    Iq = ____________________________________
    self.q_theta = self.q_theta = UnitQuaternion(Rq, self.X*Iq, self.Y*
   Iq, self.Z*Iq)
    self.q_theta_inv = ____________________________________
```

Document réponse DR8 : Question 37

angle	-55	-104	-164	158	-179	-156	170	122	100
dangle	-49	-60	322	-338	23	326	-48	-21	
dangle_desired									
correction									

Rapport sur l'épreuve de Modélisation

Le sujet portait sur l'étude d'un système implant + exosquelette destiné à permettre à des personnes paraplégiques de pouvoir se déplacer. Le sujet était découpé en 3 parties indépendantes que les candidats pouvaient traiter dans l'ordre qu'ils souhaitaient. Le sujet était long, afin de permettre un choix des candidats quant aux aspects du programme qu'ils désiraient traiter. Les trois parties suivaient le fil de l'information : la première partie portait sur l'acquisition du signal électrique en surface du cortex et sa compression, la seconde traitait des algorithmes de détection de l'intention de mouvement, la dernière partie quant à elle portait sur la commande de l'exosquelette.

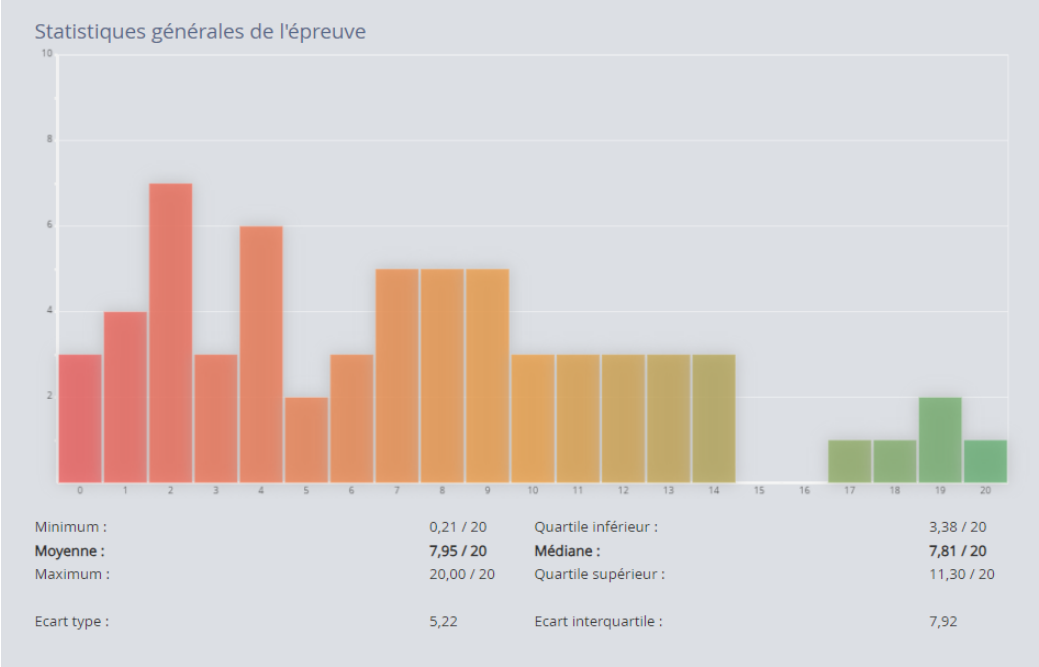


Figure 1: Histogramme de répartition des notes des candidats sur l'épreuve de modélisation d'un système.

Les notes sont réparties entre 0,21 et 20. Le sujet a permis de bien évaluer les candidats. Un petit groupe de candidats a réussi à traiter correctement plus de la moitié du sujet. Cependant, on peut regretter qu'un certain nombre de candidat ne traite qu'un nombre très réduit de questions.

Conseils généraux : Les questions en début de chaque partie devraient être à la portée d'une grande majorité de candidat. Sur le temps de l'épreuve, quelques instants devraient être utilisés afin de parcourir le sujet dans sa globalité et ainsi pouvoir identifier les questions plus accessibles.

1 Partie 1

Cette partie était très fortement orientée vers le traitement de signal. Le candidat était amené à constater l'impossibilité de transmettre l'ensemble des canaux mesurés à la fréquence d'échantillonnage maximale. Il en résulte la nécessité de compresser le signal, l'utilisation de la transformée en cosinus discrète était ensuite évaluée quant à son adéquation au cahier des charges et à la cible de calcul embarquée (micro-contrôleur)

Cette partie a été traitée par la totalité des candidats. Cependant le niveau est variable, certains bloquant dès les premières questions. Peu de candidat montre une vraie maîtrise de l'outil mathématique mais l'utilisation et la transcription de résultats démontré en code informatique était globalement satisfaisante.

Il est recommandé aux candidats de se familiariser avec les API courantes de traitement du signal, afin de pouvoir lire, comprendre ou concevoir des fonctions permettant de réaliser les opérations simples de traitement du signal. La maîtrise de certains outils (probabilités, fonctions usuelles) est indispensable en traitement du signal.

2 Partie 2

La partie 2 portait sur le traitement des signaux acquis : dans un premier temps le signal est transformé en features permettant un traitement spatio-temporel en utilisant la transformée en ondelette. Afin de réduire la complexité de ce calcul, une solution basée sur l'utilisation de la FFT était étudiée. Les vecteurs de features extraits étaient ensuite mis en entrée d'algorithmes dérivés des moindres carrés et des composantes principales. Cette partie n'a été abordée que par un faible nombre de candidats. Certaines questions simples (trouver la sortie d'un algorithme, suivi d'un schéma et d'un pseudo code, code limité à une ligne) auraient cependant permis aux candidats de marquer des points

Certains algorithmes classiques en informatique doivent faire partie de la culture scientifique des candidats. Le sujet utilisait régulièrement une étude détaillée des entrées sorties des algorithmes, qui doit être un type de raisonnement acquis à ce niveau de compétences en sciences de l'ingénieur.

3 Partie 3

Le début portait sur le codage des rotations avec des quaternions, solution classique utilisée en robotique mais aussi en informatique pour les scènes 3D (notamment dans les jeux vidéos). Elle ne nécessait pas de connaissances préalables. Après des questions de compréhension, il s'agissait de compléter un diagramme UML et du code sur des classes correspondant à l'implémentation.

On s'intéressait ensuite au motion controller, avec des questions utilisant Python. Cette partie comportait quelques questions faciles, comme le tracé dans le plan de Black, ou très classiques, comme écrire une recherche de valeur par dichotomie dans un tableau trié, qui a été de façon surprenante globalement très peu abordée, ni correctement traitée. La partie suivante faisait manipuler des fonctions de transferts codées sous forme de liste avec des algorithmes assez simples, pour arriver à un indicateur de performance de l'asservissement.

La suite mettait en place différentes méthode d'optimisation, d'abord par approche force brute, puis une méthode ad-hoc, puis une utilisation détaillée d'un algorithme génétique et enfin l'utilisation d'une librairie utilisant une approche par essaim de particules. La dernière question portait sur le choix optimum et ne nécessitait pas d'avoir fait les questions précédentes.

Épreuve d’admissibilité de conception préliminaire d'un système, d'un procédé ou d'une organisation

A. Présentation de l’épreuve

Arrêté du 28 décembre 2009 modifié

- Durée totale de l'épreuve : 6 heures
- Coefficient 1

L'épreuve est spécifique à l'option choisie.
À partir d'un dossier technique comportant les éléments nécessaires à l'étude, l'épreuve a pour objectif de vérifier les compétences d'un candidat à synthétiser ses connaissances pour proposer ou justifier des solutions de conception et d'industrialisation d'un système technique dans le domaine de la spécialité du concours dans l'option choisie.

B. Sujet

Le sujet est disponible en téléchargement sur le site du ministère à l’adresse :
https://www.devenirenseignant.gouv.fr/media/4817/download

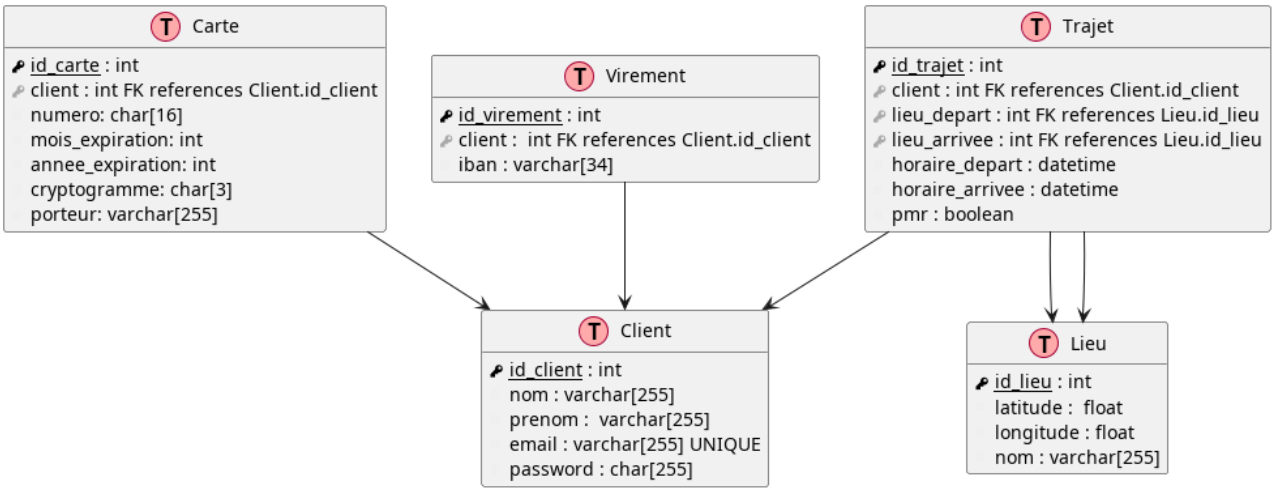
C.Eléments de correction

Partie I : Syst. d'information et accès véhicule

I.1 Modes de paiement et trajets

Q 1. Ajout des méthodes de paiement au schéma relationnel

> Ajouter une table Carte avec les numéros de carte bleue, nom etc. Ajouter une table Virement avec les IBANS. Dans chaque cas, il faut un champs supplémentaire pour la clé (l'IBAN ou le numéro de carte de sont pas des clés valides car une carte peut figurer plusieurs fois dans la base).



Q 2. Requête sur la table Client

```
SELECT nom, prenom from Client where email="john.smith@yahoo.fr";
```

Q 3. Requête sur les trajets réalisés

```
SELECT latitude, longitude
  from Client join Trajet on Trajet.client=Client.id_client
          join Lieu on Trajet.lieu_depart = Lieu.id_lieu
  where Trajet.horaire_arrivee < now()
     AND email="john.smith@yahoo.fr";
```

Q 4. Requête de récupération du prochain trajet

```
SELECT ld.latitude, ld.longitude, la.latitude, la.longitude,
    t.horaire_depart, t.horaire_arrivee
  FROM Client as c, Trajet as t, Lieu as ld, Lieu as la
  WHERE c.email=<EMAIL-CLIENT> AND c.id_client=t.client AND
     t.horaire_depart >= now() AND t.lieu_depart = ld.id_lieu AND
     t.lieu_arrivee=la.id_lieu
     ORDER BY t.horaire_depart
  LIMIT 1
```

Q 5. Accès à la BDD en Python

```
class Database:
```

```
    def get_client_name_from_email(self, email: str) \
                        -> Optional[Tuple[str, str]]:
        """
        Renvoie le nom et le prénom d'un client à partir de son email.
        Si le client n'est pas trouvé, renvoie None
        >>> db.get_client_name_from_email("john.smith@yahoo.fr")
        ("Smith", "John")
        """
        req = """\
            SELECT nom, prenom
              FROM Client as c
              WHERE c.email=?
              LIMIT 1
            """ # LIMIT 1 facultatif
        res = self._select(req, (email,))
        if not res:
            rep = None
        else:
            rep = (res[0][0], res[0][1])
        return rep
```

Q 6. Intérêt du hachage des mots de passe

Afin qu'une fuite de la base de données se solde uniquement par la divulgation des condensés des mots de passe, et non des mots de passe eux-même. Le calcul de condensés étant irréversible, et la recherche d'un antécédent difficile en pratique, on ne pourra pas retrouver les mots de passe à partir des condensés facilement, surtout si le mot de passe n'est pas courant.

Q 7. Intérêt du salage des mots de passe

Certains mots de passe sont très courants. On peut ainsi construire des tables contenant les condensés de mots de passe courants, et les comparer aux condensés volés. Le salage consiste à rajouter une chaîne aléatoire, connue et différente, à chaque mot de passe haché. Ainsi, deux mots de passe identiques donneront lieu à des condensés salés différents, ce qui ralentit l'utilisation d'un dictionnaire de mots de passes hachés préparé à l'avance ainsi que l'attaque simultanée de plusieurs comptes (car ils auront tous un sel différent).

Q 8. Complétion du code Python pour le stockage des mots de passe.

```
# dans outils.py
def hash_with_salt(salt: str, password: str) -> str:
    """
    >>> hash_with_salt("S_8u", "Wikipedia")
    'S_8u:33bf32c0e27f6361d21287df83f19a23c5b2d0e7f25e18...'
    """
    b_password = password.encode("utf8")
    b_salt = salt.encode("utf8")
    hashed_password = hashlib.sha256(b_salt + b_password).hexdigest()
    return salt + ":" + hashed_password

# dans database.py
class Database:

    def store_password(self, client_email: str, password: str) -> bool:
        salt = outils.random_chars(16)
        return self.update_pwd(client_email,
                               outils.hash_with_salt(salt, password))
```

Q 9. Complétion du code Python d'authentification d'un utilisateur

```
class Database:

  def check_password(self, client_email: str, password: str) -> bool:
    try:
      stored_hashed_password = \
          self.get_hashed_password(client_email)
    except DatabaseValueError:
      return False
    salt = stored_hashed_password.split(":")[0]
    return outils.hash_with_salt(salt, password) == \
          stored_hashed_password
```

I.2 Génération du code QR

Q 10. Désérialisation de la structure de données CourseClient.

```
class CourseClient:

  @classmethod
  def from_bytes(cls, b: bytes) -> "CourseClient":
    """
    Méthode de classe qui renvoie un nouvel objet `CourseClient`
    à partir de sa représentation compacte
    """
    course, client = struct.unpack(">ii", b[:8])
    pos_depart = Position.from_bytes(b[8:16])
    h_depart = struct.unpack(">hhhhh", b[16: 26])
    pos_arrivee = Position.from_bytes(b[26:34])
    h_arrivee = struct.unpack(">hhhhh", b[34:44])
    heure_depart = datetime.datetime(*h_depart)
    heure_arrivee = datetime.datetime(*h_arrivee)
    return cls(course, client, pos_depart, heure_depart,
        pos_arrivee, heure_arrivee)
```

On autorise aussi :

```
pos_arrivee = struct.unpack(">ff", b[26:34])
```

Q 11. Taille maximum du bloc de données

On doit s'assurer que la valeur de n est codable sur 16 bits, car il y a 2 octets seulement réservés pour coder la taille des données. 0xffff est la valeur du plus grand entier codable en non signé sur 16 bits.

Q 12. Vérification de la validité d'une signature

```
def check_sign(data_sig: bytes) -> bool:
  """
  Vérifie que les données sont correctement signées
  (ont été correctement produites par `sign`)
  """
  sig_type = data_sig[0]
  if sig_type == 1:
    data_size = data_sig[1] * 256 + data_sig[2]
    data = data_sig[3:3 + data_size]
    sig = hmac.HMAC(SECRET_KEY, msg=data,
```

```
                digestmod=hashlib.sha256).digest()
        return sig == data_sig[3 + data_size:]
    else:
        return False
```

ou mieux

```
def check_sign(data_sig: bytes) -> bool:
    """
    Vérifie que les données sont correctement signées
    (ont été correctement produites par `sign`)
    """
    sig_type = data_sig[0]
    if sig_type == 1:
        data_size = data_sig[1] * 256 + data_sig[2]
        data = data_sig[3:3 + data_size]
        return sign(data) == data_sig
    else:
        return False
```

Q 13. Encodage des données en base45

```
def to_b45(a: int, b: Optional[int] = None) -> Union[Tuple[int, int],
                                                     Tuple[int, int, int]]:
    if b is None:
        n = a
        c, d = n % 45, n // 45
        return (c, d)
    else:
        n = a * 256 + b
        c, d, e = n % 45, (n // 45) % 45, n // 45 // 45
        return (c, d, e)

def b45_to_str(b45lst: list) -> str:
    chars = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ $%*+-./:"
    return "".join(chars[k] for k in b45lst)

def base45_encode(data: bytes) -> str:
    """
    Encode une séquence d'octets en chaîne Base45
    """
    lst_res = []
    lst_bytes = list(data)
    if len(data) % 2 == 1:
        lst_bytes.append(None)
    for a, b in zip(lst_bytes[0::2], lst_bytes[1::2]):
        b45lst = to_b45(a, b)
        lst_res.append(b45_to_str(b45lst))
    return "".join(lst_res)
```

Q 14. Création de l'image d'un code QR

```
class CourseClient:

    def create_qr_code(self) -> Image:
```

```
        """
        Crée l'image d'un code QR signé à partir des informations
        d'un trajet (CourseClient)
        """
        data_bin = self.to_bytes()
        data_bin_signed = outils.sign(data_bin)
        data_b45 = outils.base45_encode(data_bin_signed)
        image = outils.qrencode_alpha(data_b45)
        return image
```

Q 15. Décodage de l'image d'une code QR

```
class CourseClient:

    @classmethod
    def read_qr_code(cls, img: Image) -> Optional["CourseClient"]:
        """
        Si l'image du code QR est valide, et que le code QR est
        correctement signé, renvoie l'objet CourseClient contenu
        dans le code QR. Sinon, renvoie None
        """
        data_b45 = outils.qrdecode_alpha(img)
        if data_b45 is None:
            return None
        data_bin_signed = outils.base45_decode(data_b45)
        if not outils.check_sign(data_bin_signed):
            return None
        else:
            lg_data = data_bin_signed[1] * 256 + data_bin_signed[2]
            data = data_bin_signed[3:3+lg_data]
        return cls.from_bytes(data)
```

Partie II : Équipement et communications

II.1 Système eCall

Q 16. Type de trame NMEA 0183 utilisée pour récupérer la position du véhicule ?

Ce sont les trames GGA qui nous intéressent.

Q 17. Calcul des caractères manquants à la fin de la trame donnée ci-dessous ?

$GPGSA,A,3,06,07,16,20,14,,,24,,07,,,2.5,1.3,2.1*

Il manque la somme de contrôle .On réaliser un ou exclusif (XOR) sur tous les octets à partir de GP.... Le résultat donne 0x34. La trame complète est donc : GPGSA,A,3,06,07,16,20,14,,,24,,07,,,2.5,1.3,2.1*34. Pour réaliser ce calcul rapidement, on utilise le fait que $x \text{XOR} x = 0$. Il suffit donc de prendre en compte les caractères en nombre impair : ,.5PS. Un ou exclusif entre : 2c, 2e, 35, 50, 53 (utilisation de la table ASCII en annexe) donne le résultat.

Q 18. Complétion du code d'analyse d'une trame NMEA 0183, qui renvoie une structure contenant les informations d'horodatage et de position.

```
typedef struct {
  float latitude, longitude;
  int heures, minutes;
  float secondes;
```

```
} position;

float read_deg(char str[]) {
   float tmp;
   int deg;
   float minutes;
   sscanf(str, "%f", &tmp);
   deg = tmp / 100;
   minutes = tmp - deg * 100;
   return deg + minutes / 60;
}

void read_latitude(char str[], float * latitude) {
   *latitude = read_deg(str);
   if (str[10] == 'S') *latitude = -*latitude;
}
void read_longitude(char str[], float * longitude) {
   *longitude = read_deg(str);
   if (str[11] == 'W') *longitude = -*longitude;
}

void read_timestamp(char str[], int * h, int * m, float *s) {
   sscanf(str, "%2d%2d%f", h, m, s);
}

int check_crc(char str[]) {
   int sum = 0;
   int chk = 0;
   int i = 0;
   for (i = 1; i < strlen(str) && str[i] != '*'; i++) {
      sum ^= str[i];
   }
   if (str[i] != '*') return 0;
   sscanf(&str[i+1], "%x", &chk);
   return chk == sum;
}

/* Fonction principale, prenant en paramètre une trame NMEA 0183
   complète et un pointeur vers une structure de position.
   Cette structure est remplie par la fonction, qui renvoie 1
   si un décodage a été effectué, et 0 sinon (par exemple si
   ce n'est pas une trame du bon type)
*/
int get_position(char nmea_frame[], position * pos) {
  if (nmea_frame[0] != '$') return 0;
  if (!check_crc(nmea_frame)) return 0;
  if (strncmp(&nmea_frame[3], "GGA,", 4)) return 0;
  read_timestamp(&nmea_frame[7], &(pos->heures),
              &(pos->minutes), &(pos->secondes));
  read_latitude(&nmea_frame[18], &(pos->latitude));
  read_longitude(&nmea_frame[30], &(pos->longitude));
  return 1;
  }
```

II.2 Couche réseau et transports - GeoNetworking

Adressage GeoNetworking

Q 19. On donne le début d'une adresse en notation hexadécimale pointée : 15.0C.xx.xx.xx.xx.xx.xx Indiquer le type de station.

Il suffit d'écrire les deux premiers octets de l'adresse en binaire :

```
|   octet 0   |   octet 1   |
0 0 [0 1 0] 1 [0 1 0 0 0 0 1 1 0 0]
 | |    | |_ 268 : Portugal
 | |    |_ véhicule privé
 | |_ 2 : Car
 |_ 0 : Véhicule
```

Vecteurs de Position

Q 20. Le champs TST des *Long POsition Vector* propose un horodatage des données de position. Du fait du nombre de bits utilisés pour encoder ce champs à quelle période, exprimée en jours, repasse-t-il à 0 ?

Le champ est un nombre de millisecondes encodées sur 32 bits. Le nombre de jours max est donc : $(2^{32} - 1)/1000/3600/24 = 49.71$j.

Trames GeoNetworking

Q 21. En supposant que l'unité est la seconde, si le champ LT de l'entête GeoNetworking code un nombre entier sur un octet, quelle est la durée maximum représentable (vmax) et quelle est la plus petite durée non nulle représentable (vmin) ?

Sur un octet, on a une valeur non nulle entre 1 et 255. Donc : vmin = 1 seconde, vmax = 255 secondes.

Q 22. En utilisant le codage réel du champs LT, quelles sont les nouvelles valeurs de vmin et vmax ? Quels octets (donner des nombres entiers en base 10) permettent de représenter ces deux valeurs extrêmes ?

La plus petite valeur représentable (vmin) est $1 \times 50ms = 50ms$. Sa représentation est : $00000100 = 4$ La plus grande valeur représentable (vmax) est $63 \times 100\text{s} = 6300\text{s}$. Sa représentation est : $11111111 = 255$

Algorithmes de routage

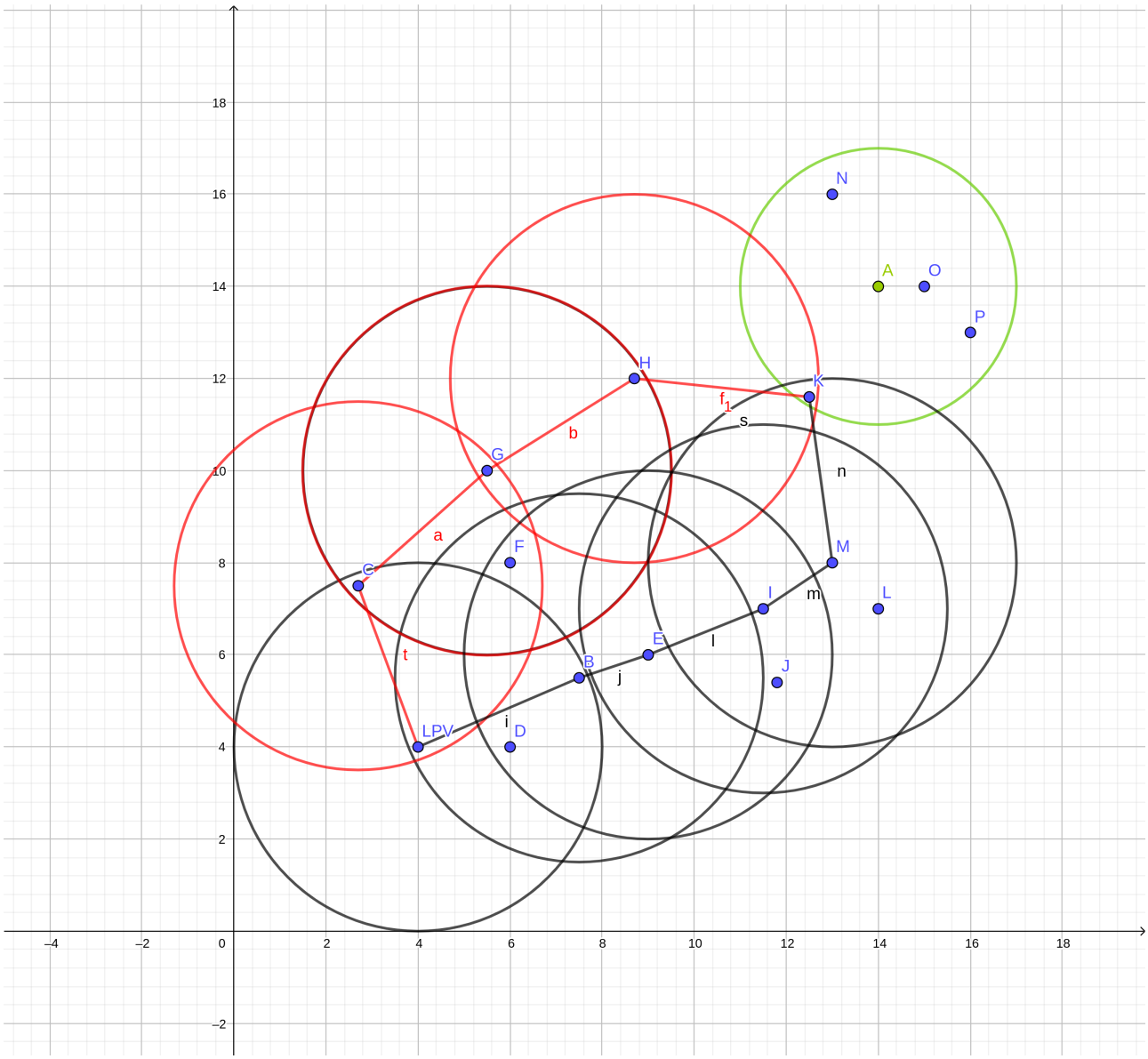
Q 23. Proposer une fonction *F* qui satisfait les conditions :

- $F(x, y) = 1$ si $M(x, y) = C$
- $F(x, y) > 0$ si $M(x, y)$ appartient au disque ouvert de centre C et de rayon r
- $F(x, y) = 0$ si $M(x, y)$ appartient au cercle de centre C et de rayon r
- $F(x, y) < 0$ si $M(x, y)$ n'appartient pas au disque fermé de centre C et de rayon r

$$F(x, y) = 1 - \frac{1}{r^2} \times ((x - x_c)^2 + (y - y_c)^2)$$

Q 24. Quels sont les nœuds par lesquels transite l'information de LPV jusqu'à une zone de rayon 3 centrée autour de A ?

Le trajet suivi est celui en noir. Le trajet en rouge est le plus court, mais l'algorithme *Simple Geobroadcast Forwarding Algorithm* (stratégie gloutonne) trouvera pas ici le trajet le plus court.



Le trajet suivi est : LPV, B, E, I, M, K. Ensuite les données sont broadcastées.

Q 25. La solution précédente est-elle optimale ?

Cf le trajet optimal sur l'image précédente. Il existe donc un trajet comportant moins de sauts : LPV, C, G, H, K. Toutefois, en utilisant uniquement une information locale, l'algorithme (glouton) donne très rapidement une solution proche de l'optimum. Le calcul de la solution optimale serait ici plus coûteux (temps de calcul et nécessité d'avoir plus qu'une information locale), pour un bénéfice faible.

Q 26. Dans le document DT12, l'algorithme nommé GF ALGORITHM est qualifié de glouton (*greedy* en anglais). Pour quelle raison ?

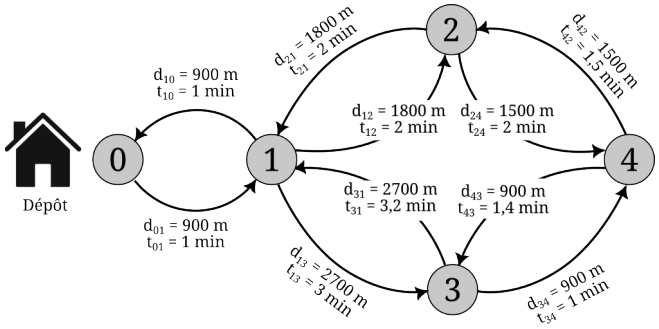
Les stratégies gloutonnes sont des stratégies qui visent à obtenir de façon rapide une solution peut-être approchée à un problème d'optimisation. Ici, pour rejoindre la zone visée avec le moins de sauts, il serait nécessaire d'envisager tous les chemins, ce qui est coûteux. Choisir comme premier saut celui qui rapproche le plus de la zone cible est un compromis qui, à défaut de toujours conduire à la solution optimale, donnera un résultat acceptable la plupart du temps, et ceci de manière extrêmement plus rapide que l'exploration systématique. Une fois le premier saut ainsi sélectionné, ce choix n'est plus remis en cause par la suite, d'où le nom d'algorithme « glouton ».

Q 27. Pour quelle(s) raison(s) le protocole IP n'est-il pas suffisant pour assurer les services de la couche réseau et en quoi le protocole GeoNetworking permet-il de régler ce problème ?

L'adressage IP est un adressage hiérarchique qui reflète la **topologie** du réseau, et non la **position** (géométrie) des différents nœuds. On peut facilement trouver des cas d'usage dans lesquels les informations topologiques ne sont pas suffisantes : gestion de la congestion dans une zone géographique, signalisation de bord de route… Le protocole GeoNetworking propose un mode de diffusion de l'information basé sur la position, et non sur la topologie. Il est ainsi possible de diffuser des informations de signalisation, par exemple, uniquement dans la zone géographique concernée par cette signalisation.

Partie III : Système d'optimisation des missions

Q 28. Dessiner un graphe représentant le réseau de transport.



Q 29. Donner en pseudo-code un algorithme permettant de calculer le chemin le plus rapide entre un nœud particulier v^* et tous les autres nœuds du graphe. Quelle est la complexité temporelle de l'algorithme proposé ?

N'importe quel algorithme de plus court chemin, par exemple Dijkstra, fait l'affaire. Il faut que l'algorithme se fonde sur les temps de parcours (t_{ij}) et non les distances (d_{ij})

```
Entrées : G=(V,A), Nœud origine v*
P := ∅
Pour chaque nœud v dans V
tt[v] := +∞
Fin Pour

tt[v*] := 0
Tant qu'il existe un nœud hors de P
  Choisir un nœud v hors de P de plus petite distance tt[v]
  Mettre v dans P
  Pour chaque nœud w hors de P voisin de v
  Si tt[w] > tt[v] + t_wv
    tt[w] := tt[v]+t_wv
    prédécesseur[w] := v
  Fin Pour
Fin Tant Que
```

Q 30. Donner en pseudo-code l'algorithme de insertion_missions permettant le traitement d'une nouvelle requête passager selon l'heuristique décrite précédemment.

```
Algorithme insertion_missions
Entrée : une requête r=(o, d, early_t, late_t, nb, PMR)
Sortie : selected_veh
min_marginal_cost := +∞

selected_veh := null

Pour tous les véhicules veh
```

```
    marginal_cost := insertion(r, veh)
    // insertion(r,veh) insère la requête r dans la tournée de veh et
    // renvoie le coût marginal de cette insertion (le detour).
    // Elle renvoie l'infini si la requête ne peut être insérée sans
    // violer les contraintes temporelles, ou si le véhicule
    // est inadapté (PMR)

    Si marginal_cost < min_marginal_cost alors
      min_marginal_cost := marginal_cost
      selected_veh := veh
    Fin Si
  Fin Pour

  Retourner selected_veh
```

Q 31. Proposer une idée de modification de l'algorithme insertion_missions, qui serait plus efficace selon vous en limitant sa myopie (argumenter).

(Toute idée plausible est recevable)

Par exemple : Pendant l'exécution, les véhicules peuvent régulièrement essayer de se débarrasser de requêtes insérées précédemment en les proposant à d'autres véhicules. Si le coût marginal de l'insertion dans le plan de l'autre véhicule est inférieur au coût marginal du véhicule, l'échange est effectué.

Ou encore : Les requêtes peuvent régulièrement être annulées et immédiatement re-soumises. Si un autre véhicule a un coût marginal inférieur au coût marginal du véhicule courant, la requête est insérée dans son plan.

Q 32. Décrire une solution permettant de déterminer si un re-calcul de missions est nécessaire.

Il faut parcourir les tournées des véhicules, et vérifier si les fenêtres temporelles ne sont pas violées, avec les temps de parcours courants. Il faut ajouter un attribut booléen enPanne spécifiant si le véhicule est en panne. Si l'une de ces deux conditions est vérifiée, on relance le calcul d'itinéraires. L'annulation d'une requête ne remet pas en question la faisabilité des autres requêtes déjà insérées.

Q 33. Proposer une solution permettant de recalculer les missions en cas de congestion.

Si lors de la procédure précédente, on découvre qu'une fenêtre temporelle est violée pour cause de changement de temps de parcours, la requête est annulée et re-soumise immédiatement pour trouver un autre véhicule qui peut l'insérer.

Q 34. Proposer une solution permettant de recalculer les missions en cas de panne.

Toutes les requêtes du véhicules sont annulées et re-soumises immédiatement pour trouver d'autres véhicules qui peuvent les insérer. Les passagers à bord soumettent des requêtes dont le nœud d'origine est le lieu de la panne (le nœud le plus proche).

Q 35. Proposer une solution permettant de recalculer les missions en cas d'annulation de requête.

Il ne faut rien à faire dans ce cas, car l'annulation d'une requête n'impacte pas la faisabilité des autres requêtes déjà insérées.

Q 36. Proposer une manière d'intégrer dans l'algorithme insertion(r, veh) des données d'historique, pour minimiser les recalculs d'itinéraires?

Il suffit d'utiliser le temps de parcours maximal observé (plutôt que minimal ou moyen) lors du calcul de la faisabilité des insertions et de la construction des plans / itinéraires des véhicules.

Q 37. Comment peut-on utiliser la possibilité pour les véhicules d'attendre sur des places de stationnement plutôt qu'au dépôt pour améliorer l'efficacité du système d'optimisation ?

> On peut utiliser l'historique des requêtes reçues avec les temps associés et positionner les véhicules libres (non-engagés pour des requêtes) et qui ont assez d'énergie à la place de stationnement la plus proche du barycentre des points d'origines de ces requêtes.

Q 38. Proposer au moins deux idées pour optimiser l'autonomie énergétique des véhicules (recharge). Quels véhicules recharger au dépôt et quand ?

> Toute idée plausible est recevable. Par exemple :
>
> - Recharger tous les véhicules la nuit, et donc commencer la journée avec une charge maximale de tous les véhicules.
> - Lorsqu'un véhicule se rapproche du dépôt dans son itinéraire et qu'il y a de la place dans les bornes de recharge, annuler le reste de ses requêtes et l'envoyer en recharge.
> - Éviter d'associer des requêtes qui emmènent les véhicules les moins chargés loin du dépôt (car ils ne pourront pas en prendre d'autres et doivent rentrer se recharger).

> L'idée générale est de minimiser les parcours à vide pour aller se recharger

Q 39. Proposer une idée pour le positionnement des bornes de recharge ailleurs qu'au dépôt.

> Toute idée plausible est recevable.
>
> On peut calculer l'historique du manque à gagner dû au manque d'énergie des véhicules (qui ont dû rentrer se recharger alors qu'il y avait des requêtes proches qu'ils auraient pu desservir). Placer les bornes de recharge dans des endroits proches de la position des requêtes concernées.

D.Commentaires du jury

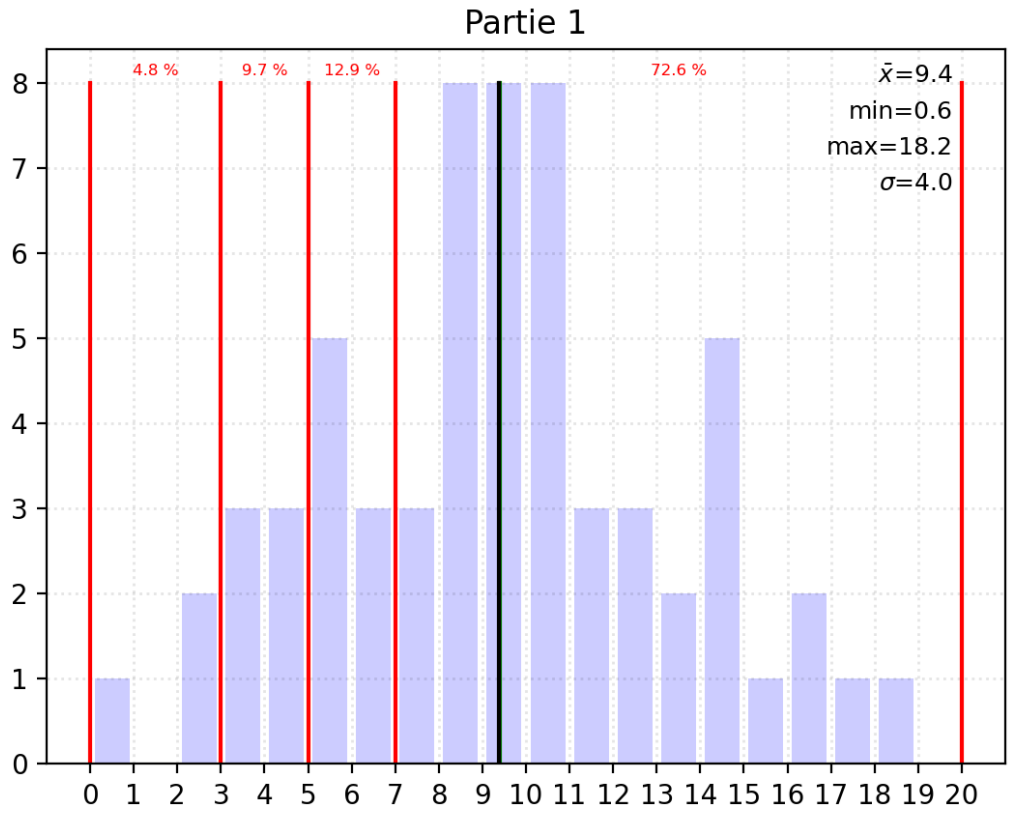
Ce sujet était composé de trois parties indépendantes et traite d'une société fictive louant les services d'une flotte de véhicules autonomes.

La première partie proposait tout d'abord aux candidat.e.s d'étudier le système d'information clients, et la gestion de codes d'accès signés pour accéder à des véhicules. L'objectif de cette partie était de vérifier les connaissances des candidat.e.s en matière de système d'information : conception d'une base de données et requêtes, stockage des informations sensibles (mots de passe). Les candidat.e.s étaient amenés à utiliser des connaissances en conception de bases de données, d'une part, et devaient d'autre part exploiter des documents techniques sur l'encodage utilisé dans les codes QR ainsi que sur les principes de la signature numérique HMAC 256.

Une première adaptation du système d'information, pour prendre en compte plusieurs moyens de paiement (ajout de tables) a été correctement réalisée par la plupart des candidat.e.s, qui ont aussi pu produire des requêtes simples (simple projection, sans jointure). Les requêtes plus complexes, nécessitant l'utilisation de plusieurs tables, dont certaines en double, ont été généralement mal traitées. Il n'est pas rares que certaines jointures soient manquantes alors que le produit cartésien est correctement formé. Enfin dans le cas de requête un peu longues, la syntaxe des requêtes est souvent approximative (alternance de WHERE et JOIN au sein d'une même requête)

Quelques éléments sur la compréhension du stockage des éléments d'authentification (mot de passe) étaient demandés, suite à la lecture de documents sur le sujet (hachage et salage). Les candidat.e.s ont généralement pu s'approprier ces connaissances, et en proposer une synthèse correcte. La réalisation technique des procédure d'authentification et stockage des mots de passe (Python) a été moins bien réussie que la partie compréhension.

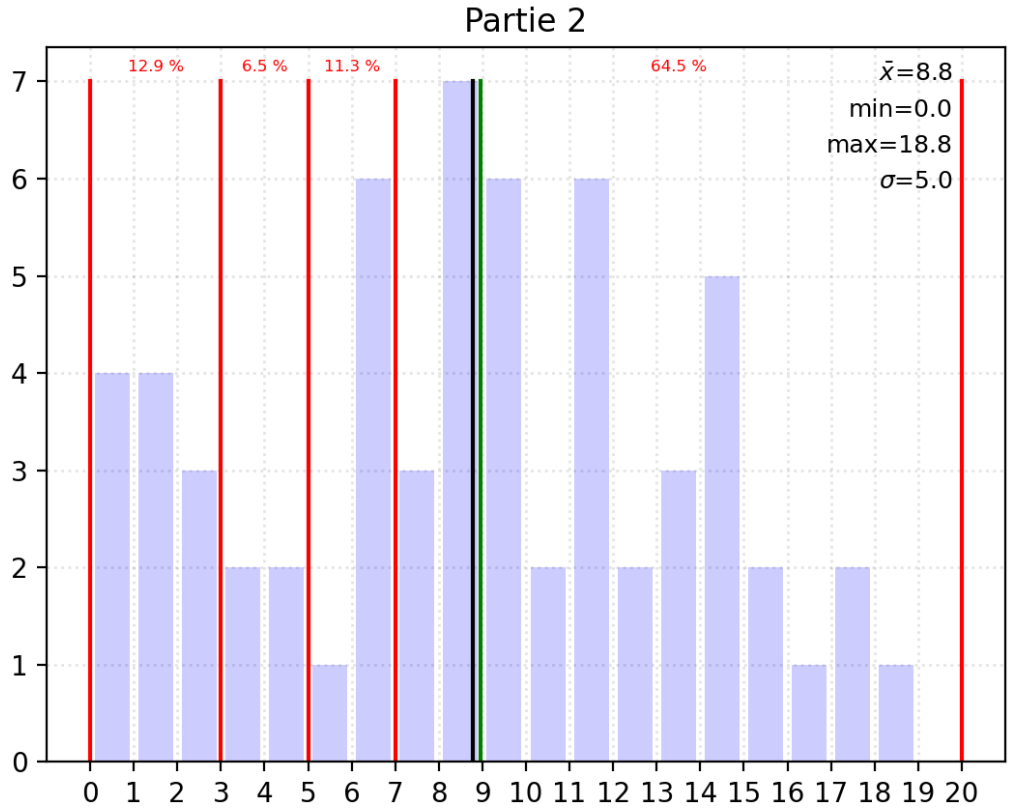
Enfin, il était demandé aux candidat.e.s, en s'appuyant sur des documents fournis, de comprendre les mécanismes de signature de données (HMAC 256) ainsi que l'encodage des données signées dans des codes QR (utilisation de base45). Un certain nombre de fonctions (Python) réalisant ces opérations devaient être produites ou complétées. L'objectif était de vérifier la maîtrise du langage Python, ainsi que l'adaptation à un sujet *a priori* inconnu à partir de documents techniques. Le passage en base45, quoi que documenté, a généralement comporté des erreurs (extraction des valeurs des chiffres en base 45) qui le rendraient inopérant, bien que la structure générale du code ait le plus souvent été correcte.



Dans le deuxième partie, il était demandé aux candidat.e.s d'analyser certaines parties des couches plus basses d'un véhicule : trames GPS d'une part, système d'adressage et contenu des trames GeoNetworking ainsi qu'un des algorithmes de routage GeoNetworking.

L'objectif de cette partie était de vérifier la connaissance des candidat.e.s sur l'encodage de trames (GPS ou GeoNetworking), ainsi que les capacités des candidat.e.s à produire du code C (dans le cadre de l'analyse de trames GPS). Les notions de routage étaient aussi abordées.

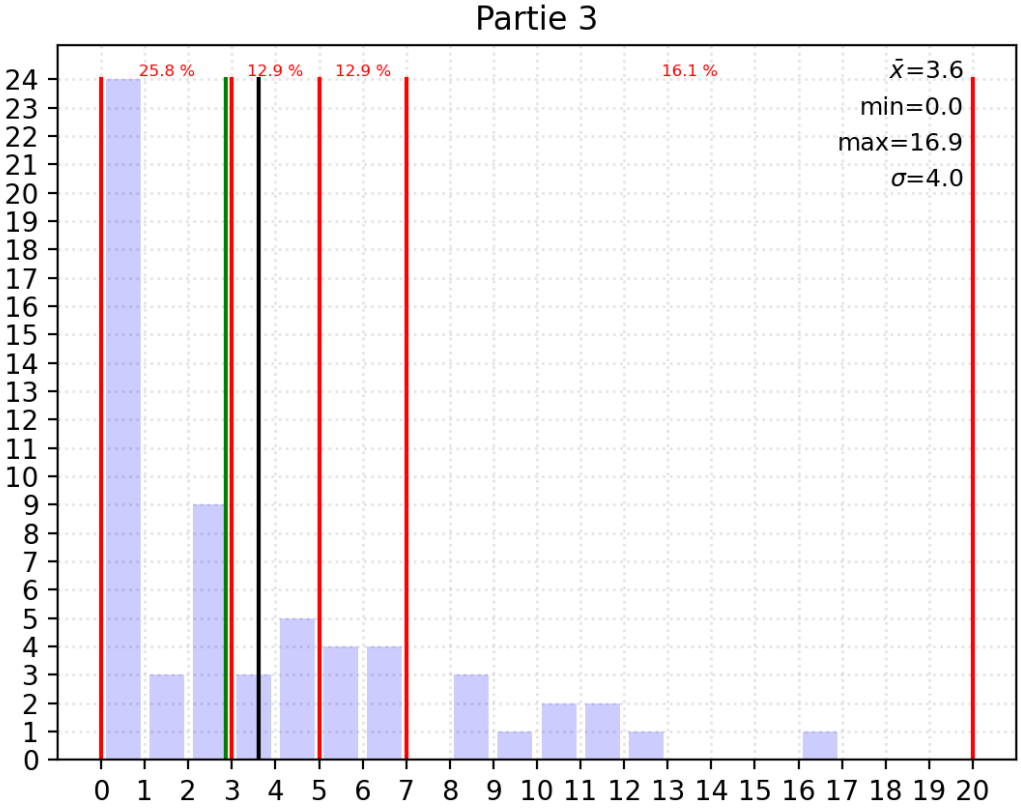
Cette partie a été globalement réussie, même si les question demandant du recul (Pourquoi cet algorithme est qualifié d'algorithme glouton ? Pour quelle raison GeoNetworking est utilisé à la place d'IP ?) n'ont pas donné lieu à des réponses satisfaisantes.

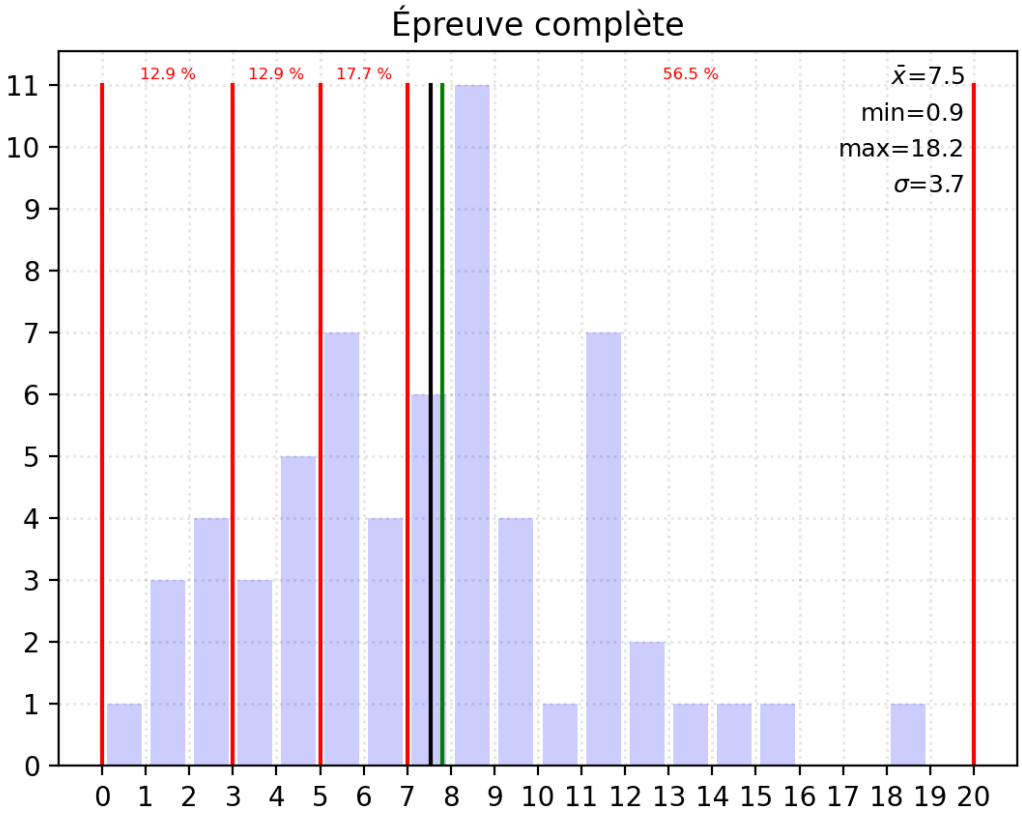


Enfin, la dernière partie, orientée algorithmique, avait pour objet de proposer des solutions aux multiples problèmes de gestion des missions des véhicules d'une flotte complète, avec pour objectif de minimiser les trajets et de respecter des contraintes horaires, tout en s'adaptant aux aléas des retards et des annulations.

L'objectif de cette partie était de vérifier la maîtrise par les candidat.e.s des algorithmes sur les graphes, de la résolution de problèmes, de la pensée critique, de la créativité et de la capacité à proposer des solutions innovantes pour améliorer l'efficacité et la durabilité du système de transport innovant proposé.

Certains candidat.e.s ont pu comprendre la problématique, visiblement par analogie à d'autres problèmes d'optimisation classiques comme le voyageur de commerce et les tournées de véhicules. Ces candidat.e.s ont généralement pu proposer des solutions pertinentes et parfois innovantes pour améliorer l'efficacité du système. En revanche, les candidat.e.s qui n'ont pas pu rentrer dans la problématique n'ont pas réussi à proposer des solutions, ou alors ont proposé des solutions qui ne conviennent pas du tout.


Partie 3
25.8 %
12.9 %
12.9 %
16.1 %
$\bar{x}$=3.6
min=0.0
max=16.9
σ=4.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24


Épreuve complète
12.9 %
12.9 %
17.7 %
56.5 %
$\bar{x}$=7.5
min=0.9
max=18.2
σ=3.7
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Épreuve d'admission d'exploitation pédagogique d'une activité pratique relative à l'approche globale d'un système pluritechnologique

A. Présentation de l'épreuve

Textes de référence

http://www.devenirenseignant.gouv.fr/cid98734/les-epreuves-de-l-agregation-externe-section-sciences-industrielles-de-l-ingenieur.html

Arrêté du 24 juin 2019 modifiant l'arrêté du 28 décembre 2009 fixant les sections et les modalités d'organisation des concours de l'agrégation

- Durée totale 6 heures (activités pratiques 4 heures, préparation de l'exposé 1 heure, exposé 30 minutes maximum, entretien 30 minutes).
- Coefficient 2.
- 10 points sont attribués à la partie liée aux activités pratiques et 10 points à la partie liée à l'exposé et à l'entretien avec le jury.

L'épreuve fait appel à des connaissances technologiques et scientifiques communes à l'ensemble des options.

Le candidat est amené au cours de cette épreuve à élaborer **une séquence pédagogique dont le contexte est imposé**. Il doit notamment y intégrer le développement d'une séance à caractère expérimental. Elle sera construite autour d'activités pratiques proposées par le candidat, sur un support didactique imposé.

Pour la session 2023, l'exploitation pédagogique demandée est relative aux enseignements non spécifiques de la spécialité ingénierie, innovation et développement durable du cycle terminal sciences et technologies de l'industrie et du développement durable (STI2D) ou de l'enseignement des sciences de l'ingénieur du lycée général et des classes préparatoires aux grandes écoles.

Le support didactique fourni est un système pluritechnologique qui permet une analyse systémique globale.

Au cours de l'entretien, le candidat est conduit à préciser certains points de sa présentation. Il est amené à expliquer et justifier les choix didactique et pédagogique qu'il a opérés notamment dans l'élaboration de la séquence de formation présentée ainsi que pour les contenus de la séance à caractère expérimental conçue.

Déroulement de l'épreuve

L'objectif de la première épreuve d'admission de l'agrégation de SII est de permettre d'évaluer chez les candidats leurs compétences pour s'inscrire dans la démarche d'un agrégé de sciences industrielles de l'ingénieur capable d'élaborer une exploitation pédagogique à partir d'une activité pratique relative à l'approche globale d'un système pluritechnologique.

Le titre d'une séquence pédagogique étant imposé, le candidat doit d'abord préparer la trame détaillée de celle-ci en respectant le niveau de formation visé et les effectifs de la classe. Il doit ensuite répondre à une problématique technique et scientifique comprenant des activités pratiques. Le candidat prépare enfin une séance à caractère expérimental s'inscrivant dans la séquence imposée. **Les activités expérimentales proposées doivent être différentes de celles déjà effectuées pour répondre à la**

problématique technique et scientifique et être adaptées au niveau de formation visé précisé dans l'énoncé du sujet.

Les compétences attendues par le jury sont pédagogiques, comportementales et scientifiques.
Le candidat doit montrer ses aptitudes à :

- concevoir, organiser et décrire une séquence dans un contexte pédagogique imposé ;
- s'approprier un système réel ou un équipement et son environnement ;
- élaborer, justifier, conduire et exploiter un protocole expérimental ;
- analyser le comportement d'un système à partir d'un modèle ;
- maitriser, conduire et exploiter une simulation numérique ;
- formuler des conclusions pour choisir et décider ;
- mener des démarches avec rigueur et évoluer avec autonomie.

L'évaluation du candidat s'effectue en trois phases.

Phase 1 – Conception et organisation d'une séquence de formation à un niveau imposé (durée 4h00)
Cette première phase d'une durée totale de 4h00 compte quatre parties.
Elle se déroule dans un laboratoire où sont mis à disposition du candidat un support d'étude, un environnement numérique de travail connecté à Internet, des moyens de mesure ou de simulation et si besoin des logiciels spécifiques d'acquisition.

- **Première partie (durée 0h45) – Réflexions pédagogiques sur la séquence imposée**

Pour cette première partie, le candidat doit réfléchir et proposer une séquence de formation parmi deux qui lui sont proposées. Pour chacune d'entre elles, le contexte pédagogique est imposé. Ce dernier est composé :

- du titre d'une des deux séquences imposées ;
- du niveau de formation visé ;
- d'une proposition de progression didactique liée à la formation visée ;
- du programme du niveau de formation visé ;
- d'une liste non exhaustive de supports matériels pédagogiques d'un laboratoire de Sciences de l'Ingénieur.

Le candidat doit recenser les compétences à développer, en intégrant les savoir-faire et savoirs du programme du niveau imposé en lien avec le titre d'une des deux séquences proposées au choix. Puis il doit proposer une trame détaillée de celle-ci (activités, durée, coordination). Les pré-requis de la séquence doivent être identifiés vis-à-vis de la progression didactique proposée et présentée. Le candidat doit justifier ses choix pédagogiques et didactiques (TP, TD, cours, projet...). L'ensemble de ces éléments doit être rédigé sur un support de présentation numérique, qui sera présenté et évalué lors de la troisième phase.

- **Deuxième partie (durée 0h30) – Prise en main du support**

Pour cette deuxième partie, les manipulations proposées ont pour objectif de faciliter la compréhension du fonctionnement global du système. À la fin de cette première partie, l'examinateur s'assure que le candidat s'est bien approprié le support de TP. L'objectif de cette partie est de faire émerger une problématique technique et scientifique à résoudre.

- **Troisième partie (durée 2h00) – Expérimentations pour répondre à une problématique technique et scientifique imposée**

Pour cette partie, le candidat doit répondre aux activités à caractère expérimental proposées afin de résoudre la problématique technique et scientifique, par la mobilisation de compétences caractéristiques du niveau de l'agrégation. L'exploitation des résultats obtenus (hypothèses, modèles, résultats expérimentaux, valeurs numériques…), la mise en évidence des écarts entre les performances souhaitées, les performances mesurées et les performances simulées et la proposition de solutions pour les réduire doivent permettre d'apporter une réponse aux problèmes posés.

- **Quatrième partie (durée 0h45) – Élaboration du scénario d'une séance à caractère expérimental**

Pour cette quatrième partie, le candidat doit décrire une séance à caractère expérimental s'insérant dans la séquence pédagogique en :

- situant la séance à caractère expérimental dans sa proposition de séquence pédagogique ;
- précisant l'organisation matérielle et pédagogique de la séance (nombre d'élèves, systèmes utilisés, travail en îlots,…) ;
- décrivant la (ou les) démarche(s) pédagogique(s) retenue(s) (démarche d'investigation, de résolution de problème technique, de projet,...) ;
- détaillant le scénario des activités que doivent réaliser les élèves ;
- proposant et en mettant en œuvre au moins un protocole expérimental différent de ceux qu'il a effectués dans la troisième partie ;
- explicitant clairement l'apport de la séance proposée dans le développement des compétences des élèves.

Pendant toute la durée de cette partie, le candidat a accès aux logiciels de simulation, au système et aux matériels de travaux pratiques. Le candidat doit donc entreprendre de réaliser de nouvelles simulations ou expérimentations utiles pour étayer et créer la trame de sa séance. Les examinateurs n'évaluent pas durant cette partie mais sont disponibles en tant qu'assistant technique.

Phase 2 – Préparation de l'exposé (durée 1h00)

Le candidat prépare son intervention devant le jury permanent en complétant son support de présentation numérique. Le candidat n'a plus accès au matériel de travaux pratiques, c'est-à-dire, ni au système, ni aux modèles associés, ni aux logiciels de simulation, mais conserve à sa disposition l'ensemble des ressources associées au sujet. Il dispose d'un poste informatique connecté à Internet et doté des logiciels courants de bureautique, et des résultats obtenus lors de la phase précédente qu'il aura stockés dans un espace dédié sur un serveur.

Phase 3 – Exposé oral et entretien avec le jury en salle (durée 1h00)

Le candidat a à sa disposition un tableau, un ordinateur et un vidéoprojecteur pour la présentation devant le jury.

L'exposé du candidat devant le jury a une durée de 30 minutes maximum sans intervention du jury. L'exposé doit comporter :

- la description du contexte pédagogique imposé ;
- la présentation de ses réflexions pédagogiques et la justification de ses choix de modalités pédagogiques ;
- la présentation de la trame de la séquence pédagogique en y intégrant l'évaluation ;
- la présentation des savoir-faire et savoirs à transmettre dans chaque séance ;
- la justification de la pertinence du support didactisé dans un contexte pédagogique (durée maximale 5 minutes) ;
- la démarche mise en œuvre dans la séance à caractère expérimental ;
- la présentation d'une ou des activités que devraient mener les élèves durant la séance d'activités à caractère expérimental ;

- la présentation de la valeur ajoutée pédagogique dans la formation de la séance proposée.

Il est à noter que durant la présentation des travaux devant le jury, il n'est absolument pas attendu des candidats qu'ils présentent à nouveau les résultats aux activités menées dans le cadre des deuxième et troisième parties de la phase 1. En effet, ceux-ci ont déjà conduit à une évaluation par le jury en salle de TP. Seule est attendue la présentation des activités envisagées de faire réaliser aux élèves lors de la séance à caractère expérimentale incluse dans la séquence pédagogique exposée. Néanmoins les résultats expérimentaux ou de simulation numérique peuvent être utilisés afin d'illustrer la séquence ou la séance expérimentale.

L'exposé du candidat est suivi d'un entretien avec le jury d'une durée de 30 minutes.

Le jury est amené à interroger les candidats, afin d'apprécier leur connaissance des principes fondamentaux du système éducatif et du cadre réglementaire de l'école, sur la manière dont ils envisagent d'accompagner les élèves dans leur parcours de formation, ou bien sur leur positionnement au sein de la communauté éducative.

Au cours de l'entretien, les candidats sont amenés à :
- préciser certains points de leurs présentations ;
- expliciter et justifier les choix de nature didactique et/ou pédagogique qu'ils ont opérés.

Utilisation des logiciels pendant l'interrogation
Aucun pré-requis ne peut être exigé du candidat concernant l'utilisation d'un logiciel. Les consignes d'utilisation sont indiquées dans le sujet ou fournies oralement. Les modeleurs volumiques ne sont pas utilisés comme outil de conception de formes mais comme un outil de lecture de documents.

Les supports retenus lors de la session 2023 étaient les suivants :
- volet roulant ;
- système de travelling ;
- imprimante 3D ;
- axe linéaire didactisé ;
- robot d'assistance à la chirurgie laparoscopique ;
- ventilation mécanique contrôlée double flux ;
- système de déplacement de caméra ;
- attelle de remobilisation du genou.

Ces supports ont permis aux candidats de mettre en œuvre leurs compétences à haut niveau scientifique sur les activités suivantes :
- élaboration et mise en œuvre d'un protocole expérimental ;
- identification des comportements de constituants ou d'un système ;
- mesure de comportement de constituants ou d'un système ;
- détermination des paramètres significatifs d'une chaîne de mesure ;
- détermination des paramètres significatifs d'une chaîne d'information ;
- détermination des paramètres significatifs d'une chaîne de puissance ;
- détermination des paramètres significatifs d'une modélisation ;
- analyse d'algorithmes simples ou de quelques lignes de programmes simples (en langage python, arduino, etc) ;
- recalage d'un modèle multiphysique ou non ;
- choix des modèles de comportement ou de connaissance ;
- validation de modèles ;
- simulation et prédiction de performance ;
- évaluation des écarts.

B. Commentaires du jury

- **Analyse des résultats**

Les candidats préparés mobilisent à bon escient leurs compétences pour répondre à la problématique pédagogique demandée. Le déroulement de la séquence pédagogique est structuré et cohérent. Ils positionnent convenablement la séance à caractère expérimental en présentant de nouvelles activités pratiques qu'ils ont réalisées durant la quatrième partie de la première phase. Les pré-requis, les objectifs, les démarches pédagogiques et d'évaluation sont bien assimilés et correctement décrits lors de l'exposé oral. Ces candidats ont généralement produit une présentation orale de qualité. La conduite des expérimentations pour répondre à la problématique technique et scientifique est traitée par la majorité des candidats.

Certains candidats présentent une séquence pédagogique qui ne respecte pas le contexte imposé, se plaçant ainsi hors sujet. Lors de l'exposé oral, quelques candidats présentent le système et les résultats obtenus pendant la troisième partie de la première phase, or ce ne sont pas les attendus de l'épreuve. De même, les activités pratiques réalisées pendant la troisième partie de la première phase sont souvent reprises dans la séance, alors que le jury en attend de nouvelles. L'explication de la pertinence du système, dans le cadre de la séance expérimentale proposée est souvent oubliée.

- **Commentaires sur les réponses apportées et conseils aux futurs candidats**

Phase 1 – Première partie : réflexions pédagogiques sur la séquence imposée
Le jury constate que :

- les déroulements des séquences sont souvent imprécis et peu approfondis ;
- les compétences visées sont peu ciblées ;
- le choix des stratégies pédagogiques mises en œuvre est rarement pertinent et justifié ;
- l'évaluation est souvent absente de la séquence.

Le jury attend une séquence pédagogique structurée en lien avec une thématique sociétale. Elle doit faire apparaître les pré-requis, les compétences et connaissances associées, le positionnement temporel, le déroulement des différentes séances la constituant et l'évaluation adéquate.
Il est proposé au candidat le choix entre deux séquences pédagogiques associant des compétences différentes d'un même niveau. Cette possibilité laissée au candidat est bien appréhendée. Pour tous les sujets, l'une et l'autre des propositions ont été traitées.
Les outils et méthodes de l'ingénierie pédagogique doivent être connus et maîtrisés. Le jury ne peut se satisfaire d'un exposé de pédagogie formel ou d'une récitation d'un extrait de programme. Il souhaite qu'il soit fait preuve d'imagination et de créativité dans le contenu pédagogique présenté afin de susciter l'intérêt et la motivation des élèves.

Pour les futures sessions, le jury conseille aux candidats d'étudier préalablement et attentivement les programmes et les objectifs des formations dont peuvent être issus les contextes pédagogiques imposés : enseignement de spécialité « sciences de l'ingénieur », enseignements technologiques de spécialités du cycle terminal STI2D et enseignement « sciences de l'ingénieur » des CPGE. Cette étude, ainsi que la lecture des documents « ressources pour faire la classe » et des guides d'équipement, leur permettront de proposer une exploitation pédagogique en adéquation avec le niveau

imposé. Une réflexion pédagogique sur les objectifs de formation de ces séries et classes post-bac est indispensable pour réussir cette partie de l'épreuve.

Le jury engage les candidats à clairement indiquer la ou les démarches pédagogiques qui structureront l'organisation pédagogique retenue (démarche d'investigation, démarche de résolution de problème technique, démarche scientifique ou encore démarche de projet technologique).

Phase 1 – Deuxième partie : prise en main du support

Pour cette partie, les manipulations ainsi que les activités proposées ont pour objectif de faciliter la compréhension du fonctionnement global du système, de s'approprier le support du travail pratique et la problématique technique et scientifique proposée. Les candidats disposent d'un dossier technique, d'un dossier ressource, ainsi que diverses ressources numériques. Le système proposé au candidat peut être le système réel ou un système didactisé.

Les manipulations proposées sont très guidées de sorte que le candidat peut rapidement appréhender l'environnement logiciel et matériel du support. Certains candidats se trouvent en difficulté dès cette phase de prise en main. Le jury leur recommande de se confronter plus régulièrement à la manipulation de systèmes réels et/ou didactisés.

Phase 1 – Troisième partie : expérimentations pour répondre à une problématique technique et scientifique imposée

Pour cette phase, le jury tient à porter à l'attention des candidats les points suivants :

- la maîtrise du raisonnement scientifique et la caractérisation des échanges d'énergie, de matière et d'information à un niveau de généralités permettent de s'adapter à une large diversité de systèmes ;
- l'extraction des informations pertinentes dans les ressources mises à disposition constitue un préalable indispensable à l'appropriation du système et de la problématique ;
- la problématique scientifique et technique doit être comprise afin d'y répondre. Elle permet d'appréhender correctement le fil directeur des activités et manipulations proposées ;
- les analyses externes et internes des systèmes gagnent en pertinence lorsqu'elles sont appuyées sur des outils formalisés (schéma des chaînes de puissance et d'information, diagrammes SysML) ;
- une bonne culture personnelle pluritechnologique, fondée sur l'observation et l'analyse de systèmes variés et modernes, est indispensable.

Le candidat est amené à :

- utiliser une instrumentation spécifique dédiée à la mesure de grandeurs physiques sur les systèmes instrumentés ;
- mettre en œuvre différents outils informatiques (logiciels de pilotage et/ou d'acquisition dédiés aux supports, logiciels de simulation, modeleur, logiciel de calculs par éléments finis, tableurs, traitements de textes, logiciels de calcul ou de visualisation, environnements de programmation…).

Le jury assiste le candidat en cas de difficultés matérielles ou de mise en œuvre des différents outils informatiques. La maitrise de ces logiciels n'est pas exigée.

Lors des activités pratiques, le jury souhaite que les candidats s'attachent à :

- lire et analyser l'ensemble du sujet proposé ;
- maitriser la durée consacrée à chaque activité ;
- maitriser les outils d'analyse courants (structurels, fonctionnels et comportementaux) ;

- exploiter et interpréter l'ensemble des résultats des expérimentations et des mesures dans leur totalité et de façon rigoureuse ;
- corréler les résultats des simulations et des expérimentations en les associant à des phénomènes physiques et à des solutions technologiques ;
- effectuer une analyse critique des résultats expérimentaux ;
- vérifier la cohérence et la pertinence des résultats expérimentaux ;
- mettre en œuvre une démarche de résolution du problème technique et scientifique posé ;
- réfléchir à de nouvelles activités pratiques pouvant nourrir la séance expérimentale qui sera présentée.

Le jury précise que les supports de travaux pratiques sont principalement issus des laboratoires SI, STI2D, CPGE et couvrent l'ensemble des champs disciplinaires transversaux des sciences industrielles de l'ingénieur.

Phase 1 – Quatrième partie : élaboration du scénario d'une séance à caractère expérimental
Le jury constate que :

- cette partie est la plus délicate pour un grand nombre de candidats ;
- la séance à caractère expérimental n'intègre pas toujours des activités pratiques différentes de celles réalisées lors de la troisième partie de la première phase ;
- les activités proposées aux élèves sont peu détaillées.

Le candidat doit développer une séance expérimentale contextualisée, positionnée avec pertinence dans la séquence pédagogique proposée. Il est attendu la description des activités proposées aux élèves. La configuration pédagogique choisie doit être justifiée.
Les manipulations et protocoles de mesures insérés dans la séance doivent être adaptés au niveau requis. Ils doivent être différents de ceux réalisés lors de la troisième partie « expérimentations » tout en utilisant le système de l'épreuve. Des modalités d'évaluation doivent également être présentées et justifiées pédagogiquement.

Phase 3 – Exposé oral et entretien avec le jury en salle
La majorité des candidats n'utilise pas le temps imparti pour la présentation de 30 minutes. L'utilisation de la durée prévue leur permettrait de préciser leurs réflexions pédagogiques trop souvent formatées.
Il n'est pas attendu des candidats la présentation des activités réalisées lors de la première phase de cette épreuve, déjà évaluées au cours de la première phase.

Le jury attend lors de cette phase de présentation de la séquence pédagogique que le candidat soit capable de :

- présenter le contexte pédagogique imposé ;
- situer la séquence de formation en l'inscrivant dans la formation au niveau requis ;
- expliciter les connaissances et les compétences visées par la séquence ;
- décrire le déroulement de la séquence ;
- situer la séance expérimentale dans la séquence pédagogique ;
- expliciter les connaissances et compétences visées dans la séance ;
- présenter la pertinence du système pour les activités pratiques de la séance ;
- définir l'enchaînement des activités réalisées par les élèves dans la séance ainsi que les résultats attendus ;
- justifier les choix pédagogiques retenus ;
- présenter les moyens de l'évaluation des connaissances et compétences acquises par les élèves ou étudiants.

Le jury attend également du candidat qu'il mette en œuvre des compétences professionnelles telles que :

- produire un discours clair, précis et rigoureux en sachant attirer l'attention du jury ;
- être pertinent et réactif aux questions posées ;
- être capable de dégager l'essentiel, de donner du sens aux connaissances développées et de captiver l'auditoire.

Le jury conseille aux candidats qui n'en auraient pas eu la possibilité au cours de leur formation, de prendre contact avec un établissement scolaire dispensant les filières de formation visées par le concours et de s'y déplacer afin de prendre connaissance des réalités matérielles, humaines et organisationnelles du terrain.

Comportement des candidats

Les candidats doivent être méthodiques et rigoureux pour appréhender un système pluritechnologique dans sa globalité et dans sa complexité. L'exploitation pédagogique d'une activité pratique relative à l'approche globale et transversale d'un système pluritechnologique ne s'improvise pas. Elle doit se préparer tout au long des formations conduisant à l'agrégation. Les candidats doivent éviter les présentations stéréotypées ne permettant pas de mettre en valeur la qualité de leur réflexion personnelle. Les contenus scientifiques des séquences doivent être maitrisés par les candidats, l'accès à Internet étant toujours possible sur des sites publics.

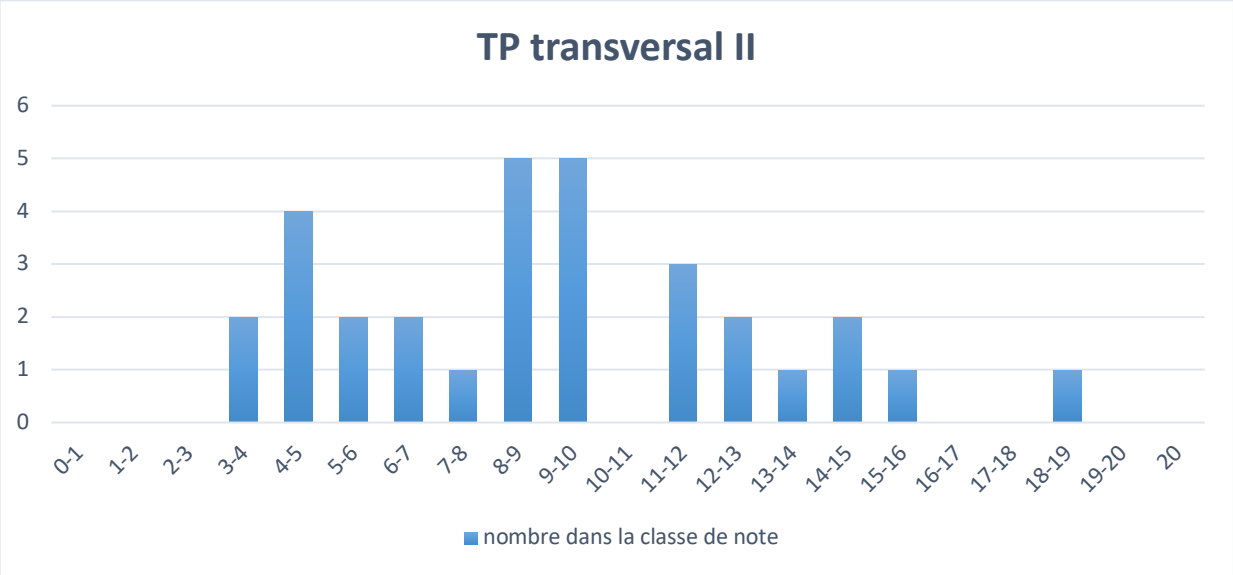
Le jury apprécie lorsque les diapositives sont numérotées lors de la présentation orale : cela lui permet de se référer plus facilement aux éléments de présentation. Le jury déplore les fautes d'orthographe dans les supports numériques présentés ; elles nuisent à l'image du candidat. Les candidats se présentant au concours de l'agrégation se destinent à être de futurs cadres de l'éducation nationale. Ils se doivent d'avoir un vocabulaire, un comportement et une tenue en adéquation avec le métier d'enseignant.

Conclusion

La session 2023 de l'agrégation externe SII confirme que l'usage d'un laboratoire unique, composé de systèmes pluritechnologiques, commun aux quatre options de l'agrégation de sciences industrielles de l'ingénieur, impose au candidat de s'approprier tous les champs disciplinaires transversaux liés au triptyque « matière, énergie et information ». Au-delà même de la juxtaposition des savoirs pluridisciplinaires indispensables pour réussir cette épreuve, cette session met aussi en évidence toute l'importance, pour un candidat, d'être réellement apte à appréhender les systèmes dans leur globalité. Enfin, les compétences en ingénierie pédagogique attendues lui imposent une connaissance approfondie des différents programmes, des objectifs de formation associés et des stratégies pédagogiques préconisées. Une préparation prenant en compte les recommandations du rapport de jury est nécessaire à la réussite de cette épreuve.

C. Résultats

La moyenne des notes obtenues est de 9,12/20. L'écart-type est 3,75. La meilleure note est 18,01/20. La plus faible est 3,67/20. La médiane est 8,88/20.



Exemple de sujet pour l'exploitation pédagogique d'une activité pratique relative à l'approche globale d'un système pluritechnologique

Robot porte-laparoscope

Phase 1 – Conception et organisation d'une séquence de formation à un niveau imposé (durée : 4h00)

Partie 1.1 – Conception de l'architecture de la séquence de formation imposée (0h45)

Objectifs : s'approprier le besoin pédagogique imposé par le jury et concevoir l'architecture de la séquence de formation.

Contexte pédagogique de la séquence de formation imposée

La séquence pédagogique à construire est associée à un des deux contextes pédagogiques suivants, à choisir par le candidat :

	Choix 1	**Choix 2**
Titre de la séquence	Analyser, modéliser et résoudre pour vérifier les performances cinématiques des transmetteurs de mouvement.	Analyser, modéliser et résoudre pour valider le comportement temporel des Systèmes Linéaires Continus Invariants.
Niveau de formation visé	Première année PCSI – CPGE – Enseignement de Sciences Industrielles de l'Ingénieur	
Supports pédagogiques	Les supports suivants sont disponibles dans le laboratoire de sciences de l'ingénieur. Ils sont choisis judicieusement pour répondre au besoin pédagogique de la séquence imposée : - Portail automatisé ; - Robot cueilleur de fruits (MaxPid) ; - Simulateur de course ; - Pilote hydraulique de bateau ; - Robot haptique ; - Direction assistée électrique ; - Plateforme 6 axes ; - Robot porte-laparoscope ; - Télescope Astrolab.	
Effectif	Classe de 40 à 42 élèves, groupe à effectif réduit de 20-21 élèves.	
Volume horaire	4 heures hebdomadaires (1 h cours + 1 h TD + 2h TP).	

Les documents suivants sont fournis et accessibles dans le dossier « contexte pédagogique » :

- le programme du niveau de formation visé (fichier Programme-pcsi_sii.pdf) ;
- une proposition de séquences adaptée au niveau de formation visé (fichier Progression didactique PCSI-PSI.xlsx).

Production attendue

Une architecture de séquence pédagogique doit être proposée en s'assurant de la cohérence, de la faisabilité et de la pertinence des choix effectués après avoir :

- contextualisé la séquence pédagogique dans une grande thématique ;
- recensé les compétences à développer et les savoir-faire et savoirs à faire acquérir aux élèves ;
- identifié les prérequis et le positionnement temporel de la séquence dans une progression pédagogique (vis-à-vis de la proposition de liste de séquences fournie) ;

- spécifié les modalités pédagogique et didactique (TP, TD, cours, projet, évaluation, remédiation, ...), leurs coordinations et leurs organisations.

Partie 1.2 – prise en main du support didactisé (durée : 0h30)

Objectif : s'approprier l'environnement et la structure du support didactisé du laboratoire.

Le candidat dispose des éléments suivants :
- un robot porte-laparoscope Evolap ;
- les logiciels ServeurEvolap (déjà lancé à l'arrivée du candidat) et ClientEvolap installés sur l'ordinateur.

Le système présent dans le laboratoire est un robot porte-laparoscope utilisé pour remplacer un assistant lors des opérations chirurgicales. Un ensemble de capteurs équipe le robot et permet différentes acquisitions.
Un dossier ressources est fourni sous forme papier et numérique. Il comprend notamment :
- la définition du contexte d'utilisation du robot ;
- la présentation documentée des principaux constituants du produit (documents constructeurs,…) ;
- les protocoles expérimentaux pour les parties 1.2 et 1.3 de l'épreuve.

Un logiciel de pilotage et d'acquisition installé sur le poste informatique permet, entre autres :
- de réaliser des acquisitions et voir la zone de travail à l'aide de la caméra équipée en bout de laparoscope ;
- de piloter le robot selon deux axes en boucle ouverte ou asservis en vitesse ;
- de traiter des résultats expérimentaux.

Prendre connaissance de l'annexe « 1 - Principe de la laparoscopie » du dossier ressources.

Activité 1 Compléter le document réponse 1 permettant d'identifier les constituants du système.

Prendre connaissance du protocole expérimental n°1 fourni dans le dossier ressources.

Activité 2 Mettre en œuvre le protocole expérimental n°1. Conclure quant à la possibilité pour le chirurgien de mouvoir le laparoscope et les outils simultanément à l'aide des images fournies par la caméra.

Prendre connaissance de l'annexe « 2 - Chaînes fonctionnelles du porte-laparoscope » du dossier ressources.

Activité 3 Situer sur le système du laboratoire les différents constituants de la chaîne de puissance et d'information fournis sur le document réponse 2, et compléter les éléments manquants. Repérer sur le système réel les angles α, β, θ et φ.

Partie 1.3 – expérimentations pour répondre à la problématique technique et scientifique (durée : 2h00)

Problématique technique et scientifique : Comment le robot porte-laparoscope permet-il d'assister le chirurgien durant une opération chirurgicale ?

Analyse et mise en œuvre du suivi automatique de l'instrument chirurgical

Analyse du mode de fonctionnement souhaité

Prendre connaissance de l'annexe « 3 - Diagramme d'états » dans le dossier ressources.

Activité 4 Analyser le diagramme d'états, et identifier le mode mis en œuvre dans le protocole n°1. Décrire qualitativement le comportement attendu en « Mode suivi ».

Mise en œuvre du suivi automatique de l'instrument chirurgical

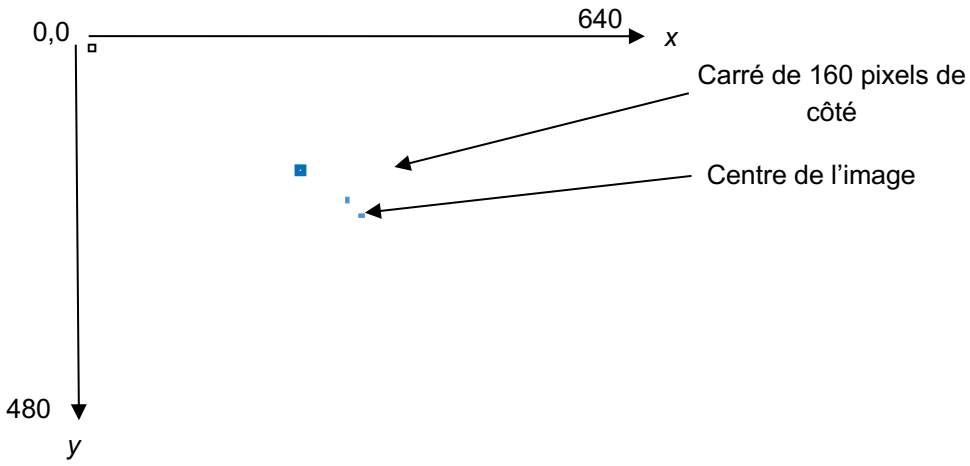
En situation d'usage dans une salle d'opération, le mode de pilotage du porte-laparoscope est le mode suivi automatique. Ce mode consiste à positionner le laparoscope automatiquement par rapport à l'instrument chirurgical grâce au robot porte-laparoscope. L'objectif de cette partie est de proposer un algorithme pour ce suivi automatique de l'instrument chirurgical.

Prendre connaissance du protocole expérimental n°2 fourni dans le dossier ressources.

Activité 5 Mettre en œuvre le protocole expérimental n°2 et décrire le comportement observé.

La suite de cette partie consiste à réaliser le suivi automatique de l'instrument chirurgical, c'est-à-dire, le pilotage des 2 actionneurs en fonction de la position de l'instrument chirurgical dans l'image acquise par la caméra. Les dimensions de l'image sont de 640x480 pixels (640 est la dimension de l'image selon l'horizontale et 480 est la dimension de l'image selon la verticale).

La logique de commande des actionneurs du robot porte-laparoscope retenue est décrite ci-dessous :
- le centre de la zone rouge de l'instrument chirurgical est repéré à partir de l'image fournie par la caméra ;
- lorsque celui-ci est en dehors d'un carré de 160 pixels de côté construit autour du centre de l'image, l'algorithme doit permettre de le ramener dans le carré.



La logique de commande est implantée dans la carte de commande et est programmée en langage Python, par le programme « programme_suivi.py » commenté en fin de dossier ressources.
La fonction « `ma_fct` » à compléter est contenue dans une boucle exécutée à intervalle de temps régulier. Lorsque le laparoscope doit être mis en mouvement, l'amplitude de la commande des mouvements est prise égale à 5.
Les noms des variables et fonctions à utiliser sont :
- `pos_g` est une liste à deux dimensions contenant les coordonnées du barycentre de la surface rouge détectée automatiquement par la caméra. Pour accéder à la coordonnée de ce point selon l'axe horizontal, la syntaxe à utiliser est `pos_g[0]` (et `pos_g[1]` pour la coordonnée selon l'axe vertical) ;
- `mvt_gd(vitesse, output)` est une fonction permettant de déplacer le centre de l'image captée par la caméra de gauche à droite à la vitesse désirée. Ainsi pour :
  - déplacer le laparoscope vers la gauche, l'instruction est `mvt_gd(5, output)` ;
  - déplacer le laparoscope vers la droite, l'instruction est `mvt_gd(-5, output)` ;
  - l'arrêter, il faut utiliser une vitesse nulle en argument ;
- `mvt_hb(vitesse, output)` est une fonction permettant de déplacer le centre de l'image captée par la caméra de haut en bas à la vitesse désirée. Ainsi pour :
  - déplacer le laparoscope vers le bas, l'instruction est `mvt_hb(5, output)` ;

- déplacer le laparoscope vers le haut, l'instruction est `mvt_hb(-5, output)` ;
- l'arrêter, il faut utiliser une vitesse nulle en argument.

Prendre connaissance du protocole expérimental n°3 fourni dans le dossier ressources.

Activité 6 Réaliser le protocole expérimental n°3 fourni dans le dossier ressources et valider expérimentalement le fonctionnement observé par rapport au comportement attendu.

À ce stade de l'étude, nous disposons d'une solution permettant de piloter le porte-laparoscope en mode suivi automatique de l'instrument chirurgical, à l'aide des fonctions `mvt_gd` et `mvt_hb`. Or, les actionneurs (motoréducteurs M1 et M2) ne permettent pas directement de piloter les mouvements haut-bas et gauche-droite.

L'objectif de la suite de cette partie consiste à déterminer la loi de commande des 2 motoréducteurs permettant de générer les mouvements haut-bas (variation de l'angle θ) et gauche-droite (variation de l'angle φ) du porte-laparoscope.

Analyse et mise en œuvre de la commande des actionneurs

Analyse de la commande des actionneurs

Prendre connaissance du protocole expérimental n°4 fourni dans le dossier ressources.

Activité 7 Mettre en œuvre le protocole expérimental n°4. Conclure en identifiant le ou les motoréducteurs à commander afin de faire évoluer l'angle θ, ainsi que le ou les actionneurs à commander pour faire varier l'angle φ. Valider les définitions des fonctions `mvt_gd` et `mvt_hb` du fichier Python « programme_suivi.py ».

Analyse et validation d'un modèle de simulation de la chaîne de puissance

Activité 8 Ouvrir dans Matlab/Simulink le modèle « Evolap_Act8.slx ». Analyser le modèle complet en proposant une correspondance entre les éléments modélisés et le système réel.

Activité 9 Lancer des simulations avec des vitesses de consigne des motoréducteurs M1 et M2, telles que définies ci-dessous :

- consignes de vitesse angulaire des 2 motoréducteurs égales à 20 rad/s (à saisir dans le champ Final Value) ;
- consigne de vitesse angulaire du motoréducteur 2 égale à 20 rad/s et consigne de vitesse angulaire du motoréducteur 1 nulle.

Visualiser, pour chaque essai ci-dessus, les évolutions des angles θ et φ en fonction du temps. Conclure quant à la pertinence du modèle complet vis-à-vis de la commande identifiée à l'activité 7.

Activité 10 Analyser, à partir des simulations, le découplage entre les évolutions des grandeurs angulaires θ et φ vis-à-vis de la commande des motoréducteurs. Valider cette analyse par observation sur le système réel. Conclure sur les conséquences dans le cadre du suivi de l'outil chirurgical en cas de non-découplage.

Élaboration d'un modèle de simulation de la chaîne de puissance

Activité 11 Ouvrir dans Matlab/Simulink le modèle « Evolap_Act11.slx ». Compléter le modèle associé au bloc « Différentiels » (n'utiliser qu'un seul des blocs mis à disposition), et valider le modèle.

À ce stade de l'étude, nous disposons de la loi de commande des motoréducteurs afin de mouvoir le laparoscope en mode suivi d'un instrument chirurgical. L'objectif de la fin de cette partie consiste à analyser la commande en vitesse des 2 motoréducteurs afin de s'assurer de la netteté des images fournies par la caméra située à l'extrémité du laparoscope.

Étude de la commande de vitesse des motoréducteurs

Une commande en boucle ouverte de chaque motoréducteur est envisagée dans un premier temps.

Analyse et mise en œuvre de la commande en boucle ouverte des motoréducteurs

Prendre connaissance du protocole expérimental n°5 fourni dans le dossier ressources.

Activité 12 Compléter le document réponse 3 en précisant le nom des composants manquants. Mettre en œuvre le protocole expérimental n°5, et commenter les résultats obtenus. Conclure quant à la possibilité d'utiliser une commande en boucle ouverte sur le robot porte-laparoscope.

Analyse et mise en œuvre de la commande en boucle fermée des motoréducteurs

Prendre connaissance du protocole expérimental n°6 fourni dans le dossier ressources.

Activité 13 Mettre en œuvre le protocole n°6. Conclure quant à la nécessité de mettre en œuvre un asservissement de vitesse angulaire de chaque motoréducteur.

Activité 14 Ouvrir dans Matlab/Simulink le modèle « Evolap_Act8.slx » et double cliquer sur le bloc « Moteur M1 asservi en vitesse angulaire ». Analyser et décrire les différents blocs utilisés dans l'asservissement de vitesse angulaire. Réaliser une simulation sur 1s et observer l'évolution de la position angulaire des motoréducteurs en double cliquant sur le scope. Après avoir effectué une prise d'origine, réaliser l'expérience de l'activité 13 sur le robot porte-laparoscope, et comparer les performances en vue de valider le modèle de simulation.

Synthèse

Activité 15 À la lumière de toutes les activités réalisées dans ce TP, répondre à la problématique technique et scientifique.

Partie 1.4 – élaboration du scénario d'une séance à caractère expérimental (durée : 0h45)

Objectif : développer une séance à caractère expérimental s'intégrant dans la séquence pédagogique proposée dans la partie 1.1.

Production attendue

Une séance à caractère expérimental pertinente doit être proposée après avoir :

- situé cette séance dans la séquence pédagogique (objectifs et prérequis) ;
- décrit l'organisation matérielle et pédagogique de la séance (nombre d'élèves, systèmes utilisés, travail en îlots ou autres) ;
- décrit et justifié la (ou les) démarche(s) pédagogique(s) retenue(s) (démarche d'investigation, de résolution de problème technique, de projet ...) ;
- détaillé le scénario des activités que doivent réaliser les élèves sur le support didactisé à l'aide des documents fournis ci-après ;
- réalisé concrètement au moins une des activités expérimentales proposées dans la séance développée. Cette activité doit être nouvelle et différente de celles réalisées dans la partie 1.3. Préciser l'objectif de la manipulation entreprise, proposer et mettre en œuvre son protocole expérimental comme le feraient les élèves et analyser les résultats obtenus ;
- explicité clairement l'apport de la séance proposée dans le développement des savoir-faire et compétences des élèves.

Phase 2 – préparation de l'exposé (durée : 1h00)

Objectif : finaliser le support de présentation pour l'exposé devant le jury.

Production attendue

Un document numérique doit être réalisé afin de :

- présenter la séquence pédagogique ;
- présenter la pertinence du support didactisé par rapport au besoin pédagogique ;
- présenter la séance à caractère expérimental.

Phase 3 – exposé oral et entretien avec le jury en salle (durée : 1h00)

Épreuve d'admission d'activité pratique et d'exploitation pédagogique relative à l'approche spécialisée d'un système pluritechnologique

A. Présentation de l'épreuve

Texte de référence

http://www.devenirenseignant.gouv.fr/cid98734/les-epreuves-de-l-agregation-externe-section-sciences-industrielles-de-l-ingenieur.html

- Durée totale : 6 heures (activités pratiques : 4 heures, préparation de l'exposé : 1 heure, exposé : 30 minutes maximum, entretien : 30 minutes maximum)
- Coefficient 2

10 points sont attribués à la première partie liée aux activités pratiques et 10 points à la seconde partie liée à la leçon.

Le support de l'activité pratique proposée permet, à partir d'une analyse systémique globale, l'analyse d'un problème technique particulier relatif à la spécialité du concours dans l'option choisie. La proposition pédagogique attendue, directement liée aux activités pratiques réalisées, est relative aux enseignements technologiques de spécialité du cycle terminal "sciences et technologies de l'industrie et du développement durable (STI2D)" du lycée et des programmes de BTS et BUT des champs couverts par l'option choisie.

L'épreuve a pour but d'évaluer l'aptitude du candidat à :

- mettre en œuvre des matériels ou équipements, associés si besoin à des systèmes informatiques de pilotage, de traitement, de simulation, de représentation ;
- conduire une expérimentation, une analyse de fonctionnement d'une solution, d'un procédé, d'un processus, dans la spécialité du concours, afin d'analyser et de vérifier les performances d'un système technique ;
- exploiter les résultats obtenus et formuler des conclusions ;
- concevoir et organiser une séquence de formation pour un objectif pédagogique imposé à un niveau de classe donné et présenter de manière détaillée un ou plusieurs points-clés des séances de formation qui la constituent. Elle prend appui sur les investigations et les analyses effectuées au préalable par le candidat au cours d'activités pratiques relatives à un système technique.

Le candidat est amené au cours de sa présentation orale à expliciter sa démarche méthodologique, à mettre en évidence les informations, les données et les résultats issus des investigations conduites au cours des activités pratiques qui lui ont permis de construire sa proposition pédagogique.

Au cours de l'entretien, le candidat est conduit plus particulièrement à préciser certains points de sa présentation ainsi qu'à expliquer et justifier les choix de nature didactique et pédagogique qu'il a opérés dans la construction de la séquence de formation présentée.

Déroulement de l'épreuve

Cette épreuve comporte trois phases.

Phase 1 – Manipulation expérimentale au laboratoire (durée 4h00)

Cette phase, d'une durée totale de 4 heures, se déroule en trois parties dans le laboratoire où sont mis à disposition les différents supports techniques qui permettent à chaque candidat de proposer une séquence pédagogique. Cette dernière s'appuie sur les activités pratiques réalisées par le candidat.

Première partie (durée ≈ 0h30)
Dans cette partie de prise en main du système, les études et activités proposées au candidat ont pour objectif de faciliter la compréhension de l'architecture globale du système et de son fonctionnement. À la fin de cette première partie, le jury s'assure que le candidat s'est bien approprié le support de l'activité pratique ainsi que la problématique proposée.

Deuxième partie (durée ≈ 2h00)
Dans cette partie d'approfondissement, le candidat doit suivre les études et les activités proposées permettant de répondre à la problématique de l'activité pratique. Cette partie doit permettre au candidat, par la mobilisation de compétences et de connaissances caractéristiques du niveau de l'agrégation, de développer/intégrer des modules logiciels, d'intégrer du code, de déverminer un programme, de résoudre les problèmes posés, puis d'en exploiter les résultats obtenus (modèles, représentation UML/SysML, algorithmes, jeu de tests, interface graphique, résultats numériques, etc).

Troisième partie (durée ≈ 1h30)
Dans cette partie, le candidat dispose librement du support de TP pour préparer la trame détaillée de sa séquence pédagogique. En s'appuyant sur les développements, les investigations et les analyses effectués durant les deux premières parties ou d'autres éléments qu'il a la possibilité de concevoir, le candidat doit proposer un ou plusieurs protocoles expérimentaux lui permettant de répondre à la fois à la problématique scientifique et à la problématique pédagogique.

Cette phase 1 se déroule dans le laboratoire dans lequel se trouve le support de TP utilisé.
L'exploitation pédagogique est relative à l'enseignement spécifique du cycle terminal :
- en lycée, de la voie technologique sciences et technologies de l'industrie et du développement durable de la spécialité système d'information et numérique (STI2D SIN) ;
- en post bac, des programmes du BTS systèmes numériques (options : informatique et réseaux, électronique et communication) et des BUT génie électrique et informatique industrielle, réseaux et télécommunication et informatique relatifs aux champs couverts par l'option ingénierie informatique.

Les candidats disposent de l'ensemble des moyens nécessaires à l'expérimentation et d'un poste informatique, relié à Internet, doté des logiciels courants de bureautique et des logiciels plus spécifiques liés au sujet qui leur est proposé.

Phase 2 – préparation de la présentation (durée 1h00)
Durant cette phase d'une heure, le candidat s'appuie sur la trame de la séquence qu'il a construite dans la troisième partie de la phase 1 et prépare l'intervention qu'il fera devant le jury. Il dispose d'un poste informatique relié à Internet doté des logiciels courants de bureautique et de tous les résultats de mesures, analyses ou investigations issus de la phase 1.

Phase 3 – présentation des travaux devant le jury (durée 1h00)
L'exposé oral est d'une durée maximale de 30 minutes. Le jury n'intervient pas pendant l'exposé du candidat. L'entretien avec le jury est d'une durée maximale de 30 minutes.
Le candidat est amené au cours de sa présentation orale à présenter :
- le système (durée maximale 5 minutes) ;
- une synthèse des activités menées dans la deuxième partie de la première phase de l'activité pratique (durée maximale 5 minutes) ;
- son exploitation pédagogique (durée maximale 20 minutes).

Au cours de l'entretien, le candidat est amené à :
- préciser certains points de sa présentation ;
- expliquer et justifier les choix de nature didactique et pédagogique.

Pour la présentation devant le jury, les candidats ont à leur disposition un tableau, un ordinateur et un vidéoprojecteur. Ils disposent d'un poste informatique doté des logiciels courants de bureautique, connecté à Internet et des résultats obtenus lors des phases 1 et 2, stockés dans l'espace qui leur est dédié.

Systèmes proposés

Pour la session 2023, les systèmes proposés sont parmi les suivants :
- drone instrumenté communicant ;
- objets connectés ;
- réception et traitement de données issues de modules météorologiques ;
- robot pédagogique 4 axes ;
- système de contrôle d'accès RFID ;
- détection et localisation de tumeurs cérébrales par traitement d'image IRM.

Ces études permettent aux candidats de mettre en œuvre leurs compétences à un haut niveau scientifique associées aux tâches suivantes :
- modélisation de systèmes (UML/SYSML …) ;
- analyse critique et validation de modèles (de calcul, de données, de logiciels, d'échange) et proposition de modifications ;
- programmation en langages C/C++, Java, Python, SQL ;
- déverminage de programmes et traces des variables ;
- configuration, déploiement et test de réseaux informatiques ;
- intégration et utilisation de bibliothèques logicielles (traitement d'images, interface graphique, protocoles réseaux, base de données …) ;
- développement d'algorithmes spécifiques et prise en compte de leurs performances.

B. Commentaires du jury

Compétences attendues par le jury

Lors de cette épreuve d'activités pratiques, le jury évalue chez les candidats les compétences suivantes :
- faire preuve d'ingénierie pédagogique en élaborant une séance d'activités pratiques dans une séquence pédagogique cohérente, inscrite dans une progression pédagogique ;
- proposer des activités pratiques ou des modélisations nouvelles, adaptées au niveau imposé et aux objectifs de formation ;
- s'approprier un support, un environnement de développement, un système d'exploitation ;
- s'approprier la problématique associée ;
- élaborer, conduire et justifier un algorithme, un développement et/ou un protocole de test ;
- exploiter et analyser des résultats de tests ;
- élaborer, justifier, analyser de manière critique un modèle ;
- évoluer en autonomie en mobilisant toutes ses connaissances et ses compétences ;
- présenter oralement ses travaux avec clarté, précision et rigueur ;
- être réactif et pertinent dans les réponses aux questions posées par les membres du jury.

Analyse des résultats

Concernant la phase 1, le jury considère que les candidats répondent favorablement aux attentes quant à leur capacité :
- à suivre des protocoles de tests en utilisant des logiciels variés et/ou en exécutant des programmes (console, analyseur de trames, déverminer) ;
- à écrire du code dans des langages variés (appels de fonctions, programmation orientée objet) ;

Cependant, durant cette phase 1, le jury a constaté que certains candidats rencontraient certaines difficultés :

- à analyser le système d'un point de vue de l'architecture logicielle et de l'infrastructure informatique ;
- à explorer rapidement les différentes parties d'un logiciel selon des méthodes de recherche intégrées dans de nombreux environnement de développement et de systèmes d'exploitation ;
- à concevoir et réaliser de nouveaux développements pour leur séquence pédagogique.
- à utiliser avec aisance le système d'exploitation GNU/Linux

Concernant la phase 3, la prestation orale des candidats lors de la présentation des travaux devant le jury se révèle souvent incomplète et/ou inconsistante pour les raisons suivantes :

- la présentation du système est souvent insuffisante et les représentations UML/SysML ne sont pas proposées ;
- l'analyse des résultats expérimentaux est trop souvent superficielle ;
- les temps de présentation préconisés ne sont pas toujours respectés (système, résultats expérimentaux, exploitation pédagogique) et les 30 minutes ne sont pas utilisées ;
- le choix des savoirs des séquences pédagogiques est très souvent cohérent mais le découpage en séances permettant d'évaluer les savoir-faire de manière progressive n'est pas maîtrisé ou proposé ;
- certains candidats présentent des activités « élèves » sans rapport avec les compétences visées ;
- l'exploitation pédagogique envisagée se limite trop souvent à proposer une séquence pédagogique incluant la seule activité de travaux pratiques conduite précédemment au laboratoire ;
- les moyens à mettre en œuvre compte tenu du nombre d'élèves ou d'étudiants présents dans la section considérée ne sont pas discutés et ne permettent pas de vérifier le réalisme de la solution pédagogique ;
- l'exploitation pédagogique n'utilise pas de manière pertinente le support proposé en travaux pratiques ou un autre support représentant des solutions techniques similaires, ce qui ne permet pas au jury d'apprécier la capacité du candidat à créer du contenu pédagogique à partir de systèmes réels ou didactiques ;
- le positionnement de la séquence pédagogique dans une progression pédagogique est très rarement proposé.
- Les prérequis de la séquence pédagogique présentée ainsi que les méthodes d'évaluation des compétences visées sont insuffisamment énumérés.

Conseils du jury

Pour réussir au mieux ces deux phases, le jury invite les candidats à maîtriser parfaitement l'utilisation des systèmes d'exploitation largement utilisés en ingénierie informatique (Windows, GNU/Linux, systèmes temps réel), les logiciels courants de bureautique (suite logicielle de conception de documents, utilitaires classiques de capture d'écran), les concepts de la programmation objet, leur modélisation en UML et leur implémentation dans les trois langages de programmation que sont le C++, Java et Python. Les futurs candidats doivent garder à l'esprit que cette agrégation demande des qualités dépassant largement le simple exercice de programmation.

Le jury conseille aux candidats d'étudier par exemple les logiciels de contrôle commande moderne qui sont très souvent construits en couches partant du bas niveau (capteur, microcontrôleur) jusqu'aux applications de haut niveau (contrôle d'une trajectoire, interface homme-machine) en passant par des couches intermédiaires (pilotes logiciels, système de messagerie, gestionnaire de tâches). Dans ces systèmes, les concepts de la programmation orientée objet sont utilisés afin de procurer à la solution informatique des caractéristiques pertinentes pour le domaine de l'ingénierie informatique.

Le jury attend que les candidats mettent à profit le temps dont ils disposent durant la troisième partie de la phase 1 (1h30) pour conduire d'autres développements (exemples de manipulations réalisées par les étudiants) que ceux demandés précédemment, sur lesquels ils pourront s'appuyer pour proposer ensuite une exploitation pédagogique originale et personnelle.

Contrairement à la première épreuve d'admission relative à l'approche globale d'un système pluritechnologique, le jury de la seconde épreuve d'admission relative à l'approche spécialisée d'un système pluritechnologique rappelle que le candidat doit présenter le système étudié, les expérimentations effectuées et les résultats obtenus. Le jury cherche à évaluer la clarté d'expression, les facultés du candidat à s'approprier, à synthétiser et à restituer avec précision et rigueur les contenus techniques et scientifiques exploités lors de la première partie.

Le jury rappelle qu'il s'agit pour le candidat de présenter une séquence pédagogique inscrite dans une progression pédagogique. Dans cette séquence, doit apparaître de façon précise un ou plusieurs points caractéristiques des séances de formation proposées. L'ensemble doit prendre appui sur les investigations et les analyses effectuées au préalable par le candidat au cours d'activités pratiques relatives au support proposé, en tenant compte du niveau de la formation visée et en adaptant l'exploitation du support à celui-ci.

Le jury rappelle que la conception d'une séquence pédagogique visant des savoir-faire et des savoirs doit suivre une méthodologie dont les étapes peuvent être les suivantes :

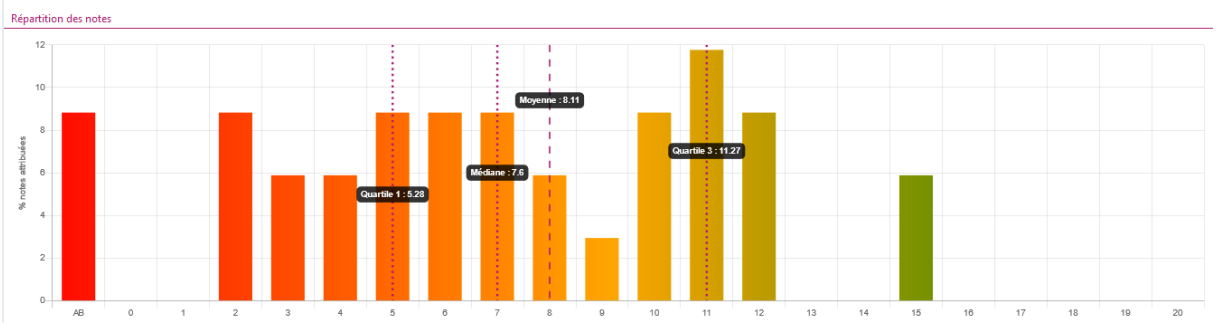
- le choix de l'objectif pédagogique qui doit s'appuyer sur un support technique représentatif des sciences de l'ingénieur. Ce dernier est imposé par le sujet de TP ;
- le choix des compétences visées par le candidat. Il n'est pas nécessaire de choisir l'ensemble des compétences définies dans le sujet de TP ;
- le choix des savoir-faire définis pour la ou les compétences choisies. L'attestation de la construction d'une compétence est obtenue en validant l'acquisition de tous les savoir-faire. Des indicateurs observables doivent permettre de les évaluer précisément ;
- le choix des savoirs associés aux savoir-faire (à positionner en prérequis, au lancement, en ressource durant les séances ou bien en synthèse) ;
- et enfin la répartition des savoir-faire et des savoirs dans chaque séance.

Le jury conseille aux candidats de préparer la partie pédagogique en utilisant les référentiels des diplômes cités dans cette épreuve.

Le jury incite également les candidats à l'agrégation à pousser la porte des lycées et des IUT ainsi que des INSPÉ ou des ENS, afin d'obtenir de la part d'enseignants de terrain ou formateurs, un maximum d'informations d'ordre pédagogique leur permettant de préparer au mieux cette épreuve.

C. Résultats

La moyenne des notes obtenues est de 8,11/20. L'écart-type est de 3,72. La meilleure note est 15,71/20 et la plus faible est 2,08/20.



AGRÉGATION DE SCIENCES INDUSTRIELLES DE L'INGÉNIEUR

TP4 : lunettes immersives

ÉPREUVE D'ADMISSION
Travaux pratiques de l'option ingénierie Informatique

PREMIÈRE PHASE : PREMIÈRE PARTIE (DURÉE ≈ 0H30)

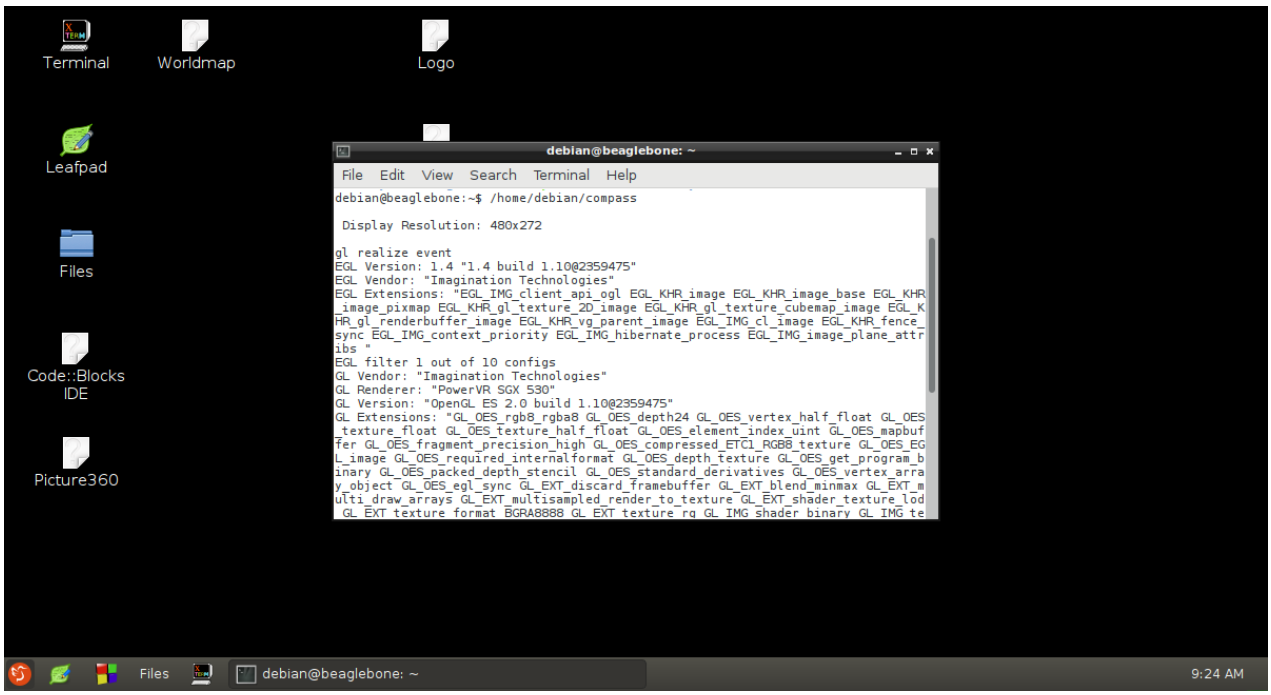
Remarque : cette partie propose trois manipulations guidées dont la seule fin est de prendre en main le support et les outils logiciels. L'analyse des résultats est proposée dans la seconde partie. Le candidat est invité à poser toute les questions relatives à la prise en main durant cette première demi-heure, de façon a être autonome par la suite.

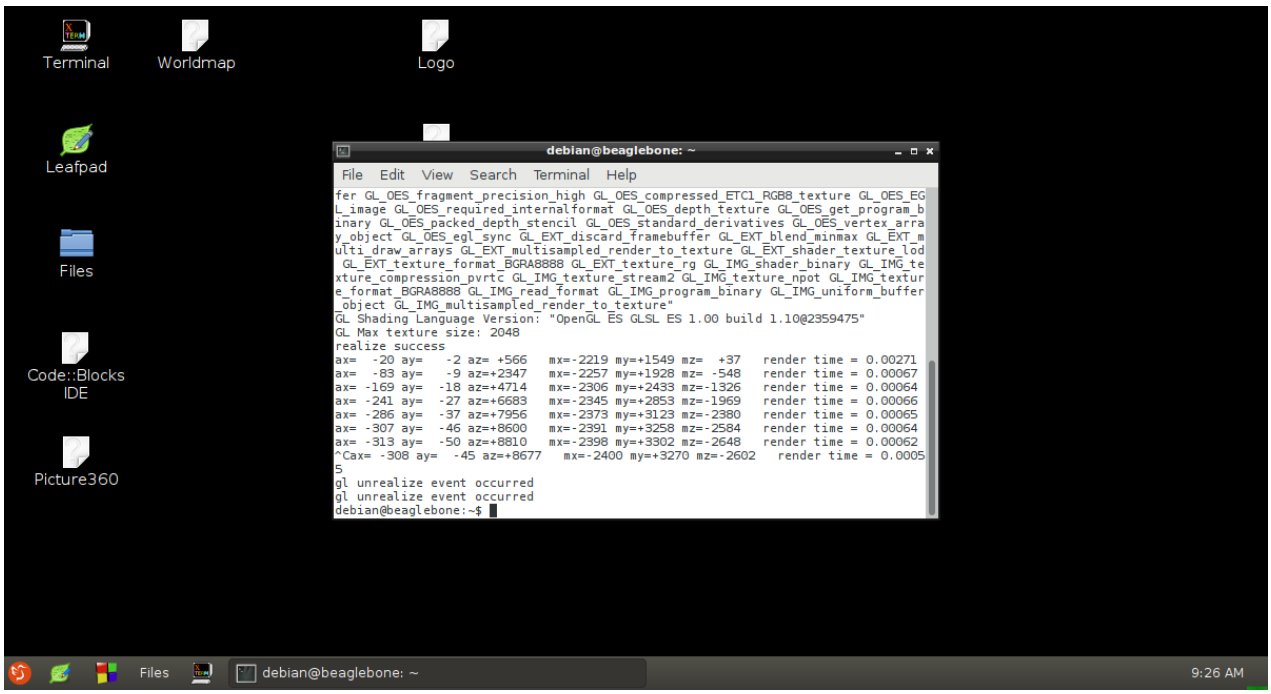
Première activité (durée ≈ 0h10)

Objectifs : lancer une application dans un terminal dans l'environnement graphique Linux, observer les valeurs brutes des capteurs

1. Démarrer l'application « compass » fournie à l'aide d'un terminal :

capture d'écran de la fenêtre candidat, les données de l'accéléromètre et du magnétomètre sont un peu plus bas dans la console après les messages d'initialisations



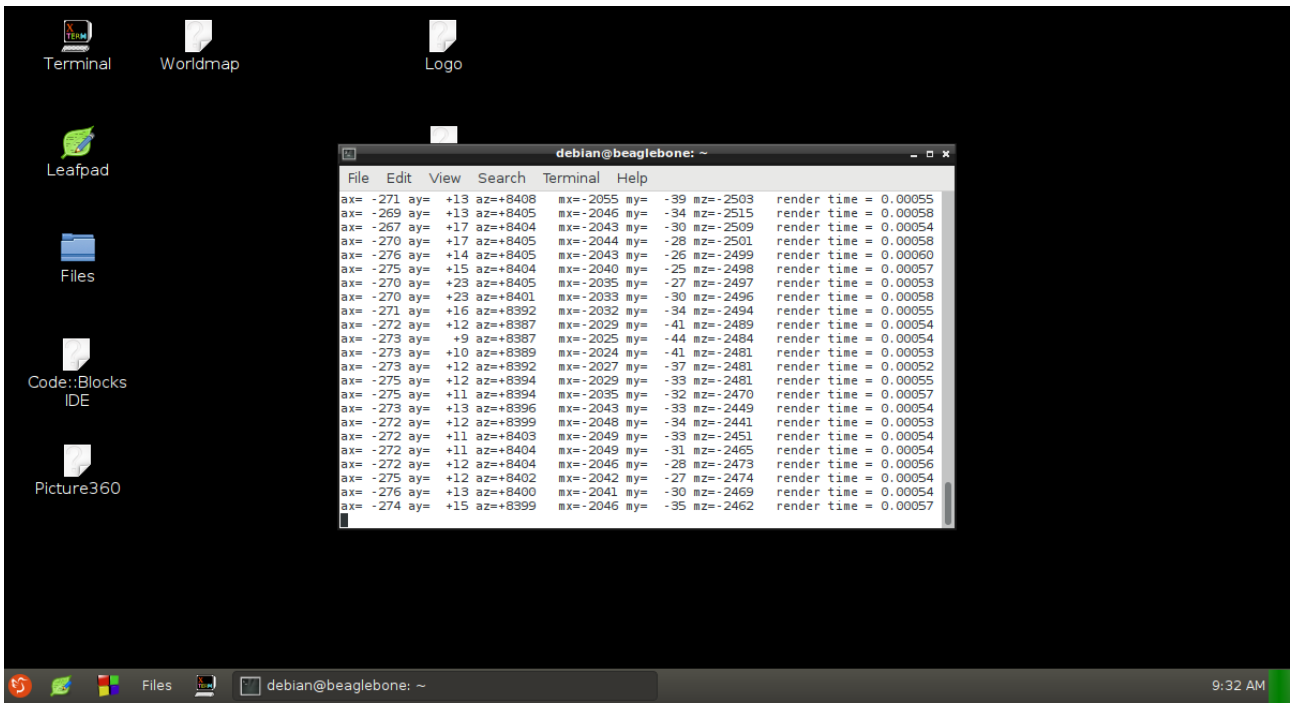


ax correspond à l'axe X de l'accéléromètre, ay son axe y, az son axe z…. mx, my,mz idem mais pour le magnétomètre

Deuxième activité (durée ≈ 0h05)

Objectifs : découvrir les ressources à disposition pour le TP (documentations, utilitaires de capture d'écrans)

2. Lorsque la carte est posée horizontalement sur la table, le nord vers le haut, réaliser une capture d'écran de l'écran LCD et de la console affichée pour cette application « compass ».



la variable my doit être très petite devant mx (composante selon x du champ magnétique terrestre) et devant mz (composante selon z du champ magnétique terrestre). A noter que les mesures sont cohérentes avec l'inclinaison moyenne à environ 45° du champ magnétique terrestre en France.

Troisième activité (durée ≈ 0h10)

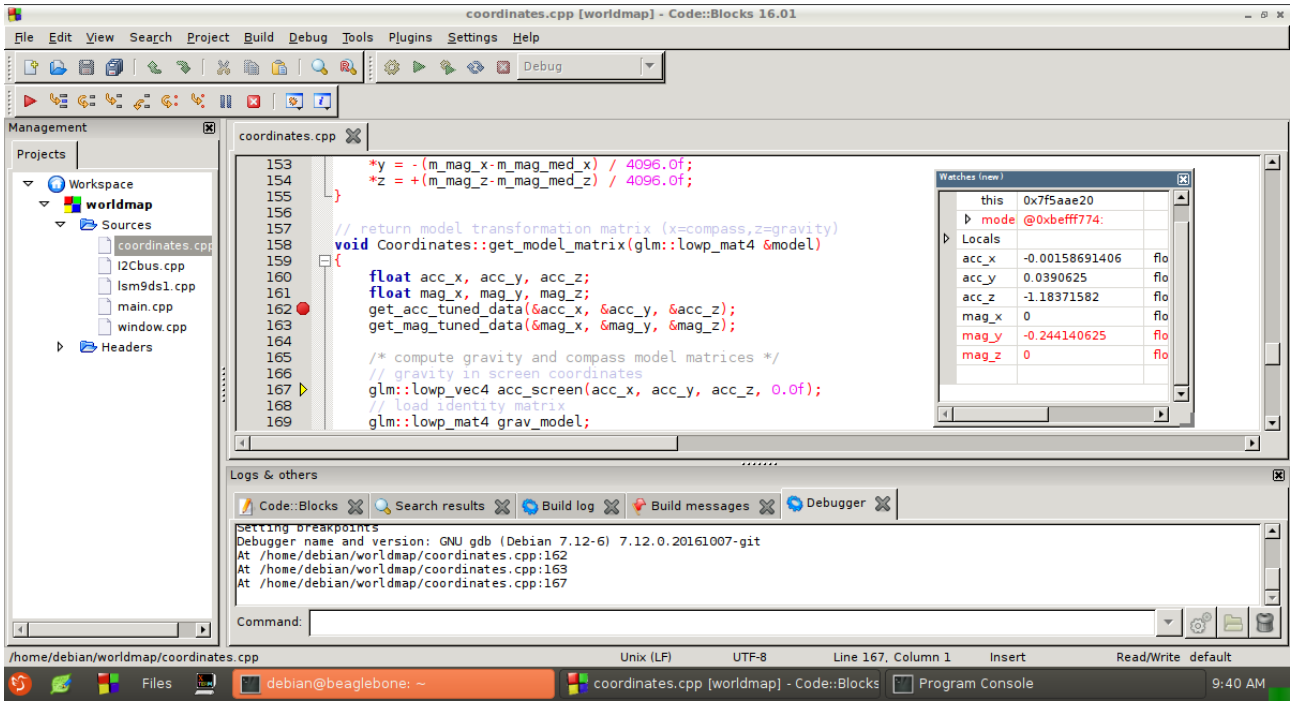
Objectifs : découvrir l'environnement de développement et les commandes de compilation et de débogage

3. Déboguer l'application « picture360 » fournie dans l'environnement CodeBlocks. Une capture d'écran est disponible en page suivante. Pour cela exécuter les tâches suivantes :

Toute l'accélération doit être selon « z » (acc_z environ -1), les valeurs de acc_x et acc_y sont très petites.

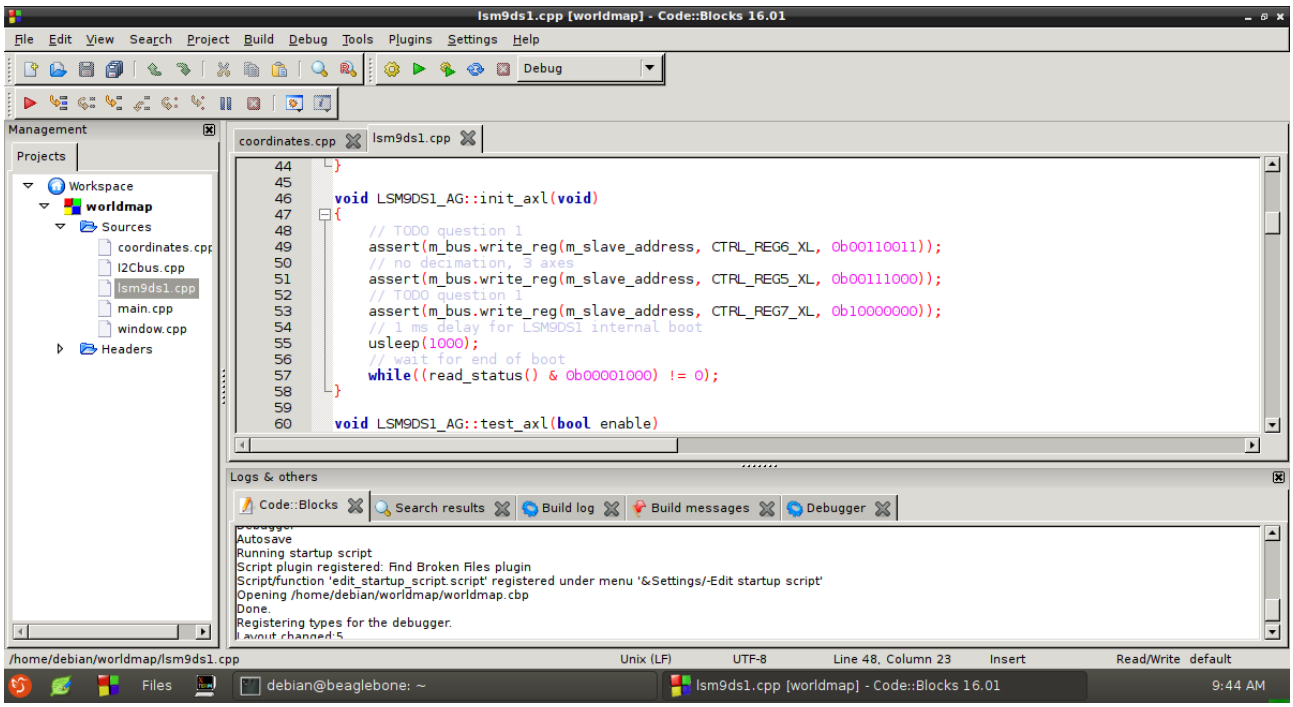
Le programme n'a conservé que la composante du vecteur « champ magnétique » normale au plan « xy » de la gravité. Il est normal d'obtenir des valeurs très petites de mag_x et mag_z.

Le champ magnétique est selon l'axe y de l'écran.


coordinates.cpp [worldmap] - Code::Blocks 16.01
File Edit View Search Project Build Debug Tools Plugins Settings Help
Debug
Management
Projects
Workspace
worldmap
Sources
coordinates.cpp
I2Cbus.cpp
lsm9ds1.cpp
main.cpp
window.cpp
Headers
coordinates.cpp
153 *y = -(m_mag_x-m_mag_med_x) / 4096.0f;
154 *z = +(m_mag_z-m_mag_med_z) / 4096.0f;
155 }
156
157 // return model transformation matrix (x=compass,z=gravity)
158 void Coordinates::get_model_matrix(glm::lowp_mat4 &model)
159 {
160 float acc_x, acc_y, acc_z;
161 float mag_x, mag_y, mag_z;
162 get_acc_tuned_data(&acc_x, &acc_y, &acc_z);
163 get_mag_tuned_data(&mag_x, &mag_y, &mag_z);
164
165 /* compute gravity and compass model matrices */
166 // gravity in screen coordinates
167 glm::lowp_vec4 acc_screen(acc_x, acc_y, acc_z, 0.0f);
168 // load identity matrix
169 glm::lowp_mat4 grav_model;
Watches (new)
this 0x7f5aae20
mode @0xbefff774:
Locals
acc_x -0.00158691406
acc_y 0.0390625
acc_z -1.18371582
mag_x 0
mag_y -0.244140625
mag_z 0
Logs & others
Code::Blocks
Search results
Build log
Build messages
Debugger
Setting breakpoints
Debugger name and version: GNU gdb (Debian 7.12-6) 7.12.0.20161007-git
At /home/debian/worldmap/coordinates.cpp:162
At /home/debian/worldmap/coordinates.cpp:163
At /home/debian/worldmap/coordinates.cpp:167
Command:
/home/debian/worldmap/coordinates.cpp
Unix (LF)
UTF-8
Line 167, Column 1
Insert
Read/Write
default
Files
debian@beaglebone: ~
coordinates.cpp [worldmap] - Code::Blocks
Program Console
9:40 AM

PREMIÈRE PHASE : DEUXIÈME PARTIE (DURÉE ≈ 2H00)

4. Localiser dans le fichier « lsm9ds1.cpp » le code réalisant l'initialisation de l'accéléromètre en réglant les registres CTRL_REG5_XL, CTRL_REG6_XL et CTRL_REG7_XL. D'après la valeur des bits actuellement écrits dans ces registres et la documentation du circuit LSM9DS1 fournie, déterminer les valeurs courantes de la fréquence d'échantillonnage et de la fréquence de coupure du filtre passe-bas.



Registre CTRL_REG5_XL : DEC=00 (no decimation), ZEN_XL=1, YEN_XL=1, XEN_XL=1 (3 axes activés)

Registre CTRL_REG6_XL : ODR=001 (10Hz), pleine échelle +-4g, BW_SCAL=0=408Hz, BW_XL=50Hz (c'est le filtrage analogique avant le convertisseur analogique-numérique).

Registre CTRL_REG7_XL : HR=1, DCF=00=ODR/50, FDS=0=digital filter disabled, HPIS1=0 (pas de filtre passe haut pour supprimer la composante statique de l'accélération).

Cette configuration par défaut est volontairement « foireuse », la fréquence d'échantillonnage (10Hz) est largement inférieure à la bande passante (50Hz), pas de filtre numérique avant la décimation.

5. Modifier le code et les commentaires associés afin de configurer l'échantillonnage à 952 Hz, le filtre anti-repliement à 408 Hz et le filtre passe-bas numérique à ODR/100 (donc une bande passante à 9,52 Hz).

// ODR=952Hz, full scale = 4g, automatic antialiasing bandwidth set to 50Hz

assert(m_bus.write_reg(m_slave_address, CTRL_REG6_XL, 0b11010011));

// no decimation, 3 axes

assert(m_bus.write_reg(m_slave_address, CTRL_REG5_XL, 0b00111000));

// high resolution enabled, filter set to ODR/100 about 10Hz, no high pass filter

assert(m_bus.write_reg(m_slave_address, CTRL_REG7_XL, 0b10100000));

6. A l'aide de la documentation fournie du LSM9DS1, des déclarations dans le fichier lsm9ds1.h et de la documentation de l'énumération LSM9DS1_M_Regs dans le fichier lsm9ds1.h, déterminer les 6 registres contenant les données du magnétomètre.

7. En vous inspirant du code existant dans la fonction LSM9DS1_AG::read_axl et dans la fonction LSM9DS1_M::read_offset_mag, réaliser l'implémentation de la méthode LSM9DS1_M::read_mag ; Valider expérimentalement son fonctionnement par l'utilisation des traces d'exécution ou du débogueur, valider l'orientation du nord magnétique.

```
void LSM9DS1_M::read_mag(int16_t *x, int16_t *y, int16_t *z)
{
    int8_t data[6];
    uint8_t unused;
    // read with autoincrement (+128)
    assert(m_bus.read_regs(m_slave_address, OUT_X_L_M + 128, (uint8_t*)data, 6, &unused));
    *x = (((int16_t)data[0]) & 0x00FF) | (((int16_t)data[1]) << 8);
    *y = (((int16_t)data[2]) & 0x00FF) | (((int16_t)data[3]) << 8);
    *z = (((int16_t)data[4]) & 0x00FF) | (((int16_t)data[5]) << 8);
}
```

Troisième étude (durée ≈ 20 minutes)

Objectif : charger des constantes de calibration depuis un fichier de configuration

Le magnétomètre a un biais de mesure (offset) à compenser. Les constantes de calibrations sont stockées dans le fichier « /home/debian/lsm9ds1.cal ».

8. La documentation de la classe KeyFile de la bibliothèque glibmm est disponible à l'adresse https://developer.gnome.org/glibmm/stable/classGlib_1_1KeyFile.html , un exemple d'utilisation de cette classe est présent dans la méthode « Coordinates::save_sensors_calibration » dans le fichier « coordinates.cpp », déterminer les méthodes de cette classe utiles pour cette opération.

Méthode « load_from_file » pour charger depuis un fichier

Méthode « get_integer » pour récupérer une valeur numérique entière dans la section spécifiée avec le nom spécifié

9. Compléter le code de la méthode « Coordinates::load_sensors_calibration » afin d'initialiser la valeur des attributs m_mag_med_x, m_mag_med_y et m_mag_med_z de cette classe. Valider expérimentalement le fonctionnement de ce code dans l'application complète.

```
// load sensor calibration
void Coordinates::load_sensors_calibration(void)
{
    // only magnetometer needs calibration
    Glib::KeyFile settings;
    settings.load_from_file("/home/debian/lsm9ds1.cal");
    // load existing calibration from setting file
    m_mag_med_x = settings.get_integer("mag","xmed");
    m_mag_med_y = settings.get_integer("mag","ymed");
    m_mag_med_z = settings.get_integer("mag","zmed");
}
```

Quatrième étude (durée ≈ 30 minutes)

Objectif : implémenter un filtrage des données issues du magnétomètre

Les signaux issus du magnétomètre sont particulièrement bruités et le périphérique ne dispose pas de filtre passe-bas intégré. L'application doit donc implémenter un filtre numérique d'équation récurrente $y_n = b_0 \cdot x_n + b_1 \cdot x_{n-1} + b_2 \cdot x_{n-2} - a_1 \cdot y_{n-1} - a_2 \cdot y_{n-2}$ dont les coefficients sont les suivants.

Coefficient	Valeur
b_0	+0,0674552739
b_1	+0,1349105478
b_2	+0,0674552739
a_1	-1,1429805025
a_2	+0,4128015981

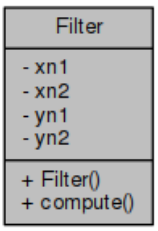
Ces coefficients ont été déterminés d'après une réponse de Butterworth, d'ordre 2, de fréquence de coupure de 2 Hz et une fréquence d'échantillonnage de 20 Hz.

Le filtrage doit être réalisé de manière indépendante sur les 3 axes du magnétomètre, l'écriture d'une classe instanciée une fois par axe permet d'obtenir rapidement la fonctionnalité voulue.

10. A l'aide du squelette de fichier « filter.h » fourni, proposer la déclaration d'une classe Filter dont le prototype de la fonction calculant l'équation récurrente est « float compute(float xn) », le paramètre x_n correspond à l'échantillon entrant dans le filtre, la valeur de retour correspond à la valeur calculée pour y_n. Déclarer dans cette classe les attributs nécessaires aux stockages des valeurs précédentes des échantillons sous la forme de nombres flottants.

```
class Filter
{
    private:
        /** place holder for x_n-1, value of the input at time-1 */
        float xn1; // x_n-1
        /** place holder for x_n-2, value of the input at time-2 */
        float xn2; // x_n-2
        /** place holder for y_n-1, value of the output at time-1 */
        float yn1; // y_n-1
        /** place holder for y_n-2, value of the output at time-2 */
        float yn2; // y_n-2
    public:
        /** instance constructor, initialize sample memories to 0.0f */
        Filter();
        /** compute output filter value from given input value.
            The input and output samples are automatically pushed into the memory
            \param[in] xn input value at current time
            \return output value at current time
        */
        float compute(float xn);
};
```

11. Donner le diagramme de classe correspondant à cette déclaration.



12. Dans un nouveau fichier « filter.cpp », implémenter ce filtre.

```
/** filter coefficient for input at current time */
const float b0 = +0.0674552739f;
/** filter coefficient for input at time - 1 */
const float b1 = +0.134910548f;
/** filter coefficient for input at time - 2 */
const float b2 = +0.0674552739f;
/** filter coefficient for output at time - 1 */
const float a1 = -1.1429805f;
/** filter coefficient for output at time - 2 */
const float a2 = +0.412801598f;

Filter::Filter()
{
    xn1 = 0.0f;
    xn2 = 0.0f;
    yn1 = 0.0f;
    yn2 = 0.0f;
}

float Filter::compute(float xn)
{
    float yn;
    yn = xn*b0 + xn1*b1 + xn2*b2 - yn1*a1 - yn2*a2;
    xn2 = xn1;
    xn1 = xn;
    yn2 = yn1;
    yn1 = yn;
    return(yn);
}
```

13. Modifier la déclaration de la classe « Coordinates » dans le fichier « coordinates.h » afin de déclarer les 3 attributs de classe « Filter ». Modifier le code de la fonction coordinates::read_sensors_data dans le fichiers « coordinates.cpp » afin d'utiliser ces trois filtres numériques,

Nouveaux attributs :

```
    /** magnetometer z axis median value (for calibration purposes) */
    int16_t m_mag_med_z;
    /** magnetometer z axis maximum value (for calibration purposes) */
    int16_t m_mag_max_z;
    /** filter for magnetometer x axis */
    Filter m_mag_filter_x;
    /** filter for magnetometer y axis */
    Filter m_mag_filter_y;
    /** filter for magnetometer z axis */
    Filter m_mag_filter_z;
public:
    /** instance constructor */
    Coordinates();
    /** instance destructor */
    ~Coordinates();
```

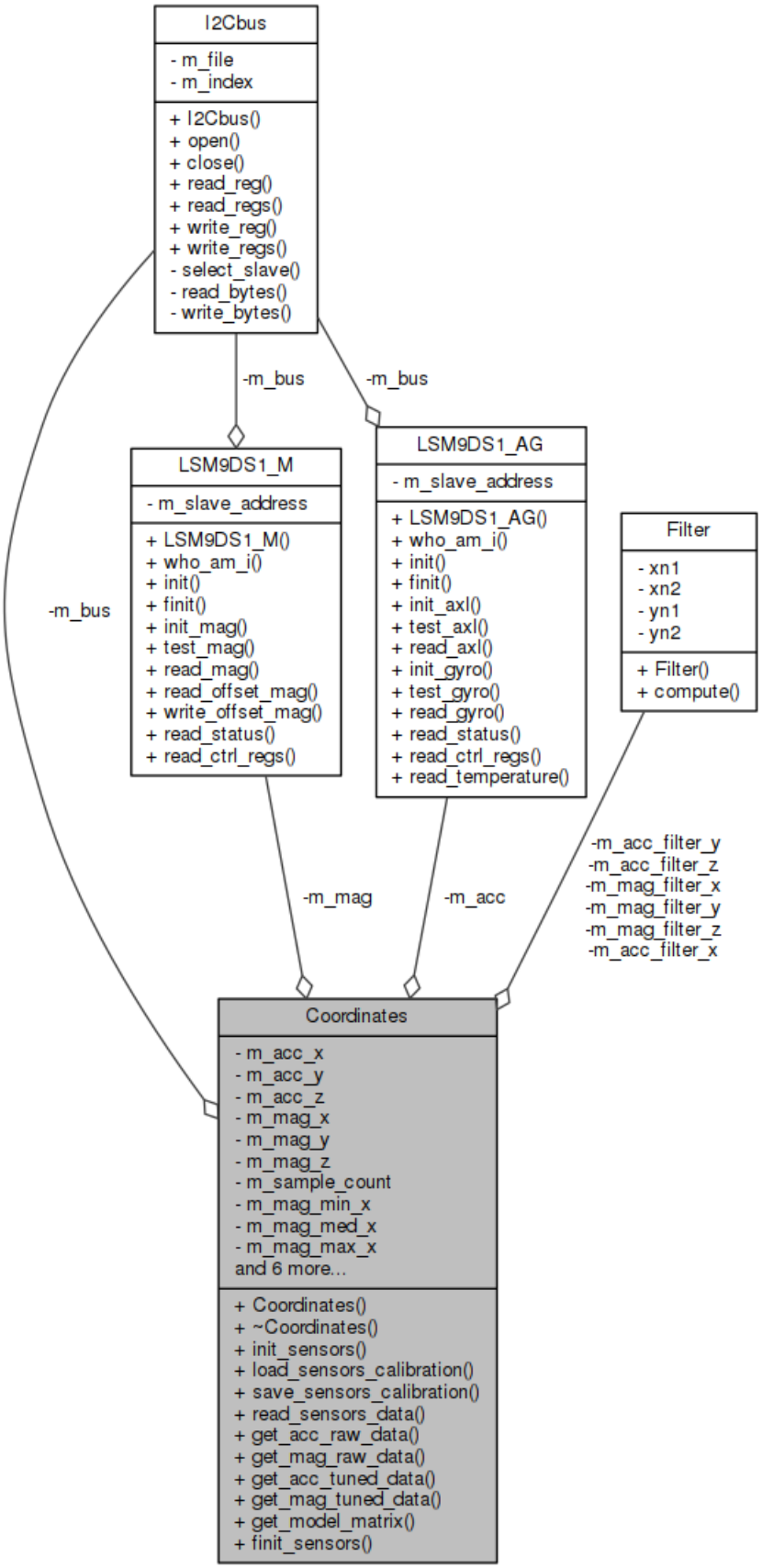
Initialisation facultative :

```
// instance constructor
Coordinates::Coordinates()
    : m_mag_filter_x(),
      m_mag_filter_y(),
      m_mag_filter_z()
{
    /* create the I2C bus object */
```

Utilisation dans la méthode coordinates::read_sensors_data :

```
    // wait for gyro data
    //while ((ag.read_status() & 0b00000010) == 0);
    //  ag.read_gyro(&gyro_x, &gyro_y, &gyro_z);*/
    // wait for magnetometer data
    while ((m_mag->read_status() & 0b00001000) == 0);
    // read magnetometer
    m_mag->read_mag(&m_mag_x, &m_mag_y, &m_mag_z);
    // update min/max data for future recalibration
    if(m_sample_count > 5)
    {
        // save for future calibration
        if(m_mag_x<m_mag_min_x) m_mag_min_x = m_mag_x;
        if(m_mag_x>m_mag_max_x) m_mag_max_x = m_mag_x;
        if(m_mag_y<m_mag_min_y) m_mag_min_y = m_mag_y;
        if(m_mag_y>m_mag_max_y) m_mag_max_y = m_mag_y;
        if(m_mag_z<m_mag_min_z) m_mag_min_z = m_mag_z;
        if(m_mag_z>m_mag_max_z) m_mag_max_z = m_mag_z;
    }
    else
          m_sample_count++;
    // filter magnetometer values
    m_mag_x = m_mag_filter_x.compute(m_mag_x);
    m_mag_y = m_mag_filter_y.compute(m_mag_y);
    m_mag_z = m_mag_filter_z.compute(m_mag_z);
```

14. Valider expérimentalement le fonctionnement. Donner le diagramme de classes correspondant aux classes « Filter », « Coordinates », « LSM9DS1_AG » et « LSM9DS1_M ». Faire apparaître les relations entre ces classes.


I2Cbus
- m_file
- m_index
+ I2Cbus()
+ open()
+ close()
+ read_reg()
+ read_regs()
+ write_reg()
+ write_regs()
- select_slave()
- read_bytes()
- write_bytes()
-m_bus
-m_bus
LSM9DS1_M
- m_slave_address
+ LSM9DS1_M()
+ who_am_i()
+ init()
+ finit()
+ init_mag()
+ test_mag()
+ read_mag()
+ read_offset_mag()
+ write_offset_mag()
+ read_status()
+ read_ctrl_regs()
LSM9DS1_AG
- m_slave_address
+ LSM9DS1_AG()
+ who_am_i()
+ init()
+ finit()
+ init_axl()
+ test_axl()
+ read_axl()
+ init_gyro()
+ test_gyro()
+ read_gyro()
+ read_status()
+ read_ctrl_regs()
+ read_temperature()
Filter
- xn1
- xn2
- yn1
- yn2
+ Filter()
+ compute()
-m_bus
-m_mag
-m_acc
-m_acc_filter_y
-m_acc_filter_z
-m_mag_filter_x
-m_mag_filter_y
-m_mag_filter_z
-m_acc_filter_x
Coordinates
- m_acc_x
- m_acc_y
- m_acc_z
- m_mag_x
- m_mag_y
- m_mag_z
- m_sample_count
- m_mag_min_x
- m_mag_med_x
- m_mag_max_x
and 6 more...
+ Coordinates()
+ ~Coordinates()
+ init_sensors()
+ load_sensors_calibration()
+ save_sensors_calibration()
+ read_sensors_data()
+ get_acc_raw_data()
+ get_mag_raw_data()
+ get_acc_tuned_data()
+ get_mag_tuned_data()
+ get_model_matrix()
+ finit_sensors()

Cinquième étude (durée ≈ 30 minutes)

Objectif : factoriser les classes existantes afin de réduire les redondances

Les classes « LSM9DS1_AG » et « LSM9DS1_M » déclarées dans le fichier lsm9ds1.h et implémentées dans le fichier lsm9ds1.cpp possèdent des redondances dans la déclaration de leurs attributs et méthodes.

15. Énumérer les attributs et les méthodes redondantes entre ces deux classes. Ces redondances sont essentiellement liée à la connexion du bus I2C, dans le fichier lsm9ds1.h déclarer une classe « I2Cslave » représentant un périphérique identifié par une adresse d'esclave connecté sur un bus.

Attributs redondants :

```
/** chip slave address */
uint8_t m_slave_address;
/** reference to the I2C bus */
I2Cbus &m_bus;
```

Le constructeur est redondant :

```
/** instance constructor, initialize members with given parameters
    \param[in] bus reference to the I2C bus
    \param[in] slave_address chip slave address (I2C_ADDR_AG by default)
*/
LSM9DS1_AG(I2Cbus &bus, uint8_t slave_address);
/** return the content of "who am i"/identification register
    \return register content, should be 0b01101000
*/
```

Facultatif : les méthodes suivantes peuvent être éventuellement virtuelles et surchargées car de noms et d'utilités identiques dans les 2 classes filles « LSM9DS1_AG » et « LSM9DS1_M » :

```
/** return the content of "who am i"/identification register
    \return register content, should be 0b01101000
*/
uint8_t who_am_i(void);
/** initialize the chip for basic operations (reset, temperature...) */
void init(void);
/** finalize and shutdown the chip */
void finit(void);
/** initialize the accelerometer with a default configuration */
```

```
/** read accelerometer/gyroscope status register
    \return status value
*/
uint8_t read_status(void);
/** read the 10 configuration registers
    \param[out] regs pointer to a 10-byte sized array filled with register values
*/
void read_ctrl_regs(uint8_t *regs);
/** read temperature raw value
    \return temperature in sensor custom scale
*/
```

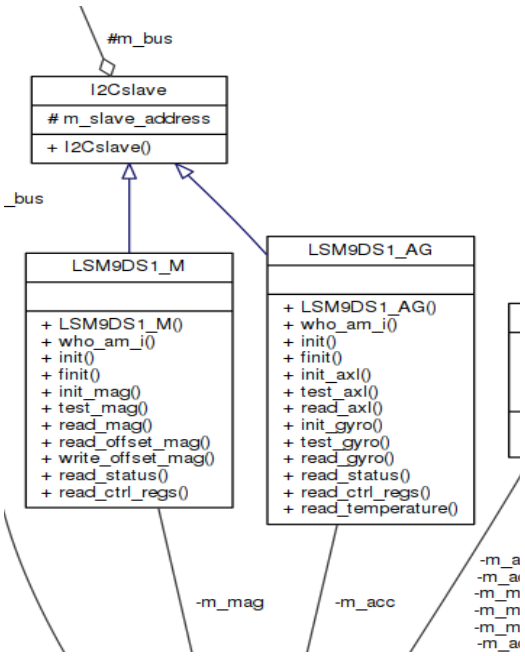
Déclaration de cette nouvelle classe I2Cslave (sans les éventuelles méthodes virtuelles) :

```
class I2Cslave
{
    protected: // ou public:
        /** chip slave address */
        uint8_t m_slave_address;
        /** reference to the I2C bus */
        I2Cbus &m_bus;
    public:
        I2Cslave(I2Cbus &bus, uint8_t slave_address);
};
```

Implémentation de cette nouvelle classe I2Cslave (sans les éventuelles méthodes virtuelles) :

```
I2Cslave::I2Cslave(I2Cbus &bus, uint8_t slave_address)
    : m_bus(bus)
{
    m_slave_address = slave_address;
}
```

16. Tracer le diagramme de classes correspondant aux classes « LSM9DS1_AG » et « LSM9DS1_M » ayant un lien d'héritage avec la classe « I2Cslave ». Modifier les déclarations et implémentations des classes LSM9DS1_AG et LSM9DS1_M afin d'ajouter un lien d'héritage vers cette nouvelle classe « I2Cslave ».



Dans le fichier lsm9ds1.h : ajout de la classe parente après le nom de la classe, suppression des attributs transférés dans la classe parente :

```
class LSM9DS1_AG: public I2Cslave
{
    public:
        /** instance constructor, initialize members with given parameters
            \param[in] bus reference to the I2C bus
            \param[in] slave_address chip slave address (I2C_ADDR_AG by default)
        */
        LSM9DS1_AG(I2Cbus &bus, uint8_t slave_address);
```

```
class LSM9DS1_M: public I2Cslave
{
    public:
        /** instance constructor, initialize members with given parameters
            \param[in] bus reference to the I2C bus
            \param[in] slave_address chip slave address (I2C_ADDR_M by default)
        */
        LSM9DS1_M(I2Cbus &bus, uint8_t slave_address);
```

Dans le fichier lsm9ds1.cpp : modification des constructeurs pour appeler le constructeur parent :

```
LSM9DS1_AG::LSM9DS1_AG(I2Cbus &bus, uint8_t slave_address)
    : I2Cslave(bus, slave_address)
{
}
```

```
LSM9DS1_M::LSM9DS1_M(I2Cbus &bus, uint8_t slave_address)
    : I2Cslave(bus, slave_address)
{
}
```

17. Déplacer cette classe dans des nouveaux fichiers dédiés I2Cslave.h et I2Cslave.cpp que vous ajouterez au projet actuel dans l'environnement CodeBlocks.

Épreuve d’admission de soutenance d’un dossier industriel

A. Présentation de l’épreuve

Texte de référence
http://www.devenirenseignant.gouv.fr/cid98734/les-epreuves-de-l-agregation-externe-section-sciences-industrielles-de-l-ingenieur.html

- Durée de la préparation des moyens de l'exposé : 30 minutes
- Durée totale de l'épreuve : 1 heure (présentation : 30 minutes maximum, entretien avec le jury : 30 minutes)
- Coefficient 2

L'épreuve consiste en la soutenance devant le jury d'un dossier technique et scientifique réalisé par le candidat dans un des domaines de l'option préparée, suivie d'un entretien.

L'épreuve a pour but de vérifier que le candidat est capable de rechercher les supports de son enseignement dans le milieu économique et d'en extraire des exploitations pertinentes pour son enseignement en cycle terminal du lycée, en classes préparatoires aux grandes écoles, en sections de techniciens supérieurs et instituts universitaires de technologie.

Le dossier présenté par le candidat est relatif à un système technique dont la dominante est choisie par le candidat. Son authenticité et son actualité sont des éléments décisifs.

L'exposé et l'entretien permettent d'apprécier l'authenticité et l'actualité du problème choisi par le candidat, sa capacité à en faire une présentation construite et claire, à mettre en évidence les questionnements qu'il suscite et à en dégager les points remarquables et caractéristiques. Ils permettent également au candidat de mettre en valeur la qualité de son dossier et l'exploitation pédagogique qu'il peut en faire dans le cadre de son enseignement.

En utilisant les moyens courants de présentation (vidéoprojecteur et informatique associée, en particulier), le candidat présente le support technique qu'il a choisi pour l'épreuve ainsi que les investigations et développements qu'il a conduits pour s'en approprier totalement le fonctionnement et les évolutions potentielles. Lors de la présentation, le candidat justifiera le choix du support d'études et les investigations conduites qui pourraient, selon lui, donner lieu à des exploitations pertinentes en collège ou en lycée.

Pendant l'entretien, le jury conduit des investigations destinées à se conforter dans l'idée que le dossier présenté résulte bien d'un travail personnel du candidat et s'en faire préciser certains points.

Les éléments constitutifs du dossier sont précisés par note publiée sur le site internet du ministère chargé de l'Éducation. Les dossiers doivent être déposés au secrétariat du jury cinq jours francs au moins avant le début des épreuves d'admission.

Le jury cherche également à apprécier la capacité du candidat, en qualité de futur agent du service public d'éducation, à se représenter la diversité des conditions d'exercice du métier et les valeurs qui le portent, dont celles de la République.

B. Commentaires du jury

Au préalable, le jury souhaite souligner :

- qu'il est incompréhensible que quelques dossiers puissent être hors du périmètre de l'ingénierie informatique ou à sa marge ce qui est fortement pénalisant ; il est rappelé que le candidat doit réaliser un dossier technique et scientifique où la part d'ingénierie informatique est prépondérante ?
- que le candidat doit s'appuyer sur des éléments authentiques du système présenté pour fonder ses développements scientifiques, technologiques et pédagogiques ;
- que le candidat doit énoncer clairement la ou les problématiques traitées ainsi que la méthodologie utilisée pour y répondre.

Cette épreuve ayant pour objectif de mesurer l'aptitude du candidat à trouver « les supports de son enseignement dans le milieu économique et d'en extraire des exploitations pertinentes pour son enseignement » il est conseillé aux candidats se représentent au concours de renouveler lesdits supports.

Les trente minutes de présentation doivent permettre au candidat d'apporter des précisions sur le dossier et montrer ainsi comment il a pu mobiliser les différentes compétences du référentiel de compétences des métiers du professorat et de l'éducation. Les candidats n'ont pas à présenter leur parcours professionnel lors de cette épreuve.

- **Principaux conseils**

L'épreuve de dossier nécessite anticipation, développements et préparation spécifique :

- le dossier réalisé par le candidat est relatif à un système authentique et actuel d'un domaine significatif de l'ingénierie informatique en sciences industrielles de l'ingénieur en lien avec des enjeux sociétaux. Les systèmes novateurs sont appréciés. Les éléments fournis doivent permettre d'attester de l'implication et des apports personnels ;
- la ou les problématiques scientifiques et/ou technologiques liées au système sont à exprimer avec clarté et précision ;
- il est attendu du candidat un développement scientifique et technologique, personnel, d'ingénierie informatique, référencé dans le dossier. Il sera traité à un niveau master 2 avec une démarche et des fondements scientifiques (théories, publications référencées, brevets, …). La modélisation, l'implémentation, la programmation, l'architecture des systèmes, la communication (réseaux, IoT, IHM, IA, …), sont des éléments essentiels de l'ingénierie informatique. Il est attendu du candidat qu'il utilise les outils « métiers » de l'ingénierie informatique adaptés aux problèmes traités ;
- il est attendu du candidat une analyse critique (qualitative, quantitative), une prise de recul, et des éléments de réponse et d'amélioration en rapport avec la problématique initiale annoncée ;
- une simple analyse d'un produit est hors sujet ;
- un partenariat réel avec l'entreprise est demandé. En retour, les études et expertises conduites par le candidat sont communiquées à l'entreprise en vue d'une valorisation. Les éléments témoignant des échanges entre le ou la candidate et l'entreprise sont un gage d'authenticité. Il est souhaitable, dans la mesure du possible, qu'un accord de confidentialité soit signé entre les parties prenantes en vue de protéger les intérêts réciproques. En cas de confidentialité réclamée, il est impératif de la mentionner clairement dans le dossier ;

- les exploitations pédagogiques proposées doivent être en cohérence avec le système support, la problématique, et les développements scientifiques et technologiques ;
- les exploitations pédagogiques développées doivent être liées aux référentiels ou programmes choisis au regard de la spécialité ingénierie informatique, les compétences et connaissances associées et l'organisation matérielle des activités d'enseignement.
- les séquences proposées doivent s'inscrire dans une progression générale formalisée dans le cycle de formation choisi en lien avec le référentiel ou programme ciblé ;
- au moins une activité est développée en cohérence avec une des séquences pédagogiques proposées.
- les documents proposés aux élèves sont à présenter avec si possible des réponses d'élèves qu'il convient d'analyser ;
- la description des activités doit permettre aux membres du jury de percevoir ce que font les élèves ;
- une attention particulière sur les stratégies pédagogiques est attendue ;
- des propositions d'exploitation pédagogique dans une perspective d'activités pluritechnologiques ou interdisciplinaires sont appréciées ;
- il est attendu la mise en place de modalités d'évaluation des compétences argumentées prenant en compte la diversité des élèves et l'inclusion scolaire ;
- l'innovation pédagogique utilisant des outils numériques « métiers » et connexes est appréciée s'ils apportent une valeur ajoutée pédagogique ;
- le développement pédagogique est l'occasion de mettre en évidence comment faire partager les valeurs de la république.

- **Réalisation du dossier**

La forme et la qualité de réalisation du dossier montrent que le candidat a réfléchi à la teneur du message qu'il souhaite communiquer aux membres du jury. Pour cela :

- les éléments figurant dans la clé USB doivent être cités dans le corps du texte du dossier chaque fois que nécessaire et référencés dans une annexe. Il est conseillé de distinguer les parties scientifique/technologique et pédagogique en deux sous-répertoires différents ;
- les règles de citation des sources (des tables, illustrations, articles, références ...) doivent être respectées ;
- le dossier doit comporter un titre, un sommaire, une conclusion et être paginé ;
- les tableaux, graphiques et annexes doivent être correctement référencés et récapitulés dans des tables dédiées ;
- le dossier ne doit en aucun cas être un manuel d'utilisation ou une documentation commerciale ;
- le dossier doit être rédigé dans une langue française soignée en tenant compte des règles grammaticales et orthographiques ;
- les développements et résultats des expérimentations et mesures réalisées doivent être référencés dans le dossier et présents en annexes ;
- le dossier doit résulter d'un travail personnel du candidat ;
- les dossiers doivent être parvenus au secrétariat du jury cinq jours ouvrés avant le début des épreuves d'admission.

- **Présentation orale**

Durant les 30 minutes de l'exposé, le candidat doit mettre en valeur ses qualités de communicant pour expliquer ses choix, ses démarches et ses analyses, dans le champ de l'ingénierie informatique. Le jury veillera à ce que le candidat ne dépasse pas les 30 minutes autorisées. L'échange avec le jury permet d'approfondir certains points présentés dans le dossier ou durant l'exposé. Cet échange porte tant sur les développements scientifiques et technologiques que sur les propositions d'exploitation pédagogique qui en découlent. Le choix des éléments présentés à l'oral est important.

Le candidat peut utiliser tout support permettant d'attester des réalisations (vidéo de présentation ou lien vers des démonstrations). Les documents vidéoprojetés doivent être de qualité et lisibles. Il est conseillé de numéroter les diapositives pour faciliter les échanges avec les membres du jury.
Le jury élargit son questionnement pour vérifier que le candidat a entrepris une réelle réflexion sur :

- les finalités de l'enseignement des sciences industrielles de l'ingénieur pour répondre aux problématiques sociétales ;
- les liens avec d'autres disciplines et les démarches pédagogiques ;
- les compétences (référentiel métier) qu'un enseignant doit développer ainsi que sur les missions qui lui sont confiées ;
- les situations au cours desquelles, en tant qu'agent du service public d'éducation, il est en position de faire partager les valeurs et les principes de la République.

Rapport sur la transmission des valeurs et principes de la République

« Lors des épreuves d'admission, outre les interrogations relatives aux sujets et à la discipline, le jury pose les questions qu'il juge utiles lui permettant d'apprécier la capacité du candidat, en qualité de futur agent du service public d'éducation, à prendre en compte dans le cadre de son enseignement la construction des apprentissages des élèves et leurs besoins, à se représenter la diversité des conditions d'exercice du métier, à en connaitre de façon réfléchie le contexte, les différentes dimensions (classe, équipe éducative, établissement, institution scolaire, société) et les valeurs qui le portent, dont celles de la République.
Le jury peut, à cet effet, prendre appui sur le référentiel des compétences professionnelles des métiers du professorat et de l'éducation fixé par l'arrêté du 1er juillet 2013. »
Texte de référence (http://www.devenirenseignant.gouv.fr/cid98734/les-epreuves-de-l-agregation-externe-section-sciences-industrielles-de-l-ingenieur.html)

Le candidat doit prendre en compte ces exigences dans la conception des séquences pédagogiques présentées au jury. Il s'agit de faire acquérir, à l'élève, des compétences alliant des connaissances scientifiques et technologiques et des savoir-faire associés, mais également d'installer des comportements responsables et respectueux des valeurs républicaines.

Cet objectif exigeant induit une posture réflexive du candidat lors de la préparation et de la présentation d'une séquence pédagogique. En particulier, les stratégies pédagogiques proposées devront permettre d'atteindre l'objectif de formation visé dans le cadre de « l'école inclusive ». Il est indispensable de donner du sens aux enseignements en ne les déconnectant pas d'un contexte sociétal identifiable. Cela doit contribuer à convaincre les élèves du bien-fondé des valeurs républicaines et à se les approprier.
L'éducation aux valeurs républicaines doit conduire à adopter des démarches pédagogiques spécifiques, variées et adaptées. Il s'agit en particulier de doter chaque futur citoyen d'une culture faisant de lui un acteur éclairé et responsable de l'usage des technologies et des enjeux éthiques associés. À dessein, il est nécessaire de lui faire acquérir des comportements fondateurs de sa réussite personnelle et le conduire à penser et construire son rapport au monde. Les modalités pédagogiques, déployées en sciences industrielles de l'ingénieur, sont nombreuses et sont autant d'opportunités offertes à l'enseignant pour apprendre aux élèves :

- à travailler en équipe et coopérer à la réussite d'un projet ;
- à assumer une responsabilité individuelle et collective ;
- à travailler en groupe à l'émergence et à la sélection d'idées issues d'un débat et donc favoriser le respect de l'altérité ;
- à développer des compétences relationnelles en lui permettant de savoir communiquer une idée personnelle ou porter la parole d'un groupe ;
- à comprendre les références et besoins divers qui ont conduit à la création d'objets ou de systèmes à partir de l'analyse des « modes », des normes, des lois… ;
- à différencier, par le déploiement de démarches rigoureuses, ce qui relève des sciences et de la connaissance de ce qui relève des opinions et des croyances. L'observation de systèmes réels, l'analyse de leur comportement, de la construction ou de l'utilisation de modèles multi-physiques participent à cet objectif ;
- à observer les faits et situations divers suivant une approche systémique et rationnelle ;
- à adopter un positionnement citoyen assumé au sein de la société en ayant une connaissance approfondie de ses enjeux au sens du développement durable. L'impact environnemental, les

coûts énergétiques, de transformation et de transport, la durée de vie des produits et leur recyclage, sont des marqueurs associés à privilégier ;
- à réfléchir collectivement à son environnement, aux usages sociaux des objets et aux conséquences induites ;
- à comprendre les enjeux sociétaux liés au respect de l'égalité républicaine entre hommes et femmes ;
- …

Ces différentes approches permettent d'évaluer la posture du candidat par rapport au besoin de transmettre les valeurs et les principes de la République à l'école. La dimension civique de l'enseignement doit être explicite.

Cette déontologie professionnelle suppose au moins l'appropriation par le candidat des ressources et textes suivants :
- les droits et obligations du fonctionnaire présentés sur le portail de la fonction publique (https://www.fonction-publique.gouv.fr/droits-et-obligations) ;
- les articles L 111-1 à L 111-4 et l'article L 442-1 du code de l'Éducation ;
- le vade-mecum « la laïcité à l'École » (https://eduscol.education.fr/1618/la-laicite-l-ecole) ;
- le vade-mecum « Agir contre le racisme et l'antisémitisme » (https://eduscol.education.fr/1720/agir-contre-le-racisme-et-l-antisemitisme) ;
- « Qu'est-ce que la laïcité ? », Conseil des sages de la laïcité, janvier 2020 (https://www.education.gouv.fr/le-conseil-des-sages-de-la-laicite-41537) ;
- le parcours magistère « faire vivre les valeurs de la République » (https://magistere.education.fr/f959) ;
- « L'idée républicaine aujourd'hui », Conseil des sages de la laïcité ;
- « La République à l'École », Inspection générale de l'éducation, du sport et de la recherche ;
- le site IH2EF (https://www.ih2ef.gouv.fr/laicite-et-services-publics).