En guise de brève introduction

> *50 kilos de patates, un sac de sciure de bois, il te sortait 25 litres de 3 étoiles à l'alambic. Un vrai magicien ce Jo. Et c'est pour ça que je me permets...*
>
> Michel Audiard – *Les tontons flingueurs*

Voici le recueil de la plupart des sujets d'informatique — ou comportant au moins une question d'informatique — posés aux concours des grandes écoles scientifiques en 2015 et disponibles actuellement.

Le recueil est partagé en trois grandes parties. D'abord, les épreuves de l'option info de nos classes, auxquels s'ajoutent les épreuves d'informatique des seconds concours des ENS. Ensuite, les épreuves d'informatique commune. Enfin, les épreuves des disciplines autres que l'informatique (mathématiques, modélisation, sciences de l'ingénieur) comportant au moins une question d'informatique.

Comme l'an dernier, le bulletin des concours Informatique ne sera proposé aux adhérents que sous forme électronique.

Dans ce recueil, le nom du fichier contenant chaque sujet figure dans la table des matières et en tête de chaque page. Les fichiers sont disponibles sur le site de l'UPS, soit à l'adresse http://prepas.org/ups.php?module=Maths&voir=recherche, soit dans la rubrique Ressources / Informatique http://prepas.org/ups.php?rubrique=146 (pour les sujets de modélisation ou de sciences de l'ingénieur).

Vous trouverez ce recueil, au format pdf, sur le site de l'UPS, à l'adresse http://prepas.org/ups.php?rubrique=146. Les ajouts d'énoncés y seront signalés en allant.

Je ne peux pas terminer sans remercier chaleureusement, au nom de l'association toute entière, Évelyne Latrémolière-Quercia qui s'occupait de ce recueil depuis une dizaine d'années et qui a mis en place la formidable machinerie LaTeX qui rend le travail de ses successeurs si facile et efficace. Qu'il est loin le temps des ciseaux et des bâtons de colle dans le local du 3, rue de l'École Polytechnique !

Philippe Patte philippe.patte@prepas.org

5 août 2015

ÉCOLE POLYTECHNIQUE – ÉCOLES NORMALES SUPÉRIEURES

CONCOURS D'ADMISSION 2015 **FILIÈRE MP — SPÉCIALITÉ INFO**

COMPOSITION D'INFORMATIQUE – A – (XULCR)

(Durée : 4 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation pour cette épreuve est Caml Light.

⋆ ⋆ ⋆

Ordonnancement de graphes de tâches

Introduction

Voici un problème important qui se pose à chacun et chacune de nous tous les jours ! J'ai un certain nombre de tâches à *exécuter* aujourd'hui ; comment planifier à quelle heure chacune va être exécutée ? Par exemple, je peux avoir aujourd'hui à terminer un devoir en mathématiques, faire ma lessive à la laverie, avancer un projet en informatique, aller faire quelques courses et repasser mon linge. Chacune de ces tâches va me prendre une certaine durée, que je peux estimer. Même si ce n'est pas tout à fait le cas dans la vie courante, on peut ici supposer que les tâches ne sont pas interrompues ; je ne commence une nouvelle tâche que lorsque la tâche en cours est terminée.

Dépendances. Le plus souvent, il existe des *dépendances* entre les tâches, en ce sens qu'il est nécessaire d'exécuter une tâche A avant d'exécuter une tâche B. Par exemple, il est nécessaire de laver le linge avant de le repasser. Mais, si je n'ai plus de lessive, il est nécessaire de passer faire les courses avant d'aller à la laverie. Ces dépendances peuvent être modélisées par un graphe orienté. Les nœuds sont les tâches, les arcs orientés sont les dépendances. Il y a un arc $A \to B$ si la tâche A doit être terminée avant que la tâche B puisse commencer. Notez qu'il n'est pas du tout nécessaire de passer reprendre son linge à la laverie dès que la machine s'arrête. La tâche B peut être ordonnancée longtemps après que la tâche A sera terminée.

Ordonnancement. Un *ordonnancement* est la donnée d'une heure d'exécution pour chacune des tâches qui respecte cette contrainte qu'une tâche commence seulement lorsque toutes celles dont elle dépend sont terminées. Notez qu'il peut y avoir des moments de repos où aucune tâche n'est en exécution. La mesure intéressante est alors la *durée d'exécution totale*, c'est-à-dire la durée écoulée entre l'heure à laquelle l'exécution de la première tâche commence et l'heure à laquelle celle de la dernière tâche se termine.

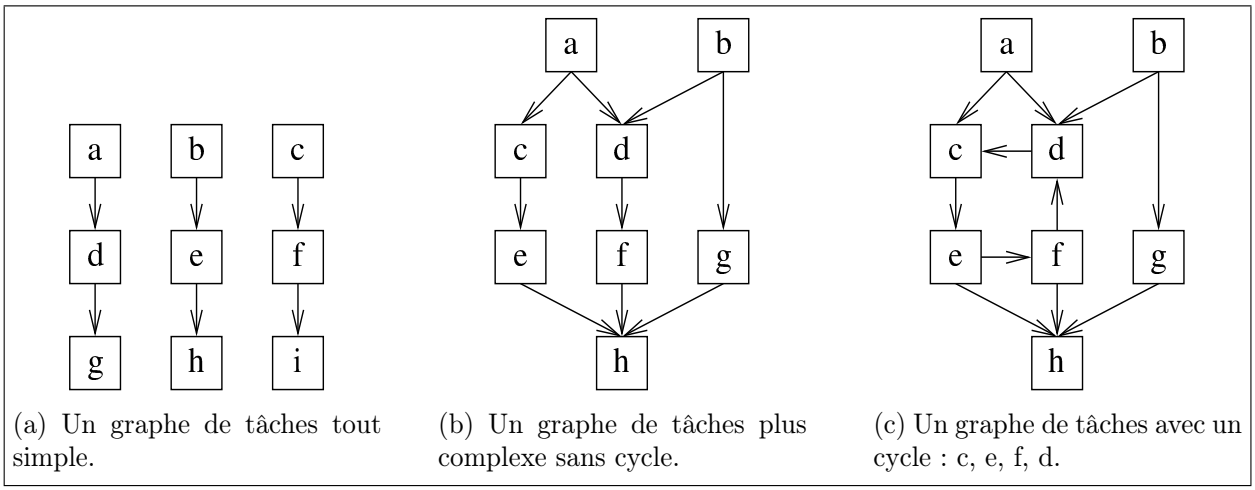
Parallélisme. Si je suis seul, je ne peux exécuter qu'une tâche à la fois. Mais je peux aussi me faire aider ! Il est alors possible d'exécuter plusieurs tâches en *parallèle*. Par exemple, je peux demander à quelqu'un de faire ma lessive à la laverie, pour aller faire mes courses pendant ce temps et ensuite revenir la prendre pour la repasser. En termes informatiques, on parle de multiples *processeurs* qui collaborent pour le traitement des tâches. Dans notre exemple, deux processeurs travaillent en parallèle : l'un pour faire la lessive, l'autre pour faire les courses. À chaque instant, plusieurs tâches peuvent être exécutées, au plus autant que de processeurs. Notez qu'il est cependant possible qu'un processeur soit forcé de rester inactif un certain temps, par exemple parce que la tâche qu'il doit exécuter dépend d'une autre tâche qui est en cours d'exécution sur un autre processeur.

Défi algorithmique. Les exemples ci-dessus sont bien sûr des illustrations simplifiées. En pratique, on va considérer des graphes de tâches de taille gigantesque, par exemple l'ensemble des actions nécessaires pour assembler un avion. Ces graphes comptent des millions de tâches avec des dépendances complexes. Les tâches seront allouées à des ouvriers. Ceux-ci travaillent à l'intérieur d'horaires fixés, avec des périodes de repos, des vacances, des arrêts imprévus pour cause de maladie ou de panne de machine. L'objectif sera de trouver le meilleur ordonnancement possible pour l'assemblage selon une mesure donnée. Notez que dans ce cas, le graphe est fixe, mais le nombre d'ouvriers peut varier : on peut par exemple embaucher plus d'ouvriers pour réduire la durée d'exécution totale. Cela augmente le coût de production mais réduit les délais de livraison. Trouver un ordonnancement optimal est alors un défi algorithmique majeur.

Plan du sujet proposé. La partie I introduit la notion d'ordonnancement d'un graphe de tâches. La partie II s'intéresse à quelques propriétés des graphes de tâches acycliques. La partie III étudie une première approche pour la recherche d'un ordonnancement d'un graphe de tâches. L'ordonnancement produit est optimal avec $p = 1$ processeur, mais pas avec $p > 1$. La partie IV étudie comment modifier cette approche pour produire un ordonnancement optimal avec $p = 2$ processeurs dans le cas particulier des arbres arborescents entrants. La partie V décrit comment compléter cette approche et obtenir un ordonnancement optimal avec $p = 2$ processeurs dans le cas général.

Les parties peuvent être traitées indépendamment. Néanmoins, chaque partie utilise des notations et des fonctions introduites dans les parties précédentes.

La complexité, ou le coût, d'un algorithme ou d'une fonction Caml est le nombre d'opérations élémentaires nécessaires à son exécution dans le pire cas. La notion d'opération élémentaire sera précisée dans chaque cas par le sujet. Lorsque cette complexité dépend d'un ensemble de



(a) Un graphe de tâches tout simple.

(b) Un graphe de tâches plus complexe sans cycle.

(c) Un graphe de tâches avec un cycle : c, e, f, d.

FIGURE 1 – Quelques exemples de graphes de tâches.

paramètres $(n, p, \dots)$, on pourra donner cette estimation sous forme asymptotique. On rappelle qu'une application $c(n, p, \dots)$ est dans la classe $\mathcal{O}(f)$ s'il existe une constante $\alpha > 0$ telle que $|c(n, p, \dots)| < \alpha \times f(n, p, \dots)$, pour toutes les valeurs de $n, p, \dots$ assez grandes.

I Définitions de base

Graphe de tâches. Un *graphe de tâches* $G = (T, D, \delta)$ est constitué d'un ensemble *fini* et *non vide* T de tâches notées u, v, etc.

L'application δ associe à chaque tâche du graphe sa durée. Dans tout ce problème, on supposera que la durée d'une tâche est unitaire : $\delta(u) = 1$.

L'ensemble $D \subseteq T \times T$ est un ensemble d'arcs (dirigés) entre ces tâches (D pour *dépendance*, voir plus loin). L'existence d'un arc (u, v) dans D est notée $u \to v$. On suppose qu'il n'y a pas d'arc entre une tâche et elle-même : $D \cap \{(u, u) | u \in T\} = \emptyset$.

Il y a un arc dirigé (u, v) d'une tâche u vers une autre tâche v distincte, $u \neq v$, si la tâche u doit être exécutée avant la tâche v. On dit alors que la tâche v *dépend* de la tâche u en ce sens que v ne peut commencer qu'une fois que u est terminée. On dit alors que u *précède* v ou que la tâche v *succède* à la tâche u. On peut donc parler de l'ensemble des tâches qui précèdent v et de l'ensemble des tâches qui succèdent à u. Notez que ces ensembles peuvent être vides.

Une tâche qui n'a aucun prédécesseur s'appelle une *racine*. Une tâche qui n'a aucun successeur s'appelle une *feuille*.

La *taille* d'un graphe de tâches est le nombre de ses tâches.

La figure 1 propose quelques exemples de graphes de tâches de petite taille.

Ordonnancement. Un *ordonnancement* σ d'un graphe de tâches $G = (T, D, \delta)$ est une application σ à valeurs entières qui associe une date d'exécution à chaque tâche du graphe, dans le respect des contraintes de dépendance spécifiées par la formule (1) ci-dessous.

Une tâche $u \in T$ est exécutée de manière ininterrompue par le processeur qui en a la charge aux instants t tels que $\sigma(u) \leq t < \sigma(u)+\delta(u)$. Une tâche v qui dépend de u ne peut être exécutée qu'après la terminaison de u, donc à partir de l'instant $\sigma(u) + \delta(u)$. Un ordonnancement doit donc respecter la contrainte suivante :

$$\forall u, v \in T,\ (u \to v) \Rightarrow (\sigma(u) + \delta(u) \leq \sigma(v)) \tag{1}$$

L'instant a_σ du début de l'ordonnancement σ du graphe de tâches $G = (T, D, \delta)$ est l'instant où la première tâche est exécutée : $a_\sigma = \min_{u \in T} \sigma(u)$.

L'instant b_σ de la fin de l'ordonnancement est l'instant où la dernière tâche est terminée : $b_\sigma = \max_{u \in T} (\sigma(u) + \delta(u))$.

La *durée d'exécution totale* de l'ordonnancement σ est $b_\sigma - a_\sigma$.

L'ensemble S_t des tâches en cours d'exécution à l'instant t est défini par :

$$S_t = \{u \mid \sigma(u) \leq t < \sigma(u) + \delta(u)\}$$

Dans notre cas, $\delta(u) = 1$ pour toute tâche u et cette formule se simplifie :

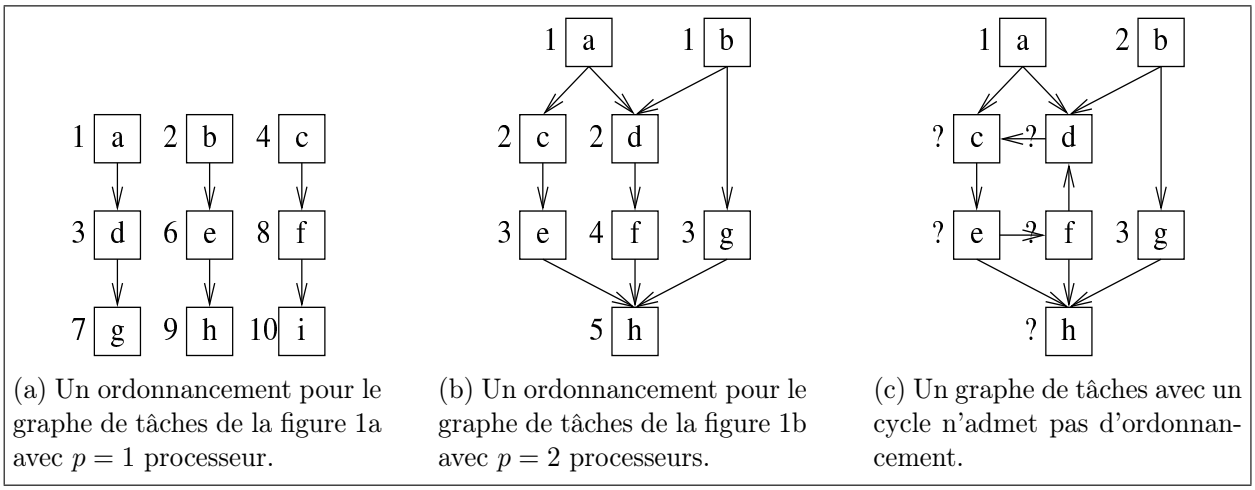
$$S_t = \{u \mid \sigma(u) = t\}$$

On dit qu'un ordonnancement utilise (au plus) p processeurs si $\mathsf{cardinal}(S_t) \leq p$ à tout instant t.

On dit qu'un ordonnancement est optimal pour un graphe de tâches G avec p processeurs si sa durée d'exécution est minimale parmi tous les ordonnancements possibles de G avec p processeurs.

La figure 2 propose quelques exemples d'ordonnancements de graphes de tâches. On rappelle que chaque tâche u dure une unité de temps : $\delta(u) = 1$. La date d'exécution de chaque tâche est indiquée à gauche de cette tâche.

1. L'ordonnancement présenté en figure 2a a une durée d'exécution totale de 10 avec $p = 1$ processeur. Il n'est pas optimal. En effet, aucune tâche n'est exécutée à l'instant 5 où la tâche e est prête ; il serait donc possible de réduire la durée d'exécution totale à 9. Ce serait alors optimal puisqu'il y a 9 tâches, chacune de durée 1.
2. L'ordonnancement présenté en figure 2b a une durée d'exécution totale de 5 avec $p = 2$ processeurs. Comme il y a 8 tâches pour 2 processeurs, tout ordonnancement a une durée au moins égale à 4. Cependant, les contraintes de dépendance ne permettent pas de réaliser un ordonnancement de durée 4. Un ordonnancement de durée d'exécution totale 5 est donc optimal.
3. Le graphe de tâches de la figure 2c comporte un cycle : $c \to e \to f \to d$. Aucune tâche d'un cycle ne peut être exécutée en respectant les contraintes de dépendance.



(a) Un ordonnancement pour le graphe de tâches de la figure 1a avec $p = 1$ processeur.

(b) Un ordonnancement pour le graphe de tâches de la figure 1b avec $p = 2$ processeurs.

(c) Un graphe de tâches avec un cycle n'admet pas d'ordonnancement.

FIGURE 2 – Quelques exemples d'ordonnancement de graphes de tâches.

Objectif. L'objectif de ce problème va être de déterminer des ordonnancements *optimaux* pour certaines classes de graphes de tâches pour un nombre p donné de processeurs.

II Graphe de tâches acyclique

Un *chemin* de dépendance dans un graphe de tâches d'une tâche u à une tâche v est une suite de tâches $(u_0, u_1, \ldots, u_n)$, $n \geq 0$, avec $u_0 = u$ et $u_n = v$ et en dépendance successive : $u_0 \to u_1$, $u_1 \to u_2$, …, $u_{n-1} \to u_n$.

La *longueur* du chemin est le nombre d'arcs de dépendance, c'est-à-dire l'entier n. La tâche initiale du chemin est u_0, sa tâche terminale u_n. Notez qu'une tâche peut apparaître plusieurs fois dans un chemin. Notez aussi qu'un chemin peut être de longueur nulle. Il a alors la même tâche initiale et terminale.

Une tâche u est *atteignable* depuis une tâche u_0 s'il existe un chemin qui a pour tâche initiale u_0 et pour tâche terminale u.

Un *cycle* dans un graphe de tâches est un chemin $(u_0, u_1, \ldots, u_n)$ de longueur $n > 0$ qui a même tâche initiale et terminale : $u_0 = u_n$.

Un graphe de tâches est dit *acyclique* s'il ne possède pas de cycle. Les graphes de tâches des figures 1a et 1b sont acycliques, celui de la figure 1c possède un cycle.

On veut maintenant montrer qu'il n'existe pas d'ordonnancement pour un graphe de tâches qui possède un cycle.

Question 1. *Soit $G = (T, D, \delta)$ un graphe de tâches qui admet un ordonnancement σ. Démontrez que s'il existe un chemin de dépendance de longueur non nulle d'une tâche u vers une tâche v dans G, alors $\sigma(u) < \sigma(v)$. En déduire que G est nécessairement acyclique.*

On pourra procéder par récurrence sur la longueur n du chemin après avoir soigneusement spécifié la propriété $H(n)$ démontrée par récurrence.

let graph = make_graph ();;		
let a = make_task 1 "a";;	add_task a graph;;	add_dependence a c graph;;
let b = make_task 1 "b";;	add_task b graph;;	add_dependence a d graph;;
let c = make_task 1 "c";;	add_task c graph;;	add_dependence b d graph;;
let d = make_task 1 "d";;	add_task d graph;;	add_dependence b g graph;;
let e = make_task 1 "e";;	add_task e graph;;	add_dependence c e graph;;
let f = make_task 1 "f";;	add_task f graph;;	add_dependence d f graph;;
let g = make_task 1 "g";;	add_task g graph;;	add_dependence e h graph;;
let h = make_task 1 "h";;	add_task h graph;;	add_dependence f h graph;;
		add_dependence g h graph;;

TABLE 1 – Comment construire le graphe de la figure 1b avec la bibliothèque Graph.

Avant de chercher un ordonnancement d'un graphe de tâches, il faut donc vérifier qu'il est acyclique. La propriété suivante fournit une condition nécessaire. On rappelle qu'un graphe de tâches est constitué d'un nombre fini et non nul de tâches.

Question 2. *Soit $G = (T, D, \delta)$ un graphe de tâches. Montrez que si G est acyclique, alors G a nécessairement au moins une racine et une feuille.*

Dans toute la suite du problème, on suppose qu'on dispose d'une certaine bibliothèque Caml appelée Graph qui permet de manipuler des graphes de tâches. Cette bibliothèque regroupe des fonctions qui sont disponibles pour les utilisateurs, même s'ils n'en connaissent pas le code. On pourra donc utiliser librement toutes les fonctions de cette bibliothèque sans les réécrire. Cette bibliothèque définit deux types de données : graph pour les graphes de tâches, et task pour les tâches. Elle propose les fonctions listées en table 2 pour manipuler ces deux types. La table 1 montre comment on pourrait construire le graphe de la figure 1b avec ces fonctions.

On rappelle quelques fonctions Caml Light permettant de manipuler les tableaux et d'imprimer.

- make_vect n a renvoie un nouveau tableau de n éléments qui contiennent tous la même valeur a.
- vect_length tab renvoie la longueur (le nombre d'éléments) n du tableau tab. Ceux-ci sont indexés de 0 à $n - 1$ inclus.
- tab.(i) renvoie la valeur de l'élément d'indice i du tableau tab.
- tab.(i) <− a affecte la valeur a à l'élément d'indice i du tableau tab.
- sub_vect tab i n renvoie le sous-tableau de tab constitué de n éléments à partir de l'indice i inclus.
- print_int n imprime l'entier n.
- print_string s imprime la chaîne s.
- print_newline () passe à la ligne suivante.

Voici par exemple comment imprimer l'ensemble des successeurs d'une tâche :

```
let print_successors t =
  let tab = get_successors t in
  for i = 0 to (vect_length tab) − 1 do
    print_string (get_name (tab.(i))); print_string "␣"
  done;
  print_newline ();;
```

make_graph: unit −> graph	Crée un graphe vide.
make_task: int −> string −> task	Crée une tâche d'une durée donnée avec un nom donné. (Attention, une erreur se produit si le nom a déjà été utilisé pour la création d'une autre tâche.)
empty_task: task	Une tâche vide, différente de toutes les tâches créées par la fonction make_task. (Attention, une erreur se produit si on applique les fonctions de manipulation de tâches ci-dessous autres que is_empty_task à cette tâche.)
is_empty_task: task −> bool	Teste si une tâche est la tâche empty_task
get_duration: task −> int	Renvoie la durée d'une tâche.
get_name: task −> string	Renvoie le nom d'une tâche.
add_task: task −> graph −> unit	Ajoute une tâche au graphe.
get_tasks: graph −> task vect	Renvoie le tableau des tâches du graphe.
add_dependence: task −> task −> graph −> unit	Ajoute un arc de dépendance au graphe, de la première vers la seconde tâche. (Attention, une erreur se produit si les tâches n'existent pas dans le graphe.)
get_successors: task −> graph −> task vect	Renvoie le tableau des tâches successeurs d'une tâche donnée. (Attention, une erreur se produit si la tâche n'existe pas dans le graphe.)
get_predecessors: task −> graph −> task vect	Renvoie le tableau des tâches prédécesseurs d'une tâche donnée. (Attention, une erreur se produit si la tâche n'existe pas dans le graphe.)

TABLE 2 – La table des fonctions de la bibliothèque Graph de manipulation de graphes de tâches.

Question 3. *Écrivez en Caml les fonctions suivantes :*

1. *count_tasks: graph −> int : renvoie le nombre de tâches d'un graphe de tâches.*
2. *count_roots: graph −> int : renvoie le nombre de ses tâches racines.*

Question 4. *Écrivez en Caml une fonction make_root_array: graph −> task vect qui renvoie le tableau des tâches racines du graphe. On pourra si nécessaire renvoyer un tableau plus grand que le nombre de racines. Il sera dans ce cas complété par la tâche empty_task.*

Un tableau est complété par une valeur u si cette valeur n'apparaît pas dans le tableau, ou alors, si elle apparaît, elle n'apparaît pas jusqu'à un certain indice, puis seule cette valeur apparaît au-delà de cet indice.

III Ordonnancement par hauteur

On s'intéresse à la recherche d'un ordonnancement σ d'un graphe acyclique donné G de $n \geq 1$ tâches sur un ensemble de p processeurs. On rappelle que chaque tâche u dure une unité de temps : $\delta(u) = 1$.

set_tag: int −> task −> unit	Affecte une étiquette à une tâche. (Attention, une erreur se produit si une étiquette a déjà été affectée à cette tâche.)
get_tag: task −> int	Renvoie l'étiquette d'une tâche. (Attention, une erreur se produit si aucune étiquette n'a été affectée à cette tâche.)
has_tag: task −> bool	Renvoie vrai si l'étiquette de la tâche a été définie par la fonction set_tag et faux sinon.

Table 3 – La table des fonctions complémentaires pour la manipulation des étiquettes à valeurs entières des tâches.

Une tâche peut être ordonnancée seulement si toutes les tâches dont elle dépend l'ont déjà été. Trouver un ordonnancement d'un graphe G acyclique avec un seul processeur revient donc à énumérer les tâches de ce graphe dans un ordre (total) qui respecte les contraintes de dépendance. Il est donc intéressant d'étiqueter le graphe de tâches selon ces contraintes.

Pour manipuler les étiquettes à valeurs entières des tâches, on étend la bibliothèque Graph avec les fonctions sur les tâches décrites en table 3.

Cet étiquetage fonctionne de la manière suivante : une tâche v reçoit une étiquette $\mathsf{tag}(v)$ portant un numéro strictement supérieur à celui de toutes les étiquettes $\mathsf{tag}(u)$ des tâches u telles que $u \to v$. Notez que plusieurs tâches peuvent recevoir la même étiquette.

Algorithme 1 (étiquetage par hauteur depuis les racines)**.**

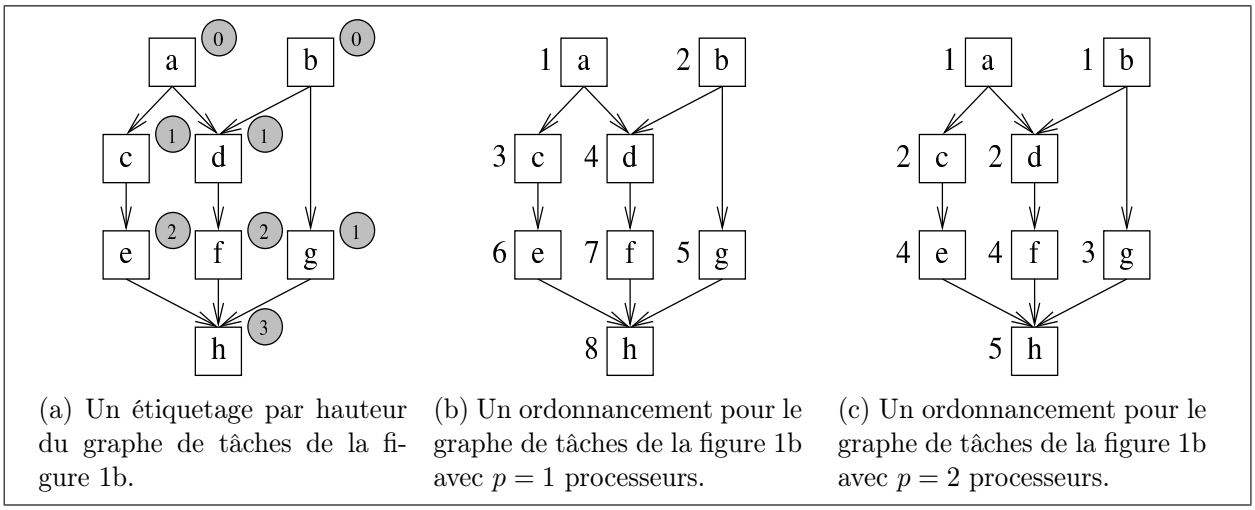
1. Initialement, aucune tâche n'est étiquetée.
2. À l'itération d'ordre $k = 0$, on parcourt l'ensemble des tâches et on affecte l'étiquette 0 aux tâches racines.
3. À l'itération $k > 0$, on parcourt l'ensemble des tâches en repérant toutes les tâches qui n'ont pas encore été étiquetées mais dont toutes les tâches prédécesseurs sont déjà étiquetées. On affecte ensuite à chacune de ces tâches l'étiquette k.
4. L'algorithme termine quand toutes les tâches sont étiquetées.

On dira d'une tâche qui a reçu l'étiquette k qu'elle *porte* l'étiquette k.

La figure 3a présente un exemple d'étiquetage selon l'algorithme 1. Les étiquettes sont notées dans les cercles grisés à droite des tâches.

Question 5. *Écrivez une fonction Caml* check_tags_predecessors: task −> bool *qui prend en paramètre une tâche et renvoie vrai si toutes ses tâches prédécesseurs dans le graphe de tâches sont étiquetées et faux sinon. En particulier, la fonction renvoie vrai si la tâche ne dépend d'aucune tâche (c'est une racine).*

Question 6. *Écrivez une fonction Caml* label_height: graph −> unit *qui prend en paramètre un graphe de tâches et affecte à chaque tâche une étiquette selon l'algorithme 1. Veillez bien à ce qu'aucune erreur ne puisse se produire lors des appels des fonctions de la bibliothèque* Graph.



(a) Un étiquetage par hauteur du graphe de tâches de la figure 1b.

(b) Un ordonnancement pour le graphe de tâches de la figure 1b avec $p = 1$ processeurs.

(c) Un ordonnancement pour le graphe de tâches de la figure 1b avec $p = 2$ processeurs.

FIGURE 3 – Un exemple d'étiquetage par hauteur et un ordonnancement associé.

Pour chaque valeur de l'étiquette k, on pourra par exemple dans un premier temps repérer l'ensemble des tâches à étiqueter, puis dans un second temps étiqueter ces tâches. Chacune de ces deux actions pourra être implémentée par une fonction auxiliaire.

Soit G un graphe de tâches. Soit u une tâche de G. Soit P_u l'ensemble des chemins de la forme $(u_0, u_1, \ldots, u_n)$, où u_0 est une racine de G et $u_n = u$. La tâche u admet un *chemin critique amont* si P_u est non vide et si l'ensemble des longueurs des chemins de P_u est majoré. Les *chemins critiques amont* de u sont alors les chemins de P_u de plus grande longueur. En particulier, le chemin critique amont d'une racine est de longueur nulle et il est unique.

Considérons un graphe de tâches G qui possède un cycle. Soit u une tâche d'un cycle de G. Supposons que P_u soit non vide. Alors, il existe une racine u_0 de G telle que u soit atteignable de cette racine. On peut produire des chemins arbitrairement longs de u_0 à u en parcourant le cycle de manière répétée. La tâche u n'admet donc pas de chemin critique amont.

Question 7. *Soit G un graphe de tâches acyclique. Montrez que toutes ses tâches admettent des chemins critiques amont.*

Question 8. *Supposons que G soit acyclique. Démontrez qu'une tâche u reçoit l'étiquette de valeur k dans l'algorithme 1 si et seulement si la longueur commune des chemins critiques amont de u est k.*

Question 9. *En déduire que l'algorithme termine si et seulement si le graphe est acyclique. Montrez par un exemple ce qui se produit si le graphe possède un cycle.*

Question 10. *Démontrez que si une tâche u porte une étiquette k, alors elle ne pourra être ordonnancée qu'au moins k unités de temps après que la première tâche a été ordonnancée, quel que soit l'ordonnancement* mais aussi quel que soit le nombre de processeurs utilisés.

Soit G un graphe de tâches acyclique. Soit $k_{\max}$ la valeur maximale des étiquettes attribuées aux tâches de G par l'algorithme 1. Soit $T_{\max}$ l'ensemble des tâches de G qui reçoivent cette étiquette $k_{\max}$. Les chemins critiques amont des tâches de $T_{\max}$ sont appelés *chemins critiques* de G. Ces chemins comportent $k_{\max} + 1$ tâches de durée unitaire. Selon la question 10, la *durée d'exécution totale* du graphe de tâches est donc minorée par $k_{\max} + 1$.

Une fois un graphe de tâches G étiqueté selon l'algorithme 1, il est possible de déterminer un ordonnancement avec p processeurs en exécutant les tâches par niveau selon la valeur de leurs étiquettes. Soit T_k l'ensemble des tâches qui portent l'étiquette k.

Algorithme 2 (algorithme d'ordonnancement par hauteur pour p processeurs)**.** Pour chaque valeur de k entre 0 et $k_{\max}$, on exécute les tâches de T_k par lots de p tâches. Pour chaque valeur k, le dernier lot pourra être incomplet. Les processeurs inutilisés sont alors inactifs.

Les figures 3b et 3c présentent des ordonnancements obtenus par cet algorithme à partir de l'étiquetage de la figure 3a, respectivement pour $p = 1$ et $p = 2$ processeurs. On notera en particulier que dans la figure 3c la tâche e reçoit l'étiquette 4 et non 3.

Un ordonnancement sera imprimé de la manière suivante. Chaque ligne entre Begin et End liste les tâches exécutées à un instant donné. Il y a une ligne par instant entre le début et la fin de l'ordonnancement. Le nombre de ligne est donc la durée totale d'exécution de l'ordonnancement.

```
Begin
u v w
x y
...
z

End
```

On pourra utiliser la fonction get_name *de la bibliothèque* Graph *pour obtenir le nom des tâches et utiliser la fonction* print_string *pour l'imprimer.*

Question 11. *Écrivez une fonction Caml* schedule_height: graph −> int −> unit *qui prend en paramètres un graphe G de tâches étiquetées par l'algorithme 1 et un nombre p de processeurs et qui imprime pour chaque instant t la liste des noms des tâches (au plus p) exécutées selon l'algorithme 2 selon le format ci-dessus.*

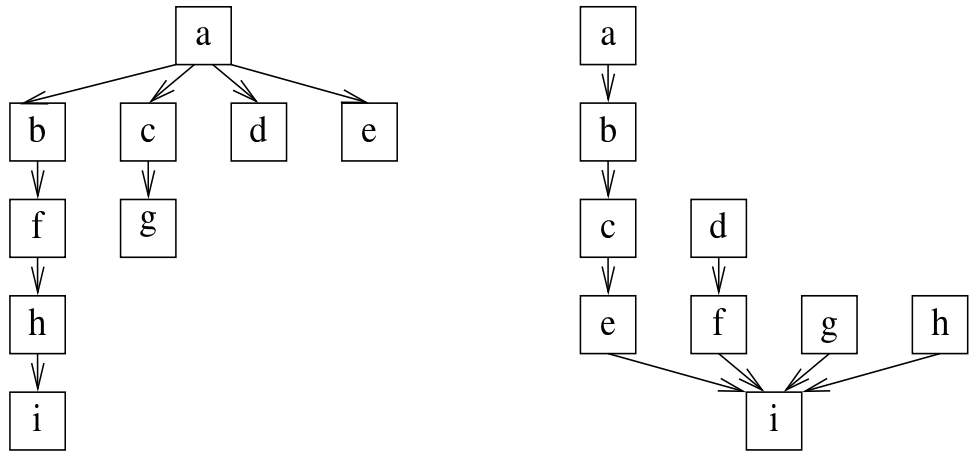
On pourra par exemple diviser le traitement en plusieurs actions implémentées par des fonctions auxiliaires. Pour chaque valeur de l'étiquette k, on extrait l'ensemble des tâches portant cette étiquette. On imprime ensuite cet ensemble par lots de p tâches, avec un lot par ligne, le dernier lot étant éventuellement incomplet.

Une opération élémentaire de l'algorithme est d'accéder à une tâche par l'une des fonctions des bibliothèques présentées dans les tables 2 et 3. On suppose que chaque opération élémentaire coûte 1.

Question 12. *Estimez la complexité de votre fonction* schedule_height *pour l'ordonnancement d'un graphe de n tâches étiquetées par l'algorithme 1 avec p processeurs.*

Question 13. *Justifiez que l'ordonnancement ainsi obtenu est optimal pour un seul processeur, c'est-à-dire quand $p = 1$.*

Un graphe de tâches acyclique *arborescent sortant* est un graphe avec une unique racine dans lequel chaque tâche sauf cette racine a exactement un prédécesseur. C'est par exemple le cas du



(a) Un graphe de tâches *arborescent sortant.*

(b) Un graphe de tâches *arborescent entrant.*

FIGURE 4 – Quelques exemples de graphes de tâches arborescents.

graphe de la figure 4a. Un graphe de tâches acyclique *arborescent entrant* est un graphe avec une unique feuille dans lequel chaque tâche sauf cette feuille a exactement un successeur. C'est par exemple le cas du graphe de la figure 4b.

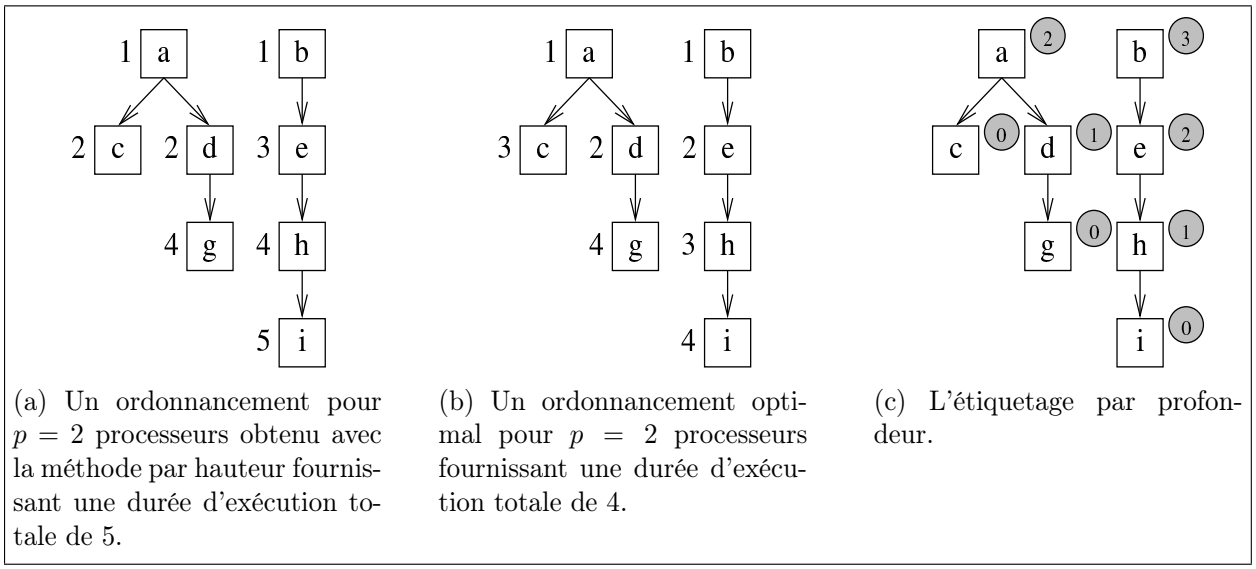
Question 14. *Appliquez l'algorithme 1 aux deux graphes de tâches de la figure 4 pour $p = 2$ processeurs. Quelle est la durée totale d'exécution des ordonnancements produits ? Montrez que ces ordonnancements ne sont pas optimaux pour $p = 2$ processeurs en décrivant pour chacun des graphes un ordonnancement dont la durée totale d'exécution est strictement plus courte.*

IV Ordonnancement par profondeur : l'algorithme de Hu

On suppose dans toute la suite du problème que tous les graphes de tâches considérés sont acycliques.

L'étiquetage par hauteur ne fournit pas assez d'informations pour ordonnancer les tâches de manière optimale car il s'appuie sur la structure du graphe en *amont* des tâches étiquetées. La figure 5 décrit par exemple deux ordonnancements du même graphe dans lequel les tâches exécutées sont exécutées dans l'ordre croissant des étiquettes de hauteur. Cependant, l'ordonnancement 5a conduit l'un des processeurs à rester inactif alors que l'ordonnancement 5b permet l'utilisation constante des deux processeurs.

L'idée de cette partie est de mettre en place un autre étiquetage, cette fois-ci fondé sur les plus longs chemins de tâches en *aval*. En effet, la question 16 ci-dessous montrera que la longueur des plus longs chemins de tâches en aval d'une tâche limite inférieurement la durée d'exécution au-delà de cette tâche. Il sera donc intéressant d'exécuter les tâches avec les plus longs chemins de tâches en aval le plus tôt possible. C'est ce que nous ferons dans l'algorithme 4 ci-dessous dû à Hu.



(a) Un ordonnancement pour $p = 2$ processeurs obtenu avec la méthode par hauteur fournissant une durée d'exécution totale de 5.

(b) Un ordonnancement optimal pour $p = 2$ processeurs fournissant une durée d'exécution totale de 4.

(c) L'étiquetage par profondeur.

FIGURE 5 – Un graphe de tâches pour lequel la méthode d'ordonnancement par hauteur de la section III ne fournit pas un ordonnancement optimal pour $p = 2$ processeurs.

Il suffit donc d'adapter l'algorithme 1 pour étiqueter les tâches à partir des feuilles au lieu des racines.

Algorithme 3 (étiquetage par profondeur depuis les feuilles)**.**

1. Initialement, aucune tâche n'est étiquetée.
2. À l'itération d'ordre $k = 0$, on parcourt l'ensemble des tâches et on affecte l'étiquette 0 aux tâches feuilles.
3. À l'itération $k > 0$, on parcourt l'ensemble des tâches en repérant toutes les tâches qui n'ont pas encore été étiquetées mais dont toutes les tâches *successeurs* sont déjà étiquetées. On affecte à chacune de ces tâches l'étiquette k.
4. L'algorithme termine quand toutes les tâches sont étiquetées.

La figure 5c présente un exemple d'étiquetage obtenu par l'algorithme 3.

Question 15. *Expliquez comment adapter la fonction* label_height *de la question 6 pour obtenir une fonction* label_depth: graph −> unit *qui affecte à chaque tâche une étiquette selon l'algorithme 3.*

Soit G un graphe de tâches. Soit u une tâche de G. Soit P'_u l'ensemble des chemins de la forme $(u_0, u_1, \ldots, u_n)$, où $u_0 = u$ et u_n est une feuille de G. La tâche u admet un *chemin critique aval* si P'_u est non vide et si l'ensemble des longueurs des chemins de P'_u est majoré. Un *chemin critique aval* de u est un chemin de plus grande longueur dans l'ensemble P'_u.

Considérons un graphe de tâches G. (On rappelle que les graphes de tâches sont supposés acycliques ici.) Comme pour les chemins critiques amont, toutes les tâches u de G admettent des chemins critiques aval. La *profondeur* d'une tâche u, $\mathsf{depth}(u)$, est la longueur commune des *chemins critiques aval* de u.

Question 16. *Soit G un graphe de tâches acyclique. Soit u une tâche de G de profondeur h. Supposons que u soit exécutée à l'instant t. Montrez que l'ordonnancement de G ne pourra pas terminer avant l'instant $t + h + 1$ quel que soit le nombre de processeurs utilisés.*

Init, Ready, Done	Les valeurs du type state.
set_state: state −> task −> unit	Affecte un état à une tâche.
get_state: task −> state	Renvoie l'état d'une tâche. (On suppose que l'état des tâches est initialisé à Init lors de leur création.)

TABLE 4 – La table des fonctions complémentaires pour la manipulation des états des tâches.

Une tâche est dite *prête* à être exécutée à un instant t si toutes les tâches dont elle dépend ont été déjà exécutées.

Algorithme 4 (algorithme de Hu pour p processeurs)**.** Soit G un graphe de tâches acyclique. On construit un ordonnancement de G pour p processeurs de manière suivante.

1. L'ordonnancement commence à l'instant $t_0 = 1$.
2. À chaque instant $t \geq t_0$, on considère l'ensemble R_t des tâches de G prêtes à être exécutées. Soit r le cardinal de cet ensemble.
3. Si $r \leq p$, on choisit pour être exécutées à l'instant t les r tâches de R_t et $p - r$ processeurs restent inactifs.
4. Sinon, on trie les tâches de R_t par ordre décroissant de profondeur et on choisit pour être exécutées à l'instant t les p premières tâches.
5. L'ordonnancement se termine quand toutes les tâches de G ont été exécutées.

On notera que le caractère acyclique de G garantit qu'il y a toujours au moins une tâche prête tant qu'il reste dans G une tâche non exécutée.

Question 17. *Appliquez l'algorithme de Hu aux graphes de tâches des figures 4a et 4b pour $p = 2$ processeurs. Quelles sont les durées d'exécution totales obtenues ?*

Pour écrire l'algorithme en Caml, il sera commode de manipuler les états des tâches grâce aux fonctions de la table 4. Ces états appartiennent à un type state qui contient les valeurs suivantes. Une tâche est dans l'état initial Init si elle n'a pas encore été traitée. C'est en particulier le cas lors de sa création par la fonction make_task. Elle est dans l'état Ready si toutes les tâches dont elle dépend ont été exécutées. Elle est dans l'état Done si elle a été exécutée.

L'état de la tâche empty_task n'est pas défini.

Question 18. *Écrivez une fonction Caml is_ready: task −> bool qui renvoie vrai si toutes les tâches dont dépend la tâche passée en argument sont dans l'état Done et faux sinon. En particulier, la fonction renvoie vrai si la tâche passée en argument ne dépend d'aucune tâche.*

Pour implémenter l'algorithme 4, il faudra trier les tâches du graphe G selon la valeur de leurs étiquettes, par ordre décroissant. On supposera donc qu'on dispose d'une fonction Caml sort_tasks_by_decreasing_tags: task vect −> task vect qui réalise une telle opération. Attention, une erreur se produit si l'étiquette d'une des tâches du tableau n'est pas définie, sauf si c'est la tâche spéciale empty_task. Cette tâche est considérée plus petite que toutes les autres tâches dans le tri.

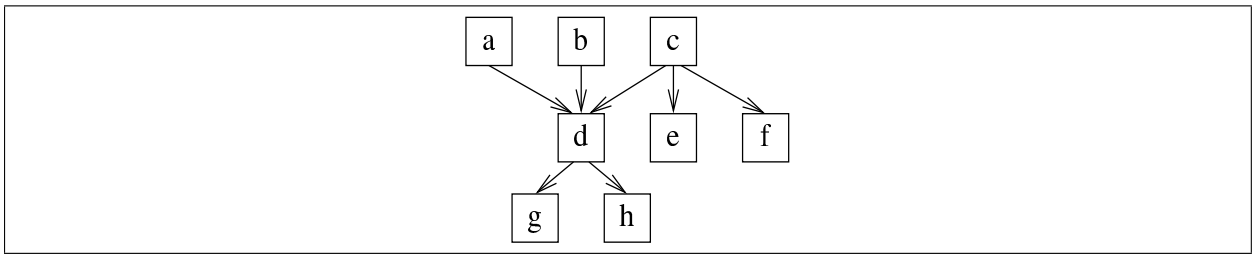


FIGURE 6 – Un graphe de tâches... un peu pathologique.

Question 19. *Écrivez une fonction Caml* schedule_Hu: graph −> int −> unit *qui prend en paramètres un graphe G de tâches étiquetées par l'algorithme 3 et un nombre p de processeurs et qui imprime pour chaque instant t la liste des tâches (au plus p) exécutées selon l'algorithme de Hu. L'impression doit avoir la même forme que pour la question 11.*

On pourra par exemple diviser le traitement en plusieurs actions implémentées par des fonctions auxiliaires.

Une opération élémentaire de l'algorithme est d'accéder à une tâche par l'une des fonctions des bibliothèques présentées dans les tables 2, 3 et 4. On suppose que chaque opération élémentaire coûte 1. On suppose que l'appel à la fonction sort_tasks_by_decreasing_tags sur un tableau de taille n coûte $n \times log_2(n)$.

Question 20. *Estimez la complexité de votre fonction* schedule_Hu *pour l'ordonnancement d'un graphe de n tâches étiquetées par l'algorithme 3 avec p processeurs.*

On peut montrer que l'algorithme de Hu est optimal pour les graphes de tâches *arborescents entrants* comme celui de la figure 4b avec un nombre de processeurs p arbitraire. La preuve de ce résultat est délicate, mais l'une des clés de la preuve est la propriété suivante.

Question 21. *Soit G un graphe de tâches* arborescent entrant *avec p processeurs. Montrez que dans l'algorithme de Hu le cardinal de l'ensemble R_t de tâches prêtes dans G ne peut pas croître au cours de l'algorithme.*

Cette propriété est spécifique du caractère arborescent entrant comme le montre l'exemple de la figure 6.

Question 22. *Montrez que l'algorithme de Hu ne conduit pas nécessairement à un ordonnancement optimal avec $p = 2$ processeurs pour le graphe de tâches de la figure 6.*

On montrera qu'il existe réarrangement trié des tâches de ce graphe qui conduit à un ordonnancement non optimal.

Question 23. *Montrez que l'algorithme de Hu est optimal pour les graphes de tâches* arborescents entrants *avec p processeurs.*

Cette preuve est délicate. Un approche possible est d'examiner l'activité des processeurs au cours d'un ordonnancement. On peut montrer qu'à partir de l'instant où les processeurs ne sont plus tous actifs, l'exécution est conditionnée par un chemin critique du graphe. Ainsi, l'ordonnancement utilise toujours au mieux les processeurs disponibles.

V Conclusion

Ce problème a été initialement étudié par *T. C. Hu* du centre de recherche IBM de Yorktown Heights en 1961. À l'époque, il s'agissait d'organiser le travail d'ouvriers sur une ligne de montage. La preuve d'optimalité dont ce sujet s'est inspiré a été proposée par *James A. M. McHugh* en 1984.

L'algorithme de Hu consiste en fait à définir une mesure de priorité sur les tâches à ordonnancer. La mesure considérée est la profondeur de la tâche. Les tâches sont ordonnancées en privilégiant celles de plus grande priorité. L'exemple de la figure 6 montre que cette approche n'est cependant pas adaptée à des graphes où des tâches ont des dépendances multiples. Dans ce cas, il est important de privilégier certaines tâches par rapport à d'autres parmi toutes les tâches de même profondeur.

En 1972, *E. G. Coffman, Jr.* et *R. L. Graham* ont proposé une amélioration de cette mesure de priorité. L'idée est de départager les tâches de profondeurs égales en privilégiant parmi elles celles qui ont les successeurs les plus profonds en un certain sens. Cet algorithme conduit à un ordonnancement optimal pour les graphes de tâches quelconques, pour $p = 2$ processeurs.

Malheureusement, il ne l'est pas pour le cas $p \geq 3$. Cependant, on peut montrer que tous les algorithmes étudiés ci-dessus conduisent à des ordonnancements dont la durée d'exécution totale n'est pas plus du double de la durée optimale. Cette approximation est heureusement suffisante dans un grand nombre d'applications.

⋆ ⋆ ⋆

ÉCOLES NORMALES SUPÉRIEURES

CONCOURS D'ADMISSION 2015 **FILIÈRE MP – SPÉCIALITÉ INFO**

COMPOSITION D'INFORMATIQUE-MATHÉMATIQUES (ULCR)

(Durée : 4 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.

⋆ ⋆ ⋆

Automates et matrices pondérés

Ce sujet comporte six pages et quatre parties. Il porte sur l'étude d'automates pondérés, de leur modélisation mathématique et de certains de leurs comportements asymptotiques.

La partie I introduit la notion d'**automate pondéré**, de langage pondéré (à chaque mot est associé un poids) et établit leurs premières propriétés. La partie II s'intéresse au poids d'un mot choisi aléatoirement. La partie III fait le lien entre un automate et sa représentation matricielle, et enfin, la partie IV s'intéresse au comportement asymptotique des automates et des matrices pondérés.

Les notions introduites en partie I sont utilisées dans les parties II et III. La partie IV s'appuie sur certains résultats de la partie III. Les résultats d'une question pourront être admis dans la suite du sujet.

Partie I. Automates pondérés

On appelle **automate pondéré** un automate fini dont les arêtes ont un poids. Plus précisément, c'est un sextuplet $\mathcal{A} = (Q, \Sigma, I, F, E, W)$, où

— Q est l'ensemble fini des **états** ;
— Σ est un alphabet fini ;
— $I, F \subseteq Q$ sont respectivement les états initiaux et terminaux de l'automate ;
— $E \subseteq Q \times \Sigma \times Q$ sont les transitions de l'automate ;
— $W : E \to \mathbb{R}$ est la **fonction de pondération** qui à chaque transition associe un nombre réel.

Un **chemin** dans l'automate $\mathcal{A}$ est une suite finie de transitions $s = (e_1, e_2, \ldots, e_k)$ avec $k \in \mathbb{N}$, $e_i = (p_i, a_i, q_i) \in E$ et tels que $\forall i \in \{1 \ldots, k-1\}$, $q_i = p_{i+1}$. Ce chemin est un **cycle** si $p_1 = q_k$ et c'est un **cycle élémentaire** s'il n'existe pas $1 \leq i < j \leq k$ tel que $p_i = p_j$. Enfin, ce chemin est **sans cycle** s'il n'existe pas $1 \leq i \leq j \leq k$ tel que $(e_i, \ldots, e_j)$ est un cycle.

L'entier k est la **longueur de ce chemin** ; le mot $u = a_1 \ldots a_k$ est le **mot (ou étiquette) de ce chemin** ; et le **poids de ce chemin** est

$$W(s) = \sum_{i=1}^{k} W(e_i), \qquad \text{avec la convention} \sum_{i=1}^{0} W(e_i) = 0.$$

Ce chemin est dit **acceptant** si $q_1 \in I$ et $q_k \in F$. Si un mot $u \in \Sigma^*$ est **accepté** par l'automate s'il existe un chemin acceptant de mot u. Le poids $T_{\mathcal{A}}(u)$ d'un mot u dans l'automate est le poids maximal d'un chemin acceptant de mot u. Si un tel chemin n'existe pas, alors par convention, le poids du mot est $-\infty$:

$$T_{\mathcal{A}}(u) = \max \{W(s) \mid s \text{ chemin acceptant de mot } u \text{ dans } \mathcal{A}\}.$$

Un **langage pondéré** L sur un alphabet fini Σ^* est un ensemble tel qu'il existe $w : \Sigma^* \to \mathbb{R} \cup \{-\infty\}$ tel que $L = \{(u, w(u)) \mid u \in \Sigma^*\}$. Le poids de $u \in \Sigma^*$ dans L est $L(u) = w(u)$, et on note $Supp(L) = \{u \in \Sigma^* \mid L(u) \neq -\infty\}$ le support du langage.

Le langage $L_{\mathcal{A}} = \{(u, T_{\mathcal{A}}(u)) \mid u \in \Sigma^*\}$ est le langage pondéré reconnu par l'automate : pour tout mot $u \in \Sigma^*$, $L_{\mathcal{A}}(u) = T_{\mathcal{A}}(u)$.

On supposera dans tout le problème que tous les états des automates pondérés sont utiles : pour tout état $q \in Q$, il existe $i \in I$ et $f \in F$ tels qu'il existe un chemin de i à q et un chemin de q à f.

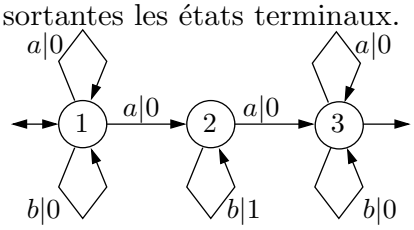
Question 1

a. Montrer que le support du langage d'un automate pondéré est un langage reconnu par un automate fini.

b. Réciproquement, montrer que si L est un langage reconnu par un automate fini, il est possible de trouver un automate pondéré $\mathcal{A}$ de support L tel que $u \in L \Leftrightarrow L_{\mathcal{A}}(u) = 0$.

Question 2 On pose $\Sigma = \{a, b\}$.

a. Quel est le langage pondéré sur Σ^* reconnu par l'automate pondéré suivant ? La notation "$a|w$" signifie que la transition est étiquetée par a et de poids w. Les flèches entrantes désignent les états initiaux et les flèches sortantes les états terminaux.



b. Donner un automate pondéré sur Σ^* reconnaissant le langage $\{(u, \max(|u|_a, |u|_b)), u \in \Sigma^*\}$, où $|u|_a$ (respectivement $|u|_b$) est le nombre d'occurrences de a (respectivement b) dans u.

On fixe $c \in \mathbb{R}$. On s'intéresse au problème de savoir si tous les poids des mots reconnus par un automate pondéré sont inférieurs à c.

On note L le langage pondéré reconnu par l'automate pondéré $\mathcal{A}$.

Question 3

a. Montrer que pour tout mot u, $L(u) \leq c$ si et seulement si tous les chemins acceptants de $\mathcal{A}$ d'étiquette u sont de poids inférieur ou égal à c.

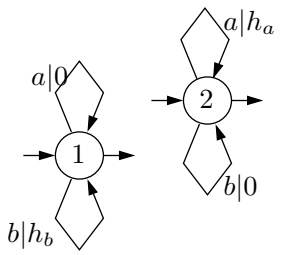
b. Montrer que l'on a l'équivalence entre les deux assertions :

(i) $\forall u \in \Sigma^*$, $L(u) \leq c$.

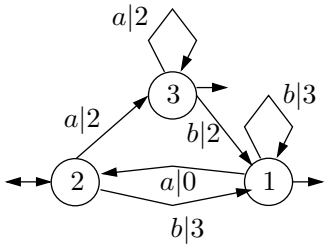
(ii) tous les cycles élémentaires de l'automate pondéré sont négatifs ou nuls et tous les chemins acceptants sans cycle sont de poids inférieur ou égal à c.

Partie II. Poids asymptotique d'un mot choisi aléatoirement

Dans cette partie on s'intéresse aux deux automates $\mathcal{A}_1$ et $\mathcal{A}_2$ suivants, où $h_a, h_b \in \mathbb{N}$.



$\mathcal{A}_1$



$\mathcal{A}_2$

Soient $\Sigma = \{a, b\}$ et $k \in \mathbb{N} \setminus \{0\}$. Considérons l'espace probabilisé $(\Sigma^k, \mathcal{P}(\Sigma^k), \mathbf{P})$ tel que pour tout $i \in \{0, \ldots, k-1\}$, $\mathbf{P}(\Sigma^i a \Sigma^{k-i-1}) = p$ et $\mathbf{P}(\Sigma^i b \Sigma^{k-i-1}) = 1-p = q$ et tel que les événements $E_i = \Sigma^i a \Sigma^{k-i-1}$ forment une famille mutuellement indépendante. Si X est une variable aléatoire sur $\mathbb{N}$, on note $\mathbf{E}[X]$ son espérance et $\mathbf{V}\mathrm{ar}(X)$ sa variance.

Question 4 Soit $u \in \Sigma^k$. Calculer $\mathbf{P}(u)$.

Soit X_k une variable aléatoire sur Σ^k.

Question 5 On s'intéresse dans cette question à l'automate $\mathcal{A}_1$. On pose $\alpha = \max(p\, h_a, q\, h_b)$.

a. Calculer $\mathbf{E}[|X_k|_a]$ et $\mathbf{V}\mathrm{ar}(|X_k|_a)$. (On rappelle que $|X_k|_a$ est le nombre d'occurrences de a dans X_k.)

b. Montrer que pour tout $\beta > p$, il existe $\delta_1 > 0$ tel que $\mathbf{P}(|X_k|_a \geq \beta k) \leq \frac{\delta_1}{k}$ et que pour tout $\gamma < p$, il existe $\delta_2 > 0$ tel que $\mathbf{P}(|X_k|_a \leq \gamma k) \leq \frac{\delta_2}{k}$.

c. Quel est le langage pondéré reconnu par l'automate $\mathcal{A}_1$?

d. Déduire des questions précédentes que $\lim_{k\to\infty} \frac{\mathbf{E}[L_{\mathcal{A}_1}(X_k)]}{k} = \alpha$ (Indication : on pourra d'abord montrer que pour tout $\epsilon > 0$, il existe $\delta > 0$ tel que $\mathbf{P}(|L_{\mathcal{A}_1}(X_k) - \alpha k| \geq \epsilon k) \leq \frac{\delta}{k}$).

Question 6 On s'intéresse maintenant à l'automate $\mathcal{A}_2$. On note Y_i l'état obtenu après lecture du mot $X[i]$, le préfixe de X_k de longueur i, (ici, on a donc $Y_0 = 2$).

a. Pour tout $i \in \{1, \ldots, k\}$, calculer $\mathbf{P}(Y_i = 1)$.

b. En déduire $\mathbf{P}(Y_i = 2)$ et $\mathbf{P}(Y_i = 3)$.

c. En déduire $\lim_{k\to\infty} \frac{\mathbf{E}[L_{\mathcal{A}_2}(X_k)]}{k}$. (Indication : on pourra s'intéresser à $\mathbf{E}[L_{\mathcal{A}_2}(X[i]) - L_{\mathcal{A}_2}(X[i-1])]$).

Partie III. Matrices pondérées

Soit $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty\}$. On appelle **matrice pondérée** une matrice à coefficients dans $\overline{\mathbb{R}}$. Soient $B, C \in \overline{\mathbb{R}}^{n,n}$ deux matrices pondérées. On définit les opérations suivantes :

— $B \oplus C$ par $\forall i, j \in \{1, \ldots, n\}$, $(B \oplus C)_{i,j} = \max(B_{i,j}, C_{i,j})$;

— $B \otimes C$ par $\forall i, j \in \{1, \ldots, n\}$, $(B \otimes C)_{i,j} = \max_{k\in\{1,\ldots,n\}}(B_{i,k} + C_{k,j})$.

Un **vecteur pondéré** est un élément de $\overline{\mathbb{R}}^{1,n}$ (vecteur ligne) ou $\overline{\mathbb{R}}^{n,1}$ (vecteur colonne). On définit de la même manière

— pour $B \in \overline{\mathbb{R}}^{n,n}$ et $u \in \overline{\mathbb{R}}^{n,1}$, $B \otimes u$ par $\forall i \in \{1, \ldots, n\}$, $(B \otimes u)_i = \max_{j\in\{1,\ldots,n\}}(B_{i,j} + u_j)$;

— pour $B \in \overline{\mathbb{R}}^{n,n}$ et $u \in \overline{\mathbb{R}}^{1,n}$, $u \otimes B$ par $\forall i \in \{1, \ldots, n\}$, $(u \otimes B)_i = \max_{j\in\{1,\ldots,n\}}(u_j + B_{j,i})$.

Question 7

a. Montrer que l'opération $\oplus$ est associative : pour tous $B, C, D \in \overline{\mathbb{R}}^{n,n}$, $(B \oplus C) \oplus D = B \oplus (C \oplus D)$.

b. Montrer que l'opération $\oplus$ est commutative : pour tous $B, C \in \overline{\mathbb{R}}^{n,n}$, $B \oplus C = C \oplus B$.

c. Montrer que l'opération $\otimes$ est associative : pour tous $B, C, D \in \overline{\mathbb{R}}^{n,n}$, $(B \otimes C) \otimes D = B \otimes (C \otimes D)$.

d. Montrer que l'opération $\otimes$ est distributive sur $\oplus$: $B, C, D \in \overline{\mathbb{R}}^{n,n}$, $B \otimes (C \oplus D) = (B \otimes C) \oplus (B \otimes D)$ et $(C \oplus D) \otimes B = (C \otimes B) \oplus (D \otimes B)$.

e. Donner une matrice $U \in \overline{\mathbb{R}}^{n,n}$ telle que pour tout $B \in \overline{\mathbb{R}}^{n,n}$, $B \otimes U = U \otimes B = B$.

Soit $M \in \overline{\mathbb{R}}^{n,n}$. On pose $M^0 = U$ et pour tout $k \in \mathbb{N}$, $M^{k+1} = M \otimes M^k$.

Une matrice $M \in \overline{\mathbb{R}}^{n,n}$ peut être représentée par un graphe orienté et pondéré $\mathcal{G}(M) = (S, A, w)$ dont l'ensemble des sommets est $S = \{1, \ldots, n\}$, l'ensemble des arcs est $A = \{(i, j) \mid M_{i,j} \neq -\infty\}$, et la fonction de pondération est $w : A \to \mathbb{R}; (i, j) \mapsto M_{i,j}$.

Question 8 Montrer que $(M^k)_{i,j}$ est égal au poids maximal d'un chemin de longueur k de i vers j dans $\mathcal{G}(M)$ (on utilisera la convention qu'un chemin de longueur 0 est de poids nul et que s'il n'y a pas de chemin de i vers j de longueur k, ce poids est $-\infty$).

Soit $\mathcal{A} = (Q, \Sigma, I, F, E, W)$ un automate pondéré avec $Q = \{1, \ldots, n\}$. Pour tout $a \in \Sigma$, soit $\mathcal{M}(a)$ la matrice du graphe pondéré obtenu en ne gardant que les transitions d'étiquette

$a : \mathcal{G}(\mathcal{M}(a)) = (Q, A, w)$, avec $A = \{(p, q) \mid (p, a, q) \in E\}$ et $w : (p, q) \mapsto W(p, a, q)$. Pour tout $u = a_1 \dots a_k \in \Sigma^*$, on pose $\mathcal{M}(u) = \mathcal{M}(a_1) \otimes \cdots \otimes \mathcal{M}(a_k)$ avec la convention $\mathcal{M}(u) = U$ si u est le mot vide.

Question 9

a. Soient $p, q \in Q$. Montrer que le poids maximum d'un chemin d'étiquette $u \in \Sigma^*$ de p vers q est égal à $\mathcal{M}(u)_{p,q}$.

b. En déduire une expression matricielle pour $L_{\mathcal{A}}(u)$, c'est-à-dire de la forme $M_1 \otimes \cdots \otimes M_\ell$ où M_i sont des matrices ou des vecteurs.

On suppose dans la question suivante que tous les cycles dans l'automate sont de poids négatif ou nul.

Question 10

a. Considérons la matrice $M = \oplus_{a \in \Sigma} \mathcal{M}(a)$ et $M^* = \oplus_{k \in \mathbb{N}} M^k$. Montrer que M^* est bien définie (c'est-à-dire que ses coefficients sont dans $\overline{\mathbb{R}}$).

b. Donner une expression matricielle pour le poids du mot le plus lourd reconnu par l'automate $\mathcal{A}$.

c. Soit $M_{i,j}^{\leq p}$ le poids maximal d'un chemin de i vers j passant uniquement par des sommets intermédiaires q tels que $q \leq p$ (i et j non compris). On pose $M^{\leq p} = (M_{i,j}^{\leq p})_{i,j \leq n}$. Exprimer $M_{i,j}^{\leq p}$ en fonction de $M^{\leq p-1}$. En déduire un algorithme par programmation dynamique qui calcule M^*.

d. Adapter cet algorithme pour qu'il réponde `vrai` si tous les mots reconnus par l'automate sont de poids inférieur ou égal à c et `faux` sinon, sans faire d'hypothèse a priori sur les poids des cycles de l'automate.

Partie IV. Croissance asymptotique du poids des mots

On dit qu'une matrice pondérée est **irréductible** si le graphe orienté qui lui est associé est fortement connexe. Soit M une matrice pondérée irréductible. On suppose dans les questions 11 à 14 que le cycle de poids maximum dans ce graphe est **de poids exactement égal à zéro**.

On appelle vecteur propre de M associé à la valeur propre $\lambda \in \mathbb{R}$ un vecteur $v \neq (-\infty, \dots, -\infty)$ tel que $M \otimes v = \lambda \otimes v$, où $\lambda \otimes v$ est le vecteur $(\lambda + v_1, \dots, \lambda + v_n)$.

Soient $v, v' \in \overline{\mathbb{R}}^{n,1}$. On note $v \leq v'$ si pour tout $i \in \{1, \dots, n\}$, $v_i \leq v'_i$ et de la même manière, on note $v \geq v'$ si pour tout $i \in \{1, \dots, n\}$, $v_i \geq v'_i$.

Question 11 Soient $v \neq (-\infty, \dots, -\infty)$ et $\lambda \in \mathbb{R}$ tels que $M \otimes v \leq \lambda \otimes v$.

a. Montrer que pour tout $k \in \mathbb{N}$, $M^k \otimes v \leq (k\lambda) \otimes v$ puis que $\forall i \in \{1, \dots, n\}$, $v_i \neq -\infty$.

b. En déduire que $\lambda \geq 0$.

Question 12 Soient $v \neq (-\infty, \dots, -\infty)$ et $\lambda \in \mathbb{R}$ tels que $M \otimes v \geq \lambda \otimes v$.

a. Montrer que pour tout $i \in \{1, \ldots, n\}$ il existe $j \in \{1, \ldots, n\}$ tel que $\lambda + v_i \leq M_{i,j} + v_j$.
b. En déduire que $\lambda \leq 0$.

Soit I^* l'ensemble des sommets de $\mathcal{G}(M)$ qui sont sur un cycle de poids 0.

Question 13 Montrer que pour $i \in I^*$, le i-ème vecteur colonne de M^* est un vecteur propre pour M. Quel est l'ensemble des valeurs propres de M ?

Pour $i, j \in \{1, \ldots, n\}$ et $i^* \in I^*$, on note $(M^k)_{i,i^*,j} = \max_{k_1+k_2=k}(M^{k_1})_{i,i^*} + (M^{k_2})_{i^*,j}$ le poids du chemin le plus lourd de i vers j de longueur k et passant par i^*.

Question 14
a. Montrer qu'il existe $d \in \mathbb{N} \setminus \{0\}$ tel que pour tout $i^* \in I^*$, $(M^d)_{i^*,i^*} = 0$.
b. Soient $i, j \in \{1, \ldots, n\}$ et $i^* \in I^*$. Montrer qu'il existe $K > 0$ tel que pour tout $k \geq K$, $(M^{(k+1)d})_{i,i^*,j} = (M^{kd})_{i,i^*,j}$.
c. Soient $i, j \in \{1, \ldots, n\}$. Montrer qu'il existe K' tel que pour tout $k \geq K'$, $M^{kd}_{i,j} = \max_{i^* \in I^*}(M^{kd})_{i,i^*,j}$.
d. En déduire qu'il existe K_0 et d tels que pour tout $k \geq K_0$, $M^{k+d} = M^k$.

On se place maintenant dans le cas où le cycle de poids maximal n'est plus égal à 0, mais la matrice M est toujours irréductible. On pose

$$\rho(M) = \max_{k \leq n-1} \frac{\max_{i \in \{1,\ldots,n\}}(M^k)_{i,i}}{k}.$$

Question 15
a. Donner une interprétation de $\rho(M)$ en terme de graphe.
b. Soit $\lambda \in \mathbb{R}$. Montrer que λ est une valeur propre de M si et seulement si 0 est valeur propre de la matrice $(-\lambda) \otimes M$, où $(-\lambda) \otimes M = (M_{i,j} - \lambda)_{i,j \in \{1,\ldots,n\}}$.
c. En déduire que $\rho(M)$ est l'unique valeur propre de M.
d. Montrer qu'il existe K et d tels que pour tout $k \geq K$, $M^{k+d} = d\rho(M) \otimes M^k$.

Question 16 Montrer que $\rho(M)$ est valeur propre de M même si M n'est pas irréductible.

Soit $\mathcal{A}$ un automate pondéré dont tous les états sont terminaux. Pour toute suite $(u_k)_{k \geq 1} \in \Sigma^{\mathbb{N}}$, on note $u[k] = u_1 \cdots u_k$.

Question 17 Montrer qu'il existe λ tel que :
(i) pour toute suite $(u_k)_{k \geq 1}$, $\lim_{k \to \infty} \frac{L_{\mathcal{A}}(u[k])}{k} \leq \lambda$ (si cette limite existe) ;
(ii) il existe une suite $(u_k)_{k \geq 1}$ telle que $\lim_{k \to \infty} \frac{L_{\mathcal{A}}(u[k])}{k} = \lambda$.

* *
*

Problème : Automates d'arbre

Préliminaire concernant la programmation

Il faudra coder des fonctions à l'aide du langage de programmation Caml, tout autre langage étant exclu. Lorsque le candidat écrira une fonction, il pourra faire appel à d'autres fonctions définies dans les questions précédentes ; il pourra aussi définir des fonctions auxiliaires. Quand l'énoncé demande de coder une fonction, il n'est pas nécessaire de justifier que celle-ci est correcte, sauf si l'énoncé le demande explicitement. Enfin, si les paramètres d'une fonction à coder sont supposés vérifier certaines hypothèses, il ne sera pas utile dans l'écriture de cette fonction de tester si les hypothèses sont bien vérifiées.

Dans les énoncés du problème, un même identificateur écrit dans deux polices de caractères différentes désignera la même entité, mais du point de vue mathématique pour la police en italique (par exemple n) et du point de vue informatique pour celle en romain avec espacement fixe (par exemple `n`).

Fonctions utilitaires

Dans cette partie, on code quelques fonctions générales qui seront utiles par la suite. On ne cherchera pas à proposer l'implémentation la plus efficace possible de chaque fonction.

Quand il est question de donner la complexité d'une fonction, il s'agit de calculer la complexité asymptotique en temps, en notation $O(\cdot)$, de cette fonction dans le pire des cas. Il est inutile de donner une preuve de cette complexité.

☐ 1 – Coder une fonction Caml `contient: 'a list -> 'a -> bool` telle que `contient li x` renvoie un booléen qui vaut *Vrai* si et seulement si la liste `li` contient l'élément `x`. Donner la complexité de cette fonction.

☐ 2 – En utilisant la fonction `contient`, coder une fonction Caml `union: 'a list -> 'a list -> 'a list` telle que `union l1 l2`, où `l1` et `l2` sont deux listes d'éléments sans doublon dans un ordre arbitraire, renvoie une liste sans doublon contenant l'union des éléments des deux listes, dans un ordre arbitraire. Donner la complexité de cette fonction.

☐ 3 – En utilisant la fonction `union`, coder une fonction Caml `fusion: 'a list list -> 'a list` telle que `fusion l`, où `l` est une liste de listes d'éléments, chacune de ces listes étant sans doublon, renvoie une liste de tous les éléments contenus dans au moins une des listes de la liste `l`, sans doublon et dans un ordre arbitraire. En notant $l = (l_1, \ldots, l_k)$ la liste codée par `l` et en posant $L \coloneqq \sum_{j=1}^{k} |l_j|$, donner la complexité de la fonction `fusion` en fonction de L.

☐ 4 – Coder `produit: 'a list -> 'b list -> ('a * 'b) list`, telle que `produit l1 l2` renvoie une liste de tous les couples `(x,y)` avec `x` un élément de `l1` et `y` un élément de `l2`. On supposera les listes `l1` et `l2` sans doublon. La liste résultante doit avoir pour longueur le produit des longueurs des deux listes. Donner la complexité de cette fonction.

Arbres binaires étiquetés

Soit $\Sigma = \{\alpha_0, \ldots, \alpha_{m-1}\}$ un ensemble fini non vide de m symboles, appelé *alphabet*. En Caml, on représentera le symbole α_k (pour $0 \leqslant k \leqslant m-1$) par l'entier k. Cet alphabet sera supposé fixé dans tout l'énoncé.

Un *arbre binaire* étiqueté par Σ (simplement appelé, dans ce problème, *arbre*) est soit *l'arbre vide* (noté ε), soit un quintuplet (S, r, λ, g, d), où :

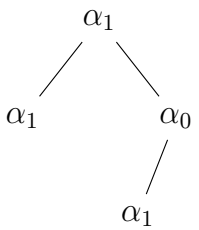
(i) S est un ensemble fini non vide dont les éléments sont appelés *nœuds* ;
(ii) $r \in S$ est la *racine* de S ;
(iii) $\lambda : S \to \Sigma$ est une application associant à chaque nœud de S une *étiquette* de Σ ;
(iv) $g : S_g \to S \backslash \{r\}$, où $S_g \subseteq S$, est une application injective associant à un nœud u de S_g un nœud appelé *fils gauche* de u ;
(v) $d : S_d \to S \backslash \{r\}$, où $S_d \subseteq S$, est une application injective associant à un nœud u de S_d un nœud appelé *fils droit* de u.

On dit qu'un nœud v est *descendant* d'un nœud u s'il existe une séquence de nœuds $u_0, u_1, \ldots, u_p \in S$ avec $p > 0$ telle que $u_0 = u$, $u_p = v$ et, pour tout $0 \leqslant k \leqslant p-1$, soit $u_{k+1} = g(u_k)$, soit $u_{k+1} = d(u_k)$.

On requiert que tout nœud sauf la racine soit le fils gauche ou droit d'un unique nœud, et qu'aucun nœud ne soit à la fois fils gauche et fils droit :

$$\forall u \in S \backslash \{r\}, |g^{-1}(\{u\})| + |d^{-1}(\{u\})| = 1.$$

Par ailleurs, tout nœud de $S \backslash \{r\}$ doit être descendant de r.



Un arbre admet une représentation graphique naturelle. Par exemple, l'arbre $t_0 = (\{u_0, u_1, u_2, u_3\}, u_0, \lambda, g, d)$ est représenté ci-contre, avec :
— $g(u_0) = u_1$, $g(u_2) = u_3$ et $d(u_0) = u_2$;
— $\lambda(u_0) = \lambda(u_1) = \lambda(u_3) = \alpha_1$ et $\lambda(u_2) = \alpha_0$.

On utilisera en Caml le type de données suivant pour coder un arbre :

```
type Arbre = Noeud of Noeud | Vide
 and Noeud = { etiquette: int; gauche: Arbre; droit: Arbre; };;
```

Dans ce codage, un arbre non vide est représenté par une instance du type `Noeud` décrivant sa racine ; un nœud est décrit par son étiquette (codée comme un entier), son fils gauche, son fils droit ; le fils gauche et le fils droit peuvent, à nouveau, être décrits par une instance du type `Noeud`, ou par le constructeur `Vide`, qui décrit leur absence.

Par exemple, l'arbre t_0 pourra être décrit par la variable `t0` comme suit :

```
let t0 = Noeud {
  etiquette=1;
  gauche=Noeud
    {etiquette=1; gauche=Vide; droit=Vide};
  droit=Noeud
    {etiquette=0;
     gauche=Noeud {etiquette=1; gauche=Vide; droit=Vide};
     droit=Vide}
};;
```

☐ 5 – Pour simplifier l'écriture d'arbres, coder en Caml une fonction `arbre` telle que si `x` représente une étiquette x et `ag` et `ad` représentent deux arbres a_g et a_d, alors `arbre x ag ad` représente un arbre dont la racine est étiquetée par x, avec pour fils gauche la racine de a_g (avec ses propres fils) et pour fils droit la racine de a_d (avec ses propres fils). Ainsi, cette fonction doit permettre de construire t_0 avec :

```
let t0 = arbre 1 (arbre 1 Vide Vide)
                 (arbre 0 (arbre 1 Vide Vide) Vide);;
```

☐ 6 – Coder en Caml une fonction `taille_arbre` prenant en argument une variable `t` représentant un arbre t et renvoyant le nombre de nœuds de l'arbre t.

Langages d'arbres

Soit $\mathcal{T}^\Sigma$ l'ensemble de tous les arbres étiquetés par Σ. Un *langage d'arbres* sur un alphabet Σ est un ensemble (fini ou infini) d'arbres étiquetés par Σ, c'est-à-dire un sous-ensemble de $\mathcal{T}^\Sigma$.

On considère dans ce problème certains langages particuliers, tous définis sur l'alphabet $\{\alpha_0, \alpha_1\}$:

— L_0 est l'ensemble des arbres dont au moins un nœud est étiqueté par α_0.

— Un arbre est *complet* s'il ne contient aucun nœud ayant un seul fils (c'est-à-dire, tout nœud a un fils gauche si et seulement s'il a un fils droit) ; conventionnellement, ε est considéré comme complet. Le langage L_{complet} est l'ensemble de tous les arbres complets.

— Un arbre (S, r, λ, g, d) est un *arbre-chaîne* s'il a uniquement des fils gauches : $d^{-1}(S \backslash \{r\}) = \varnothing$; conventionnellement, ε est également un arbre-chaîne. Le langage $L_{\text{chaîne}}$ est l'ensemble de tous les arbres-chaînes.

— Un arbre (S, r, λ, g, d) est *impartial* s'il a autant de fils gauches que de fils droits, c'est-à-dire si on a $|g(S)| = |d(S)|$; conventionnellement, ε est également impartial. On note $L_{\text{impartial}}$ l'ensemble de tous les arbres impartiaux.

☐ 7 – Pour chacun des quatre langages L_0, L_{complet}, $L_{\text{chaîne}}$, $L_{\text{impartial}}$, donner (sans justification) un exemple d'arbre avec au moins deux nœuds qui appartient au langage, et un exemple d'arbre avec au moins deux nœuds qui n'y appartient pas.

☐ 8 – Démontrer que tout arbre complet est impartial, mais que la réciproque est fausse.

☐ 9 – Démontrer que tout arbre impartial non vide a un nombre impair de nœuds.

Automates d'arbres descendants déterministes

Un *automate d'arbres descendant déterministe* (ou simplement *automate descendant déterministe*) sur l'alphabet Σ est un quadruplet $\mathcal{A}^{\downarrow} = (Q, q_0, F, \delta)$ où :

(i) Q est un ensemble fini non vide dont les éléments sont appelés *états* ;
(ii) $q_0 \in Q$ est appelé *état initial* ;
(iii) $F \subseteq Q$ est un ensemble dont les éléments sont appelés *états finals* ;
(iv) $\delta : Q \times \Sigma \to Q \times Q$ est une application appelée *fonction de transition* ; pour tout $q \in Q$, pour tout $\alpha \in \Sigma$, $\delta(q, \alpha)$ est un couple d'états (q_g, q_d).

Un automate descendant déterministe $\mathcal{A}^{\downarrow} = (Q, q_0, F, \delta)$ *reconnaît* un arbre t si :

— soit $t = \varepsilon$ et $q_0 \in F$;
— soit $t = (S, r, \lambda, g, d)$ et il existe une application $\varphi : S \to Q$ avec :
  (i) $\varphi(r) = q_0$;
  (ii) pour tout $u \in S$, si $(q_g, q_d) = \delta(\varphi(u), \lambda(u))$:
    — si $g(u)$ est défini ($u \in S_g$), alors on a $\varphi(g(u)) = q_g$, sinon on a $q_g \in F$;
    — si $d(u)$ est défini ($u \in S_d$), alors on a $\varphi(d(u)) = q_d$, sinon on a $q_d \in F$.

Noter que quand une telle application φ existe, elle est nécessairement unique.

Le *langage reconnu* par un automate descendant déterministe $\mathcal{A}^{\downarrow}$, noté $\mathcal{L}(\mathcal{A}^{\downarrow})$, est l'ensemble de tous les arbres reconnus par $\mathcal{A}^{\downarrow}$.

☐ 10 – Donner un automate descendant déterministe reconnaissant le langage $L_{\text{chaîne}}$; aucune justification n'est demandée.

☐ 11 – Montrer qu'il n'existe pas d'automate descendant déterministe qui reconnaît L_0.

En Caml, un état q_i de $Q = \{q_0, \ldots, q_{n-1}\}$ est codé par l'entier i. L'ensemble des états finals F est codé par un vecteur de booléens `finals_desc` de taille n, tel que `finals_desc.(i)` contient *Vrai* si et seulement si $q_i \in F$. Enfin, les transitions sont codées par une matrice de couples d'entiers, telle que `transitions_desc.(i).(k)` est le couple `(g,d)` vérifiant $(q_g, q_d) = \delta(q_i, \alpha_k)$.

On représente ainsi en Caml un automate descendant déterministe (Q, q_0, F, δ) avec le type suivant :

```
type Automate_Descendant_Deterministe = {
  finals_desc: bool vect;
  transitions_desc: (int*int) vect vect
};;
```

☐ 12 – Pour un automate descendant déterministe $\mathcal{A}^{\downarrow} = (Q, q_0, F, \delta)$ et $q \in Q$, on note $\mathcal{A}_q^{\downarrow}$ l'automate descendant déterministe (Q, q, F, δ) identique à $\mathcal{A}^{\downarrow}$ sauf pour l'état initial. Coder une fonction `applique_desc` telle que `applique_desc add q t`, où `add` représente un automate descendant déterministe $\mathcal{A}^{\downarrow} = (Q, q_0, F, \delta)$, `q` un état $q \in Q$ et `t` un arbre $t = (S, r, \lambda, g, d)$, renvoie un booléen qui vaut *Vrai* si et seulement $\mathcal{A}_q^{\downarrow}$ reconnaît t.

☐ 13 – En utilisant `applique_desc`, coder une fonction `evalue_desc` telle que `evalue_desc add t`, où `add` représente un automate descendant déterministe $\mathcal{A}^{\downarrow}$ et `t` un arbre t, renvoie un booléen qui vaut *Vrai* si et seulement si $\mathcal{A}^{\downarrow}$ reconnaît t.

Automates descendants et langages rationnels de mots

À tout mot non vide $x = x_1 \dots x_l$ avec $x_1, \dots, x_l \in \Sigma$, on associe un arbre-chaîne $\mathit{chaîne}(x) = (\{u_1 \dots u_l\}, u_1, \lambda, g, d)$ vérifiant : pour $1 \leqslant i \leqslant l, \lambda(u_i) = x_i$ et pour $1 \leqslant i \leqslant l-1, g(u_i) = u_{i+1}$, d n'étant défini pour aucun u_i, et $g(u_l)$ étant non défini. Par convention, $\mathit{chaîne}(\varepsilon) = \varepsilon$ (où le premier ε est le mot vide, le second l'arbre vide).

Pour un langage de mots L, on définit le langage d'arbres $\mathit{chaîne}(L) \coloneqq \{\mathit{chaîne}(x) \mid x \in L\}$.

☐ 14 – Soit L un langage de mots, supposé rationnel. Il existe donc un automate de mots déterministe $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ reconnaissant L. Soient $q_1', q_2' \notin Q$. On construit l'automate d'arbres descendant déterministe $\mathcal{A}^{\downarrow} = (Q \cup \{q_1', q_2'\}, q_0, F \cup \{q_1'\}, \delta')$ avec pour $(q, \alpha) \in Q \times \Sigma, \delta'(q, \alpha) \coloneqq (\delta(q, \alpha), q_1')$ et, pour $q \in \{q_1', q_2'\}$ et pour $\alpha \in \Sigma$, $\delta'(q, \alpha) \coloneqq (q_2', q_2')$. Démontrer que $\mathcal{A}^{\downarrow}$ reconnaît $\mathit{chaîne}(L)$.

☐ 15 – Montrer que pour tout langage de mots L, si $\mathit{chaîne}(L)$ est reconnu par un automate d'arbres descendant déterministe, alors L est rationnel.

☐ 16 – Soit $L_{\text{égal}}$ le langage de mots sur l'alphabet $\{\alpha_0, \alpha_1\}$ formé des mots contenant autant de α_0 que de α_1. Supposons par l'absurde qu'il existe un automate (de mots) déterministe $\mathcal{A}_{\text{égal}}$ reconnaissant $L_{\text{égal}}$ et soit k le nombre d'états de $\mathcal{A}_{\text{égal}}$. En considérant le mot $x = \alpha_0^k \alpha_1^k$, montrer que l'on aboutit à une contradiction, et que donc $L_{\text{égal}}$ n'est pas un langage rationnel.

En déduire qu'il n'existe aucun automate descendant déterministe reconnaissant $\mathit{chaîne}(L_{\text{égal}})$.

Automates d'arbres ascendants

Un *automate d'arbres ascendant* (ou simplement *automate ascendant*) sur l'alphabet Σ est un quadruplet $\mathcal{A}^{\uparrow} = (Q, I, F, \Delta)$ où :

(i) Q est un ensemble fini non vide dont les éléments sont appelés *états* ;
(ii) $I \subseteq Q$ est un ensemble dont les éléments sont appelés *états initiaux* ;
(iii) $F \subseteq Q$ est un ensemble dont les éléments sont appelés *états finals* ;
(iv) $\Delta : Q \times Q \times \Sigma \to \mathcal{P}(Q)$, où $\mathcal{P}(X)$ désigne l'ensemble des parties de X, est une application appelée *fonction de transition* ; pour tout $(q_g, q_d) \in Q \times Q$, et tout $\alpha \in \Sigma$, $\Delta(q_g, q_d, \alpha)$ est un ensemble d'états.

Par exemple, on définit un automate ascendant $\mathcal{A}_0^{\uparrow} = (Q, I, F, \Delta)$ sur l'alphabet $\{\alpha_0, \alpha_1\}$ avec :

(i) $Q = \{q_0, q_1\}$;
(ii) $I = \{q_0\}$;
(iii) $F = \{q_1\}$;
(iv) Δ est donnée par la table de transition suivante :

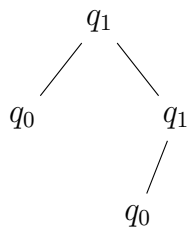
	Σ	
$Q \times Q$	α_0	α_1
(q_0, q_0)	$\{q_1\}$	$\{q_0\}$
(q_0, q_1)	$\{q_1\}$	$\{q_1\}$
(q_1, q_0)	$\{q_1\}$	$\{q_1\}$
(q_1, q_1)	$\{q_1\}$	$\{q_1\}$

Un automate ascendant $\mathcal{A}^{\uparrow} = (Q, I, F, \Delta)$ *reconnaît* un arbre t si :

— soit $t = \varepsilon$ et $I \cap F \neq \varnothing$;
— soit $t = (S, r, \lambda, g, d)$ et il existe une application $\varphi : S \to Q$ avec :
  (i) $\varphi(r) \in F$;
  (ii) pour tout $u \in S$, il existe $(q_g, q_d) \in Q \times Q$ tels que $\varphi(u) \in \Delta(q_g, q_d, \lambda(u))$ et :
    — si $g(u)$ est défini ($u \in S_g$), alors on a $\varphi(g(u)) = q_g$, sinon $q_g \in I$;
    — si $d(u)$ est défini ($u \in S_d$), alors on a $\varphi(d(u)) = q_d$, sinon $q_d \in I$.

Noter que, contrairement au cas des automates descendants déterministes, quand une telle application φ existe, elle n'est pas nécessairement unique.

On observe que $\mathcal{A}_0^{\uparrow}$ ne reconnaît pas ε (car $\{q_0\} \cap \{q_1\} = \varnothing$) et que $\mathcal{A}_0^{\uparrow}$ reconnaît l'arbre t_0 défini page 3 *via* l'application représentée ci-dessous :



Le *langage reconnu* par un automate ascendant $\mathcal{A}^{\uparrow}$, noté $\mathcal{L}(\mathcal{A}^{\uparrow})$, est l'ensemble de tous les arbres reconnus par $\mathcal{A}^{\uparrow}$. On dit qu'un langage d'arbres L est *rationnel* s'il existe un automate ascendant $\mathcal{A}^{\uparrow}$ qui reconnaît L.[1]

On dit qu'un automate ascendant (Q, I, F, Δ) est *déterministe* si $|I| = 1$ et, pour tout $(q_g, q_d, \alpha) \in Q \times Q \times \Sigma$, $|\Delta(q_g, q_d, \alpha)| = 1$.

☐ 17 – Montrer que l'on a $\mathcal{L}(\mathcal{A}_0^{\uparrow}) = L_0$.

☐ 18 – Soit L un langage d'arbres. Montrer que s'il existe un automate descendant déterministe $\mathcal{A}^{\downarrow}$ reconnaissant L, alors L est un langage d'arbres rationnel.

En Caml, un état q_i de $Q = \{q_0, \ldots, q_{n-1}\}$ est codé par l'entier i. L'ensemble des états initiaux I est codé par leur liste `initiaux_asc`, dans un ordre arbitraire ; l'ensemble des états finals F est codé par un vecteur de booléens `finals_asc` de taille n, dont la composante de position i contient *Vrai* si et seulement si $q_i \in F$. Finalement, les transitions sont codées par un tableau tridimensionnel de listes d'entiers, telle que `transitions_asc.(i).(j).(k)` est une liste dans un ordre arbitraire des états `q` avec $q \in \Delta(q_i, q_j, \alpha_k)$.

On représente ainsi en Caml un automate ascendant (Q, I, F, Δ) avec le type ci-dessous :

```
type Automate_Ascendant = {
  initiaux_asc: int list;
  finals_asc: bool vect;
  transitions_asc: int list vect vect vect
};;
```

1. La notion de langages d'arbres rationnels est distincte de la notion de langages de mots rationnels.

L'automate $\mathcal{A}_0^{\uparrow}$ peut alors être codé par :

```
let aa0 = {
  initiaux_asc=[0];
  finals_asc = [| false; true |];
  transitions_asc=[|
    [| [| [1]; [0] |]; [| [1]; [1] |] |];
    [| [| [1]; [1] |]; [| [1]; [1] |] |]
  |]
};;
```

☐ 19 – Coder une fonction `nombre_etats_asc` prenant en argument la représentation `aa` d'un automate ascendant et renvoyant le nombre d'états de cet automate.

☐ 20 – Coder une fonction `nombre_symboles_asc` prenant en argument la représentation `aa` d'un automate ascendant et renvoyant le nombre de symboles de l'alphabet sur lequel cet automate est défini.

☐ 21 – Coder une fonction Caml `applique_asc` telle que `applique_asc aa t`, où `aa` représente un automate ascendant $\mathcal{A}^{\uparrow} = (Q, I, F, \Delta)$ et `t` un arbre t, renvoie une liste sans doublon des états q pour lesquels il existe une application $\varphi : S \to Q$ avec $\varphi(r) = q$ qui vérifie la condition (ii) de la définition de reconnaissance d'un arbre par un automate ascendant page 7. Si $t = \varepsilon$, la fonction `applique_asc` doit renvoyer la liste des états initiaux de $\mathcal{A}^{\uparrow}$. On pourra utiliser les fonctions utilitaires des questions 1 à 4.

☐ 22 – En utilisant `applique_asc`, coder une fonction `evalue_asc` telle que `evalue_asc aa t`, où `aa` représente un automate ascendant $\mathcal{A}^{\uparrow}$ et `t` un arbre t, renvoie un booléen qui vaut *Vrai* si et seulement si $\mathcal{A}^{\uparrow}$ reconnaît t. On pourra utiliser la fonction `contient`.

☐ 23 – Montrer qu'un langage d'arbres L est un langage d'arbres rationnel si et seulement s'il existe un automate ascendant *déterministe* reconnaissant L.

☐ 24 – Coder deux fonctions Caml `identifiant_partie: int list -> int` et `partie_identifiant: int -> int list` réciproques l'une de l'autre, codant une bijection entre les parties de $[\![0\,;n-1]\!]$ (une partie étant représentée par une liste d'entiers sans doublon, dans un ordre arbitraire) et les entiers de 0 à $2^n - 1$. On rappelle qu'en Caml l'expression `1 lsl i` calcule l'entier 2^i.

☐ 25 – En s'appuyant sur `identifiant_partie` et `partie_identifiant` et sur la réponse à la question 23, coder une fonction `determinise_asc` prenant en argument la représentation `aa` d'un automate ascendant $\mathcal{A}^{\uparrow}$ et renvoyant la représentation d'un automate ascendant déterministe reconnaissant le même langage que $\mathcal{A}^{\uparrow}$.

☐ 26 – Montrer que si L est un langage d'arbres rationnel, alors $\mathcal{T}^{\Sigma}\backslash L$ est un langage d'arbres rationnel.

☐ 27 – Coder une fonction `complementaire_asc` prenant en entrée la représentation `aa` d'un automate ascendant reconnaissant un langage L et renvoyant la représentation d'un automate ascendant reconnaissant $\mathcal{T}^{\Sigma}\backslash L$.

☐ 28 – Montrer que si L_1 et L_2 sont deux langages d'arbres rationnels, alors $L_1 \cup L_2$ est un langage d'arbres rationnel.

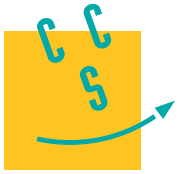
☐ 29 – Coder une fonction `union_asc` prenant en entrée les représentations `aa1` et `aa2` de deux automates ascendants reconnaissant respectivement les langages L_1 et L_2 et renvoyant la représentation d'un automate ascendant reconnaissant $L_1 \cup L_2$.

☐ 30 – Montrer que si L_1 et L_2 sont deux langages d'arbres rationnels, alors $L_1 \cap L_2$ est un langage d'arbres rationnel.

☐ 31 – Coder une fonction `intersection_asc` prenant en entrée les représentations `aa1` et `aa2` de deux automates ascendants reconnaissant respectivement les langages L_1 et L_2 et renvoyant la représentation d'un automate ascendant reconnaissant $L_1 \cap L_2$.

☐ 32 – Sans chercher à utiliser les propriétés de clôture par union, complémentation ou intersection, montrer que le langage $L_{\text{impartial}}$ n'est pas un langage d'arbres rationnel.

Fin de l'épreuve



CONCOURS CENTRALE•SUPÉLEC

Option informatique

MP

2015

4 heures Calculatrices autorisées

Les candidats devront répondre aux questions de programmation en utilisant le langage Caml. Ils devront donner le type, ou la signature, de chaque fonction écrite, sauf lorsqu'il est indiqué par le sujet : dans ce cas la réponse doit être d'un type compatible avec la signature proposée. Par exemple, la fonction `cons` *définie par*

```
let cons x l = x :: l
```

est du type `'a -> 'a list -> 'a list` *qui est compatible avec la signature* `int -> int list -> int list`. *L'énoncé indique la signature attendue, toute réponse de type compatible est acceptée.*

I Graphes d'intervalles

On considère le problème concret suivant : des cours doivent avoir lieu dans un intervalle de temps précis (de 8h à 9h55, ...) et on cherche à attribuer une salle à chaque cours. On souhaite qu'à tout moment une salle ne puisse être attribuée à deux cours différents et on aimerait utiliser le plus petit nombre de salles possibles.

Ce problème d'allocation de ressources (ici les salles) en fonction de besoins fixes (ici les horaires des cours) intervient dans de nombreuses situations très diverses (allocation de pistes d'atterrissage aux avions, répartition de la charge de travail sur plusieurs machines, ...).

I.A – Représentation du problème

On modélise le problème ainsi :
- chaque besoin est représenté par un segment $[a, b]$ où $a, b \in \mathbb{N}$ et $a \leqslant b$;
- deux besoins I et J sont en conflit quand $I \cap J \neq \emptyset$.

La donnée du problème est une suite finie $(I_0, \cdots, I_{n-1})$ de n segments où $n \in \mathbb{N}^*$.

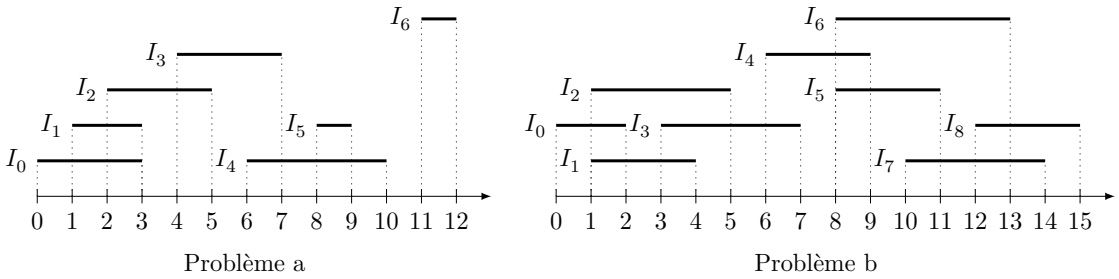


Figure 1 Deux exemples de problèmes

On représente un segment en Caml par un couple d'entiers, la donnée du problème est une valeur du type `(int*int) vect`. Le problème a de la figure 1 est représenté par le tableau

```
[| (0,3); (1,3); (2,5); (4,7); (6,10); (8,9); (11,12) |]
```

I.A.1) Écrire une fonction ayant pour signature

```
conflit : int * int -> int * int -> bool
```

telle que `conflit I J` renvoie `true` si et seulement si I et J sont en conflit.

I.B – Graphe simple non orienté

On appelle graphe simple non orienté un couple $G = (S, A)$ où
- S est un ensemble fini dont les éléments sont appelés les sommets du graphe ;
- A est un ensemble de paires d'éléments distincts de S. Lorsque $\{x, y\} \in A$ on dit que x et y sont reliés dans G et $\{x, y\}$ est appelée une arête de G. Les sommets reliés à un sommet x sont appelés les voisins de x.

Étant donnée une énumération de S sous la forme d'une suite finie $(x_0, \cdots, x_{n-1})$ on représente A en Caml par un élément du type `int list vect` ainsi : pour $i \in \{0, \cdots, n-1\}$, la liste `A.(i)` contient les j tels que x_i soit relié à x_j dans G.

On représente graphiquement le graphe G par un diagramme où les arêtes sont représentées par des traits entre les sommets.

Les arêtes du graphe dont une représentation graphique est donnée figure 2 sont représentées en Caml par le tableau :

```
[| [1;2;3]; [0;2;3]; [0;1;3;4]; [0;1;2]; [2] |]
```

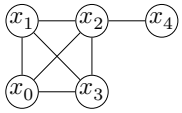


Figure 2

Une telle liste d'arêtes suffit pour déterminer un graphe lorsque l'énumération des sommets est connue car on peut alors identifier un sommet à son indice. Dans la suite de ce problème on identifiera ainsi un graphe à sa liste d'arêtes.

I.C – Graphe d'intervalles

Soit $\overline{I} = (I_0, ..., I_{n-1})$ une suite finie de segments, on appelle graphe d'intervalles associé à $\overline{I}$ le graphe $G(\overline{I})$
– dont les sommets sont les segments $I_0, ..., I_{n-1}$
– et où, pour $i, j \in \{0, ..., n-1\}$, avec $i \neq j$, les sommets I_i et I_j sont reliés si et seulement si ils sont en conflit.
Le graphe d'intervalles qui correspond au problème a de la figure 1 admet la représentation graphique de la figure 3.

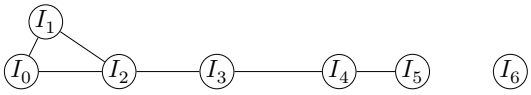


Figure 3

I.C.1) Donner une représentation graphique du graphe d'intervalles associé au problème b de la figure 1.

I.C.2) Écrire une fonction ayant pour signature

```
construit_graphe : (int * int) vect -> int list vect
```

qui étant donné le tableau des segments $\overline{I} = (I_0, ..., I_{n-1})$, énumérés dans cet ordre, renvoie la représentation des arêtes de $G(\overline{I})$.

I.D – Coloration

Soit $G = (S, A)$ un graphe simple non orienté dont les sommets sont $x_0, ..., x_{n-1}$. On appelle coloration de G une suite finie d'entiers naturels $(c_0, ..., c_{n-1})$ telle que

$$\forall (i, j) \in \{0, ..., n-1\}^2, \quad \{x_i, x_j\} \in A \Longrightarrow c_i \neq c_j$$

L'entier c_i est appelé la couleur du sommet x_i et la condition se traduit ainsi : deux sommets reliés ont des couleurs distinctes. Dorénavant, le terme couleur sera synonyme d'entier naturel.

La suite finie $(0, 1, 2, 3, 0)$ est une coloration du graphe de la figure 2.

Lorsqu'une coloration utilise le plus petit nombre de couleurs distinctes possibles, on dit qu'elle est optimale. On note alors $\chi(G)$ ce nombre minimum de couleurs, appelé le nombre chromatique de G.

En associant une salle à chaque couleur, on peut répondre au problème initial à l'aide d'une coloration de son graphe d'intervalles associé.

I.D.1) Déterminer des colorations optimales pour les graphes d'intervalles associés aux deux problèmes de la figure 1. On attribuera à chaque fois la couleur 0 à l'intervalle I_0.

I.D.2) Couleur disponible

a) Écrire une fonction de signature

```
appartient : int list -> int -> bool
```

telle que l'appel à `appartient l x` envoie `true` si et seulement si l'entier x est présent dans la liste l.

b) Écrire une fonction de signature

```
plus_petit_absent : int list -> int
```

telle que l'appel à `plus_petit_absent l` renvoie le plus petit entier naturel non présent dans `l`.

c) On considère ici une coloration progressive des sommets d'un graphe. Pour cela, une coloration partielle est un tableau `couleurs : int vect` tel que `couleurs.(i)` contient la couleur de i s'il est coloré et -1 sinon, ce qui ne pose pas de problème car les couleurs sont toujours positives.

Écrire une fonction de signature

```
couleurs_voisins : int list vect -> int vect -> int -> int list
```

telle que l'appel à `couleurs_voisins aretes couleurs i` renvoie la liste des couleurs des voisins colorés du sommet d'indice i dans le graphe décrit par `aretes` où le tableau `couleurs` décrit une coloration partielle.

d) En déduire, une fonction de signature

```
couleur_disponible : int list vect -> int vect -> int -> int
```

telle que l'appel à `couleur_disponible aretes couleurs i` renvoie la plus petite couleur pouvant être attribuée au sommet `i` afin qu'il n'ait la couleur d'aucun de ses voisins dans le graphe décrit par `aretes`.

I.E – Cliques

Soit $G = (S, A)$ un graphe.

Un sous-ensemble $C \subset S$ est appelé une clique de G lorsqu'il vérifie

$$\forall x, y \in C, \quad x \neq y \Longrightarrow \{x, y\} \in A$$

Le nombre d'éléments de C est appelé sa taille. La taille de la plus grande (celle qui possède le plus grand nombre d'éléments) clique de G est notée $\omega(G)$.

I.E.1) Déterminer $\chi(G)$ et $\omega(G)$ lorsque

a) G ne possède pas d'arête (c'est à dire $A = \emptyset$).

b) G est un graphe complet à n sommets, c'est à dire $|S| = n$ et pour tous $u, v \in S$ distincts, $\{u, v\} \in A$.

I.E.2) Comparer $\chi(G)$ et $\omega(G)$ pour un graphe G quelconque.

I.E.3) Écrire une fonction de signature

```
est_clique : int list vect -> int list -> bool
```

telle que `est_clique aretes xs` renvoie `true` si et seulement si la liste `xs` est une liste d'indices de sommets formant une clique dans le graphe décrit par `aretes`.

II Algorithme glouton pour la coloration

Étant donnée une liste de segments $\overline{I} = (I_0, I_1, ..., I_{n-1})$ de longueur $n \geqslant 1$, on se propose de déterminer une coloration optimale de son graphe d'intervalles associé. On appelle coloration de $\overline{I}$ une suite finie d'entiers naturels $(c_0, ..., c_{n-1})$ telle que

$$\forall (i, j) \in \{0, ..., n\}^2, \quad I_i \cap I_j \neq \emptyset \Longrightarrow c_i \neq c_j$$

On suppose dans cette partie que les segments $I_k = [a_k, b_k]$, pour $k \in \{0, ..., n-1\}$, sont énumérés dans l'ordre croissant de leur extrémités gauches, c'est-à-dire que

$$a_0 \leqslant a_1 \leqslant ... \leqslant a_{n-1}$$

On propose l'algorithme suivant :

> Pour k variant de 0 à $n-1$, colorer l'intervalle I_k avec la plus petite couleur non encore utilisée dans la coloration des intervalles I_j, avec $0 \leqslant j < k$, qui ont une intersection non vide avec I_k.

Ainsi, l'intervalle I_0 est toujours coloré avec la couleur 0, l'intervalle I_1 reçoit la couleur 0 si $I_0 \cap I_1 = \emptyset$, et la couleur 1 sinon, etc.

II.A – L'algorithme sur un exemple

Déterminer la coloration renvoyée par l'algorithme pour le problème b décrit sur la figure 1.

II.B – Coloration

Écrire une fonction de signature

```
coloration : (int * int) vect -> int list vect -> int vect
```

telle que l'appel `coloration segments aretes`, où `segments` est un tableau contenant des segments triés par ordre croissant de leurs extrémités gauches et où `aretes` représente les arêtes du graphe d'intervalles associé à ces segments, renvoie la coloration obtenue avec l'algorithme ci-dessus.

II.C – Preuve de l'algorithme

On se propose maintenant de démontrer que l'algorithme ci-dessus fournit une coloration optimale de l'ensemble de segments. Soit k un entier entre 0 et $n-1$. On suppose qu'à la k-ième étape de l'algorithme, le segment I_k reçoit la couleur c.

II.C.1) L'extrémité gauche du segment I_k appartient à un certain nombre de segments parmi $I_0, I_1, ..., I_{k-1}$. Combien au moins ?

II.C.2) Prouver que l'ensemble constitué de I_k et de ses voisins d'indice inférieur à k constitue une clique de taille au moins $c+1$ dans le graphe d'intervalles associé.

II.C.3) En déduire que le nombre de couleurs nécessaires à une coloration de l'ensemble des segments est au moins égal à $c+1$.

II.C.4) Conclure.

II.D – Complexité

Déterminer la complexité de la fonction `coloration` en fonction du nombre m d'arêtes du graphe d'intervalles associé à la liste $\overline{I}$.

III Graphes munis d'un ordre d'élimination parfait

On introduit ici la notion d'ordre d'élimination parfait, dont on montre qu'il existe toujours pour un graphe d'intervalles, et qui permet de proposer un algorithme glouton pour le problème de la coloration d'un graphe.

Soient $G = (S, A)$ un graphe et $(x_0, ..., x_{n-1})$ une énumération des sommets de G. Pour tout $i \in \{0, ..., n-1\}$ on note $G_i = (S_i, A_i)$ où $S_i = \{x_0, ..., x_i\}$ et

$$\forall k, l \in \{0, ..., n-1\}, \quad \{x_k, x_l\} \in A_i \Longleftrightarrow k \leqslant i \text{ et } l \leqslant i \text{ et } \{x_k, x_l\} \in A$$

G_i est ainsi le graphe déduit de G en se restreignant aux sommets de x_0 à x_i.

Une énumération $(x_0, ..., x_{n-1})$ des sommets de G est appelée un ordre d'élimination parfait si pour tout $i \in \{0, ..., n-1\}$ les voisins de x_i d'indices inférieurs à i forment une clique.

III.A – Un exemple

Déterminer un ordre d'élimination parfait pour le graphe G donné par la représentation de la figure 4.

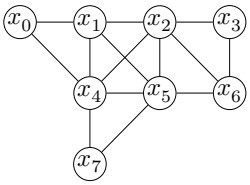


Figure 4

III.B – Vérification

III.B.1) Écrire une fonction de signature

```
voisins_inferieurs : int list vect -> int -> int list
```

telle que `voisins_inferieurs aretes x` renvoie la liste des voisins du sommet d'indice x dont l'indice est strictement inférieur à x.

III.B.2) Écrire une fonction de signature

```
est_ordre_parfait : int list vect -> bool
```

telle que `est_ordre_parfait aretes` renvoie `true` si et seulement si l'énumération associée au graphe représenté par `aretes` est un ordre d'élimination parfait.

III.C – Ordre d'élimination parfait pour un graphe d'intervalles

Montrer que l'énumération des segments $(I_0, ..., I_{n-1})$ obtenue en les triant par leurs extrémités gauches en ordre croissant est un ordre d'élimination parfait de leur graphe d'intervalles.

III.D – Coloration

On considère un graphe dont $(x_0, ..., x_{n-1})$ est une énumération des sommets.

On colore ce graphe à l'aide l'algorithme suivant :

> pour i allant de 0 à $n-1$, on colore x_i avec la plus petite couleur qui ne soit pas utilisée par un de ses voisins déjà colorés.

III.D.1) Appliquer cet algorithme de coloration au graphe G de la figure 4 muni

a) de l'ordre $(x_0, ..., x_7)$;

b) d'un ordre d'élimination parfait.

III.D.2) Écrire une fonction de signature

```
colore : int list vect -> int vect
```

telle que l'appel à `colore aretes` renvoie selon cet algorithme un tableau `c` représentant une coloration valide du graphe décrit par `aretes` où la couleur du i-ème sommet est donnée par `c.(i)`.

III.D.3) Soit $(c_0, ..., c_{n-1})$ la coloration obtenue par cet algorithme pour un graphe G dont l'énumération des sommets est un ordre d'élimination parfait.

a) Montrer que pour tout $i \in \{0, ..., n-1\}$ on a $\chi(G) \geqslant 1 + c_i$.

b) En déduire que l'algorithme de coloration renvoie une coloration optimale.

IV Ordre d'élimination parfait pour un graphe cordal

On s'intéresse ici à une nouvelle condition suffisante pour qu'un graphe admette un ordre d'élimination parfait, qui s'exprime en considérant les cycles de longueur au moins égale à 4 du graphe considéré.

Un graphe G est dit *cordal* lorsque pour tout cycle $C = (v_0, v_1, ..., v_{n-1}, v_0)$ de G de longueur $n \geqslant 4$, il existe i, j distincts entre 0 et $n-1$ tels que les sommets v_i et v_j soient reliés dans le graphe G mais non successifs dans le cycle. Une telle arête $\{v_i, v_j\}$ est appelée une *corde* du cycle C. Autrement dit, le graphe G est cordal lorsque tout cycle de G de longueur supérieure ou égale à 4 possède une corde.

IV.A – Cycles de longueur 4 dans un graphe d'intervalles

Soit G un graphe d'intervalles. Dans cette question, on se propose de démontrer par l'absurde que tout cycle de longueur 4 de G possède une corde. On suppose à cet effet que G contient un 4-cycle sans corde.

On dispose donc de 4 segments I_0, I_1, I_2, I_3 tels que $I_0 \cap I_1 \neq \emptyset$, $I_1 \cap I_2 \neq \emptyset$, $I_2 \cap I_3 \neq \emptyset$, $I_3 \cap I_0 \neq \emptyset$, et $I_0 \cap I_2 = \emptyset$, $I_1 \cap I_3 = \emptyset$. On supposera pour simplifier que les extrémités des segments sont toutes distinctes.

IV.A.1) Montrer qu'aucun des segments I_k, $k = 0, 1, 2, 3$ n'est inclus dans un autre de ces segments.

IV.A.2) On a donc par exemple $\min I_0 < \min I_1 < \max I_0 < \max I_1$. Montrer que $\min I_1 < \min I_2 < \max I_1 < \max I_2$ et de même pour I_2 et I_3.

IV.A.3) Conclure à une contradiction.

IV.B – Cordalité des graphes d'intervalles

Montrer plus généralement que tout graphe d'intervalles est cordal.

IV.C – Une enquête policière

Six personnes sont entrées dans la bibliothèque le jour où un livre rare y a été volé. Chacune d'entre elles est entrée une seule fois dans la bibliothèque, y est restée un certain temps, puis elle en est sortie. Si deux personnes étaient ensemble dans la bibliothèque à un instant donné, alors au moins l'une des deux a vu l'autre. À l'issue de l'enquête, les témoignages recueillis sont les suivants : Albert dit qu'il a vu Bernard et Édouard dans la bibliothèque. Bernard a vu Albert et Isabelle. Charlotte affirme avoir vu Didier et Isabelle. Didier dit qu'il a vu Albert et Isabelle. Édouard certifie avoir vu Bernard et Charlotte. Isabelle dit avoir vu Charlotte et Édouard. Seul le coupable a menti. Qui est-il ?

IV.D – Ordre d'élimination parfait

Un sommet v d'un graphe G est dit *simplicial* lorsque l'ensemble des voisins de v dans G est une clique.

Étant donnés un graphe $G = (S, A)$ et $S' \subset S$ un ensemble de sommets de G le *sous-graphe de G induit par S'* est le graphe $H = (S', A')$ où $A' \subset A$ est l'ensemble des arêtes de G dont les extrémités appartiennent à S'.

On représente en Caml un sous-graphe induit d'un graphe `G` possédant n sommets par le couple `(aretes, sg)` de type `int list vect * bool vect` où `aretes` est une description du graphe G et `sg` est un tableau de taille n tel que `sg.(i)` vaut `true` si le sommet d'indice i est un sommet du sous-graphe induit et `false` sinon.

IV.D.1) Écrire une fonction de signature

```
simplicial : (int list vect * bool vect) -> int -> bool
```

telle que l'appel à `simplicial (aretes, sg) k`, où le sommet d'indice k est supposé appartenir au sous-graphe induit H décrit par `(aretes, sg)`, renvoie `true` si le sommet d'indice k est simplicial dans H et `false` sinon. Déterminer la complexité de la fonction `simplicial`.

IV.D.2) Écrire une fonction de signature

```
trouver_simplicial : (int list vect * bool vect) -> int
```

telle que l'appel à `trouver_simplicial (aretes, sg)` renvoie, s'il en existe, un sommet simplicial du sous-graphe induit décrit par `(aretes, sg)`. Déterminer la complexité de la fonction `trouver_simplicial`.

IV.D.3) Écrire une fonction de signature

```
ordre_parfait : int list vect -> int list
```

telle que l'appel à `ordre_parfait aretes` renvoie un ordre d'élimination parfait du graphe décrit par `aretes`, s'il en existe un. Déterminer la complexité de la fonction `ordre_parfait`.

IV.E – Coupures minimales dans un graphe cordal

Étant donné un graphe G on appelle *coupure* de G tout ensemble $C \subseteq S$ de sommets de G, de cardinal au moins égal à 2, tel que certains sommets reliés par un chemin dans le graphe G ne le sont plus dans le sous-graphe H de G induit par $S \setminus C$.

On se donne dans cette question un graphe cordal $G = (S, A)$. Soit C une coupure de G de cardinal minimal (supérieur ou égal à 2). Soit H le sous-graphe de G induit par $S \setminus C$. Soient a et b deux sommets de G déconnectés par la coupure, et soient G_1 et G_2 les composantes connexes de a et b dans le graphe H. Soient enfin x et y deux sommets distincts de la coupure C.

IV.E.1) Montrer que x est voisin dans le graphe G d'un sommet de G_1 et d'un sommet de G_2, et de même pour y.

IV.E.2) Montrer qu'il existe un chemin $P_1 = (x, a_1, ..., a_p, y)$ dont tous les sommets hormis x et y sont des sommets de G_1 et un chemin $P_2 = (y, b_1, ..., b_q, x)$ dont tous les sommets hormis x et y sont des sommets de G_2.

IV.E.3) On prend deux tels chemins P_1 et P_2 de longueur minimale. En considérant un cycle formé à partir des chemins P_1 et P_2, montrer que x et y sont reliés dans le graphe G.

IV.E.4) Montrer que C est une clique du graphe G.

IV.F – Sommets simpliciaux dans un graphe cordal

On se propose de montrer que tout graphe cordal G possède la propriété suivante, que l'on appellera la propriété $\mathcal{P}(G)$: *G possède un sommet simplicial, et même deux sommets simpliciaux non voisins si G n'est pas complet.* On se donne dans toute la question un graphe cordal G.

IV.F.1) Montrer que si G est complet alors tous ses sommets sont simpliciaux.

IV.F.2) Montrer que la propriété $\mathcal{P}(G)$ est vérifiée si G possède 1, 2 ou 3 sommets.

IV.F.3) On suppose dans cette question que G n'est pas complet, possède au moins trois sommets et que la propriété $\mathcal{P}(G')$ est vérifiée pour tous les graphes cordaux G' ayant strictement moins de sommets que G. Soit C une coupure de G de cardinal minimal. Soient a et b deux sommets de G déconnectés par la coupure C, et G_1 et G_2 les composantes connexes de a et b dans le sous-graphe de G induit par $S \setminus C$. Soit S_1 (resp : S_2) l'ensemble des sommets de G_1 (resp : G_2). Soit enfin H_1 (resp : H_2) le sous-graphe de G induit par $S_1 \cup C$ (resp $S_2 \cup C$).

a) Justifier que le graphe H_1 est cordal.

b) On suppose que H_1 est complet. Montrer que S_1 contient un sommet simplicial du graphe H_1. Prouver que ce sommet est en fait un sommet simplicial de G.

c) On suppose que H_1 n'est pas complet. Montrer que $S_1 \cup C$ contient deux sommets simpliciaux non voisins du graphe H_1. Montrer que au moins l'un de ces deux sommets est dans S_1 et que ce sommet est un sommet simplicial de G.

d) Montrer que la propriété $\mathcal{P}(G)$ est vérifiée.

IV.G – Ordre d'élimination parfait dans un graphe cordal

Montrer que tout graphe cordal possède un ordre d'élimination parfait.

• • • FIN • • •

Partie I – Logique et calcul des propositions

De nombreux travaux sont réalisés en Intelligence Artificielle pour construire un programme qui imite le raisonnement humain et soit capable de réussir le test de Turing, c'est-à-dire qu'il ne puisse pas être distingué d'un être humain dans une conversation en aveugle. Vous êtes chargé(e)s de vérifier la correction des réponses données par un tel programme lors des tests de bon fonctionnement. Dans le scénario de test considéré, le comportement attendu est le respect de la règle suivante : pour chaque question, le programme répondra par trois affirmations dont une seule sera correcte.
Nous noterons A_1, A_2 et A_3 les propositions associées aux affirmations effectuées par le programme.

Question I.1 *Représenter le comportement attendu sous la forme d'une formule du calcul des propositions qui dépend de A_1, A_2 et A_3.*

1 Premier cas

Vous demandez au programme : *Quels éléments doivent contenir les aliments que je dois consommer pour préserver ma santé ?*
Il répond les affirmations suivantes :

A_1 Consommez au moins des aliments qui contiennent des glucides, mais pas des lipides !

A_2 Si vous consommez des aliments qui contiennent des glucides alors ne consommez pas d'aliments qui contiennent des lipides !

A_3 Ne consommez aucun aliment qui contient des lipides !

Nous noterons G, respectivement L, les variables propositionnelles qui correspondent au fait de consommer des aliments qui contiennent des glucides, respectivement des lipides.

Question I.2 *Exprimer A_1, A_2 et A_3 sous la forme de formules du calcul des propositions. Ces formules peuvent dépendre des variables G et L.*

Question I.3 *En utilisant le calcul des propositions (résolution avec les formules de De Morgan), déterminer ce que doivent contenir les aliments que vous devez consommer pour préserver votre santé.*

2 Second cas

Vous demandez au programme : *Quelles activités dois-je pratiquer si je veux préserver ma santé ?*
Suite à une coupure de courant, la dernière affirmation est interrompue.

A_1 Ne faites des activités sportives que si vous prenez également du repos !

A_2 Si vous ne faites pas d'activité intellectuelle alors ne prenez pas de repos !

A_3 Prenez du repos ou faites des activités ...!

Nous noterons S, I et R les variables propositionnelles qui correspondent au fait de faire des activitiés sportives, des activités intellectuelles et de prendre du repos.

Question I.4 *Exprimer A_1, A_2 et A_3 sous la forme de formules du calcul des propositions. Ces formules peuvent dépendre de S, I et R.*

Question I.5 *En utilisant le calcul des propositions (résolution avec les tables de vérité), déterminer quelle(s) activité(s) vous devez pratiquer pour préserver votre santé.*

Partie II – Automates et langages

Cette partie étudie l'opérateur *racine carrée* $\sqrt{\mathcal{A}}$ pour un automate fini $\mathcal{A}$.

1 Automate fini complet déterministe

Pour simplifier cette étude, nous nous limiterons au cas des automates finis complets déterministes. Les résultats étudiés s'étendent au cadre des automates finis quelconques.

1.1 Représentation d'un automate fini complet déterministe

Déf. II.1 (Alphabet, mot, langage) L'alphabet X est un ensemble de symboles. $\Lambda \notin X$ est le symbole représentant le mot vide. $X^\star$ est l'ensemble contenant Λ et les mots composés de séquences de symboles de X. Un langage sur X est un sous-ensemble de $X^\star$.

Déf. II.2 (Automate fini complet déterministe) Un automate fini complet déterministe sur X est un quintuplet $\mathcal{A} = (Q, X, i, T, \delta)$ composé :

— d'un ensemble fini d'états : Q ;
— d'un état initial : $i \in Q$;
— d'un ensemble d'états terminaux : $T \subseteq Q$;
— d'une fonction totale de transition confondue avec son graphe : $\delta \subseteq Q \times X \mapsto Q$.

Pour une transition $\delta(o, e) = d$ donnée, nous appelons o l'origine de la transition, e l'étiquette de la transition et d la destination de la transition.

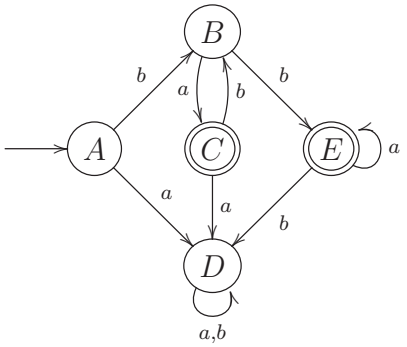
1.2 Représentation graphique d'un automate

Les automates peuvent être représentés par un schéma suivant les conventions :

— les valeurs de la fonction totale de transition δ sont représentées par un graphe orienté dont les nœuds sont les états et les arêtes sont les transitions ;
— l'état initial i est désigné par une flèche $\longrightarrow (i)$;
— tout état terminal t est entouré d'un double cercle $(\!(t)\!)$;
— une arête étiquetée par le symbole $e \in X$ va de l'état o à l'état d si et seulement si $\delta(o, e) = d$.

Exemple II.1 L'automate $\mathcal{E} = (Q, X, i, T, \delta)$ est représenté par le graphe suivant :

$$Q = \{A, B, C, D, E\} \quad X = \{a, b\} \quad i = A \quad T = \{C, E\}$$
$$\delta(A, a) = D; \quad \delta(A, b) = B;$$
$$\delta(B, a) = C; \quad \delta(B, b) = E;$$
$$\delta(C, a) = D; \quad \delta(C, b) = B;$$
$$\delta(D, a) = D; \quad \delta(D, b) = D;$$
$$\delta(E, a) = E; \quad \delta(E, b) = D.$$



1.3 Langage reconnu par un automate fini déterministe

Déf. II.3 (Transition sur un mot) L'extension $\delta^\star$ de δ à $Q \times X^\star \mapsto Q$ est définie par :

$$\forall q \in Q,\ \delta^\star(q, \Lambda) = q$$

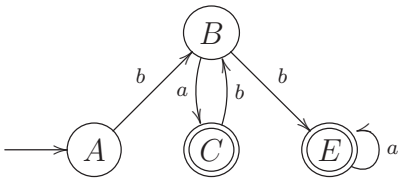
$$\begin{cases} \forall e \in X, \\ \forall m \in X^\star, \\ \forall o \in Q, \\ \forall d \in Q, \end{cases} \delta^\star(o, e.m) = d \Leftrightarrow \exists q \in Q,\ (\delta(o, e) = q) \wedge (\delta^\star(q, m) = d).$$

Déf. II.4 (Langage reconnu par un automate) Le langage sur X reconnu par un automate fini déterministe $\mathcal{A}$ est :

$$L(\mathcal{A}) = \{m \in X^\star \mid \exists d \in T,\ \delta^\star(i, m) = d\}.$$

Notons que certains états et transitions ne sont pas utiles dans la description d'un langage car ils ne permettent pas d'aller de l'état initial à un état terminal. Il s'agit, d'une part, des états qui ne sont pas accessibles ou co-accessibles et, d'autre part, des transitions dont ils sont l'origine ou la destination. Un automate dans lequel ces états et transitions inutiles ont été éliminés est appelé automate émondé.

Exemple II.2 L'automate émondé extrait de $\mathcal{E}$ (**exemple II.1, page 3**) (composé des états et transitions utiles, c'est-à-dire des états à la fois accessibles et co-accessibles) est représenté par le graphe suivant :



Question II.1 *Donner, sans la justifier, une expression régulière ou ensembliste représentant le langage $L(\mathcal{E})$ sur $X = \{a, b\}$ reconnu par l'automate $\mathcal{E}$ de* ***l'exemple II.1, page 3****.*

Question II.2 *Montrer que :*

$$\forall m \in X^\star,\ \forall n \in X^\star,\ \forall o \in Q,\ \forall q \in Q,\ \forall d \in Q,\ (q = \delta^\star(o, m)) \wedge (d = \delta^\star(q, n)) \Leftrightarrow d = \delta^\star(o, m.n).$$

2 Racine carrée d'un langage

Déf. II.5 (Racine carrée d'un langage) Le langage $\sqrt{L}$ (racine carrée de L), est défini par :

$$\sqrt{L} = \{m \in X^\star \mid m.m \in L\}.$$

Question II.3 *Donner, sans la justifier, une expression régulière ou ensembliste représentant le langage $\sqrt{L(\mathcal{E})}$ sur $X = \{a, b\}$ construit à partir du langage $L(\mathcal{E})$ reconnu par l'automate $\mathcal{E}$ de* ***l'exemple II.1, page 3****.*

Question II.4 *Soit un langage L sur un alphabet X, comparer L avec $\sqrt{L^2}$ puis avec $\left(\sqrt{L}\right)^2$.*

3 Racine carrée d'un automate

Déf. II.6 (Séquence) Une séquence s de valeurs v_i avec $i \in [d, f]$ est notée $s = \langle v_i \rangle_{i=d}^{f}$. Sa taille est notée $\mid s \mid = f - d + 1$. Si $i \in [d, f]$, sa i-ième valeur est notée $s_i = v_i$.
Si les valeurs v_i appartiennent à l'ensemble V alors les séquences de taille n appartiennent à l'ensemble $V^n = \{\langle v_i \rangle_{i=1}^{n}\}$.

Soit l'opération interne sur les automates finis complets déterministes définie par :

Déf. II.7 (Racine carrée) Soit $\mathcal{A} = (Q_\mathcal{A}, X, i_\mathcal{A}, T_\mathcal{A}, \delta_\mathcal{A})$ un automate fini complet déterministe, notons $Q_\mathcal{A} = \{q_i\}_{i=0}^{n}$ avec $i_\mathcal{A} = q_0$ et $card(Q_\mathcal{A}) = n + 1$, alors l'automate $\sqrt{\mathcal{A}} = (Q_{\sqrt{\mathcal{A}}}, X, i_{\sqrt{\mathcal{A}}}, T_{\sqrt{\mathcal{A}}}, \delta_{\sqrt{\mathcal{A}}})$ (racine carrée de $\mathcal{A}$) est défini par :

$$
\begin{aligned}
&Q_{\sqrt{\mathcal{A}}} = {Q_\mathcal{A}}^{n+1};\\
&i_{\sqrt{\mathcal{A}}} = \langle q_i \rangle_{i=0}^{n} \in Q_{\sqrt{\mathcal{A}}};\\
&T_{\sqrt{\mathcal{A}}} = \{\langle t_i \rangle_{i=0}^{n} \in Q_{\sqrt{\mathcal{A}}} \mid t_0 = q_j, t_j \in T_\mathcal{A}, j \in [0, n]\};\\
&\forall o \in Q_{\sqrt{\mathcal{A}}}, \forall d \in Q_{\sqrt{\mathcal{A}}}, \forall a \in X, d = \delta_{\sqrt{\mathcal{A}}}(o, a) \Leftrightarrow \forall i \in [0, n], d_i = \delta_\mathcal{A}(o_i, a).
\end{aligned}
$$

Question II.5 *En considérant **l'exemple II.1, page 3**, construire l'automate $\sqrt{\mathcal{E}}$ (seuls les états et les transitions utiles, c'est-à-dire accessibles depuis les états initiaux et co-accessibles depuis les états terminaux, devront être construits).*

Question II.6 *Caractériser le langage reconnu par $\sqrt{\mathcal{E}}$ par une expression régulière ou ensembliste.*

Dans la suite de cette partie, nous considérons que $\mathcal{A}$ est un automate fini complet déterministe.

Question II.7 *Montrer que $\sqrt{\mathcal{A}}$ est un automate fini complet déterministe.*

Question II.8 *Montrer que :*

$$\forall o \in Q_{\sqrt{\mathcal{A}}}, \forall m \in X^\star, {\delta_{\sqrt{\mathcal{A}}}}^\star(o, m) = \langle {\delta_\mathcal{A}}^\star(o_i, m) \rangle_{i=0}^{n}.$$

Question II.9 *Montrer que :*

$$\forall m \in X^\star,\ m \in L(\sqrt{\mathcal{A}}) \Leftrightarrow m.m \in L(\mathcal{A}).$$

Question II.10 *Quelle relation liant les automates $\mathcal{A}$ et $\sqrt{\mathcal{A}}$ pouvez-vous en déduire ?*

Partie III – Algorithmique et programmation

Cette partie combine les programmes d'*Informatique pour tous* et d'option *Informatique*. Elle doit être traitée, selon les questions, soit en utilisant le langage Python tel qu'il a été présenté dans le cadre des enseignements d'*Informatique pour tous*, soit en utilisant le langage CaML tel qu'il a été présenté dans le cadre des enseignements d'option *Informatique*.
Les fonctions écrites en langage Python ne devront pas être récursives.
Les fonctions écrites en langage CaML devront être récursives ou faire appel à des fonctions auxiliaires récursives. Elles ne devront pas utiliser d'instructions itératives (c'est-à-dire **`for`**, **`while`**, ...), ni de références, ni d'exceptions.

1 Exercice : le tri à bulles

Cet exercice étudie l'algorithme de tri à bulles d'une séquence d'entiers. Pour simplifier les notations et les preuves, les séquences manipulées ne contiennent qu'une seule fois chaque valeur. Les résultats obtenus se généralisent aux séquences quelconques.

Déf. III.1 (Séquence) Une séquence s de valeurs v_i avec $i \in [d, f] \subset \mathbb{N}$ est notée $s = \langle v_i \rangle_{i=d}^{f}$.

— Sa taille est notée $\mid s \mid = f - d + 1$;

— Si $i \in [d, f]$, sa i-ième valeur est notée $s_i = v_i$;

— Son domaine, c'est-à-dire l'intervalle des indices de ses valeurs, est noté : $dom(\langle v_i \rangle_{i=d}^{f}) = [d, f]$;

— Son co-domaine, c'est-à-dire l'ensemble de ses valeurs, est noté : $codom(\langle v_i \rangle_{i=d}^{f}) = \{v_i\}_{i=d}^{f}$;

— La séquence vide de taille 0 est notée $\langle\rangle$;

— Si $d \leq d' \leq f' \leq f$ alors $\langle v_i \rangle_{i=d'}^{f'}$ de taille $f' - d' + 1$ désigne la sous-séquence de $\langle v_i \rangle_{i=d}^{f}$ contenant les valeurs de $v_{d'}$ à $v_{f'}$.

Les valeurs contenues dans les séquences s manipulées par la suite sont toutes distinctes, c'est-à-dire que le cardinal du codomaine de s est égal à la taille de s ($card(codom(s)) = \mid s \mid$).

Une séquence sera représentée en CaML et en Python par une liste.

Question III.1 *Ecrire en CaML une fonction* `lire` *dont le type est* `int -> int list -> int` *telle que l'appel* `(lire i l)` *sur la liste d'entiers* `l` *correspondant à la séquence* $\langle v_i \rangle_{i=1}^{n}$ *de taille* n *avec* `i` $\in [1, n]$ *doit renvoyer l'entier* $v_{\texttt{i}}$. *Cette fonction devra au plus parcourir une seule fois chaque élément de la liste* `l`. *Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

Dans la suite de cet exercice, nous considérons la fonction `trier` du programme en langage Python du **listing 1, page 7**.

Question III.2 *Quelles sont les informations affichées lors de son exécution ? Que contient la variable* `resultat` *après son exécution ? Que contient la variable* `exemple` *après son exécution ? Que se passe-t-il si nous remplaçons l'instruction* `t = copy(p)` *par l'instruction* `t = p` *?*

```
from copy import copy
def trier(p):
    t = copy(p)
    for i in range(len(t)):                # entrée boucle externe
        for j in range(len(t)-i-1):        # entrée boucle interne
            if (t[j+1]<t[j]):
                t[j], t[j+1] = t[j+1], t[j]
            print( i, j, t)
        # sortie boucle interne
    # sortie boucle externe
    return t

exemple = [ 3, 1, 4, 2]
resultat = trier(exemple)
```

Listing 1: tri à bulles en Python

Question III.3 *Soient les séquences* `s` *et* `r` *avec* $dom(\mathtt{s}) = [1, m]$ *et* $dom(\mathtt{r}) = [1, n]$, *telles que* `r = trier(s)`, *montrer que :*

(i) $dom(\mathtt{r}) = dom(\mathtt{s})$;

(ii) $codom(\mathtt{r}) = codom(\mathtt{s})$;

(iii) $\forall i \in [1, n[, r_i \leq r_{i+1}$.

Vous utiliserez pour cela les invariants de boucle :

— *boucle externe :*
$\forall k \in [n - \mathtt{i}, n[, \mathtt{t}_k \leq \mathtt{t}_{k+1} \wedge dom(\mathtt{t}) = dom(\mathtt{p}) \wedge codom(\mathtt{t}) = codom(\mathtt{p})$;

— *boucle interne :*
$\forall k \in [1, j], \mathtt{t}_k \leq \mathtt{t}_{j+1} \wedge dom(\mathtt{t}) = dom(\mathtt{p}) \wedge codom(\mathtt{t}) = codom(\mathtt{p})$.

Question III.4 *Montrer que le calcul de la fonction* `trier` *se termine quelles que soient les valeurs de son paramètre* `p`.

Question III.5 *Donner des exemples de valeurs du paramètre* `p` *de la fonction* `trier` *qui correspondent au pire cas en temps d'exécution.*
Montrer que la complexité en temps d'exécution dans le pire cas de la fonction `trier` *en fonction de la taille* n *des séquences données en paramètre est de* $O(n^2)$.

Question III.6 *Ecrire en CaML une fonction* `trier` *dont le type est* `int list -> int list` *et qui utilise le même algorithme que la fonction* `trier` *du programme Python du* ***listing 1 ci-dessus*** *sans afficher les valeurs intermédiaires. L'appel* `(trier s)` *sur une liste d'entiers* `s` *doit donc renvoyer une liste d'entiers triée* `r` *qui contient les mêmes éléments que la liste* `s`, *c'est-à-dire, satisfaisant les propriétés (i), (ii) et (iii) de la* ***question III.3 ci-dessus***. *Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

2 Problème : le tri par tas

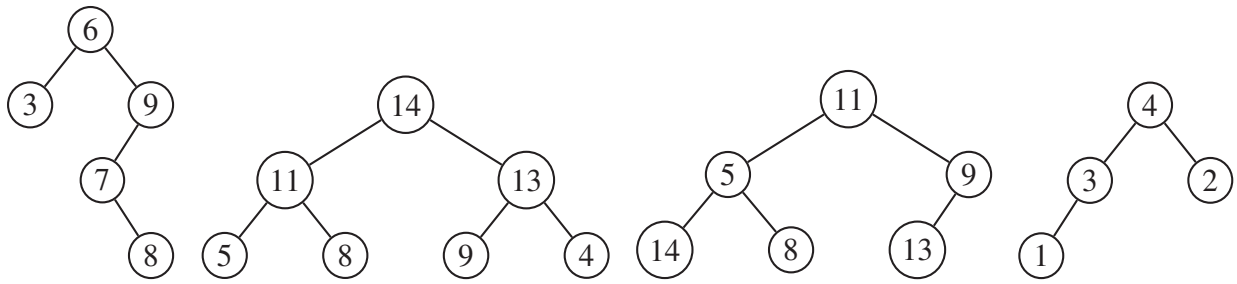
La complexité de l'algorithme de tri à bulles étudié dans l'exercice précédent ne permet pas de l'utiliser pour de gros volumes de données. Ce problème étudie un algorithme de tri plus performant basé sur la structure d'arbre en tas, c'est-à-dire d'arbre binaire parfait partiellement ordonné. Nous considérerons une implantation arborescente en langage CaML et une implantation sous la forme de tableau en langage Python.

2.1 Arbre binaire d'entiers

Déf. III.2 (Arbre binaire d'entiers) Un arbre binaire d'entiers a est une structure qui peut, soit être vide (notée $\emptyset$), soit être un nœud qui contient une étiquette entière (notée $\mathcal{E}(a)$), un sous-arbre gauche (noté $\mathcal{G}(a)$) et un sous-arbre droit (noté $\mathcal{D}(a)$) qui sont tous deux des arbres binaires d'entiers. La taille de l'arbre a, notée $\mid a \mid$ est le nombre de nœuds de l'arbre a.
Cette définition s'exprime sous la forme de la propriété $\mathcal{B}(a)$ qui caractérise les arbres binaires d'entiers a :

$$\mathcal{B}(a) \equiv (a = \emptyset) \vee \big(a \neq \emptyset \wedge \mathcal{B}(\mathcal{G}(a)) \wedge \mathcal{B}(\mathcal{D}(a)) \wedge \mathcal{E}(a) \in \mathbb{N}\big).$$

Exemple III.1 (Arbre binaire d'entiers) Voici quatre exemples d'arbres binaires étiquetés par des entiers (les sous-arbres vides qui sont fils gauche ou droit des nœuds ne sont pas représentés) :



2.1.1 Implantation en langage CaML

Un arbre binaire d'entiers est représenté en langage CaML par le type `arbre` dont la définition est :

```
type arbre =
 | Vide
 | Noeud of arbre * int * arbre;;
```

Dans l'appel `Noeud( fg, v, fd)`, les paramètres `fg`, `v` et `fd` sont respectivement le fils gauche, l'étiquette et le fils droit de la racine de l'arbre binaire d'entiers créé.

Exemple III.2 L'expression suivante :

```
Noeud(
   Noeud( Vide, 3, Vide)
   6,
   Noeud(
      Noeud(
         Vide,
         7,
         Noeud( Vide, 8, Vide)),
      9,
      Vide))
```

est alors associée au premier arbre binaire représenté graphiquement dans **l'exemple III.1, page 8**.

Question III.7 *Donner l'expression en langage CaML qui correspond au quatrième arbre binaire de* ***l'exemple III.1, page 8****.*

2.1.2 Hauteur dans un arbre binaire

Déf. III.3 (Hauteur dans un arbre binaire) Soit a un arbre binaire et n un nœud de a. La hauteur de n dans a est égale au nombre de nœuds du chemin sans cycle le plus long reliant n à un sous-arbre vide. Nous la noterons $\eta(n)$. Si n est la racine de l'arbre ($n = a$), il s'agit alors de la hauteur de l'arbre. Nous associerons la hauteur 0 à l'arbre binaire vide $\emptyset$.

Exemple III.3 (Hauteurs) Les hauteurs des quatre arbres binaires de **l'exemple III.1, page 8** sont respectivement 4, 3, 3 et 3.

Question III.8 *Donner une définition mathématique de la hauteur $\eta(a)$ d'un arbre binaire a en fonction de $\emptyset$, $\mathcal{G}(a)$ et $\mathcal{D}(a)$.*

Question III.9 *Soit un ensemble non vide d'étiquettes donné, quelle est la structure de l'arbre binaire contenant ces étiquettes dont la hauteur est maximale ? Quelle est la structure de l'arbre binaire contenant ces étiquettes dont la hauteur est minimale ? Justifier vos réponses.*

2.1.3 Profondeur d'un nœud et niveau dans un arbre binaire

Déf. III.4 (Profondeur d'un nœud, niveau dans un arbre binaire) Soit un arbre binaire a contenant un nœud n, la profondeur du nœud n dans a notée $\pi(n)$ est définie par :

$$\pi(n) = \eta(a) - \eta(n).$$

Un niveau dans un arbre binaire a est la séquence de gauche à droite des nœuds de même profondeur dans a.

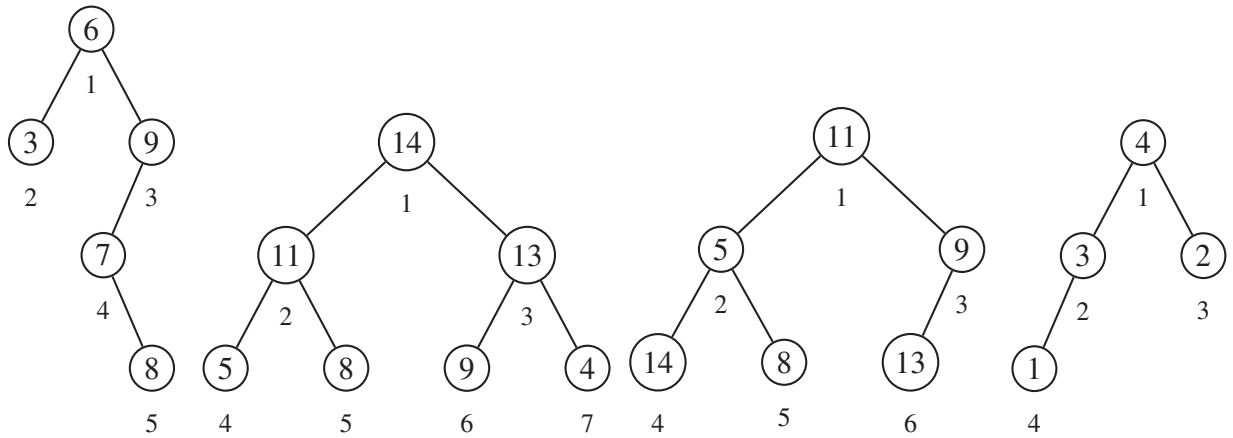
Exemple III.4 (Profondeurs) Les étiquettes des nœuds de même profondeur dans les quatre arbres de **l'exemple III.1, page 8** sont :

Profondeur	Arbre 1	Arbre 2	Arbre 3	Arbre 4
0	6	14	11	4
1	$3, 9$	$11, 13$	$5, 9$	$3, 2$
2	7	$5, 8, 9, 4$	$14, 8, 13$	1
3	8			

2.1.4 Numérotation hiérarchique des nœuds d'un arbre binaire

La numérotation hiérarchique des nœuds d'un arbre binaire a consiste à associer à chaque nœud un numéro compris entre 1 et $\mid a \mid$ par un parcours en largeur partant de la racine (numéro 1) et en parcourant chaque niveau de gauche à droite jusqu'au dernier nœud : le plus profond et le plus à droite (numéro $\mid a \mid$). Nous noterons $\mathcal{N}_i(a)$ le nœud de l'arbre binaire a de numéro i avec $i \in [1, \mid a \mid]$. Dans les exemples suivants, le numéro de chaque nœud sera noté en-dessous de son étiquette.

Exemple III.5 (Numérotation hiérarchique) La numérotation hiérarchique des nœuds des quatre arbres de **l'exemple III.1, page 8** produit les arbres numérotés suivants (les sous-arbres vides qui sont fils gauche ou droit des nœuds ne sont pas représentés) :



Question III.10 *Ecrire en CaML une fonction* `lire` *dont le type est* `int -> arbre -> int` *telle que l'appel* `(lire i a)` *sur l'arbre binaire d'entiers* `a` *avec* `i` $\in [1, \mid$ `a` $\mid]$ *doit renvoyer l'entier* $\mathcal{E}(\mathcal{N}_{\mathtt{i}}(\mathtt{a}))$. *Cette fonction devra au plus parcourir une seule fois chaque élément de l'arbre* `a`. *Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

2.2 Arbre binaire partiellement ordonné d'entiers

Déf. III.5 (Arbre binaire partiellement ordonné d'entiers) Un arbre binaire d'entiers a est partiellement ordonné si :

— les fils gauche et droit de a sont des arbres binaires partiellement ordonnés d'entiers ;

— les étiquettes de tous les nœuds composant les fils gauche et droit de a sont inférieures ou égales à l'étiquette de a.

Cette définition s'exprime sous la forme de la propriété $\mathcal{O}(a)$ qui caractérise les arbres binaires partiellement ordonnés d'entiers a :

$$\mathcal{O}(a) \equiv (a = \emptyset) \vee (a \neq \emptyset \wedge \mathcal{O}(\mathcal{G}(a)) \wedge \mathcal{O}(\mathcal{D}(a)) \wedge \mathcal{E}(\mathcal{G}(a)) \leq \mathcal{E}(a) \wedge \mathcal{E}(\mathcal{D}(a)) \leq \mathcal{E}(a)).$$

Exemple III.6 (Arbres binaires partiellement ordonnés d'entiers) Le deuxième et le quatrième arbres de **l'exemple III.1, page 8** sont des arbres binaires partiellement ordonnés d'entiers.

Question III.11 *Ecrire en CaML une fonction* `verifier` *dont le type est* `arbre -> bool` *telle que l'appel* `(verifier a)` *sur l'arbre binaire d'entiers* `a` *renvoie la valeur* `true` *si* $\mathcal{O}(\mathtt{a})$ *et la valeur* `false` *sinon. Cette fonction devra au plus parcourir une seule fois chaque nœud de l'arbre* `a`. *Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

2.3 Arbres binaires complets

Déf. III.6 (Arbre binaire complet) Un arbre binaire complet est un arbre binaire dont tous les niveaux sont complets, c'est-à-dire que tous les nœuds d'un même niveau ont deux fils non vides sauf les nœuds du niveau le plus profond qui n'ont aucun fils (c'est-à-dire deux fils vides).
Cette définition s'exprime sous la forme de la propriété $\mathcal{C}_n(a)$ qui caractérise les arbres binaires complets a de hauteur n ($\eta(a) = n$) :

$$\mathcal{C}_n(a) \equiv (a = \emptyset \wedge n = 0) \vee (a \neq \emptyset \wedge n \neq 0 \wedge \mathcal{C}_{n-1}(\mathcal{G}(a)) \wedge \mathcal{C}_{n-1}(\mathcal{D}(a))).$$

Exemple III.7 (Arbre binaire complet) Le deuxième arbre de **l'exemple III.1, page 8** est complet.

Question III.12 *Montrer que, dans un arbre binaire complet non vide a, le niveau de profondeur p contient 2^p nœuds.*

Question III.13 *Calculer le nombre n de nœuds d'un arbre binaire complet non vide a de hauteur $p = \eta(a)$.*

Question III.14 *En déduire la hauteur $p = \eta(a)$ d'un arbre binaire complet non vide a contenant n éléments ($n = \mid a \mid$).*

2.4 Arbres binaires parfaits

Déf. III.7 (Arbre binaire parfait) Un arbre binaire parfait est un arbre binaire dont tous les niveaux sont complets sauf le niveau le plus profond qui peut être incomplet auquel cas ses nœuds sont alignés à gauche de l'arbre.
Cette définition s'exprime sous la forme de la propriété $\mathcal{P}_n(a)$ qui caractérise l'arbre binaire parfait a de hauteur n :

$$\mathcal{P}_n(a) \equiv \mathcal{C}_n(a) \vee a \neq \emptyset \wedge n \neq 0 \wedge \begin{cases} & n \neq 1 \wedge \mathcal{P}_{n-1}(\mathcal{G}(a)) \wedge \mathcal{C}_{n-2}(\mathcal{D}(a)) \\ \vee & n \neq 1 \wedge \mathcal{C}_{n-1}(\mathcal{G}(a)) \wedge \mathcal{C}_{n-2}(\mathcal{D}(a)) \\ \vee & \mathcal{C}_{n-1}(\mathcal{G}(a)) \wedge \mathcal{P}_{n-1}(\mathcal{D}(a)) \end{cases}$$

Exemple III.8 (Arbres binaires parfaits) Le troisième et le quatrième arbres de **l'exemple III.1, page 8** sont parfaits. Le deuxième est également parfait car il est complet.

Soit le type énuméré `categorieArbre` en langage CaML distinguant les arbres binaires complets, parfaits et quelconques :

```
type categorieArbre = Complet| Parfait| Quelconque;;
```

Question III.15 *Ecrire en CaML une fonction* `analyser` *dont le type est* `arbre -> int * categorieArbre` *telle que l'appel* `(analyser a)` *sur l'arbre binaire* `a` *renvoie une paire contenant la hauteur $\eta(\mathtt{a})$ de l'arbre* `a` *ainsi que la valeur* `Complet` *si $\mathcal{C}_{\eta(\mathtt{a})}(\mathtt{a})$, sinon la valeur* `Parfait` *si $\mathcal{P}_{\eta(\mathtt{a})}(\mathtt{a})$ sinon la valeur* `Quelconque`*. Cette fonction devra au plus parcourir une seule fois chaque nœud de l'arbre* `a`*. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

Question III.16 *Soit un nœud de numéro n dans le niveau de profondeur p d'un arbre binaire parfait, calculer le nombre de nœuds qui se trouvent à sa gauche dans le niveau de profondeur p.*

Question III.17 *Dans le niveau de profondeur $p+1$ d'un arbre binaire parfait, quel est le nombre de nœuds qui se trouvent à la gauche des fils du nœud de numéro n (n faisant partie du niveau de profondeur p).*

Question III.18 *Soit un nœud de numéro n d'un arbre binaire parfait, calculer les numéros de ses fils gauche et droit.*

Question III.19 *Déduire de la question précédente, le numéro du père du nœud de numéro n dans un arbre binaire parfait.*

Question III.20 *Ecrire en CaML une fonction* `lire` *dont le type est* `int -> arbre -> int` *telle que l'appel* `(lire i a)` *sur l'arbre binaire parfait* `a` *avec* `i` $\in [1, |$ `a` $|]$ *doit renvoyer l'entier* $\mathcal{E}(\mathcal{N}_{\mathtt{i}}(\mathtt{a}))$. *Cette fonction devra au plus parcourir une seule fois chaque élément de la branche qui conduit de la racine de* `a` *au nœud de numéro* `i`. *Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

Soit le programme en langage CaML :

```
let construire l =
  let rec aux1 l a =
    match l,a with
      | ([],_) -> (a,l)
      | (t::q,Vide) -> (Noeud(Vide,t,Vide),q)
      | (_,Noeud(g,v,d)) ->
        match (aux1 l g) with
          | (rga,[]) ->
            ((Noeud(rga,v,d)),[])
          | (rga,rgl) ->
            (match (aux1 rgl d) with
              | (rda,rdl) ->
                ((Noeud(rga,v,rda)),rdl))
  in
  let rec aux2 l a =
    match (aux1 l a) with
      | (ra,[]) -> ra
      | (ra,rl) -> (aux2 rl ra)
  in
  (aux2 l Vide);;

let exemple = [ 1; 2; 3; 4; 5; 6];;
```

Question III.21 *Détailler les étapes du calcul de* `(construire exemple)` *en précisant pour chaque appel aux fonctions* `construire`, `aux1` *et* `aux2`, *la valeur du paramètre et du résultat.*

Question III.22 *Soient les séquences d'entiers* s *et* r *avec* $dom(\mathtt{s}) = [1, m]$ *et* $dom(\mathtt{r}) = [1, n]$*, les arbres binaires d'entiers* a *et* b *tels que* $\mathcal{C}_p(\mathtt{a})$ *et* (b,r) *est la paire renvoyée par* (aux1 s a)*, montrer que :*

(i) $\forall i, 1 \leq i < 2^p, \mathcal{E}(\mathcal{N}_i(\mathtt{b})) = \mathcal{E}(\mathcal{N}_i(\mathtt{a}))$ *;*

(ii) $m = 0 \implies (\mathcal{C}_p(b) \wedge n = 0)$ *;*

(iii) $1 \leq m < 2^p \implies (\mathcal{P}_{p+1}(b) \wedge n = 0 \wedge \forall i \in dom(s), \mathcal{E}(\mathcal{N}_{i+2^p-1}(\mathtt{b})) = s_i)$ *;*

(iv) $m \geq 2^p \implies (\mathcal{C}_{p+1}(b) \wedge n = m - 2^p \wedge \forall i \in dom(r), r_i = s_{i+2^p-1})$.

Question III.23 *Soit la séquence* s *avec* $dom(\mathtt{s}) = [1, m]$*, soit l'arbre binaire d'entiers* a*, tels que* a = (construire s)*, montrer que :*

(i) $\mathcal{P}_p(\mathtt{a})$ *;*

(ii) $\mid a \mid = m$ *;*

(iii) $2^{p-1} \leq m < 2^p$ *;*

(iv) $\forall i \in dom(s), \mathcal{E}(\mathcal{N}_i(\mathtt{a})) = s_i$.

Question III.24 *Montrer que le calcul des fonctions* aux1*,* aux2 *et* construire *se termine quelles que soient les valeurs de leurs paramètres respectant le type des fonctions.*

Question III.25 *Donner des exemples de valeurs du paramètre* s *de la fonction* construire *qui correspondent au pire cas en nombre d'appels récursifs effectués.*
Montrer que la complexité de la fonction construire *en fonction de la taille* n *des séquences données en paramètre est de* $O(n^2)$*. Cette estimation ne prend en compte que le nombre d'appels récursifs effectués.*

2.5 Arbres en tas

Déf. III.8 (Arbre en tas) Un arbre en tas est un arbre binaire parfait partiellement ordonné.
Cette définition s'exprime sous la forme de la propriété $\mathcal{T}_n(a)$ qui caractérise les arbres en tas a de hauteur n :

$$\mathcal{T}_n(a) \equiv \mathcal{P}_n(a) \wedge \mathcal{O}(a).$$

Exemple III.9 (Arbres en tas) Le deuxième et le quatrième arbres binaires de **l'exemple III.1, page 8** sont en tas.

Lorsqu'un arbre binaire parfait n'est pas partiellement ordonné, les étiquettes des nœuds peuvent être permutées pour obtenir un arbre en tas sans changer la structure d'arbre binaire parfait.

2.5.1 Implantation en langage CaML

Question III.26 *Ecrire en CaML une fonction* placer *dont le type est* arbre -> int -> arbre -> arbre *telle que l'appel* (placer g v d) *sur l'entier* v *et les arbres en tas* g *et* d *tels que* $\eta(\mathtt{g}) = \eta(\mathtt{d})$ *ou* $\eta(\mathtt{g}) = \eta(\mathtt{d}) + 1$ *(*η *est* ***définie page 9****), renvoie un arbre en tas contenant les même étiquettes que* g *et* d *ainsi que l'étiquette* v*. Cette fonction devra au plus parcourir une branche de* g *ou de* d*. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

Question III.27 *Proposer la modification la plus simple possible de la fonction* `construire` *étudiée précédemment* ***page 12*** *pour que l'arbre résultat ait une structure d'arbre en tas.*

2.5.2 Implantation par un tableau en langage Python

La structure d'arbre parfait peut être implantée très efficacement par un tableau contenant les étiquettes de l'arbre selon une numérotation hiérarchique des nœuds de l'arbre. Les nœuds de l'arbre seront désignés par leur numéro. L'indice le plus petit de la séquence doit être au moins égal à 1. Vous noterez que, dans le cas où l'indice le plus petit de la séquence est strictement plus grand que 1, toutes les cases du tableau ne correspondent pas à des nœuds de l'arbre. Il est également possible de préciser l'indice de la dernière case utilisée pour représenter le tableau. Alors, les cases suivantes ne correspondent pas à des nœuds.

Exemple III.10 (Arbres binaires parfaits représentés par un tableau) Les deuxième, troisième et quatrième arbres de **l'exemple III.1, page 8** sont parfaits. Ils sont représentés par les tableaux suivants (les entiers des lignes supérieures sont les numéros des nœuds et les entiers des lignes inférieures correspondent aux valeurs de leurs étiquettes) :

1	2	3	4	5	6	7
14	11	13	5	8	9	4

1	2	3	4	5	6
11	5	9	14	8	13

1	2	3	4
4	3	2	1

Pour le premier tableau, si l'indice correspondant à la racine de l'arbre parfait est le 3, alors seules les cases 3, 6 et 7 correspondent à des nœuds de l'arbre (**exemple III.5, page 10**).

Question III.28 *Soient :*

- `a` *une séquence d'entiers avec* $dom(\mathtt{a}) = [1, n]$ *;*
- `r` *et* `f` *des entiers avec* $1 \leq \mathtt{r} \leq \mathtt{f} \leq n$ *;*

tels que :

- `a` *représente un arbre binaire parfait avec* `f` *l'indice du dernier nœud ;*
- *si l'étiquette* $\mathcal{E}(\mathcal{N}_{\mathtt{r}}(\mathtt{a}))$ *est remplacée par le maximum de* $\mathcal{E}(\mathcal{G}(\mathcal{N}_{\mathtt{r}}(\mathtt{a})))$ *et* $\mathcal{E}(\mathcal{D}(\mathcal{N}_{\mathtt{r}}(\mathtt{a})))$ *alors* $\mathcal{N}_{\mathtt{r}}(\mathtt{a})$ *est un arbre en tas.*

Ecrire en Python une procédure `placer(r,a,f)` *qui permute le contenu de certaines cases du tableau* `a` *pour transformer l'arbre binaire parfait contenu dans* `a` *avant l'appel dont le dernier nœud est de numéro* `f`*, en un arbre binaire parfait contenu dans* `a` *après l'appel tel que* $\mathcal{N}_{\mathtt{r}}(\mathtt{a})$ *soit un arbre en tas dont le dernier nœud est de numéro* `f`*. Seules les cases de* $\mathcal{N}_{\mathtt{r}}(\mathtt{a})$ *peuvent être modifiées. Cette procédure est donc telle que :*

- a *est le contenu de* `a` *avant l'exécution de l'appel* `placer(r,a,f)` *;*
- a' *est le contenu de* `a` *après l'exécution de l'appel* `placer(r,a,f)` *;*
- $\mathcal{P}_m(a) \wedge \mathcal{P}_m(a') \wedge \mathcal{T}_p(\mathcal{N}_{\mathtt{r}}(a'))$ *;*
- $2^{m-1} \leq n < 2^m \wedge p + \mathtt{r} \in \{m-1, m\}$ *;*
- $\langle a_i \rangle_{i=1}^{\mathtt{r}-1} = \langle a'_i \rangle_{i=1}^{\mathtt{r}-1}$ *;*
- $\{a_i\}_{i=\mathtt{r}}^{\mathtt{f}} = \{a'_i\}_{i=\mathtt{r}}^{\mathtt{f}}$ *;*
- $\langle a_i \rangle_{i=\mathtt{f}+1}^{n} = \langle a'_i \rangle_{i=\mathtt{f}+1}^{n}$.

Cette fonction devra au plus parcourir une branche de l'arbre binaire parfait représenté par `a`*. Cette fonction ne devra pas être récursive ni faire appel à des fonctions auxiliaires récursives.*

Question III.29 *Ecrire en Python une procédure* `entasser(a)` *qui permute le contenu de certaines cases du tableau d'entiers* `a` *pour transformer l'arbre binaire parfait contenu dans* `a` *avant l'appel en un arbre en tas contenu dans* `a` *après l'appel. Cette procédure est donc telle que :*

— `a` *est une séquence d'entiers avec* $dom(\texttt{a}) = [1, n]$ *;*

— a *est le contenu de* `a` *avant l'exécution de l'appel* `entasser(a)` *;*

— a' *est le contenu de* `a` *après l'exécution de l'appel* `entasser(a)` *;*

— $\mathcal{P}_m(a)$ *;*

— $\mathcal{T}_m(a')$ *;*

— $2^{m-1} \leq n < 2^m$ *;*

— $codom(a) = codom(a')$.

Cette fonction ne devra pas être récursive ni faire appel à des fonctions auxiliaires récursives.

2.6 Tri par tas

La structure d'arbre en tas permet d'implanter un algorithme de tri efficace appelé tri par tas. La structure d'arbre binaire partiellement ordonné assure que l'étiquette de la racine de l'arbre en tas contient la plus grande étiquette du tas. L'algorithme répète l'étape suivante jusqu'à ce que le tas soit vide :

— l'étiquette de la racine est placée à la fin de la séquence triée ;

— le dernier nœud du tas selon la numérotation hiérarchique est enlevé du tas. Son étiquette remplace celle de la racine. L'arbre binaire ainsi obtenu est parfait. Les sous-arbres gauche et droit de sa racine sont des arbres en tas ;

— l'arbre binaire parfait ainsi obtenu est ensuite converti en arbre en tas en utilisant la fonction `entasser`.

Question III.30 *Détailler les étapes de l'algorithme du tri par tas pour le quatrième arbre de* ***l'exemple III.1, page 8****. Pour chaque étape, donner les représentations du tas sous les formes d'arbre et de tableau.*

2.6.1 Implantation en langage Python

L'utilisation d'un tableau pour représenter un tas permet de permuter l'étiquette de la racine et celle du dernier nœud en une seule étape et de construire le tableau trié à partir de la fin.

Question III.31 *Ecrire en Python une procédure* `trier(s)` *qui permute le contenu de certaines cases du tableau* `s` *pour trier ce tableau. Cette procédure est donc telle que :*

— `s` *est une séquence d'entiers avec* $dom(\texttt{s}) = [1, n]$ *;*

— s *est le contenu de* `s` *avant l'exécution de l'appel* `trier(s)` *;*

— s' *est le contenu de* `s` *après l'exécution de l'appel* `trier(s)` *;*

— $codom(s) = codom(s')$ *;*

— $\forall i \in [1, n[, s'_i \leq s'_{i+1}$.

Cette fonction ne devra pas être récursive ni faire appel à des fonctions auxiliaires récursives.

2.6.2 Implantation en langage CaML

Le langage CaML ne permet pas de permuter en une seule étape les étiquettes de la racine et du dernier nœud. Cette permutation sera donc combinée avec les opérations `placer` et `entasser` qui réorganisent l'arbre binaire pour préserver la structure de tas.

Question III.32 *Ecrire en CaML une fonction* `remonter` *dont le type est* `int -> arbre -> arbre` *telle que l'appel* `(remonter i a)` *sur l'entier* `i` *avec* $\mathtt{i} = |\mathtt{a}|$ *et l'arbre en tas* `a`*, renvoie un arbre en tas de taille* $|\mathtt{a}| - 1$ *qui contient les mêmes étiquettes que* `a` *sauf celle de la racine* $\mathcal{E}(\mathtt{a})$*. Cette fonction devra au plus parcourir une fois la branche allant de la racine au dernier nœud de* `a`*. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

Question III.33 *Ecrire en CaML une fonction* `trier` *dont le type est* `int list -> int list` *telle que l'appel* `(trier l)` *sur la liste d'entiers* `l` *renvoie une liste triée d'entiers contenant les mêmes valeurs que* `l`*. Cette fonction exploitera la technique du tri par tas. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

Fin de l'énoncé

CONCOURS ARTS ET MÉTIERS ParisTech - ESTP - POLYTECH

Épreuve d'Informatique MP

Durée 3 h

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, d'une part il le signale au chef de salle, d'autre part il le signale sur sa copie et poursuit sa composition en indiquant les raisons des initiatives qu'il est amené à prendre.

L'usage de calculatrices est interdit.

AVERTISSEMENT

La **présentation**, la lisibilité, l'orthographe, la qualité de la **rédaction, la clarté et la précision** des raisonnements entreront pour une **part importante** dans **l'appréciation des copies**. En particulier, les résultats non justifiés ne seront pas pris en compte. Les candidats sont invités à encadrer les résultats de leurs calculs.

Les différents exercices sont indépendants.

Lorsqu'on demande d'évaluer une complexité, on attend un ordre de grandeur, par exemple $O(n)$.

Pour un nombre réel x, on note $\lfloor x \rfloor$ sa partie entière.

Exercice 1

On admettra que la multiplication de deux entiers naturels se fait en temps constant.

1. On s'intéresse dans cette partie au calcul de la puissance k-ième d'un entier naturel n, pour un entier naturel $k \geq 1$.

   (a) Dans un premier temps, on utilise un algorithme naïf :

```
let rec puissance n k =
    if k = 1 then n
    else n * (puissance n (k-1));;
val puissance : int -> int -> int
```

   Quelle est la complexité de cet algorithme ? Le justifier précisément.

   (b) On utilise ensuite l'algorithme suivant :

```
let rec puissance2 n k =
    if k > 1 then
        let x = puissance2 n (k/2) in
        if k mod 2 = 0 then
            x * x
        else
            x * x * n
    else n;;
val puissance2 : int -> int -> int
```

   i. Décrire l'exécution du programme `puissance2` sur l'entrée $(2, 7)$.
   ii. Décrire l'exécution du programme `puissance2` sur l'entrée $(2, 8)$.
   iii. Démontrer que le nombre d'appels récursifs à l'intérieur du programme `puissance2` sur l'entrée (n, k) est au plus de $\log_2 k + 1$. En déduire que le programme termine.
   iv. Justifier que ce programme est correct.
   v. Evaluer la complexité de ce programme.

2. On s'intéresse ici au problème de déterminer si un entier naturel est une puissance non triviale d'un nombre entier ou non : on dit qu'un nombre entier naturel n est une *puissance entière* s'il existe deux entiers naturels k et m tous deux > 1 tels que $n = m^k$.

   (a) Ecrire la fonction :

```
test_puissance : int -> int -> bool
```

   qui prend en entrée les entiers naturels $n > 1$ et $k > 1$ et renvoie le booléen `true` s'il existe un entier naturel m tel que $N = m^k$ et `false` sinon.

   (b) i. Soit n un entier naturel. On suppose que n est une puissance entière : Soient k et m deux entiers naturels > 1 tels que $n = m^k$. Justifier que $k \leqslant \log_2(n)$.

ii. Ecrire la fonction :

```
test_puissance_entiere : int -> bool
```

qui prend en entrée l'entier naturel $n > 1$ et renvoie le booléen `true` s'il existe deux entiers naturels k et m tous deux > 1 tels que $n = m^k$ et `false` sinon.

iii. Démontrer que la complexité de ce programme est au plus $O(n \log_2(n) \log_2(k))$.

iv. En déduire la fonction :

```
liste1_puissances_entieres : int -> int list
```

qui prend en entrée l'entier naturel $n > 1$ et renvoie la liste des entiers naturels compris entre 2 et n qui sont des puissances entières.

(c) En vous inspirant du crible d'Erathosthène, écrire la fonction :

```
liste2_puissances_entieres : int -> int list
```

qui prend en entrée l'entier naturel $n > 1$ et renvoie la liste des entiers naturels compris entre 2 et n qui sont des puissances entières.

Exercice 2

Soit Σ l'alphabet des chiffres : $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
Un nombre entier naturel non nul n est considéré à l'aide de son écriture en base 10 comme un mot sur l'alphabet Σ :

$$n \text{ s'écrit } a_l a_{l-1} \cdots a_0 \Leftrightarrow n = a_l 10^l + a_{l-1} 10^{l-1} + \cdots + a_0, \text{ avec } a_l \neq 0.$$

Un mot sur Σ sera considéré comme un tableau de type `int vect` dont les éléments sont compris entre 0 et 9. Par exemple le nombre 2015 sera représenté par `[|2;0;1;5|]`.

On dit qu'un entier naturel non nul a la propriété P_{2015} si son écriture en base 10 comprend le motif 2015. On cherche à reconnaître les entiers naturels non nuls ayant cette propriété.

1. Dessiner un automate déterministe complet qui reconnait exactement les entiers naturels non nuls ayant la propriété P_{2015}.

2. On considère la fonction G qui envoie un mot $abcd$ dans Σ^4 sur l'entier naturel $a * 1000 + b * 100 + c * 10 + d$.

   (a) Combien de valeurs distinctes la fonction G peut-elle prendre ?

   (b) Etant donné un mot $abcde$ dans Σ^5, expliciter comment calculer $G(bcde)$ en fonction de $G(abcd)$ et e.

   (c) Ecrire la fonction :

   ```
   decalG : int -> int -> int
   ```

   qui prend en entrée l'entier naturel $G(abcd)$ et le chiffre e dans Σ et renvoie la valeur $G(bcde)$, pour $abcde$ un mot dans Σ^5.

(d) On utilise la fonction G pour résoudre le problème. Etant donné un entier naturel non nul n s'écrivant $a_l a_{l-1} \cdots a_0$ en base 10, on calcule la valeur prise par G sur chaque facteur $a_i a_{i-1} a_{i-2} a_{i-3}$. Ecrire la fonction :

```
testG_motif : int vect -> bool
```

qui prend en entrée le tableau défini par l'écriture en base 10 d'un entier naturel non nul n et renvoie le booléen `true` si l'entier naturel n a la propriété P_{2015} et `false` sinon. Cette fonction ne lira qu'une fois au plus chacun des caractères de l'écriture en base 10 de n.

(e) Evaluer la complexité de votre algorithme.

(f) Ecrire la fonction :

```
nbG_motif : int vect -> int
```

qui prend en entrée le tableau défini par l'écriture en base 10 d'un entier naturel non nul n et renvoie le nombre d'occurences du motif 2015 dans l'écriture en base 10 de n.

3. On considère la fonction H qui envoie un mot $abcd$ dans Σ^4 sur l'entier naturel compris entre 0 et 10 égal à $a * 1000 + b * 100 + c * 10 + d$ modulo 11.

(a) Combien de valeurs distinctes la fonction H peut-elle prendre ?

(b) Ecrire la fonction :

```
calculH : int vect  -> int
```

qui prend en entrée un mot $abcd$ dans Σ^4 et renvoie la valeur de H prise sur ce mot.

(c) Etant donné un mot $abcde$ dans Σ^5, expliciter comment calculer $H(bcde)$ en fonction de $H(abcd)$, a et e.

(d) En utilisant la fonction H, écrire la fonction :

```
testH_motif : int vect -> int
```

qui prend en entrée le tableau défini par l'écriture en base 10 d'un entier naturel non nul n et renvoie renvoie le booléen `true` si l'entier naturel n a la propriété P_{2015} et 0 (ou `false`) sinon. Cette fonction ne lira que trois fois au plus chacun des caractères de l'écriture en base 10 de n.

(e) Evaluer la complexité de votre algorithme.

Exercice 3

On souhaite analyser les résultats de sondages concernant une élection. Les candidats à l'élection sont numérotés de 0 à $k-1$. Les résultats de chaque sondage sont stockés dans un tableau : si le sondage a recueilli N réponses, le tableau comporte N cases, une pour chaque réponse : la i-ème case du tableau contient le numéro du candidat proposé par la i-ème personne sondée. On a ainsi un tableau T de longueur N qui contient des entiers naturels entre 0 et $k-1$.

Le sondage donne le candidat numéroté i élu si le nombre i est dans strictement plus de $N/2$ cases de T. Par exemple, un sondage correspondant au tableau $[2, 4, 5, 0, 4, 4, 4]$ donne le candidat 4 élu. Mais un tableau ne donne pas toujours un élu, par exemple le tableau $[1, 2, 3, 4, 6, 2, 3, 3]$.

On suppose qu'un entier k est défini.

1. (a) Écrire une fonction `nb` de type `int vect -> int -> int` telle que si `a` est un entier naturel et `tab` un tableau de taille N issu d'un tel sondage, `nb tab a` est le nombre de cases du tableau `tab` qui contiennent `a`.
   (b) Evaluer la complexité de l'appel de `nb` en fonction de N et de k.
   (c) En déduire une fonction `elu1` de type `int vect -> int` telle que `elu1 tab` est l'entier donné élu par le tableau `tab` si celui-ci existe et `-1` sinon.
   (d) Evaluer la complexité de votre algorithme en fonction de N et de k.

2. On se propose d'utiliser la stratégie diviser pour régner pour déterminer l'éventuel élu donné par un tableau. On dispose de deux fonctions, qu'on ne cherchera pas à décrire :
   - `miGauche : int vect -> int vect` qui prend en argument un tableau `tab` de longueur $N \geqslant 2$ et retourne le tableau de longueur $\lfloor N/2 \rfloor$ formé par les $\lfloor N/2 \rfloor$ premières cases de `tab`,
   - `miDroite : int vect -> int vect` qui prend en argument un tableau `tab` de longueur $N \geqslant 2$ et retourne le tableau de longueur $N - \lfloor N/2 \rfloor$ formé par les $N - \lfloor N/2 \rfloor$ dernières cases de `tab`.

   (a) Soit `tab` un tableau de longueur $N \geqslant 2$. Démontrer que si `tab` donne a comme élu alors celui-ci est aussi donné élu par le tableau `miGauche tab` ou par le tableau `miDroite tab`.
   (b) Proposer une fonction `elu2`, utilisant la stratégie diviser pour régner, de type `int vect -> int * int` ; si `tab` est un tableau, `elu2 tab` est le couple `(a,n)` si l'entier `a` est donné élu par `tab` et apparait dans `n` cases exactement de `tab`, et `(-1,0)` si `tab` ne donne pas d'élu. On pourra utiliser la fonction `nb` définie en 1(a).
   (c) Evaluer la complexité de cette fonction en fonction de N.

3. Soit T un tableau de longueur N. On dit que le nombre entier a est un *postulant* pour la valeur n du tableau T si : n est un entier strictement supérieur à $N/2$ tel que a apparait au plus (au sens large) n fois dans T et tout entier b distinct de a, apparait au plus (au sens large) $N - n$ fois dans T. Par exemple, 3 est un postulant pour $n = 5$ du tableau $[1, 2, 3, 4, 3, 2, 3, 3]$.

   On dit que le nombre entier a est un postulant du tableau T s'il existe un nombre entier $n > N/2$ tel que a est un postulant pour la valeur n du tableau T

   (a) Démontrer que si le tableau T donne a élu alors a est un postulant de T.
   (b) Démontrer que si a est un postulant de T, alors aucun autre élément de T ne pourrait être donné comme élu.
   (c) Donner un exemple de tableau qui contient un postulant mais ne donne aucun élu et un exemple de tableau n'ayant aucun postulant.
   (d) Soit T un tableau de longueur un entier pair N. On note TG le tableau de longueur $N/2$ formé par les $N/2$ premières cases de T et TD le tableau de longueur $N/2$ formé par les $N/2$ dernières cases de T.
      i. On suppose que le tableau TD ne donne pas d'élu. Soit a un postulant pour la valeur l du tableau TG. Démontrer que a est un postulant de T. On exprimera la valeur d'un entier n tel que $n > N/2$ et a est un postulant pour la valeur n du tableau T en fonction de l et N.

ii. Soient a un postulant pour la valeur l du tableau TG et b un postulant pour la valeur m du tableau TD.

A. On suppose que $a = b$. Démontrer que a est un postulant de T. On exprimera la valeur d'un entier n tel que $n > N/2$ et a est un postulant pour la valeur n du tableau T en fonction de l et m.

B. On suppose que $a \neq b$ et $m > l$. Démontrer que b est un postulant pour la valeur $N/2 + m - l$ de T.

C. On suppose que $a \neq b$ et $m = l$. Démontrer que T ne donne pas d'élu.

(e) Ecrire une fonction `postulant : int vect -> int * int` telle que, si `tab` est un tableau d'entiers naturels de longueur N, `postulant tab` est un couple `(a,n)` tel que

- lorsque `postulant tab` renvoie le couple `(-1,0)`, le tableau `tab` n'a pas d'élu,
- lorsque `postulant tab` renvoie le couple `(a,n)` avec `n` $> N/2$, `a` est un postulant pour la valeur `n` du tableau `tab`. On supposera pour simplifier que la taille du tableau est une puissance de 2. La procédure utilisera la stratégie diviser pour régner et aura une complexité linéaire, ce qu'on justifiera précisément.

(f) En déduire une fonction `elu3`, de type `int vect -> int` qui prend en argument un tableau `tab` et retourne l'entier élu de `tab` si celui-ci existe et `-1` sinon, de complexité linéaire.

1) Recherche trichotomique…

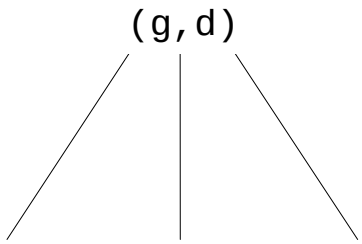
La recherche dichotomique (principe bien connu) consiste à séparer en deux parties de taille quasi-égale l'ensemble de données triées sur lequel se fait la recherche d'un élément. Pour cela on détermine un élément pivot auquel est comparé celui que l'on recherche. Si l'élément correspond, la recherche est positive, sinon on recommence sur une des deux parties précédemment déterminées. Et ainsi de suite jusqu'à trouver l'élément recherché ou constater son absence de l'ensemble.

La « trichotomie » consiste elle en une découpe de l'ensemble en trois parties quasi-égales.

a) Donnez le principe de recherche trichotomique d'un élément `x` dans une liste λ de `n` éléments triés en ordre croissant bornée de `0` à `n-1`. Votre principe doit décrire une fonction entière qui retourne le rang de `x` dans λ s'il existe et `-1` sinon.

b) Ecrivez, sous la forme d'une fonction récursive C, Caml ou Python, l'algorithme correspondant au principe du a).

c) Représentez l'arbre d'évaluation de la recherche trichotomique d'un élément dans une liste de `20` éléments. Chaque nœud de l'arbre devra présenter les valeurs de l'intervalle `(g,d)` de recherche. Comme sur l'exemple suivant:



d) En extrapolant à `n` donnez l'ordre de grandeur de la complexité, au pire, en nombre de récursions de la recherche trichotomique positive d'un élément `x` parmi une liste de `n` éléments.

e) Justifiez votre réponse.

2) Liste bitonique …

Une liste non vide est ***bitonique*** si elle contient une suite (éventuellement vide) croissante suivie d'une suite (éventuellement vide) décroissante. Une liste vide n'est pas bitonique.

Exemples :
Les listes suivantes sont bitoniques:

- `1,4,8,12,5,2`
- `12,25,40,52`
- `15,8,3`

Les listes suivantes ne sont pas bitoniques:

- `1,5,10,8,6,12`
- `15,12,11,9,10,14,16`

Écrire la fonction C, Caml ou Python `bitonique(l)` qui détermine si une liste `l` est bitonique. Si c'est le cas, votre fonction devra retourner l'index du point culminant (la valeur la plus haute) de la liste, la valeur `-1` sinon.

Par exemple, dans la liste `1,4,8,12,5,2` le point culminant est `12`. Dans la liste `12,25,40,52` c'est `52`.

3) Graphes…

Soit `G` un graphe non orienté dont les sommets sont des entiers de `1` à `8` et dont les sommets adjacents de chaque sommet sont donnés dans le tableau suivant:

Sommets	Sommets adjacents
1	(2,3,4)
2	(1,3,4)
3	(1,2,4)
4	(1,2,3,6)
5	(6,7,8)
6	(4,5,7)
7	(5,6,8)
8	(5,7)

On considérera que lors du parcours du graphe les sommets adjacents d'un sommet donné sont rencontrés dans le même ordre qu'ils sont listés dans le tableau précédent.

a) Représenter graphiquement le graphe correspondant à `G`.

b) Ce graphe est-il complet ? Ce graphe est-il connexe ?

c) Donner la séquence des sommets de `G` obtenus lors du parcours profondeur en commençant sur le sommet `1`.

d) Donner la séquence des sommets de `G` obtenus lors du parcours largeur en commençant sur le sommet `1`.

Sélection internationale Session 2014
École Normale Supérieure Paris
Épreuve de culture scientifique - **Informatique** Durée : 3 heures

Pour les candidats ayant choisi **l'informatique** comme **spécialité principale**

Si vous ne parvenez pas à répondre à une question, vous pouvez cependant l'utiliser comme hypothèse pour les questions suivantes.

Calculatrices interdites.

Exercice 1.

1. Décrire une structure de données S pour stocker des éléments pondérés qui permet les opérations suivantes.

   - Insérer(S, elem, w) ajoute un élément de poids w à la structure S,
   - Recherche-min(S) retourne un élément de poids minimum dans S ainsi que son poids,
   - Efface-min(S) enlève l'élément de poids minimum de S.

   Le temps de calcul de Insérer et Efface-min doivent être $O(\log n)$ où n est le nombre d'éléments dans S au moment de l'appel. Recherche-min doit fonctionner en $O(1)$.

   Vous pouvez supposer que les poids sont entiers.

   Donner un argument bref expliquant que vos fonctions se déroulent dans les temps souhaités et produisent des résultats corrects.

2. Combien de temps faut-il pour aller de la ville a à la ville z si le temps de voyage entre les villes sont les suivantes?

   - 1 minutes entre les villes a et b
   - 9 minutes entre les villes a et c
   - 4 minutes entre les villes a et d
   - 5 minutes entre les villes b et e
   - 12 minutes entre les villes b et f
   - 12 minutes entre les villes b et f
   - 4 minutes entre les villes c et f
   - 4 minutes entre les villes c et g
   - 2 minutes entre les villes d et g
   - 8 minutes entre les villes e et h
   - 3 minutes entre les villes f et h
   - 7 minutes entre les villes f et z
   - 2 minutes entre les villes f et i
   - 3 minutes entre les villes h et z

- 6 minutes entre les villes i et z

3. Utilisant le langage de programmation de votre choix, écrire un programme qui prends en entrée des entiers décrivant le temps de voyage entre deux villes et calcule le temps minimum qu'il faut pour aller de la ville 1 à la ville 2. Chaque ville est représenté par un numéro entre 1 et 1000000.

   Le temps de calcul de votre programme devrait être $O((m+n)\log m)$ où m est le nombre de paires de villes donnés à l'entrée et n le nombre de villes.

   Supposer que vous avez accès à la structure décrit dans la Question 1.

4. Maintenant, supposons que traverser une ville (c'est à dire arriver et repartir) prends aussi du temps. Le temps requis est la moitié du nombre de minutes restant jusqu'à la prochaine heure. Donc, si vous arrivez à 2:16 (soit 136 minutes après le départ), vous sortez de la ville dans $44/2 = 22$ minutes. Arriver pile à l'heure vous laisser sortir immédiatement.

   Décrire un algorithme pour trouver le temps minimum requis pour arriver à la ville z de la ville a si vous commencez à a à minuit (i.e., 00:00).

5. Décrire comment le temps de calcul d'une des opérations Insérer or Efface-min de la Question 1 peut être améliorée en gardant le temps de calcul des deux autres opérations.

Exercice 2.

Un *graphe* G est un paire ordonnée (V, E) où V est un ensemble appelé sommets et E est un sous ensemble des paires (non-ordonnées) de V appelé arêtes. Deux sommets $u, v \in V$ sont *adjacent* si (u, v) est un élément de E. Le *degré d'un sommet* u est le nombre d'arêtes contenant u.

Un *chemin* dans un graphe G est une suite de sommets $v_1, v_2, \ldots, v_k$ où v_i et v_{i+1} sont adjacents pour tout i and aucun sommet n'apparaît deux fois dans cette suite. Un *cycle* dans un graphe G est un chemin où v_1 et v_k sont adjacent. Un *cycle Hamiltonian* dans un graphe G est un cycle qui contient tout sommet de G.

Le but de cet exercice est de donner un algorithme polynomial pour trouver un cycle Hamiltonian dans un graphe $G = (V, E)$ avec au moins trois sommets et où tous les sommets ont degré au moins $|V|/2$. Un graphe avec ces propriétés est un *graphe de degré élevé*.

Vous pouvez directement donner un tel algorithme et démontrer qu'il est correcte et a un temps de calcul polynomial à la place de répondre aux Questions 1-5. Dans ce cas, vous devez toujours répondre à la Question 6.

1. Donner un algorithme polynomial qui prends en entrée un graphe de degré élevé G et chemin $P = v_1, \ldots, v_k$ dans G où $k < n$ et retourne soit un chemin avec $k + 1$ sommets ou un cycle contenant les sommets de P (possiblement dans un ordre différent).

2. Donner un algorithme polynomial qui prends en entrée un graphe de degré élevé G et cycle C dans G où $k < n$ et retourne un chemin avec $k + 1$ sommets contenant tous les sommets de C (possiblement dans un ordre différent).

3. Donner un algorithme polynomial qui prends en entrée un graphe de degré élevé G et retourne un cycle Hamiltonian dans G.

4. Montrer que le temps de calcul de votre algorithme de la Question 3 est bien polynomial. Écrire explicitement la valeur de k pour lequel votre algorithme a un temps de calcul $O(n^k)$.

5. Montrer que votre algorithme de la Question 3 est correcte. C'est à dire, montrer qu'il retourne bien un cycle et que celui-ci est Hamiltonian. (Selon votre algorithme, il est peut être nécessaire de montrer qu'il ne bloque pas et termine.)

6. Expliquer comment changer l'algorithme de la Question 3 en un algorithme qui prends en entrée un graphe $G = (V, E)$ avec au moins 3 sommets et où la somme des degrés de n'importe quel paire de sommets non-adjacents u, v est au moins $|V|$ et qui retourne un cycle Hamiltonian dans G.

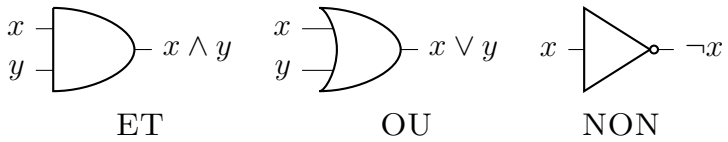
Sélection internationale Session 2014
École Normale Supérieure Paris
Épreuve de culture scientifique - **Informatique** Durée : 2 heures

Pour les candidats ayant choisi **l'informatique** comme **spécialité secondaire**

Si vous ne parvenez pas à répondre à une question, vous pouvez cependant l'utiliser comme hypothèse pour les questions suivantes.

Calculatrices interdites.

Exercice 1.
Les portes logiques sont des éléments d'un circuit dont les entrées et les sorties sont binaires. Les trois portes logiques élémentaires ET, OU and NON sont décrites ci-dessous.



Dans les questions suivantes, indiquer clairement les entrées et les sorties dans tous les diagrammes de circuit. Éviter autant que possible d'utiliser des symboles supplémentaires et si c'est le cas, indiquer dans votre réponse ce qu'ils représentent.

1. En utilisant uniquement les portes ET, OU et NON, dessiner un diagramme de circuit pour un additionneur de 1 bit qui prend en entrée deux bits a_0, b_0 et (éventuellement) une retenue r et retourne la somme s_0 de a_0, b_0 et r et la retenue s_1.

$$s_0 = a_0 + b_0 + r \bmod 2 \text{ et } s_1 = \lfloor (a_0 + b_0 + r)/2 \rfloor$$

2. En utilisant uniquement les portes ET, OU et NON et le circuit de la question précédente, dessiner un diagramme pour un additionneur de 3 bits qui prend en entrée a_2, b_2, a_1, b_1, a_0, b_0, r representant deux nombres de 3 bits a and b en binaire (noté $a_2a_1a_0$ et $b_2b_1b_0$), et un bit de retenue r. Il devra retourner la somme s de ces trois nombres en binaire sous la forme s_3, s_2, s_1, s_0.

$$\sum_{i=0}^{3} s_i 2^i = \sum_{i=0}^{2} a_i 2^i + \sum_{i=0}^{2} b_i 2^i + r$$

Expliquer comment étendre cet additionneur en un additionneur pour n bits.

3. Déterminer le nombre de portes $g(n)$ nécessaires pour le circuit de l'additionneur pour n bits de la Question 2.

4. Supposons que le temps de propagation d'un signal logique à travers une porte élémentaire est de t secondes (indépendamment de la porte). Si tous les signaux d'entrée sont altérés pour l'additionneur pour n bits de la Question 2 au temps 0, quel est le temps nécessaire pour l'obtention du résultat en sortie du circuit (en fonction de n et t).

Nous appelons un additionneur de 2^k bits *diviser-pour-régner* le circuit défini récursivement de la façon suivante :

- L'additionneur de 1-bit diviser pour régner est l'additionneur de 1-bit de la Question 1.
- Pour $k \geq 1$, l'additionneur de 2^k-bit diviser pour régner est obtenu en fusionnant deux additionneurs de 2^{k-1}-bit diviser pour régner A et B en envoyant les 2^{k-1} bits de poids faible des entrées a et b et l'éventuelle retenue r à A et tous les bits restants à B. La retenue de la sortie de A est envoyée à B.

5. Déduire le délai pour l'obtention du résultat pour ce circuit en fonction de $n = 2^k$ et de t.

Nous considérons maintenant un additionneur de 2^k bits *diviser-pour-régner amélioré* où le circuit agissant sur les bits de poids forts est remplacé par deux additionneurs, l'un supposant que la retenue entrante vaut 1 et l'autre supposant que la retenue entrante vaut 0.

6. Dessiner le diagramme de circuit utilisant trois additionneurs de 2^{k-1} bits diviser-pour-régner amélioré et quelques portes logiques élémentaires pour construire un additionneur de 2^k bits diviser-pour-régner amélioré.

7. Déduire le délai pour l'obtention du résultat pour cet additionneur de 2^k bits diviser-pour-régner amélioré en fonction de $n = 2^k$ et de t.

8. Donner une formule récursive donnant le nombre de portes logiques élémentaires dans cet additionneur de 2^k bits diviser-pour-régner amélioré.

Ce sujet comporte trois parties indépendantes. Il est recommandé de lire l'ensemble du sujet avant de commencer la rédaction. Il est également conseillé de traiter les questions dans l'ordre de l'énoncé. On pourra cependant aborder une question en admettant les résultats des questions précédentes. Les algorithmes demandés seront écrits dans un langage de programmation au choix du candidat.

Définitions et notations

Dans ce sujet, on s'intéresse à la gestion de matrices. On note $A_{i,j}$ l'élément situé sur la $i^{\text{ème}}$ ligne et la $j^{\text{ème}}$ colonne de la matrice A. On considère que tous les indices commencent à 0 (donc l'élément en haut à gauche de la matrice A est $A_{0,0}$ et celui en bas à droite est $A_{n-1,n-1}$). Toutes les matrices considérées seront de taille $n \times n$ (n lignes et n colonnes).

On considère dans ce sujet des matrices particulières, appelées *matrices creuses*, dans lesquelles la plupart des éléments sont nuls. On rencontre souvent de telles matrices en pratique, et si on y prend garde, elles nécessitent beaucoup moins de calcul que des matrices pleines pour de nombreuses opérations. Dans ce sujet, on va concevoir des structures de données et des algorithmes spécifiques permettant de réduire la complexité de certaines opérations sur ces matrices. Ces algorithmes pourront traiter n'importe quelles matrices (creuses ou pleines), mais permettront des gains importants de complexité avec les matrices contenant beaucoup de zéros.

Pour analyser la complexité des algorithmes (ainsi que la taille des données) on utilisera d'une part la taille n de la matrice et d'autre part le nombre d'éléments non-nuls : on note $nnz(A)$ le nombre d'éléments non-nuls (ou *non-zéros*) de la matrice A. En général, $nnz(A)$ est beaucoup plus petit que n^2 : on préférera donc utiliser $nnz(A)$ à la place de n^2 dans les analyses de complexité lorsque cela est possible.

Dans la description des algorithmes, nous utiliserons la notation $a \leftarrow b$ pour l'affectation de la valeur de b à la variable a. Pour un tableau T contenant m éléments, on notera $T[i]$ le $i^{\text{ème}}$ élément de T pour i allant de 0 à $m-1$.

Partie 1 Représentations et opérations de base

Pour éviter de stocker tous les éléments non nuls d'une matrice creuse, une solution simple consiste à représenter une matrice sous la forme de trois tableaux de taille $nnz(A)$ chacun : deux tableaux `ligne` et `colonne` contenant les indices des éléments non-nuls, et un tableau `valeur` contenant leur valeur, tels que $A_{\texttt{ligne}[k],\texttt{colonne}[k]} = \texttt{valeur}[k]$. On impose que `colonne` est trié par ordre croissant, ainsi que les portions de `ligne` correspondant à une même colonne. On appelle ce format la **représentation simple triée** de la matrice.

Voici un exemple de matrice 4×4 (avec les indices de lignes et colonnes) et sa représentation simple triée.

$$A = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{pmatrix} 45 & 0 & 32 & 0 \\ 31 & 0 & 0 & 9 \\ 0 & 17 & 30 & 0 \\ 35 & 4 & 0 & 10 \end{pmatrix} \end{array}$$

```
colonne = [ 0,  0,  0,  1,  1,  2,  2,  3,  3]
ligne   = [ 0,  1,  3,  2,  3,  0,  2,  1,  3]
valeur  = [45, 31, 35, 17,  4, 32, 30,  9, 10]
```

Question 1. À partir d'une matrice de taille $n \times n$ donnée sous la forme d'un tableau à deux dimensions, tel que `A[i][j]` contient l'élément $A_{i,j}$, donner un algorithme qui construit sa représentation simple triée. On précisera la complexité de l'algorithme en fonction de n et $nnz(A)$.

Pour diminuer encore la taille de la représentation d'une matrice creuse, on introduit la **représentation compacte** qui comporte :

— un tableau `colnnz` de taille $n+1$, tel que `colnnz`$[0] = 0$ et pour $1 \leqslant j \leqslant n$, `colnnz`$[j]$ est le nombre d'éléments non-nuls dans les colonnes 0 à $j-1$ de A ;

— les tableaux `ligne` et `valeur` identiques à ceux de la représentation simple triée.

Voici la représentation compacte de la matrice A de l'exemple précédent (les espaces supplémentaires dans les tableaux ci-dessous sont uniquement destinés à aider la compréhension).

```
colnnz = [ 0,          3,      5,      7,     9]
ligne  = [ 0,  1,  3,  2,  3,  0,  2,  1,  3]
valeur = [45, 31, 35, 17,  4, 32, 30,  9, 10]
```

Question 2. On considère une matrice A en représentation compacte. Étant donnés k et l tels que $0 \leqslant k \leqslant n-1$ et $0 \leqslant l \leqslant$ `colnnz`$[k+1]-$`colnnz`$[k]-1$, que vaut `valeur`[`colnnz`$[k]+l$] ?

Question 3. Comment modifier l'algorithme de la question 1 pour construire la représentation compacte d'une matrice donnée sous la forme d'un tableau à deux dimensions ?

Dans la suite de ce sujet, nous considérerons que **toutes les matrices sont données sous la forme de leur représentation compacte**.

⋆ ⋆ ⋆

Dans la suite du sujet, on cherche à prédire la position ou le nombre des éléments non-nuls d'une matrice résultat d'une opération, en se basant uniquement sur la position des éléments non-nuls des opérandes. Par exemple, considérons l'opération $C \leftarrow A + B$. Étant donnée la position des éléments non-nuls de A et B, on dit qu'un élément $C_{i,j}$ est *essentiellement nul* s'il est égal à zéro quelles que soit les valeurs des éléments non-nuls de A et B. Par exemple, pour les matrices suivantes (où les X représentent les éléments non nuls) :

$$A = \begin{pmatrix} 0 & X \\ 0 & X \end{pmatrix} \qquad B = \begin{pmatrix} 0 & X \\ X & 0 \end{pmatrix}$$

$C_{0,0}$ est le seul élément essentiellement nul, même si d'autres éléments de C peuvent s'annuler en fonction des valeurs des éléments non-nuls de A et B. Dans la suite du sujet, on ne s'intéressera pas aux valeurs des éléments non essentiellement nuls, et on notera $C_{i,j} = 0$ (respectivement $C_{i,j} \neq 0$) pour dire que $C_{i,j}$ est (resp. n'est pas) essentiellement nul.

Question 4. Écrire un algorithme qui, étant données deux matrices A et B, calcule le nombre d'éléments non essentiellement nuls dans la somme de matrices $A + B$ et dont la complexité est $O(nnz(A) + nnz(B))$.

Question 5. On considère le produit de matrices $C = AB$ où A , B et C sont trois matrices carrées de taille $n \times n$. On note $\mathcal{A}_j$, $\mathcal{B}_j$ et $\mathcal{C}_j$ l'ensemble des indices de lignes des éléments non (essentiellement) nuls de la $j^{\text{ème}}$ colonne de A, B et C. En particulier : $\mathcal{A}_j = \{i \text{ tels que } A_{i,j} \neq 0\}$. Montrer que

$$\mathcal{C}_j = \bigcup_{k \in \mathcal{B}_j} \mathcal{A}_k$$

Question 6. Donner un algorithme qui calcule le nombre d'éléments non essentiellement nuls du produit de deux matrices A et B. Donner sa complexité dans le pire cas, et dans le cas où les éléments non nuls de A sont répartis équitablement entre ses lignes et les éléments non nuls de B sont répartis équitablement entre ses colonnes.

Partie 2 Résolution d'un système triangulaire

On considère une matrice triangulaire inférieure L (c'est-à-dire telle que $L_{i,j} = 0$ pour $i < j$), et un vecteur b de taille n. On cherche x tel que $Lx = B$. On suppose que $L_{i,i} \neq 0$ pour tout $i = 0 \dots n-1$. On peut utiliser l'algorithme 1 pour effectuer cette résolution.

Algorithme 1 : Résolution de $Lx = b$

$x \leftarrow b$
pour $i = 0 \dots n-1$ **faire**
  $x_i \leftarrow \dfrac{x_i}{L_{i,i}}$
  pour $j = i+1 \dots n-1$ **faire**
    $x_j \leftarrow x_j - x_i \times L_{j,i}$

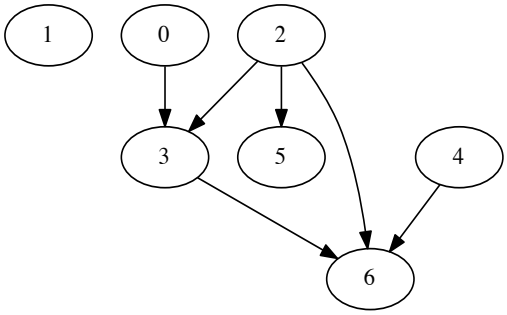
Question 7. Expliquer comment adapter l'algorithme 1 pour améliorer la complexité lorsque $nnz(L)$ est beaucoup plus petit que n^2, et que L est donnée par sa représentation compacte et b sous la forme d'un tableau de n éléments. Quelle est la complexité obtenue ? Enfin, que penser de cet algorithme si b contient beaucoup d'éléments nuls (on pourra regarder le cas $b_n = 1$ et $b_i = 0$ pour $i < n$).

On se concentre maintenant sur le cas où b contient également un grand nombre d'éléments nuls. On appelle **structure** d'une matrice (ou d'un vecteur) la position de ses éléments non-nuls (ou non essentiellement nuls dans le cas d'un résultat). On cherche à prévoir la structure de x en fonction des structures de L et b.

Question 8. On considère la valeur des coefficients de x après l'exécution de l'algorithme 1. Donner une condition nécessaire et suffisante pour que $x_i \neq 0$ en fonction de la structure de L et des x_j pour $j < i$.

Étant donnée une matrice triangulaire inférieure L, on définit le **graphe orienté** de la transposée de la matrice L, le graphe noté $G(L^T)$ où les sommets sont les entiers de 0 à $n-1$ et il existe un arc (j, i) pour tout $L_{i,j} \neq 0$ (on remarquera l'inversion des indices), que l'on pourra aussi noter $j \to i$. On donne ci-dessous l'exemple d'une matrice triangulaire inférieure L (avec indices des lignes/colonnes, les X représentant les éléments non nuls) et le graphe orienté de sa transposée $G(L^T)$.

$$L = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} X & & & & & & \\ & X & & & & & \\ & & X & & & & \\ X & & X & X & & & \\ & & & & X & & \\ & & X & & & X & \\ & & X & X & X & & X \end{pmatrix} \end{array}$$



Question 9. On appelle cycle dans un graphe orienté une suite d'arcs (u_1, u_2), (u_2, u_3), ..., (u_{k-1}, u_k), (u_k, u_1) où $k \geqslant 2$. Montrer que le graphe $G(L^T)$ ne contient pas de cycle.

Question 10. Donnez une caractérisation des nœuds i du graphe $G(L^T)$ tels que x_i n'est pas essentiellement nul. En déduire un algorithme calculant l'ensemble des indices i tels que $x_i \neq 0$.

Question 11. On suppose connus les indices i tels que $x_i \neq 0$. Donner un algorithme calculant la solution de $Lx = b$, pour b creux. L'algorithme prendra en entrée les données suivantes :

- un tableau b_ligne contentant les indices (triés par ordre croissant) des éléments non-nuls des éléments de b,
- un tableau b_valeur contentant les valeurs de ces éléments dans le même ordre,
- un tableau x_ligne contentant les indices (triés par ordre croissant) des éléments non-nuls de x, calculés grâce aux questions précédentes.

L'algorithme doit calculer un tableau x_valeur contenant les valeurs des éléments non essentiellement nuls de x. On donnera la complexité de l'algorithme proposé.

Partie 3 Décomposition de Cholesky

On suppose dans cette partie que la matrice A est symétrique (pour tout couple i, j, $A_{i,j} = A_{j,i}$). Sous certaines conditions (si la matrice est définie positive), on peut décomposer A sous la forme du produit d'une matrice triangulaire inférieure L et de sa transposée (notée L^T) : $A = LL^T$. On suppose que $A_{j,j} \neq 0$ pour tout $j = 0 \ldots n-1$. On utilise l'algorithme 2 pour effectuer ce calcul.

Algorithme 2 : Décomposition de Cholesky $A = LL^T$

pour $j = 0, \ldots, n-1$ **faire**

$$L_{j,j} \leftarrow \sqrt{A_{j,j} - \sum_{k=0}^{j-1} {L_{j,k}}^2}$$

pour $i = j+1, \ldots, n-1$ **faire**

$$L_{i,j} = \frac{1}{L_{j,j}} \left(A_{i,j} - \sum_{k=0}^{j-1} L_{i,k} L_{j,k} \right)$$

Question 12. On considère la matrice L après l'exécution de l'algorithme 2. Pour un coefficient $L_{i,j}$ avec $i > j$, donner une condition nécessaire et suffisante pour que $L_{i,j} \neq 0$ en fonction de la structure de A et des $L_{k,l}$ pour $l < j$.

⋆ ⋆ ⋆

Question 13. On considère la matrice A ci-dessous (X désigne un élément non-nul). Donner la structure du résultat L obtenu par l'algorithme 2 appliqué à A.

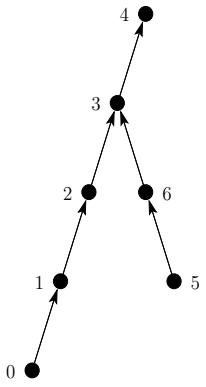
$$A = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} X & X & X & X & X \\ X & X & & & \\ X & & X & & \\ X & & & X & \\ X & & & & X \end{pmatrix} \end{array}$$

Pour une permutation de σ de $0, \ldots, n-1$, on appelle **permutation de** A **par** σ et on note $\sigma(A)$ la matrice telle que $\sigma(A)_{i,j} = A_{\sigma(i),\sigma(j)}$.

Question 14. Pour la matrice A de la question précédente, proposer une permutation σ permettant de réduire le nombre d'éléments non-nuls du résultat L obtenu par l'algorithme 2 appliqué à $\sigma(A)$.

⋆ ⋆ ⋆

Pour $0 \leqslant j < n-1$, on note $\mathit{parent}(j) = \min\{i > j \text{ tel que } L_{i,j} \neq 0\}$ (pour la matrice L calculée par l'algorithme 2). Si pour j donné, il n'existe pas de $i > j$ tel que $L_{i,j} \neq 0$, alors on définit $\mathit{parent}(j) = -1$. La fonction *parent* permet de définir l'**arbre généalogique** des entiers $0, \ldots n-1$ en fonction de la matrice L : on dit que i est un **ancêtre** de j si $i = \mathit{parent}(j)$ ou $i = \mathit{parent}(\mathit{parent}(j))$, etc. Sur la figure ci-contre, la notation $j \longrightarrow i$ signifie $i = \mathit{parent}(j)$, les ancêtres de 0 sont 1, 2, 3 et 4 et $\mathit{parent}(4) = -1$.



Question 15. Montrer que si $L_{i,j} \neq 0$, alors i est un ancêtre de j.

Question 16. Soit $i > j$. Montrer que $L_{i,j} \neq 0$ si et seulement si il existe $k \leqslant j$ tel que $A_{i,k} \neq 0$ et j est soit un ancêtre de k, soit k lui-même.

On note $\mathcal{L}_i$ l'ensemble des indices de colonne des éléments non essentiellement nuls de la $i^{\text{ème}}$ ligne de L. Formellement :

$$\mathcal{L}_i = \big\{ j \text{ tels que } L_{i,j} \neq 0 \big\}$$

Question 17. Pour i donné, connaissant l'ensemble des éléments $A_{i,k}$ non essentiellement nuls, que représente $\mathcal{L}_i$ dans l'arbre généalogique ? En déduire un algorithme qui, étant donnée la fonction *parent*, calcule le nombre d'éléments non essentiellement nuls dans chaque **colonne** de L.

On considère l'arbre généalogique construit en ne considérant que les k premières lignes et colonnes de la matrice L, et on note parent_k la fonction associée : $\mathit{parent}_k(j) = \min\{i > j \text{ tel que } i \leqslant k \text{ et } L_{i,j} \neq 0\}$ et $\mathit{parent}_k(j) = -1$ s'il n'existe pas d'indice i tel que $j < i \leqslant k$ et $L_{i,j} \neq 0$.

Question 18. Étant donné $parent_{k-1}$ et la structure de A, comment construire $parent_k$? En déduire un algorithme pour calculer $parent(j)$ pour $j = 0, \ldots, n-1$.

⋆ ⋆

⋆

ÉCOLE POLYTECHNIQUE – ÉCOLES NORMALES SUPÉRIEURES

ÉCOLE SUPÉRIEURE DE PHYSIQUE ET DE CHIMIE INDUSTRIELLES

CONCOURS D'ADMISSION 2015 **FILIÈRE MP HORS SPÉCIALITÉ INFO**

FILIÈRE PC

COMPOSITION D'INFORMATIQUE – B – (XECLR)

(Durée : 2 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation choisi par le candidat doit être spécifié en tête de la copie.

⋆ ⋆ ⋆

Enveloppes convexes dans le plan

Ce sujet a pour objectif de calculer des enveloppes convexes de nuages de points dans le plan affine, un grand classique en géométrie algorithmique. On rappelle qu'un ensemble $C \subseteq \mathbb{R}^2$ est convexe si et seulement si pour toute paire de points $p, q \in C$, le segment de droite $[p, q]$ est inclus dans C. L'enveloppe convexe d'un ensemble $P \subseteq \mathbb{R}^2$, notée $\mathrm{Conv}(P)$, est le plus petit convexe contenant P. Dans le cas où P est un ensemble fini (appelé *nuage de points*), le bord de $\mathrm{Conv}(P)$ est un polygône convexe dont les sommets appartiennent à P, comme illustré dans la figure 1.

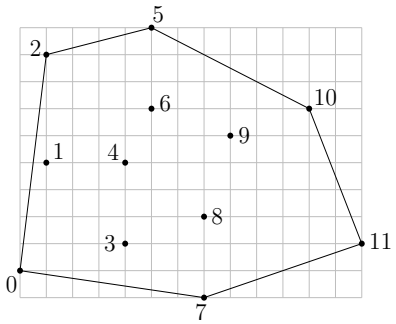


FIGURE 1 – Un nuage de points, numérotés de 0 à 11, et le bord de son enveloppe convexe.

Le calcul de l'enveloppe convexe d'un nuage de points est un problème fondamental en informatique, qui trouve des applications dans de nombreux domaines comme :

- la robotique, par exemple pour l'accélération de la détection de collisions dans le cadre de la planification de trajectoire,

- le traitement d'images et la vision, par exemple pour la détection d'objets convexes (comme des plaques minéralogiques de voiture) dans des scènes 2d,
- l'informatique graphique, par exemple pour l'accélération du rendu de scènes 3d par lancer de rayons,
- la théorie des jeux, par exemple pour déterminer l'existence d'équilibres de Nash,
- la vérification formelle, par exemple pour déterminer si une variable risque de dépasser sa capacité de stockage ou d'atteindre un ensemble de valeurs interdites lors de l'exécution d'une boucle dans un programme,

et bien d'autres encore.

Dans ce sujet nous allons écrire deux algorithmes de calcul du bord de l'enveloppe convexe d'un nuage de points P dans le plan affine. Le premier, dit *algorithme du paquet cadeau*, consiste à envelopper le nuage de points P progressivement en faisant pivoter une droite tout autour. Le deuxième, dit *de balayage*, consiste à balayer le plan horizontalement avec une droite verticale, tout en maintenant au fur et à mesure l'enveloppe convexe de la partie du nuage située à gauche de cette droite verticale. Les deux algorithmes sont illustrés respectivement dans les figures 3 et 4.

Le temps d'exécution du premier algorithme est majoré par une constante fois nm, celui du deuxième par une constante fois $n \log n$, où n désigne le nombre total de points de P et m le nombre de points de P appartenant au bord de $\mathrm{Conv}(P)$. Rappelons que le temps d'exécution d'un programme A (fonction ou procédure) est le nombre d'opérations élémentaires (comparaisons, additions, soustractions, multiplications, divisions, affectations, etc.) nécessaires à l'exécution de A. Sauf mention contraire dans l'énoncé du sujet, le candidat n'aura pas à justifier des temps de calcul de ses programmes. Toutefois, il devra veiller à ce que ces derniers ne dépassent pas les bornes prescrites.

Dans toute la suite on supposera que le nuage de points P est de taille $n \geq 3$ et en position générale, c'est-à-dire qu'il ne contient pas 3 points distincts alignés.

Ces hypothèses vont permettre de simplifier les calculs en ignorant les cas pathologiques, comme par exemple la présence de 3 points alignés sur le bord de l'enveloppe convexe. Nos programmes prendront en entrée un nuage de points P dont les coordonnées sont stockées dans un tableau *tab* à 2 dimensions, comme dans l'exemple ci-dessous qui contient les coordonnées du nuage de points de la figure 1 :

i \ j	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
0	0	1	1	4	4	5	5	7	7	8	11	13
1	0	4	8	1	4	9	6	-1	2	5	6	1

Précisons que les coordonnées, supposées entières, sont données dans une base orthonormée du plan, orientée dans le sens direct. La première ligne du tableau contient les abscisses, tandis que la deuxième contient les ordonnées. Ainsi, la colonne d'indice j contient les deux coordonnées du point d'indice j. Ce dernier sera nommé p_j dans la suite.

Partie I. Préliminaires

Question 1 Écrire une fonction plusBas(tab, n) qui prend en paramètre le tableau tab de taille $2 \times n$ et qui renvoie l'indice j du point le plus bas (c'est-à-dire de plus petite ordonnée) parmi les points du nuage P. En cas d'égalité, votre fonction devra renvoyer l'indice du point de plus petite abscisse parmi les points les plus bas.

Sur le tableau exemple précédent, le résultat de la fonction doit être l'indice 7.

Dans la suite nous aurons besoin d'effectuer un seul type de test géométrique : celui de l'orientation.

Définition 1 *Étant donnés trois points p_i, p_j, p_k du nuage P, distincts ou non, le test d'orientation renvoie $+1$ si la séquence (p_i, p_j, p_k) est orientée positivement, -1 si elle est orientée négativement, et 0 si les trois points sont alignés (c'est-à-dire si deux au moins sont égaux, d'après l'hypothèse de position générale).*

Pour déterminer l'orientation de (p_i, p_j, p_k), il suffit de calculer l'aire signée du triangle, comme illustré sur la figure 2. Cette aire est la moitié du déterminant de la matrice 2×2 formée par les coordonnées des vecteurs $\overrightarrow{p_i p_j}$ et $\overrightarrow{p_i p_k}$.

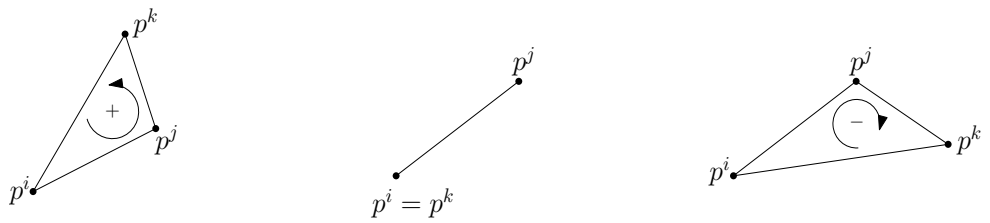


FIGURE 2 – Test d'orientation sur la séquence (p_i, p_j, p_k) : positif à gauche, nul au centre, négatif à droite.

Question 2 Sur le tableau exemple précédent, donner le résultat du test d'orientation pour les choix d'indices suivants :
- $i = 0$, $j = 3$, $k = 4$,
- $i = 8$, $j = 9$, $k = 10$.

Question 3 Écrire une fonction orient(tab, i, j, k) qui prend en paramètres le tableau tab et trois indices de colonnes, potentiellement égaux, et qui renvoie le résultat (-1, 0 ou $+1$) du test d'orientation sur la séquence (p_i, p_j, p_k) de points de P.

Partie II. Algorithme du paquet cadeau

Cet algorithme a été proposé par R. Jarvis en 1973. Il consiste à envelopper peu à peu le nuage de points P dans une sorte de paquet cadeau, qui à la fin du processus est exactement le bord de $\mathrm{Conv}(P)$. On commence par insérer le point de plus petite ordonnée (celui d'indice 7 dans l'exemple de la figure 1) dans le paquet cadeau, puis à chaque étape de la procédure on sélectionne le prochain point du nuage P à insérer.

La procédure de sélection fonctionne comme suit. Soit p_i le dernier point inséré dans le paquet cadeau à cet instant. Par exemple, $i = 10$ dans l'exemple de la figure 3. Considérons la

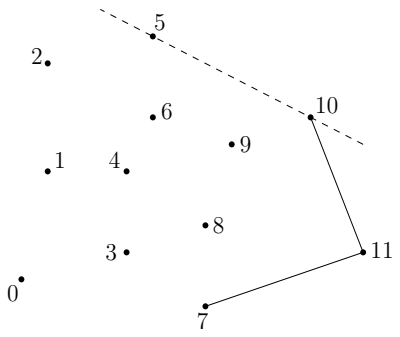


FIGURE 3 – Mise à jour du paquet cadeau après insertion du point p_{10}.

relation binaire $\preceq$ définie sur l'ensemble $P \setminus \{p_i\}$ par :

$$p_j \preceq p_k \quad \Longleftrightarrow \quad \mathsf{orient}(tab, i, j, k) \leq 0.$$

Question 4 Justifier brièvement le fait que $\preceq$ est une relation d'ordre total sur l'ensemble $P \setminus \{p_i\}$, c'est-à-dire :
- (réflexivité) pour tout $j \neq i$, $p_j \preceq p_j$,
- (antisymétrie) pour tous $j, k \neq i$, $p_j \preceq p_k$ et $p_k \preceq p_j$ implique $j = k$,
- (transitivité) pour tous $j, k, l \neq i$, $p_j \preceq p_k$ et $p_k \preceq p_l$ implique $p_j \preceq p_l$,
- (totalité) pour tous $j, k \neq i$, $p_j \preceq p_k$ ou $p_k \preceq p_j$.

Ainsi, le prochain point à insérer (le point d'indice 5 dans la figure 3) est l'élément maximum pour la relation d'ordre $\preceq$. Il peut se calculer en temps linéaire (c'est-à-dire majoré par une constante fois n) par une simple itération sur les points de $P \setminus \{p_i\}$.

Question 5 Décrire une réalisation en Python de la procédure. Elle prendra la forme d'une fonction $\mathsf{prochainPoint}(tab, n, i)$, qui prend en paramètre le tableau tab de taille $2 \times n$ ainsi que l'indice i du point inséré en dernier dans le paquet cadeau, et qui renvoie l'indice du prochain point à insérer. Le temps d'exécution de votre fonction doit être majoré par une constante fois n, pour tous n et i. La constante doit être indépendante de n et i, et on ne demande pas de la préciser.

Question 6 Décrire à la main le déroulement de la procédure prochainPoint sur l'exemple de la figure 3. Plus précisément, indiquer la séquence des points de $P \setminus \{p_{10}\}$ considérés et la valeur de l'indice du maximum à chaque itération.

On peut maintenant combiner la fonction prochainPoint avec la fonction plusBas de la question 1 pour calculer le bord de l'enveloppe convexe de P. On commence par insérer le point p_i d'ordonnée la plus basse, puis on itère le processus de mise à jour du paquet cadeau jusqu'à ce que le prochain point à insérer soit de nouveau p_i. À ce moment-là on renvoie le paquet cadeau comme résultat sans insérer p_i une seconde fois.

Un détail technique : comme la taille du paquet cadeau augmente peu à peu lors du processus, et qu'à la fin elle peut être petite par rapport au nombre n de points de P, nous stockerons les

indices des points du paquet cadeau dans une liste. Par exemple, sur le nuage de la figure 1, le résultat sera la liste [7, 11, 10, 5, 2, 0].

Question 7 Écrire une fonction convJarvis(tab, n) qui prend en paramètre le tableau tab de taille $2 \times n$ représentant le nuage P, et qui renvoie une liste contenant les indices des sommets du bord de l'enveloppe convexe de P, sans doublon. Le temps d'exécution de votre fonction doit être majoré par une constante fois nm, où m est le nombre de points de P situés sur le bord de $\mathrm{Conv}(P)$.

Question 8 Justifier brièvement le temps d'exécution de l'algorithme du paquet cadeau.

Intermède : piles d'entiers

Dans la suite nous aurons besoin d'utiliser des piles d'entiers, dont on rappelle la définition ci-dessous :

Définition 2 *Une pile d'entiers est une structure de données permettant de stocker des entiers et d'effectuer les opérations suivantes en temps constant (indépendant de la taille de la pile) :*

- *créer une nouvelle pile vide,*
- *déterminer si la pile est vide,*
- *insérer un entier au sommet de la pile,*
- *déterminer la valeur de l'entier au sommet de la pile,*
- *retirer l'entier au sommet de la pile.*

Nous supposerons fournies les fonctions suivantes, qui réalisent les opérations ci-dessus et s'executent chacune en temps constant :

- newStack(), qui ne prend pas d'argument et renvoie une pile vide,
- isEmpty(s), qui prend une pile s en argument et renvoie True ou False suivant que s est vide ou non,
- push(i, s), qui prend un entier i et une pile s en argument, insère i au sommet de s (c'est-à-dire à la fin de la liste), et ne renvoie rien,
- top(s), qui prend une pile s (supposée non vide) en argument et renvoie la valeur de l'entier au sommet de s (c'est-à-dire à la fin de la liste),
- pop(s), qui prend une pile s (supposée non vide) en argument, supprime l'entier au sommet de s (c'est-à-dire à la fin de la liste) et renvoie sa valeur.

Dans la suite il est demandé aux candidats de manipuler les piles uniquement au travers de ces fonctions, sans aucune hypothèse sur la représentation effective des piles en mémoire.

Partie III. Algorithme de balayage

Cet algorithme a été proposé par R. Graham en 1972. Nous allons écrire la variante (plus simple) proposée par A. Andrew quelques années plus tard.

La première étape consiste à trier les n points du nuage P par ordre croissant d'abscisse, en conservant tous les points de même abscisse dans un ordre arbitraire.

Question 9 Parmi les algorithmes de tri que vous connaissez, mentionnez-en un qui a un temps d'exécution majoré par une constante fois $n \log n$ sur les entrées de taille n.

À partir de maintenant, on supposera que les points fournis en entrée sont triés par abscisse croissante, comme c'est le cas dans l'exemple du tableau *tab* donné au début du sujet.

L'idée de l'algorithme est de balayer le nuage de points horizontalement de gauche à droite par une droite verticale, tout en mettant à jour l'enveloppe convexe des points de P situés à gauche de cette droite, comme illustré dans la figure 4.

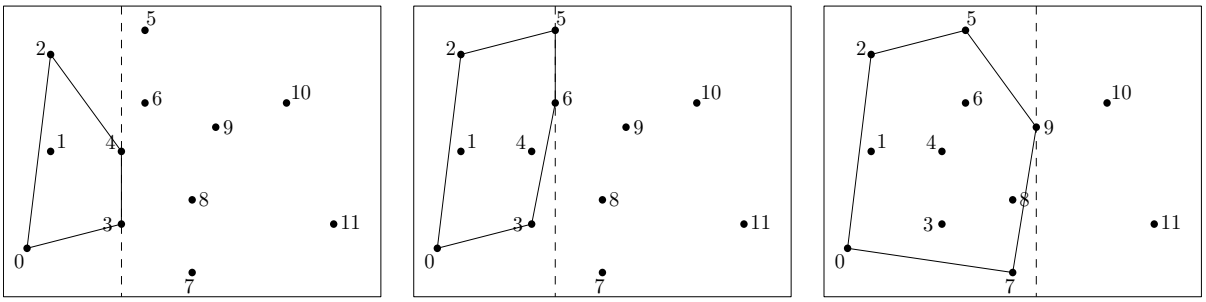


FIGURE 4 – Diverses étapes dans la procédure de balayage. La droite de balayage est en tirets.

Plus précisément, l'algorithme visite chaque point de P une fois, par ordre croissant d'abscisse (donc par ordre croissant d'indice de colonne dans le tableau *tab* car celui-ci est trié). À chaque nouveau point p_i visité, il met à jour le bord de l'enveloppe convexe du sous-nuage $\{p_0, \cdots, p_i\}$ situé à gauche de p_i. On remarque que les points p_0 et p_i sont sur ce bord, et on appelle *enveloppe supérieure* la partie du bord de $\mathrm{Conv}\{p_0, \cdots, p_i\}$ située au-dessus de la droite passant par p_0 et p_i (p_0 et p_i compris), et *enveloppe inférieure* la partie du bord de $\mathrm{Conv}\{p_0, \cdots, p_i\}$ située au-dessous (p_0 et p_i compris). Le bord de $\mathrm{Conv}\{p_0, \cdots, p_i\}$ est donc constitué de l'union de ces deux enveloppes, après suppression des doublons de p_0 et p_i.

Par exemple, dans le cas du nuage P de la figure 4 gauche, le sous-nuage $\{p_0, p_1, p_2, p_3, p_4\}$ a pour enveloppe supérieure la séquence (p_0, p_2, p_4) et pour enveloppe inférieure la séquence (p_0, p_3, p_4), le bord de son enveloppe convexe étant donné par la séquence (p_0, p_3, p_4, p_2).

Informatiquement, les indices des sommets des enveloppes inférieure et supérieure seront stockés dans deux piles d'entiers séparées, *ei* (pour *enveloppe inférieure*) et *es* (pour *enveloppe supérieure*).

La mise à jour de l'enveloppe supérieure est illustrée dans la figure 5 : tant que le point visité (p_9 dans ce cas) et les deux points dont les indices sont situés au sommet de la pile *es* (dans l'ordre : p_8 et p_5) forme une séquence (p_9, p_8, p_5) d'orientation négative (voir la définition 1 pour rappel de l'orientation), on dépile l'indice situé au sommet de *es* (8 dans ce cas). On poursuit ce processus d'élimination jusqu'à ce que l'orientation devienne positive ou qu'il ne reste plus qu'un seul indice dans la pile. L'indice du point visité (p_9 dans ce cas) est alors inséré au sommet de *es*. La mise à jour de l'enveloppe inférieure s'opère de manière symétrique.

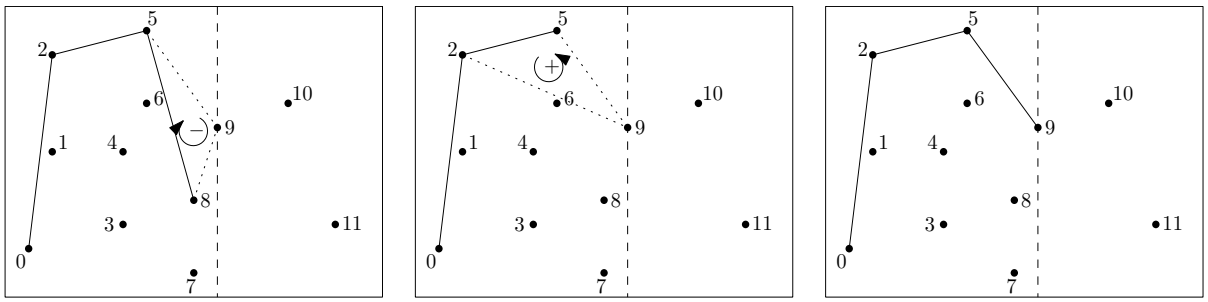


FIGURE 5 – Mise à jour de l'enveloppe supérieure lors de la visite du point p_9.

Question 10 Écrire une fonction majES(tab, es, i) qui prend en paramètre le tableau tab ainsi que la pile es et l'indice i du point à visiter, et qui met à jour l'enveloppe supérieure du sous-nuage. Le temps d'exécution de votre fonction doit être majoré par une constante fois i.

Question 11 Écrire une fonction majEI(tab, ei, i) qui effectue la mise à jour de l'enveloppe inférieure, avec le même temps d'exécution.

Question 12 Écrire maintenant une fonction convGraham(tab, n) qui prend en paramètre le tableau tab de taille $2 \times n$ représentant le nuage P, et qui effectue le balayage des points de P comme décrit précédemment. On supposera les colonnes du tableau tab déjà triées par ordre croissant d'abscisse. La fonction doit renvoyer une pile s contenant les indices des sommets du bord de $\mathrm{Conv}(P)$ triés dans l'ordre positif d'orientation, à commencer par le point p_0.

Par exemple, sur le nuage de la figure 1, le résultat de la fonction convGraham doit être la pile s contenant la suite d'incides $0, 7, 11, 10, 5, 2$ dans cet ordre, l'indice 0 se trouvant au fond de la pile s et l'indice 2 au sommet de s.

Question 13 Analyser brièvement le temps d'exécution de l'algorithme de balayage décrit précédemment, en supposant une fois encore que les points du nuage fourni en entrée sont déjà triés par abscisse croissante. En déduire que le temps d'exécution total de l'algorithme de Graham-Andrew est bien majoré par une constante fois $n \log n$.

⋆ ⋆ ⋆

ÉCOLE POLYTECHNIQUE — ÉCOLES NORMALES SUPÉRIEURES

CONCOURS D'ADMISSION 2015 **FILIÈRES PSI et PT**

ÉPREUVE D'INFORMATIQUE (XCR)

(Durée : 2 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.
Le langage de programmation sera **obligatoirement** Python.

⋆⋆⋆

Quand la taille n'est pas un problème

Notations. On désignera par $[\![n]\!]$ l'ensemble des entiers de 1 à n : $[\![n]\!] = \{1, \ldots, n\}$.

Objectif. Le but de cette épreuve est de décider s'il existe, entre deux villes données, un chemin passant par exactement k villes intermédiaires *distinctes*, dans un plan contenant au total n villes reliées par m routes. L'algorithme d'exploration naturel s'exécute en temps $O(n^k m)$. L'objectif est d'obtenir un algorithme qui s'exécute en un temps $O(f(k) \times n(n+m))$, qui croit polynomialement en la taille $(n+m)$ du problème quelle que soit la valeur de k demandée.

Complexité. La complexité, ou le temps d'exécution, d'un programme Π (fonction ou procédure) est le nombre d'opérations élémentaires (addition, multiplication, affectation, test, etc...) nécessaires à l'exécution de Π. Lorsque cette complexité dépend de plusieurs paramètres n, m et k, on dira que Π a une complexité en $O(\phi(n, m, k))$, lorsqu'il existe quatre constantes absolues A, n_0, m_0 et k_0 telles que la complexité de Π soit inférieure ou égale à $A \times \phi(n, m, k)$, pour tout $n \geqslant n_0$, $m \geqslant m_0$ et $k \geqslant k_0$.

Lorsqu'il est demandé de préciser la complexité d'un programme, le candidat devra justifier cette dernière si elle ne se déduit pas directement de la lecture du code.

Implémentation. On suppose que l'on dispose d'une fonction `creerTableau(n)` qui alloue un tableau de taille `n` indexé de 0 à `n` − 1 (les valeurs contenues dans le tableau initialement sont arbitraires). L'instruction `b = creerTableau(n)` créera un tableau de taille `n` dans la variable `b`.

On pourra ainsi créer un tableau a de p tableaux de taille q par la suite d'instructions suivante :

```
a = creerTableau(p)
for i in range(p):
    a[i] = creerTableau(q)
```

On accédera par l'instruction a[i][j] à la j-ème case du i-ème tableau contenu dans le tableau a ainsi créé. Par exemple, la suite d'instructions suivante remplit le tableau a avec les sommes des indices i et j de chaque case :

```
for i in range(p):
    for j in range(q):
        a[i][j] = i+j
```

On supposera l'existence de deux valeurs booléennes True et False.

On supposera l'existence d'une procédure : affiche(...) qui affiche le contenu de ses arguments à l'écran. Par exemple, x = 1; y = x+1; affiche("x = ",x," et y = ",y); affiche à l'écran :

```
x = 1 et y = 2
```

Dans la suite, nous distinguerons *fonction* et *procédure* : les fonctions renvoient une valeur (ou un tableau) tandis que les procédures ne renvoient aucune valeur.

La plus grande importance est donnée à la lisibilité du code produit par les candidats. Ainsi, les candidats sont encouragés à introduire des procédures ou fonctions intermédiaires lorsque cela en simplifie l'écriture.

Partie I. Préliminaires : Listes sans redondance

On souhaite stocker en mémoire une liste *non-ordonnée* d'au plus n entiers *sans redondance* (i.e. où aucun entier n'apparaît plusieurs fois). Nous nous proposons d'utiliser un tableau liste de longueur $n+1$ tel que :

— liste[0] contient le nombre d'éléments dans la liste

— liste[i] contient le i-ème élément de la liste (non-ordonnée) pour $1 \leqslant$ i $\leqslant$ liste[0]

Question 1. Écrire une fonction creerListeVide(n) qui crée, initialise et renvoie un tableau de longueur n + 1 correspondant à la liste vide ayant une capacité de n éléments.

Question 2. Écrire une fonction estDansListe(liste, x) qui renvoie True si l'élément x apparaît dans la liste représentée par le tableau liste, et renvoie False sinon.

Quelle est la complexité en temps de votre fonction dans le pire cas en fonction du nombre

maximal n d'éléments dans la liste ?

Question 3. Écrire une procédure `ajouteDansListe(liste, x)` qui modifie de façon appropriée le tableau `liste` pour y ajouter `x` si l'entier `x` n'appartient pas déjà à la liste, et ne fait rien sinon.

Quel est le comportement de votre procédure si la liste est pleine initialement ? (On ne demande pas de traiter ce cas)

Quelle est la complexité en temps de votre procédure dans le pire cas en fonction du nombre maximal n d'éléments dans la liste ?

Partie II. Création et manipulation de plans

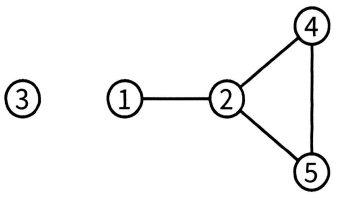
Un *plan* P est défini par : un ensemble de n villes numérotées de 1 à n et un ensemble de m routes (toutes à double-sens) reliant chacune deux villes ensemble. On dira que deux villes $x, y \in [\![n]\!]$ sont *voisines* lorsqu'elles sont reliées par une route, ce que l'on notera par $x \sim y$. On appellera *chemin* de longueur k toute suite de villes $v_1, \ldots, v_k$ telle que $v_1 \sim v_2 \sim \cdots \sim v_k$. On représentera les villes d'un plan par des ronds contenant leur numéro et les routes par des traits reliant les villes voisines (voir Fig. 1).

Structure de données. Nous représenterons tout plan P à n villes par un tableau `plan` de $(n+1)$ tableaux où :

— `plan[0]` contient un tableau à deux éléments où :
  · `plan[0][0] =` n contient le nombre de villes du plan
  · `plan[0][1] =` m contient le nombre de routes du plan

— Pour chaque ville `x` $\in [\![n]\!]$, `plan[x]` contient un tableau à n éléments représentant la liste à au plus $n-1$ éléments des villes voisines de la ville `x` dans P dans un ordre arbitraire en utilisant la structure de liste sans redondance définie dans la partie précédente. Ainsi :
  · `plan[x][0]` contient le nombre de villes voisines de `x`
  · `plan[x][1]`, ..., `plan[x][plan[x][0]]` sont les indices des villes voisines de x.

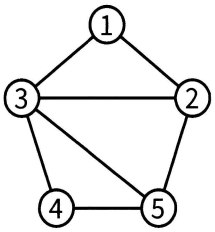
La figure 1 donne un exemple de plan et d'une représentation possible sous la forme de tableau de tableaux (les `*` représentent les valeurs non-utilisées des tableaux).



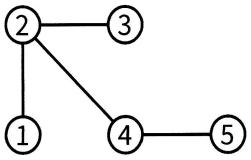

```
plan = [ [5  4],
         [1  2   *   *   *],
         [3  4   1   5   *],
         [0  *   *   *   *],
         [2  2   5   *   *],
         [2  4   2   *   *] ]
```

FIGURE 1 – Un plan à 5 villes et 4 routes et une représentation possible en mémoire sous forme d'un tableau de tableaux `plan`.

Question 4. Représenter sous forme de tableaux de tableaux les deux plans suivants :



Plan 1



Plan 2

On pourra utiliser dans la suite, les fonctions et procédures de gestion de listes définies dans la partie précédente.

Question 5. Écrire une fonction `creerPlanSansRoute(n)` qui crée, remplit et renvoie le tableau de tableaux correspondant au plan à `n` villes n'ayant aucune route.

Question 6. Écrire une fonction `estVoisine(plan, x, y)` qui renvoie `True` si les villes `x` et `y` sont voisines dans le plan codé par le tableau de tableaux `plan`, et renvoie `False` sinon.

Question 7. Écrire une procédure `ajouteRoute(plan, x, y)` qui modifie le tableau de tableaux `plan` pour ajouter une route entre les villes `x` et `y` si elle n'était pas déjà présente et ne fait rien sinon. (On prendra garde à bien mettre à jour toutes les cases concernées dans le tableau de tableaux `plan`.)

Y a-t-il un risque de dépassement de la capacité des listes ?

Question 8. Écrire une procédure `afficheToutesLesRoutes(plan)` qui affiche à l'écran la liste des routes du plan codé par le tableau de tableaux `plan` où chaque route apparaît exactement une seule fois. Par exemple, pour le graphe codé par le tableau de tableaux de la figure 1, votre procédure pourra afficher à l'écran :

```
Ce plan contient 4 route(s): (1-2) (2-4) (2-5) (4-5)
```

Quelle est la complexité en temps de votre procédure dans le pire cas en fonction de n et m ?

Partie III. Recherche de chemins arc-en-ciel

Étant données deux villes distinctes s et $t \in [\![n]\!]$, nous recherchons un chemin de s à t passant par exactement k villes intermédiaires toutes distinctes. L'objectif de cette partie et de la suivante est de construire une fonction qui va détecter en temps linéaire en $n(n+m)$, l'existence d'un tel chemin avec une probabilité indépendante de la taille du plan $n+m$.

Le principe de l'algorithme est d'attribuer à chaque ville $i \in [\![n]\!] \setminus \{s,t\}$ une couleur aléatoire codée par un entier aléatoire uniforme `couleur[`i`]` $\in \{1,\dots,k\}$ stocké dans un tableau `couleur` de taille $n+1$ (la case 0 n'est pas utilisée). Les villes s et t reçoivent respectivement les couleurs spéciales 0 et $k+1$, i.e. `couleur[`s`] = 0` et `couleur[`t`] =` k`+1`. L'objectif de cette partie est d'écrire une procédure qui détermine s'il existe un chemin de longueur $k+2$ allant de s à t dont la j-ème ville intermédiaire a reçu la couleur j. Dans l'exemple de la figure 2, le chemin

$6 \sim 7 \sim 8 \sim 3 \sim 4$ de longueur $5 = k + 2$ qui relie $s = 6$ à $t = 4$ vérifie cette propriété pour $k = 3$.

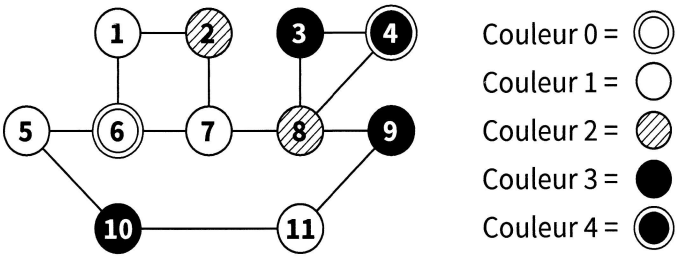


FIGURE 2 – Exemple de plan colorié pour $k = 3$, $s = 6$, $t = 4$.

On suppose l'existence d'une fonction `entierAleatoire(k)` qui renvoie un entier aléatoire uniforme entre 1 et `k` (i.e. telle que $\Pr\{\texttt{entierAleatoire(k)} = c\} = 1/\texttt{k}$ pour tout entier $c \in \{1, \ldots, \texttt{k}\}$).

Question 9. Écrire une procédure `coloriageAleatoire(plan, couleur, k, s, t)` qui prend en argument un plan de n villes, un tableau `couleur` de taille $n + 1$, un entier `k`, et deux villes s et $t \in [\![n]\!]$, et remplit le tableau `couleur` avec : une couleur aléatoire uniforme dans $\{1, \ldots, \texttt{k}\}$ choisie indépendamment pour chaque ville $i \in [\![n]\!] \setminus \{s, t\}$; et les couleurs 0 et $\texttt{k} + 1$ pour s et t respectivement.

Nous cherchons maintenant à écrire une fonction qui calcule l'ensemble des villes de couleur c voisines d'un ensemble de villes donné. Dans l'exemple de la figure 2, l'ensemble des villes de couleur 2 voisines des villes $\{1, 5, 7\}$ est $\{2, 8\}$.

Question 10. Écrire une fonction `voisinesDeCouleur(plan, couleur, i, c)` qui crée et renvoie un tableau codant la liste sans redondance des villes de couleur `c` voisines de la ville `i` dans le plan `plan` colorié par le tableau `couleur`.

Question 11. Écrire une fonction `voisinesDeLaListeDeCouleur(plan, couleur, liste, c)` qui crée et renvoie un tableau codant la liste sans redondance des villes de couleur `c` voisines d'une des villes présente dans la liste sans redondance `liste` dans le plan `plan` colorié par le tableau `couleur`.

Quelle est la complexité de votre fonction dans le pire cas en fonction de n et m ?

Question 12. Écrire une fonction `existeCheminArcEnCiel(plan, couleur, k, s, t)` qui renvoie `True` s'il existe dans le plan `plan`, un chemin $s \sim v_1 \sim \cdots \sim v_k \sim t$ tel que `couleur[`v_j`] = `j pour tout $j \in \{1, \ldots, k\}$; et renvoie `False` sinon.

Quelle est la complexité de votre fonction dans le pire cas en fonction de k, n et m ?

Partie IV. Recherche de chemin passant par exactement k villes intermédiaires distinctes

Si les couleurs des villes sont choisies aléatoirement et uniformément dans $\{1, \ldots, k\}$, la probabilité que j soit la couleur de la j-ème ville d'une suite fixée de k villes, vaut $1/k$ indépendamment pour tout j. Ainsi, étant données deux villes distinctes s et $t \in [\![n]\!]$, s'il existe dans le plan `plan` un chemin de s à t passant par exactement k villes intermédiaires toutes distinctes et si le coloriage `couleur` est choisi aléatoirement conformément à la procédure `coloriageAleatoire(plan, couleur, k, s, t)`, la procédure `existeCheminArcEnCiel(plan, couleur, k, s, t)` répond `True` avec probabilité au moins k^{-k} ; et répond toujours `False` sinon. Ainsi, si un tel chemin existe, la probabilité qu'une parmi k^k exécutions indépendantes de `existeCheminArcEnCiel` réponde `True` est supérieure ou égale à $1 - (1 - k^{-k})^{k^k} = 1 - \exp(k^k \ln(1 - k^{-k})) \geqslant 1 - 1/e > 0$ (admis).

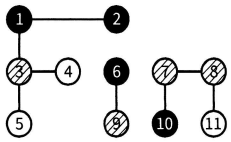
Question 13. Écrire une fonction `existeCheminSimple(plan, k, s, t)` qui renvoie `True` avec probabilité au moins $1 - 1/e$ s'il existe un chemin de s à t passant par exactement k villes intermédiaires toutes distinctes dans le plan `plan` ; et renvoie toujours `False` sinon.

Quelle est sa complexité en fonction de k, n et m dans le pire cas ? Exprimez-la sous la forme $O(f(k) \times g(n, m))$ pour f et g bien choisies.

Question 14. Expliquer comment modifier votre programme pour renvoyer un tel chemin s'il est détecté avec succès.

Note : L'algorithme présenté partiellement dans les parties III et IV de ce sujet est dû à Dániel Marx. Pour en savoir plus, on pourra consulter sa page web (`http://www.cs.bme.hu/~dmarx`) et plus particulièrement les transparents sur la complexité paramétrée (Part 1: Algorithmic techniques, page 66 et suivantes).

En utilisant une meilleure structure de liste sans redondance que celle proposée dans le sujet, on peut facilement obtenir une complexité qui soit linéaire en la taille $(n+m)$ du problème quelle que soit la valeur de k demandée.



Tests de validation d'une imprimante

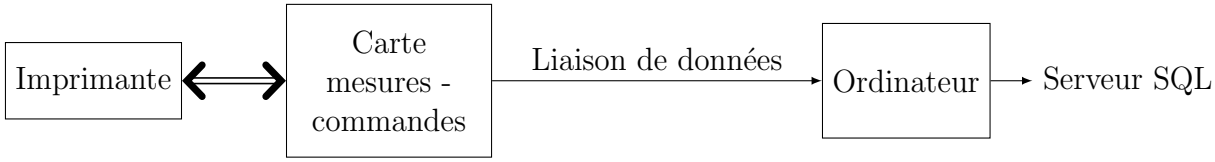
Le sujet comporte des questions de programmation. *Le langage à utiliser est* `Python` *dans les parties I à IV. Dans la partie V, on demande d'utiliser* `Scilab`.

Introduction

Les imprimantes sont des systèmes mécatroniques fabriqués en grande série dans des usines robotisées. Pour améliorer la qualité des produits vendus, il a été mis en place différents tests de fin de chaîne pour valider l'assemblage des produits. Pour un de ces tests, un opérateur connecte l'outil de test sur la commande du moteur de déplacement de la tête d'impression et sur la commande du moteur d'avance papier. Une autre connexion permet de récupérer les signaux issus des capteurs de position.

Différentes commandes et mesures sont alors exécutées. Ces mesures sont envoyées par liaison de données sous la forme d'une suite de caractères ASCII vers un ordinateur.

Cet ordinateur va effectuer différentes mesures pour valider le fonctionnement de l'électromécanique de l'imprimante. L'ensemble des mesures et des analyses est sauvegardé dans un fichier texte. Cette sauvegarde s'effectue dans une base de données et, afin de minimiser l'espace occupé, les fichiers sont compressés. La base de données permet à l'entreprise d'améliorer la qualité de la production par diverses études statistiques.



Rappels et définitions :

- Une liste commence par un crochet `[` et se termine par un crochet `]`. Les éléments d'une liste sont ordonnés (indexés).
- On pourra utiliser la surcharge de l'opérateur + : `['a']+['b']=['a','b']`.
- Un dictionnaire définit une relation une à une entre des clés et des valeurs. Celui-ci se note entre accolades `{}`.
- Un tuple est une collection d'éléments ordonnés comme dans une liste mais une fois le tuple créé, ses éléments ne peuvent pas être modifiés indépendamment les uns des autres. Il se note entre parenthèses `()`. Exemple : (4,'e',[1,3])

I Réception des données issues de la carte d'acquisition

Les mesures sont faites à l'aide de convertisseurs analogique/numérique (CAN). Le résultat de conversion est codé sur 10 bits signés en complément à 2.

Q1. Quelle plage de valeurs entières pourra prendre le résultat de la conversion ?

Q2. Si on considère que les valeurs analogiques converties s'étendent en pleine échelle de -5V à 5V, quelle est la résolution de la mesure en volt ?

Une liaison série asynchrone permet la communication entre la carte de commande/acquisition et le PC. Les échantillons correspondant à une mesure sont envoyés par la carte électronique sous la forme d'une trame (suite de caractères ASCII). Cette suite de caractères se présente sous la forme suivante :

- un entête qui permet d'identifier la mesure sur un caractère ('U' tension moteur, 'I 'courant moteur, 'P' position absolue),
- le nombre de données envoyées (3 caractères),
- les données constituées des mesures brutes issues de la conversion analogique-numérique, chaque mesure étant codée à l'aide du caractère '+' ou '-' suivi de 3 caractères pour la valeur absolue,
- un checksum, somme des valeurs absolues des données précédentes modulo 10000 sur 4 caractères. Le nombre de données transmises n'est pas inclus dans le checksum.

Exemple : Mesure de la tension sur 5 échantillons.

Caractères reçus : 'U' '0' '0' '5' '+' '0' '1' '2' '+' '0' '0' '4' '-' '0' '2' '3' '-' '0' '0' '2' '+' '0' '4' '2' '0' '0' '8' '3' $\rightarrow t$

La commande `carac_recus=com.read(nbre_car)` permet de récupérer *nbre_car* caractères reçus sous la forme d'une chaîne de caractères. En supposant que les caractères reçus correspondent à l'exemple précédent, après l'exécution de `carac_recus=com.read(5)`, la variable `carac_recus` contiendra la chaîne "U005+".

Après une nouvelle exécution de `carac_recus=com.read(3)`, la variable `carac_recus` contiendra la chaîne "012".

Q3. Écrire une fonction `lect_mesures()` en langage `Python` qui retourne une liste contenant : le type de la mesure ('U','I' ou 'P'), une liste contenant l'ensemble des valeurs des mesures reçues et le checksum. Exemple : ['U',[12,4,-23,-2,42],83]. Cette fonction doit attendre que le caractère d'entête reçu soit correct ('U','I' ou 'P') avant de réaliser le stockage des informations dans la liste qui sera retournée.

Q4. On suppose que toutes les mesures sont disponibles dans la liste `mesures[]`, et le checksum reçu dans la variable `CheckSum`. Écrire une fonction `check(mesure,CheckSum)` en langage `Python` qui retourne True si la transmission présente un checksum valide et False sinon.

Q5. Les mesures étant dans la liste `mesures`, écrire une fonction `affichage(mesure)` en langage `Python` qui produit un affichage graphique comme représenté en Figure 1, sachant que la résolution de la conversion analogique-numérique du courant est de 4 mA et que les mesures ont été effectuées toutes les 2 ms. On ne demande pas de légender les axes ni de donner un titre à la figure. On suppose que les bibliothèques nécessaires ont été importées.

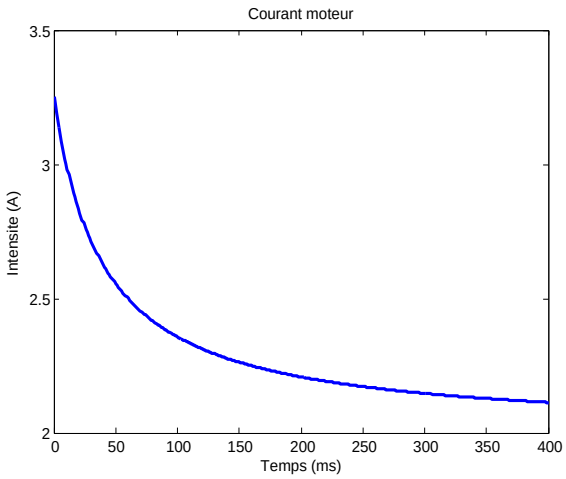


FIGURE 1 – Affichage de la mesure.

II Analyse des mesures

La suite des valeurs de mesure du courant en Ampère du moteur de la tête d'impression est contenue dans une liste. Les mesures ont été effectuées toutes les 2 ms. Ces mesures sont disponibles dans la liste `mesure`. Deux traitements permettent de valider le fonctionnement de l'imprimante :

- Le calcul de la valeur moyenne I_{moy} du signal $I(t)$ sur la durée d'acquisition.

$$I_{moy} = \frac{1}{t_{final}} \int_0^{t_{final}} I(t)\,\mathrm{d}t$$

- Le calcul de l'écart type I_{ec} du signal $I(t)$ sur la durée d'acquisition.

$$I_{ec} = \sqrt{\frac{1}{t_{final}} \int_0^{t_{final}} (I(t) - I_{moy})^2\,\mathrm{d}t}$$

Q6. Écrire une fonction en langage `Python` qui retourne I_{moy} après l'avoir calculée par la méthode des trapèzes.

Q7. Écrire une fonction en langage `Python` qui retourne I_{ec} après l'avoir calculé en utilisant la fonction précédente.

III Base de données

Une représentation simplifiée de deux tables de la base de données qu'on souhaite utiliser est donnée ci-dessous :

testfin

nSerie	dateTest	...	Imoy	Iec	...	fichierMes
230-588ZX2547	2012-04-22 14-25-45		0.45	0.11		mesure31025.csv
230-588ZX2548	2012-04-22 14-26-57		0.43	0.12		mesure41026.csv
⋮	⋮	⋮	⋮	⋮	⋮	⋮

production

Num	nSerie	dateProd	type
20	230-588ZX2547	2012-04-22 15-52-12	JETDESK-1050
21	230-588ZX2549	2012-04-22 15-53-24	JETDESK-3050
⋮	⋮	⋮	⋮

Après son assemblage et avant les différents tests de validation, un numéro de série unique est attribué à chaque imprimante. A la fin des tests de chaque imprimante, les résultats d'analyse ainsi que le fichier contenant l'ensemble des mesures réalisées sur l'imprimante sont rangés dans la table `testfin`. Lorsqu'une imprimante satisfait les critères de validation, elle est enregistrée dans la table `production` avec son numero de série, la date et l'heure de sortie de production ainsi que son type.

Q8. Rédiger une requête SQL permettant d'obtenir les numéros de série des imprimantes ayant une valeur de Imoy comprise strictement entre deux bornes Imin et Imax.

Q9. Rédiger une requête SQL permettant d'obtenir les numéros de série, la valeur de l'écart type et le fichier de mesures des imprimantes ayant une valeur de Iec strictement inférieure à la valeur moyenne de la colonne Iec.

Q10. Rédiger une requête SQL qui permettra d'extraire à partir de la table `testfin` le numéro de série et le fichier de mesures correspondant aux imprimantes qui n'ont pas été validées en sortie de production.

IV Préparation du fichier texte avant envoi : la compression

Le fichier de résultat va être stocké sous la forme d'un fichier binaire. Une des étapes de l'algorithme de compression utilise le codage de Huffman.

IV.1 Présentation :

Le codage de Huffman utilise un code à longueur variable pour représenter un symbole de la source (par exemple un caractère dans un fichier). Le code est déterminé à partir d'une estimation des probabilités d'apparition des symboles de source, un code court étant associé aux symboles de source les plus fréquents. La première étape du codage de Huffman consiste à créer un

dictionnaire contenant la liste des caractères présents dans le texte, associé à leur fréquence dans ce texte. Exemple : "AABCDCCEF" donnera `{'A':2,'B':1,'C':3,'D':1,'E':1,'F':1}`. La deuxième étape consiste à construire un arbre de Huffman qui permet ensuite de coder chaque caractère.

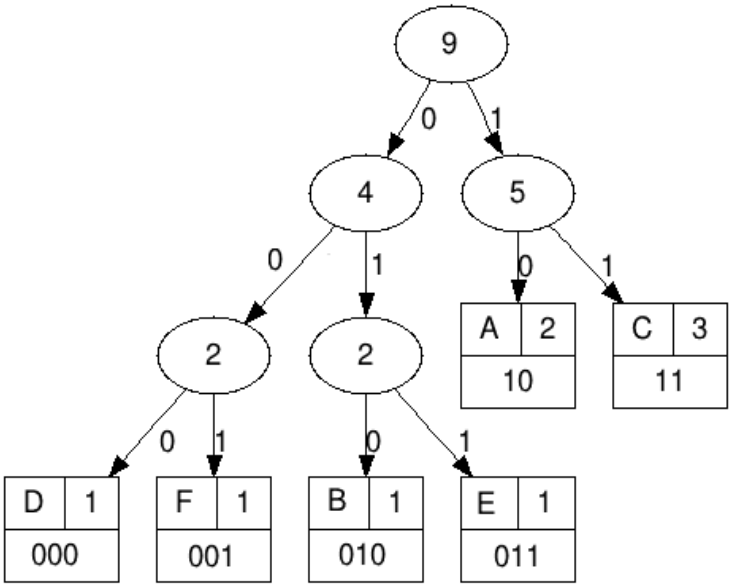


FIGURE 2 – Arbre de Huffman.

Notre texte "AABCDCCEF" de 9 caractères ASCII (72 bits) sera ainsi codé en binaire "10 10 010 11 000 11 11 011 001" (22 bits).

Chaque caractère constitue une des feuilles de l'arbre à laquelle on associe un poids valant son nombre d'occurrences. Puis l'arbre est créé suivant un principe simple : on associe à chaque fois les deux nœuds de plus faibles poids pour créer un nœud dont le poids équivaut à la somme des poids de ses fils jusqu'à n'en avoir plus qu'un, la racine. On associe ensuite par exemple le code 0 à la branche de gauche et le code 1 à la branche de droite.

Pour obtenir le code binaire de chaque caractère, on descend sur la Figure 2 de la racine jusqu'aux feuilles en ajoutant à chaque fois au code un 0 ou un 1 selon la branche suivie.

Pour pouvoir être décodé par l'ordinateur, l'arbre doit aussi être transmis.

Le code `Python` permettant de construire un arbre de Huffman et de coder chaque caractère est fourni en *annexe*.

Les feuilles de l'arbre de Huffman (leaf) sont codées sous la forme de tuple avec comme premier élément le caractère et comme deuxième élément le poids. Pour l'arbre donné en exemple en Figure 2, on aura 6 tuples : `('A',2), ('B',1), ('C',3), ('D',1), ('E',1) et ('F',1)`.

Q11. La documentation de l'instruction `isinstance(object,classinfo)` décrit le fonctionnement suivant : "Return True if the object is an instance of the classinfo argument. If object is not a class instance of the given type, the function returns False." Décrire succinctement le rôle des fonctions suivantes et indiquer le type de la variable retournée :

- `test()`
- `get1()`
- `get2()`

IV.2 Analyse des fonctions `make_huffman_tree()` et `freq_table()`

Q12. Donner le contenu des variables `node1` et `node2` suite à l'exécution des commandes
`node1=make_huffman_tree(('F',1),('E',1))`.
`node2=make_huffman_tree(('D',1),('B',1))`.

Q13. De même, donner le contenu de la variable `node3` suite à l'éxécution de la commande `node3=make_huffman_tree(node1,node2)`.

Q14. Donner le contenu de la variable `f` suite à l'éxécution de la commande `f=freq_table('AABBCB')`.

IV.3 Analyse de la fonction `insert_item()`

Cette fonction permet d'insérer un nœud ou une feuille dans une liste de nœuds et de feuilles triés par poids croissant, en conservant l'ordre de tri.

Q15. Quelle est la particularité de cette fonction ?

Q16. Montrer qu'un invariant d'itération est "tous les éléments de la sous liste `lst[0 à pos-1]` ont un poids inférieur à celui de la variable item". On démontrera qu'il est vrai à l'initialisation, puis à chaque itération, sachant que cette fonction doit être appelée avec l'argument d'entrée `pos=0` produisant ainsi une liste vide.

IV.4 Analyse de `build_huffman_tree()`

Q17. D'après la description précédente et le résultat de la question Q13, commenter la fonction de manière à expliquer comment l'arbre de Huffman est construit. On demande de proposer des rédactions pour les commentaires correspondant aux lignes ## 5, ## 6, ## 7, ## 8 dans la fonction `build_huffman_tree()`.

Q18. Donner le nombre de tests dans le meilleur des cas et dans le pire des cas effectués dans la fonction `insert_item()` pour une liste `lst` contenant n éléments. Les résultats devront être justifiés.

Q19. Donner la complexité en temps dans le meilleur des cas et dans le pire des cas de la fonction `build_huffman_tree` pour une liste `lst` contenant n éléments en tenant compte de la fonction `insert_item`. On négligera le coût en complexité de la fonction `lst.sort`. Les résultats devront être justifiés.

Q20. Donner le contenu de la variable `htree` après l'éxécution de la commande `htree=build_huffman_tree("ZBBCB")`.

Q21. Dessiner l'arbre de Huffman correspondant à `htree` de la question précédente en vous inspirant de la Figure 2.

V Simulation physique

Une possible source de pannes dans chaque imprimante est la défectuosité d'un moteur particulier. On suppose disposer d'enregistrements du signal d'entrée e et du signal de sortie s de cet élément. Ces variables sont supposées satisfaire une équation différentielle linéaire du premier ordre

$$\frac{d}{dt}s = -k\frac{s-e}{10}$$

où k est un paramètre réel strictement positif. On va chercher à vérifier le bon fonctionnement du moteur en analysant des simulations.

Q22. Écrire en langage Scilab une fonction qui calcule de manière approchée la solution de l'équation différentielle précédente pour le signal $e(t) = \sin(t)$, avec $s(0) = 0$, sur l'intervalle $t \in [0,10]$ pour une valeur quelconque de k. La fonction doit retourner une estimation de $s(t)$ aux instants [0, 0.1, 0.2,..., 10].

Q23. Trois valeurs sont possibles pour le paramètre k : 0.5, 1.1 et 2. On décide de déclarer le moteur défectueux si aucune de ces valeurs ne permet d'expliquer l'enregistrement s réalisé expérimentalement aux instants [0, 0.1, 0.2,..., 10]. Définir un code Scilab utilisant la fonction précédemment définie pour réaliser une validation ou une invalidation de la défectuosité du moteur suivant un critère à proposer.

Annexe

Huffman tree :

```
##########################################################################
### Auto Generation of Huffman Trees
##########################################################################

def make_leaf(symbol, weight):
    return (symbol, weight)

def test(x):
    return isinstance(x, tuple) and \
            len(x) == 2 and \
            isinstance(x[0], str) and \
            isinstance(x[1], int)

def get1(x):
    return x[0]

def get2(x):
    return x[1]

def get3(huff_tree):
    return huff_tree[0]

def get4(huff_tree):
    return huff_tree[1]

def get5(huff_tree):
    if test(huff_tree):
        return [get1(huff_tree)] #Attention le symbole est dans une liste
    else:
        return huff_tree[2]

def get6(huff_tree):
    if test(huff_tree):
        return get2q(huff_tree)
    else:
        return huff_tree[3]

def make_huffman_tree(left_branch, right_branch):
    return [left_branch,
            right_branch,
            get5(left_branch) + get5(right_branch),
            get6(left_branch) + get6(right_branch)]

### entr\'{e}e : string txt et retourne un dictionnaire contenant chaque
### caractere avec son occurrence dans le string txt
def freq_table(txt):
    ftble = {}
    for c in txt:
        if c not in ftble:
            ftble[c] = 1
        else:
            ftble[c] += 1
    return ftble
```

```
### Fonction de comparaison qui permet de comparer
### les noeuds de Huffman entre eux selon leur occurrence.
def freq_cmp(node1, node2):
    freq1, freq2 = get6(node1), get6(node2)
    if freq1 < freq2:
        return -1
    elif freq1 > freq2:
        return 1
    else:
        return 0

### ins\`{e}re un item \`{a} sa place appropri\'{e}e dans une liste de
   noeuds et feuilles
def insert_item(item, lst, pos):
    if pos == len(lst):
        lst.append(item)
    elif freq_cmp(item, lst[pos]) <= 0 :
        lst.insert(pos, item)
    else:
        insert_item(item, lst, pos+1)
    return

### Construction de l'arbre de Huffman
def build_huffman_tree(txt):
    ### 1. construire une table des occurrences \`{a} partir de txt
    ftble = freq_table(txt)
    ### 2. obtenir la liste des feuilles de Huffman
    lst = list(ftble.items())
    ### 3. classer leaf_lst par occurrence de la plus petite \`{a} la plus
    ###    grande
    lst.sort(key=lambda lst: lst[1])
    ### 4. construction de l'arbre de huffman
    if len(lst) == 0:
        return None
    elif len(lst) == 1:
        return lst[0]
    else:
        ## 5. ____________________________________________________________
          while len(lst) > 2:
            ## 6. _________________________________________________________
            new_node = make_huffman_tree(lst[0], lst[1])
            ## 7. _________________________________________________________
            del lst[0]
            del lst[0]
            ## 8. _________________________________________________________
            insert_item(new_node, lst, 0)
        else:
            return make_huffman_tree(lst[0], lst[1])
```

Manual and Auto Generation of Huffman Trees in Python, Vladimir Kulyukin http://vkedco.blogspot.fr/2012/02/manual-and-auto-generation-of-huffman.html
Huffman tree generator http://huffman.ooz.ie/

CONCOURS CENTRALE•SUPÉLEC

Informatique

MP,PC,PSI,TSI

3 heures Calculatrices autorisées

2015

Autour de la dynamique gravitationnelle

Modéliser les interactions physiques entre un grand nombre de constituants mène à l'écriture de systèmes différentiels pour lesquels, en dehors de quelques situations particulières, il n'existe aucune solution analytique. Les problèmes de dynamique gravitationnelle et de dynamique moléculaire en sont deux exemples. Afin d'analyser le comportement temporel de tels systèmes, l'informatique peut apporter une aide substantielle en permettant leur simulation numérique. L'objet de ce sujet, composé de quatre parties, est l'étude de solutions algorithmiques en vue de simuler une dynamique gravitationnelle afin, par exemple, de prédire une éclipse ou le passage d'une comète.

Les programmes doivent être écrits en langage python et les requêtes de base de données en langage SQL. Les candidats sont libres de définir et de programmer toute fonction auxiliaire dont ils estiment avoir besoin pour répondre aux questions posées. Ils veilleront dans ce cas à définir précisément, le rôle de chaque fonction introduite, ses paramètres et son résultat. Ils peuvent également utiliser librement les fonctions de la bibliothèque standard Python, en particulier celles du module `math`.

Lorsque le sujet demande l'écriture d'une fonction python, la réponse doit commencer par l'entête de la fonction (instruction `def`*). D'autre part, si le sujet précise que la fonction prend un paramètre d'un certain type ou qui répond à une certaine condition, la fonction n'a pas à vérifier la conformité de l'argument reçu.*

La lisibilité des codes produits, tant en python qu'en SQL, est un élément important d'appréciation.

I Quelques fonctions utilitaires

I.A – Donner la valeur des expressions python suivantes :

I.A.1) `[1, 2, 3] + [4, 5, 6]`

I.A.2) `2 * [1, 2, 3]`

I.B – Écrire une fonction python `smul` à deux paramètres, un nombre et une liste de nombres, qui multiple chaque élément de la liste par le nombre et renvoie une nouvelle liste : `smul(2, [1, 2, 3])` → `[2, 4, 6]`.

I.C – ***Arithmétique de listes***

I.C.1) Écrire une fonction python `vsom` qui prend en paramètre deux listes de nombres de même longueur et qui renvoie une nouvelle liste constituée de la somme terme à terme de ces deux listes :
`vsom([1, 2, 3], [4, 5, 6])` → `[5, 7, 9]`.

I.C.2) Écrire une fonction python `vdif` qui prend en paramètre deux listes de nombres de même longueur et qui renvoie une nouvelle liste constituée de la différence terme à terme de ces deux listes (la première moins la deuxième) : `vdif([1, 2, 3], [4, 5, 6])` → `[-3, -3, -3]`.

II Étude de schémas numériques

Soient y une fonction de classe C^2 sur $\mathbb{R}$ et $t_{\min}$ et $t_{\max}$ deux réels tels que $t_{\min} < t_{\max}$. On note I l'intervalle $[t_{\min}, t_{\max}]$. On s'intéresse à une équation différentielle du second ordre de la forme :

$$\forall t \in I \qquad y''(t) = f(y(t)) \tag{II.1}$$

où f est une fonction donnée, continue sur $\mathbb{R}$. De nombreux systèmes physiques peuvent être décrits par une équation de ce type.

On suppose connues les valeurs $y_0 = y(t_{\min})$ et $z_0 = y'(t_{\min})$. On suppose également que le système physique étudié est conservatif. Ce qui entraine l'existence d'une quantité indépendante du temps (énergie, quantité de mouvement, ...), notée E, qui vérifie l'équation (II.2) où $g' = -f$.

$$\forall t \in I \qquad \frac{1}{2}y'(t)^2 + g(y(t)) = E \tag{II.2}$$

II.A – ***Mise en forme du problème***

Pour résoudre numériquement l'équation différentielle (II.1), on introduit la fonction $z : I \to \mathbb{R}$ définie par $\forall t \in I, z(t) = y'(t)$.

II.A.1) Montrer que l'équation (II.1) peut se mettre sous la forme d'un système différentiel du premier ordre en $z(t)$ et $y(t)$, noté (S).

II.A.2) Soit n un entier strictement supérieur à 1 et $J_n = [\![0, n-1]\!]$. On pose $h = \dfrac{t_{\max} - t_{\min}}{n-1}$ et $\forall i \in J_n$, $t_i = t_{\min} + ih$. Montrer que, pour tout entier $i \in [\![0, n-2]\!]$,

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} z(t)\,\mathrm{d}t \qquad \text{et} \qquad z(t_{i+1}) = z(t_i) + \int_{t_i}^{t_{i+1}} f(y(t))\,\mathrm{d}t \tag{II.3}$$

La suite du problème exploite les notations introduites dans cette partie et présente deux méthodes numériques dans lesquelles les intégrales précédentes sont remplacées par une valeur approchée.

II.B – Schéma d'Euler explicite

Dans le schéma d'Euler explicite, chaque terme sous le signe intégrale est remplacé par sa valeur prise en la borne inférieure.

II.B.1) Dans ce schéma, montrer que les équations (II.3) permettent de définir deux suites $(y_i)_{i\in J_n}$ et $(z_i)_{i\in J_n}$, où y_i et z_i sont des valeurs approchées de $y(t_i)$ et $z(t_i)$. Donner les relations de récurrence permettant de déterminer les valeurs de y_i et z_i connaissant y_0 et z_0.

II.B.2) Écrire une fonction `euler` qui reçoit en argument les paramètres qui vous semblent pertinents et qui renvoie deux listes de nombres correspondant aux valeurs associées aux suites $(y_i)_{i\in J_n}$ et $(z_j)_{i\in J_n}$. Vous justifierez le choix des paramètres transmis à la fonction.

II.B.3) Pour illustrer cette méthode, on considère l'équation différentielle $\forall t \in I, y''(t) = -\omega^2 y(t)$ dans laquelle ω est un nombre réel.

a) Montrer qu'on peut définir une quantité E, indépendante du temps, vérifiant une équation de la forme (II.2).

b) On note E_i la valeur approchée de E à l'instant t_i, $i \in J_n$, calculée en utilisant les valeurs approchées de $y(t_i)$ et $z(t_i)$ obtenues à la question II.B.1. Montrer que $E_{i+1} - E_i = h^2\omega^2 E_i$.

c) Qu'aurait donné un schéma numérique qui satisfait à la conservation de E ?

d) En portant les valeurs de y_i et z_i sur l'axe des abscisses et l'axe des ordonnées respectivement, quelle serait l'allure du graphe qui respecte la conservation de E ?

e) La mise en œuvre de la méthode d'Euler explicite génère le résultat graphique donné figure 1 à gauche. Dans un système d'unités adapté, les calculs ont été menés en prenant $y_0 = 3$, $z_0 = 0$, $t_{\texttt{min}} = 0$, $t_{\texttt{max}} = 3$, $\omega = 2\pi$ et $n = 100$.

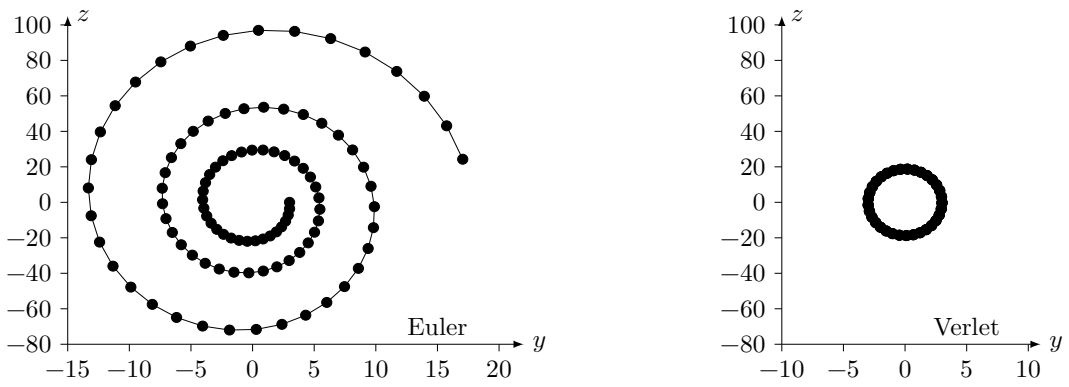


Figure 1

En quoi ce graphe confirme-t-il que le schéma numérique ne conserve pas E ? Pouvez-vous justifier son allure ?

II.C – Schéma de Verlet

Le physicien français Loup Verlet a proposé en 1967 un schéma numérique d'intégration d'une équation de la forme (II.1) dans lequel, en notant $f_i = f(y_i)$ et $f_{i+1} = f(y_{i+1})$, les relations de récurrence s'écrivent

$$y_{i+1} = y_i + hz_i + \frac{h^2}{2} f_i \qquad \text{et} \qquad z_{i+1} = z_i + \frac{h}{2}\left(f_i + f_{i+1}\right)$$

II.C.1) Écrire une fonction `verlet` qui reçoit en argument les paramètres qui vous semblent pertinents et qui renvoie deux listes de nombres correspondant aux valeurs associées aux suites $(y_i)_{i\in J_n}$ et $(z_j)_{i\in J_n}$.

II.C.2) On reprend l'exemple de l'oscillateur harmonique (question II.B.3) et on compare les résultats obtenus à l'aide des schémas d'Euler et de Verlet.

a) Montrer que dans le schéma de Verlet, on a $E_{i+1} - E_i = O(h^3)$.

b) La mise en œuvre du schéma de Verlet avec les mêmes paramètres que ceux utilisés au II.B.3e donne le résultat de la figure 1 à droite. Interpréter l'allure de ce graphe.

c) Que peut-on conclure sur le schéma de Verlet ?

III Problème à N corps

On s'intéresse à présent à la dynamique d'un système de N corps massifs en interaction gravitationnelle. Dans la suite, les corps considérés sont assimilés à des points matériels P_j de masses m_j où $j \in [\![0, N-1]\!]$, $N \geqslant 2$ étant un entier positif donné. Le mouvement de ces points est étudié dans un référentiel galiléen muni d'une base orthonormée. L'interaction entre deux corps j et k est modélisée par la force gravitationnelle. L'action exercée par le corps k sur le le corps j est décrite par la force $\overrightarrow{F}_{k/j} = G\dfrac{m_j m_k}{r_{jk}^3}\overrightarrow{P_jP_k}$ où r_{jk} est la distance séparant les corps j et k ($r_{jk} = \|\overrightarrow{P_jP_k}\|$) et $G = 6{,}67 \times 10^{-11}$ N·m^2·kg^{-2} la constante de gravitation universelle.

À tout instant t_i avec $i \in [\![0, n]\!]$, chaque corps de masse m_j est repéré par ses coordonnées cartésiennes (x_{ij}, y_{ij}, z_{ij}) et les composantes de son vecteur vitesse $(v_{xij}, v_{yij}, v_{zij})$ dans le référentiel de référence.

Trois listes sont utilisées pour représenter ce système en python

- `masse` conserve les masses de chaque corps : `masse[j] =` m_j ;
- `position` contient les positions successives de chaque corps : `position[i][j] = [`x_{ij}, y_{ij}, z_{ij}`]` ;
- `vitesse` mémorise les vitesses successives de chaque corps : `vitesse[i][j] = [`$v_{xij}, v_{yij}, v_{zij}$`]`.

L'objet de la suite du problème est de construire ces listes en mettant en œuvre l'algorithme de Verlet.

III.A – Position du problème

III.A.1) Exprimer la force $\overrightarrow{F}_j$ exercée sur le corps P_j par l'ensemble des autres corps P_k, avec $k \neq j$.

III.A.2) Écrire une fonction python `force2(m1, p1, m2, p2)` qui prend en paramètre les masses (`m1` et `m2` en kilogrammes) et les positions (`p1` et `p2`, sous forme de listes de trois coordonnées cartésiennes en mètres) de deux corps 1 et 2 et qui renvoie la valeur de la force exercée par le corps 2 sur le corps 1, sous la forme d'une liste à trois éléments représentant les composantes de la force dans la base de référence, en newtons.

III.A.3) Écrire une fonction `forceN(j, m, pos)` qui prend en paramètre l'indice `j` d'un corps, la liste des masses des N corps du système étudié ainsi que la liste de leurs positions et qui renvoie $\overrightarrow{F}_j$, la force exercée par tous les autres corps sur le corps `j`, sous la forme d'une liste de ses trois composantes cartésiennes.

III.B – Approche numérique

III.B.1) Expliciter la structure et la signification de `position[i]` et `vitesse[i]`.

III.B.2) Écrire une fonction `pos_suiv(m, pos, vit, h)` qui prend en paramètres la liste des masses des N corps du système étudié (en kilogrammes), la liste de leurs positions (en mètres) à l'instant t_i, la liste de leurs vitesses (en mètres par seconde) au même instant et le pas d'intégration h (en secondes) et qui renvoie la liste des positions des N corps à l'instant t_{i+1} calculées en utilisant le schéma de Verlet.

III.B.3) Écrire une fonction `etat_suiv(m, pos, vit, h)` qui prend les mêmes paramètres que la fonction `pos_suiv` et qui renvoie la liste des positions (en mètres) et la liste des vitesses (en m/s) des N corps à l'instant t_{i+1} calculées en utilisant le schéma de Verlet.

III.B.4) En notant τ_N la durée des calculs pour un nombre N de corps, la mise en œuvre de la fonction `etat_suiv` a donné le résultat graphique de la figure 2 où on a porté $\ln(N)$ en abscisse et $\ln(\tau_N)$ en ordonnée.

a) Quelle relation simple peut-on établir entre $\ln(\tau_N)$ et $\ln(N)$ à partir de la figure 2 ?

b) Quelle hypothèse peut-on émettre quant à la complexité de l'algorithme étudié ?

III.B.5)

a) Estimer la complexité temporelle de la fonction `etat_suiv` sous la forme $O(N^\alpha)$.

b) Comparer avec le résultat obtenu à la question III.B.4.

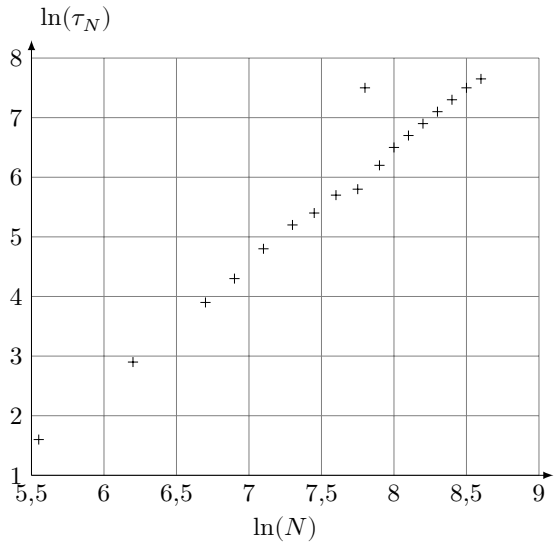


Figure 2

IV Exploitation d'une base de données

À partir de mesures régulièrement effectuées par différents observatoires, une base de données des caractéristiques et des états des corps célestes de notre Système solaire est maintenue à jour. L'objectif de cette partie est d'extraire de cette base de données les informations nécessaires à la mise en œuvre des fonctions développées dans les parties précédentes, puis de les utiliser pour prévoir les positions futures des différentes planètes. Les données à extraire sont les masses des corps étudiés et leurs états (position et vitesse) à l'instant $t_{\min}$ du début de la simulation.

Une version simplifiée, réduite à deux tables, de la base de données du Système solaire est donnée figure 3. Les masses sont exprimées en kilogrammes, les distances en unités astronomiques ($1\,\text{au} = 1{,}5 \times 10^{11}\,\text{m}$) et les vitesses en kilomètres par seconde. Le référentiel utilisé pour exprimer les composantes des positions et des vitesses est galiléen, orthonormé et son centre est situé à proximité du Soleil.

CORPS

id_corps	nom	masse
⋮	⋮	⋮

ETAT

id_corps	datem	x	y	z	vx	vy	vz
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 3 Schéma de la base de données

La table `CORPS` répertorie les corps étudiés, elle contient les colonnes
- `id_corps` (clé primaire) entier identifiant chaque corps ;
- `nom`, chaine de caractères, désigne le nom usuel du corps ;
- `masse` de type flottant, contient la masse du corps.

La table `ETAT` rassemble l'historique des états successifs (positions et vitesses) des corps étudiés. Elle est constituée de huit colonnes :
- `id_corps` de type entier, identifie le corps concerné ;
- `datem` est la date de la mesure, sous forme d'un entier donnant le nombre de secondes écoulées depuis un instant d'origine ;
- trois colonnes de type flottant pour les composantes de la position `x`, `y`, `z` ;
- trois colonnes de type flottant pour les composantes de la vitesse `vx`, `vy`, `vz`.

IV.A – Écrire une requête SQL qui renvoie la liste des masses de tous les corps étudiés.

IV.B – Les états des différents corps ne sont pas forcément tous déterminés exactement au même instant. Nous allons assimiler l'état initial (à la date $t_{\min}$) de chaque corps à son dernier état connu antérieur à $t_{\min}$.

Dans toute la suite, on supposera que la valeur de $t_{\min}$, sous le format utilisé dans la table `ETAT`, est accessible à toute requête SQL via l'expression `tmin()`.

IV.B.1) On souhaite d'abord vérifier que tous les corps étudiés disposent d'un état connu antérieur à `tmin()`. Le nombre de corps présents dans la base est obtenu grâce à la requête `SELECT count(*) FROM corps`. Écrire une requête SQL qui renvoie le nombre de corps qui ont au moins un état connu antérieur à `tmin()`.

IV.B.2) Écrire une requête SQL qui renvoie, pour chaque corps, son identifiant et la date de son dernier état antérieur à `tmin()`.

IV.B.3) Le résultat de la requête précédente est stocké dans une nouvelle table `date_mesure` à deux colonnes :
- `id_corps` de type entier, contient l'identifiant du corps considéré ;
- `date_der` de type entier, correspond à la date du dernier état connu du corps considéré, antérieur à `tmin()`.

Pour simplifier la simulation, on décide de négliger l'influence des corps ayant une masse strictement inférieure à une valeur fixée `masse_min()` et de ne s'intéresser qu'aux corps situés dans un cube, centré sur l'origine du référentiel de référence et d'arête `arete()` donnée. Les faces de ce cube sont parallèles aux plans formés par les axes du référentiel de référence.

Écrire une requête SQL qui renvoie la masse et l'état initial (sous la forme `masse, x, y, z, vx, vy, vz`) de chaque corps retenu pour participer à la simulation. Classez les corps dans l'ordre croissant par rapport à leur distance à l'origine du référentiel.

IV.C – On dispose des variables python `t0`, `p0`, `v0` et `masse` initialisées à partir du résultat de la requête précédente. `t0` est un entier qui donne la date des conditions initiales : il correspond à $t_{\min}$ et à `tmin()`. `p0` est une liste de longueur N, chaque élément de `p0` est une liste à 3 éléments de la forme `[x, y, z]` représentant la position initiale d'un corps, en unité astronomique. `v0` a une structure identique mais indique les vitesses initiales des corps considérés, en km/s. `masse` est décrite en partie III.

Écrire la fonction python `simulation_verlet(deltat, n)` qui prend en paramètre un incrément de temps en secondes (`deltat > 0`) et un nombre d'itérations (`n > 0`) et qui renvoie la liste des positions des corps considérés pour chaque instant `t0`, `t0 + deltat`, ..., `t0 + n*deltat` (cf variable `position` définie en partie III). Les calculs seront menés en utilisant le schéma d'intégration de Verlet, le résultat sera fourni en unité astronomique.

• • • FIN • • •

CONCOURS COMMUNS POLYTECHNIQUES

EPREUVE SPECIFIQUE - FILIERE PSI

INFORMATIQUE

Durée : 3 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Les calculatrices sont interdites

Le sujet comporte 12 pages dont :
- 9 pages de texte de présentation et énoncé du sujet ;
- 3 pages d'annexe.

Toute documentation autre que celle fournie est interdite.

REMARQUES PRELIMINAIRES

L'épreuve peut être traitée en langage **Python** ou langage **Scilab**. Il est demandé au candidat de bien préciser sur sa copie le choix du langage et de rédiger l'ensemble de ses réponses dans ce langage. Les syntaxes Python et Scilab sont rappelées en annexe **V.1**, page 10.

Les différents algorithmes doivent être rendus dans leur forme définitive sur la copie en respectant les éléments de syntaxe du langage choisi (les brouillons ne seront pas acceptés).

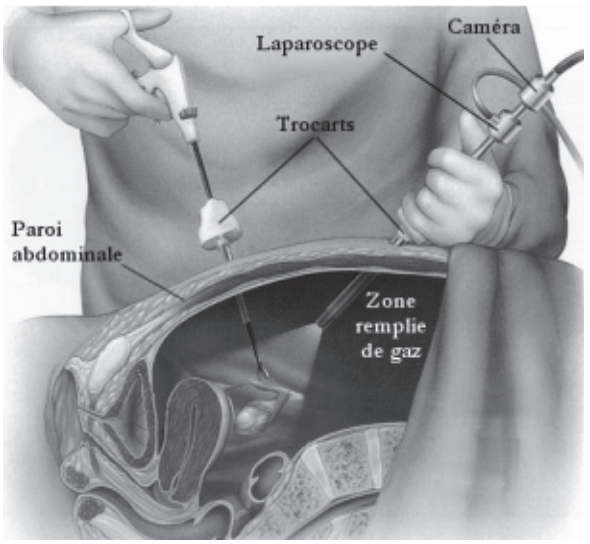
Il est demandé au candidat de bien vouloir rédiger ses réponses **en précisant bien le numéro de la question traitée et, si possible, dans l'ordre des questions.** La réponse ne doit pas se cantonner à la rédaction de l'algorithme sans explication, les programmes doivent être expliqués et commentés.

Robot Evolap - Suivi d'instrument chirurgical

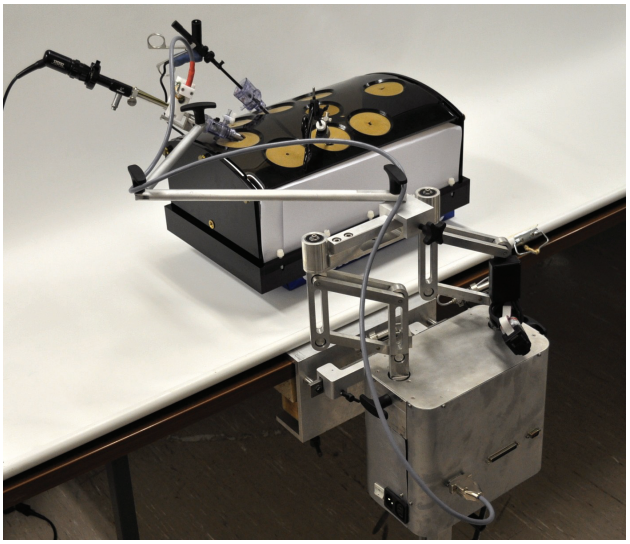
I Présentation

La laparoscopie est une technique chirurgicale mini-invasive de diagnostic et d'intervention abdominale. Une ou plusieurs petites incisions sont réalisées dans la paroi abdominale pour y insérer, au travers de canules appelées trocarts, de longs instruments chirurgicaux et un endoscope rigide - appelé laparoscope - surmonté d'une caméra. Ce laparoscope est également connecté à une source de lumière froide pour éclairer la cavité abdominale. Le robot EVOLAP est un robot d'assistance à la chirurgie laparoscopique développé par l'Université Catholique de Louvain (UCL). Il permet d'éviter à l'assistant chirurgical de porter le laparoscope pendant les opérations et améliore le bon déroulement de celles-ci.

Objectif
L'objectif de l'étude proposée est de réaliser le programme de suivi de l'instrument chirurgical par le robot EVOLAP, ce qui permet au chirurgien une vision optimale de la zone d'intervention.



a - Principe de la laparoscopie



b - Robot Evolap avec simulateur abdominal

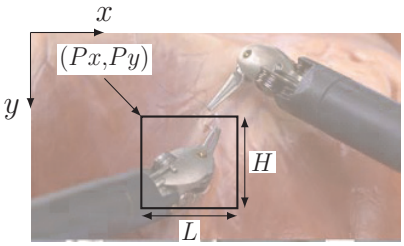
Figures 1 – Laparoscope manuel et robotisé.

Le robot possède deux moteurs qui permettent d'orienter le laparoscope selon deux angles indépendants autour du point de passage de l'appareil.

Le programme de suivi d'un instrument chirurgical à partir de l'analyse d'image de la zone de travail est constitué de plusieurs parties fondamentales :
- définition du motif à suivre dans l'image,
- acquisition d'une nouvelle image (issue du flux vidéo de la caméra),
- recherche du motif dans la nouvelle image,
- pilotage des moteurs pour aligner la caméra sur le motif détecté.

Dans tout le sujet, il sera supposé que les modules et bibliothèques sont déjà importés dans le programme.

II Définition du motif à suivre dans l'image



A partir du flux vidéo, un module permet d'extraire une image en couleur (RGB) à chaque instant de l'analyse. On notera `image` cette image. On définit sa largeur par la variable `image_width` et sa hauteur par la variable `image_height` définies en pixels (ou points) de 800×600. Le point supérieur gauche de l'image a pour coordonnées [0,0], le point inférieur droit [799,599] en Python et respectivement (1,1) et (800,600) en Scilab.

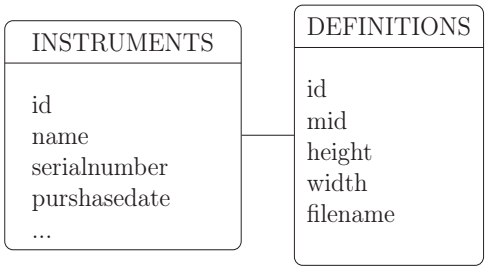
Pour définir le motif à suivre dans l'image, l'utilisateur doit spécifier les coordonnées (en pixels) du point supérieur gauche (Px,Py) d'une fenêtre ainsi que sa largeur L et sa hauteur H.

La fonction `demande_valeur(text)` est utilisée pour récupérer **une** valeur entière demandée à l'utilisateur. La vérification du type de donnée renvoyée par cette fonction est implantée dans celle-ci. L'argument `text` est le message affiché à l'écran pour demander la valeur à l'utilisateur.

Q1. Définir une fonction `demande_fenetre()` qui utilise la fonction `demande_valeur(text)` pour demander à l'utilisateur de définir la zone d'intérêt et qui renvoie une variable sous forme de tableau contenant les 4 informations recueillies `[Px,Py,L,H]`. Dans le programme principal, le retour de cette fonction sera nommé `fenetre`.

Q2. Définir une fonction `verification_fenetre(image_width,image_height,fenetre)` qui vérifie que la fenêtre soit bien à l'intérieur de l'image et qui renvoie un booléen `True` si la fenêtre est définie correctement et `False` sinon.

Plutôt que de spécifier manuellement la zone à suivre, l'utilisateur peut choisir un motif correspondant à un instrument chirurgical donné à partir d'une base de données, qui permet de connaître les vues d'un instrument particulier quelle que soit son orientation, sa taille...



Cette base de données est constituée de deux tables.
La table INSTRUMENTS contient les différents types d'instruments avec les attributs :
- id : identifiant de type entier, clé primaire,
- name : nom de l'instrument de type texte,
- serialnumber : numéro de série du produit,
- purshasedate : date d'achat de l'instrument,
- autres attributs non détaillés...

La table DEFINITIONS contient des entités qui correspondent à la définition d'un instrument pour une configuration donnée (nommée motif). Elle contient les attributs :
- id : identifiant de type entier, clé primaire,
- mid : identifiant de l'instrument correspondant à la définition de type entier,
- height : hauteur en pixels du motif de type entier,
- width : largeur en pixels du motif de type entier,
- filename : nom du fichier image de type texte (stocké sur réseau) correspondant à ce motif.

A un instrument peut correspondre une ou plusieurs définitions.

Q3. Ecrire une requête SQL permettant de récupérer les noms de toutes les images qui correspondent à l'instrument dont le nom est `"pince"`.

III Pilotage des moteurs pour déplacer la caméra

La caméra envoie à chaque instant t une image en couleur. L'image à l'instant $t + dt$ est appelée image courante tandis que l'image à l'instant t est appelée image précédente.

La partie suivante va permettre de comprendre comment extraire de l'image courante une nouvelle fenêtre homothétique de la fenêtre définie dans l'image précédente.

La nouvelle fenêtre étant déterminée, on cherche ensuite à recentrer la caméra sur l'image, c'est-à-dire que la fenêtre doit se trouver au centre de l'image. On suppose que, sur la première image, la fenêtre autour de l'instrument est bien centrée.

Q4. Ecrire une fonction `centre(fenetre)` permettant de calculer le centre de la fenêtre définie par `fenetre=[Px,Py,L,H]`.

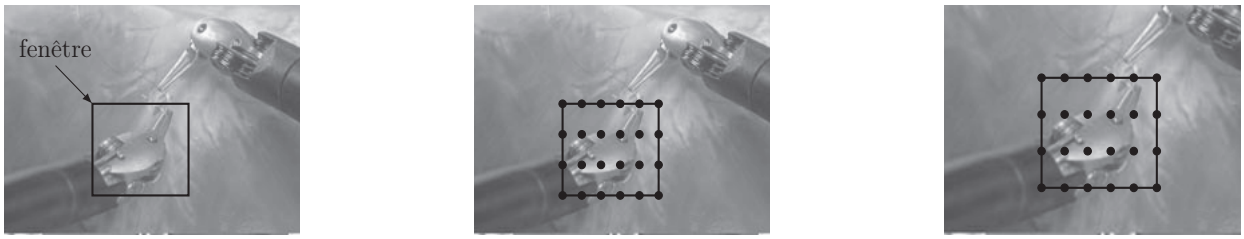
Cela permet de calculer la loi de mouvement à imposer aux moteurs du robot qui porte le laparoscope afin de suivre la fenêtre d'étude.

IV Recherche d'un motif dans une image

La difficulté de l'algorithme est de déterminer la fenêtre dans la nouvelle image à partir de l'analyse du mouvement des pixels de la fenêtre entre l'ancienne image et la nouvelle.

Après un traitement de l'image courante (conversion en niveaux de gris), connaissant la fenêtre dans l'image précédente (figure 2a), on extrait tout d'abord de la fenêtre d'étude quelques pixels d'intérêt répartis régulièrement (figure 2b). On recherche ensuite le déplacement moyen de chacun de ces pixels d'une image à l'autre. Il est alors nécessaire de procéder à une élimination des pixels pour lesquels le déplacement obtenu est incohérent. Plusieurs vérifications sont donc mises en place pour décider de l'élimination de ces pixels aberrants. Une fois le motif identifié dans l'image courante, on estime le grossissement possible de ce motif et on renvoie la nouvelle fenêtre d'étude qui englobe ces pixels (figure 2c).

Les parties suivantes vont permettre de spécifier les algorithmes et fonctions nécessaires à la recherche du motif dans l'image courante.



a - Image instant t et fenêtre entourant le motif

b - Image instant t et fenêtre avec les pixels d'intérêt

c - Image instant $t + dt$ et fenêtre avec les pixels d'intérêt

Figures 2 – Illustrations de la recherche de la fenêtre définie dans une image dans une nouvelle image.

IV.1 Traitement de l'image courante

L'image acquise par la caméra est une image en couleur constituée de trois couches (RGB). Les données de l'image sont stockées dans un tableau à trois dimensions : la première dimension correspond à la couleur (rouge, vert ou bleu, indice 0, 1 ou 2), la seconde correspond à la coordonnée selon $\vec{x}$ et la troisième à la coordonnée selon $\vec{y}$. Ainsi, les dimensions du tableau sont : $3 \times m \times n$ où $m = 800$ et $n = 600$.

La valeur associée à chaque pixel est un entier compris entre 0 et 255.

Q5. Donner la quantité de mémoire nécessaire en octets pour stocker le tableau représentant l'image émise par la caméra en justifiant le codage retenu pour un pixel d'une couche.

La première étape de l'algorithme de suivi d'image est de convertir l'image en couleur en niveaux de gris. La méthode consiste à rechercher le maximum et le minimum pour chaque pixel sur les trois couches, puis à faire la moyenne de ces maxima et minima et ainsi obtenir la valeur du nouveau pixel en niveaux de gris.

On note `imagecolor` le tableau représentant une image en couleur.

Q6. Ecrire une fonction `grayscale(imagecolor)` qui renvoie une image en niveaux de gris (qui sera notée `image` dans l'algorithme principal), sous forme de tableau à deux dimensions de taille $m \times n$ contenant des valeurs entières comprises entre 0 et 255, en suivant l'algorithme décrit ci-dessus. Remarque : il est possible d'utiliser les fonctions `min` et `max` internes au langage choisi.

IV.2 Extraction des pixels d'intérêt de la fenêtre d'étude

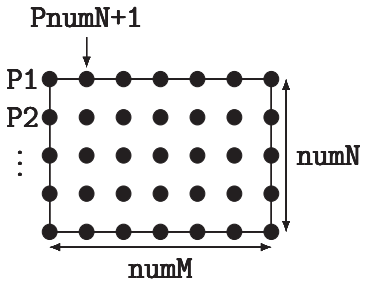
Pour déterminer le déplacement du motif, il est nécessaire d'extraire les pixels de la fenêtre définie initialement. Pour améliorer l'efficacité de la stratégie, on ne sélectionne pas tous les pixels mais uniquement un petit nombre, nommés pixels d'intérêt, de cette fenêtre (figure 2b).

`numM` > 1 est le nombre de points souhaités horizontalement.

`numN` > 1 est le nombre de points souhaités verticalement.

Les points doivent être équirépartis sur la zone décrite par la fenêtre.

Un point est défini par ses coordonnées `x` et `y`, respectivement l'indice de la colonne et l'indice de la ligne, en prenant comme origine le coin supérieur gauche de l'image, bord inclus.



Q7. Ecrire une fonction `construction_coordonnees_pts(fenetre,numM,numN)` qui renvoie un tableau de points noté `pts=[P1x,P1y,P2x,P2y,...]` dans l'algorithme principal (le parcours des points se fait de haut en bas puis de la gauche vers la droite). On ne traitera pas les cas pour lesquels `numN` ou `numM` sont égaux à 1. On rappelle que `fenetre=[Px,Py,L,H]`.

IV.3 Recherche des pixels d'intérêt dans l'image courante

On note `imgJ` le tableau à deux dimensions permettant d'accéder aux valeurs des pixels en niveau de gris de l'image courante (au temps $t + dt$) et `imgI` celui pour l'image précédente (temps t).

On utilise la méthode de Lucas-Kanade pour trouver les pixels d'intérêt de l'image précédente dans l'image courante.

Cette méthode suppose que le déplacement d'un point de l'image entre deux instants consécutifs est petit et se fait à vitesse constante. Pour estimer ce déplacement, on considère une zone de quelques pixels centrée autour du pixel d'intérêt étudié et on suppose que le déplacement est

le même pour tous les pixels de cette zone. La résolution de ce problème se fait par la méthode des moindres carrés explicitée dans la suite du sujet.

IV.3.1 Définition de l'algorithme

Pour mettre en place la méthode de Lucas-Kanade, il est nécessaire de partir d'une définition continue de l'image et de l'algorithme.

On note $I(t)$ l'image en niveaux de gris à l'instant t et $I(t+dt)$ l'image en niveaux de gris obtenue à un instant $t+dt$. Pour repérer un point de l'image, on utilise la notation $I(t)[P]$ qui désigne la valeur en niveaux de gris de l'image à l'instant t au point P de coordonnées $[x,y]$.

Soit u un point de coordonnées $[ux,uy]$ sur l'image $I(t)$.

L'algorithme permet de trouver les coordonnées de ce point dans l'image $I(t+dt)$ (notées $v=[vx,vy]=u+d$) en supposant que la valeur du pixel ne change pas, c'est-à-dire :
$I(t)[u]=I(t+dt)[v]$ (on parle de stationnarité de la fonctionnelle $I(x,y,t)$).

Le vecteur $d=[dx=V_x dt, dy=V_y dt]$ représente le déplacement du pixel à la vitesse constante $V=[V_x,V_y]$. Les variations dx, dy autour d'un point sont exprimées en pixels. En pratique, le déplacement n'est que de quelques pixels.

Ainsi, étudier la stationnarité du pixel revient à résoudre le problème suivant :

$$Ix\,dx + Iy\,dy = -It\,dt$$

où $Ix = \dfrac{\partial I(t)[u]}{\partial x}$ et $Iy = \dfrac{\partial I(t)[u]}{\partial y}$ sont les dérivées spatiales (dérivées en un point de coordonnées u suivant les directions x ou y de l'image $I(t)$) et $It = \dfrac{\partial I(t)[u]}{\partial t}$ est la dérivée temporelle de l'image (dérivée en un point de coordonnée u entre l'image $I(t)$ et $I(t+dt)$).

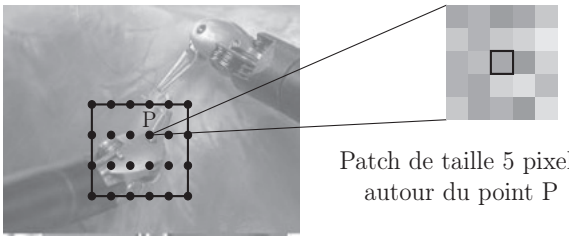
Q8. Proposer des approximations pour définir numériquement les termes Ix, Iy et It en fonction de `imgI`, `imgJ`, `ux`, `uy` et `dt`. On ne traitera pas les cas où le point u est sur le bord de l'image. Pour définir une dérivée numérique spatiale, on prendra des variations dx, dy égales à ± 1 pixel.

On note $Itdt$ le produit It par le pas de temps dt défini par le rafraîchissement des images de la caméra. On définit la fonction `calc_LK_terms(pixP,imgI,imgJ)` qui permet de calculer et de retourner les termes de l'équation de stationnarité, `Ix,Iy,Itdt`, définie précédemment pour un pixel `pixP` donné :

$$Ix\,dx + Iy\,dy = -Itdt.$$

On suppose cette fonction connue ce qui permet de répondre à la suite du problème indépendamment de la spécification de la question précédente.

IV.3.2 Implantation et résolution



Patch de taille 5 pixels autour du point P

L'algorithme indique que la relation précédente est écrite pour quelques pixels dans un voisinage proche d'un pixel d'intérêt P appartenant au motif étudié (question **Q7**, page 5). Il est donc nécessaire de définir ce voisinage autour du point P que l'on appellera `patch`. Ce voisinage contiendra 5×5 points contigus dont le point central sera le point P. On fait l'hypothèse que le point P n'est pas près du bord et qu'il y a toujours suffisamment de pixels disponibles autour du point P pour définir le patch.

Q9. Ecrire une fonction `creation_patch(P,patch_size)` où `P=[Px,Py]` est le pixel étudié et `patch_size=5` est le nombre de points choisis horizontalement et verticalement dans le voisinage du point P. Cette fonction renverra un tableau de coordonnées `patch=[P1x,P1y,P2x,P2y...]` classé du haut vers le bas et de gauche à droite comme à la question **Q7**, page 5. On supposera que la variable `patch_size` est impaire.

En écrivant l'équation de stationnarité pour tous les pixels du patch et en supposant que le déplacement est le même que celui du point P, on obtient un système composé de `patch_size` $\times$ `patch_size` équations à 2 inconnues ($d = [dx,dy]$). On doit ainsi résoudre un système qui s'écrit sous la forme $A.d = b$.

Q10. Ecrire une fonction `calc_Ab(imgI,imgJ,patch)`, utilisant la fonction `calc_LK_terms`, qui renvoie un tableau de taille (`patch_size` $\times$ `patch_size`) $\times$ 2 (noté `A` dans la suite) et un vecteur (noté `b`) de taille (`patch_size` $\times$ `patch_size`) $\times$ 1.

Le système $A.d = b$ n'a pas de solution. On cherche cependant un déplacement d qui minimise au sens des moindres carrés la norme quadratique de $Ad - b$. Ceci revient à résoudre le système matriciel 2×2 : $A^T A.d = A^T.b$ où A^T représente la transposée de A.

Q11. Ecrire une fonction `resoud_LK(A,b)` qui renvoie les deux composantes dx et dy de d. On supposera que la matrice $A^T A$ est inversible et on ne fera donc aucune vérification concernant son inversibilité. Détailler la résolution sans utiliser de solveur interne au langage choisi en séparant et en commentant correctement chaque partie de l'algorithme.

IV.3.3 Synthèse : algorithme de recherche de pixels déplacés

Les sous-fonctions définies dans les parties précédentes permettent de déterminer le déplacement des pixels d'intérêt de la fenêtre de l'image précédente `imgI` et ainsi de retrouver ces pixels dans l'image courante `imgJ`.

On définit alors le tableau des pixels trouvés noté `fpts` dans l'image courante `imgJ` et stocké de la même manière que le tableau `pts` défini à la question **Q7**, page 5.

Q12. Définir une fonction `recherche_points(imgI,imgJ,pts)` qui utilise les fonctions précédentes et qui renvoie un tableau de points qui correspondent aux pixels déplacés. La résolution du système pouvant donner des valeurs non entières de pixels, on approchera les solutions obtenues par leur partie entière. On supposera de plus que cette résolution fournit toujours une solution.

Q13. Expliquer, en quelques phrases, les limites de l'algorithme proposé en discutant des cas qui pourraient ne pas fournir de solution.

IV.4 Vérification des points obtenus

Les points obtenus par résolution du système ne correspondent pas forcément à chaque pixel déplacé. C'est pourquoi il est nécessaire de mettre en place une ou plusieurs vérifications pour éliminer les points qui ne conviennent pas.

La première vérification consiste à s'assurer qu'il est possible de retrouver les points initiaux de `imgI` en partant des points obtenus dans l'image courante `imgJ` par la même méthode. Par exemple, `P1` dans `imgI` donne `P1n` dans `imgJ`, qui donne `P1nn` dans `imgI`. Si `P1nn` est différent de `P1`, alors il est éliminé, sinon il est conservé.

Q14. Ecrire la partie du programme principal qui permet d'obtenir un tableau de booléen (noté `statut` dans le programme principal) de la taille du nombre de points représentant le motif. Chaque case contient `True` si le point a été retrouvé et `False` sinon. On utilisera les variables définies tout au long du sujet et la fonction `recherche_points`.

La deuxième vérification fait appel à la fonction suivante qui utilise les pixels sélectionnés (points1) de l'image précédente et les pixels trouvés (points2) de l'image courante.

Définition Python

```
def calcul1(points1,points2):
  return sqrt((points1[0::2]-points2[0::2])**2+(points1[1::2]-points2[1::2])**2)
```

Définition Scilab

```
function calcul1(points1,points2)
  return sqrt((points1(1:2:$)-points2(1:2:$))^2+(points1(2:2:$)-points2(2:2:$))^2)
endfunction
```

Q15. Indiquer quelle opération mathématique est réalisée par cette fonction `calcul1` et préciser le type et la dimension de ce que renvoie la fonction.

Pour éliminer des points, on calcule la médiane des valeurs obtenues et on supprime les points dont la valeur est trop éloignée de la médiane.

Q16. Ecrire une fonction `mediane(a)`, basée sur un algorithme de tri de votre choix, qui renvoie la valeur médiane du tableau de valeurs `a`. On suppose que les valeurs de `a` sont strictement positives et que la longueur de `a` est impaire.

Q17. Nommer l'algorithme de tri utilisé et donner sa complexité dans le meilleur et le pire des cas.

Q18. Ecrire une fonction `verification_pts(pts,fpts,statut)` qui renvoie un nouveau tableau de points (noté `nouveaux_pts` dans le programme principal) pour lesquels le premier critère est vérifié et la valeur obtenue par le deuxième critère est inférieure ou égale à la médiane.

Il est possible d'ajouter un troisième critère pour plus de précision concernant cette détection. On peut par exemple calculer une corrélation croisée (CCORR) entre les images `imgI` et `imgJ` sur des patchs autour de chaque point de la fenêtre d'étude. Des librairies permettent de réaliser cette corrélation croisée normée. La documentation de la fonction correspondante est détaillée dans l'annexe **V.2**.

Q19. En utilisant cette documentation, expliquer quelles commandes taper pour calculer cette corrélation croisée normée pour chaque point. Préciser ce que renvoie la fonction. On utilisera à nouveau la fonction `creation_patch(P,patch_size)`.

IV.5 Définition de la nouvelle fenêtre

Le tableau des points trouvés dans l'image courante peut être inclus dans une nouvelle fenêtre image de la fenêtre définie initialement. Celle-ci a subi une homothétie appelée facteur de grossissement.

On donne en annexe **V.3** la fonction `determine_fenetre` permettant de renvoyer la nouvelle fenêtre dans l'image `imgJ` et le facteur de grossissement à partir des points valides trouvés dans la nouvelle image. Cette fonction a été développée par une autre personne. Le développeur a utilisé sa propre spécification des variables : c'est-à-dire que la façon de stocker les grandeurs d'entrées et de sorties est différente de celle définie dans le programme principal et adoptée depuis le début du sujet. On sait que `bb0` correspond à la fenêtre dans l'image précédente, `pt0` aux points dans l'image précédente et `pt1` aux points dans l'image courante.

Q20. Préciser les variables qui ont été définies différemment et indiquer ce qui doit être modifié afin que la fonction puisse être utilisée dans le programme principal. Il n'est pas demandé de réécrire cette fonction.

Q21. Indiquer sur votre copie ce que font les parties de programmes entre les zones de commentaires en précisant le numéro du commentaire (`com1` à `com6`). Des commentaires d'une à deux lignes maximum sont attendus.

Fin de l'énoncé

V Annexes

V.1 Rappels des syntaxes en Python et Scilab

Remarque : sous Python, l'import du module numpy permet de réaliser des opérations pratiques sur les tableaux : from numpy import *. Les indices de ces tableaux commencent à 0.

Remarque : sous Scilab, les indices des tableaux commencent à 1.

	Python	Scilab
tableau à une dimension	`L=[1,2,3]` (liste) `v=array([1,2,3])` (vecteur)	`v=[1, 2, 3]` ou `[1 2 3]`
accéder à un élément	`v[0]` renvoie 1	`v(1)` renvoie 1
ajouter un élément	`L.append(5)` uniquement sur les listes	`v($+1) = 5`
tableau à deux dimensions (matrice) :	`M=array(([1,2,3],[3,4,5]))`	`M=[1,2,3;3,4,5]`
accéder à un élément	`M[1,2]` ou `M[1][2]` donne 5	`M(2,3)` donne 5
extraire une portion de tableau (2 premières colonnes)	`M[:,0:2]`	`M(:,1:2)`
tableau de 0 (2 lignes, 3 colonnes)	`zeros((2,3))`	`zeros(2,3)`
dimension d'un tableau `T` de taille (i,j)	`T.shape` donne `[i,j]`	`length(T)` donne `(i,j)`
séquence équirépartie quelconque de 0 à 10.1 (exclus) par pas de 0.1	`arange(0,10.1,0.1)`	`[0:0.1:10]`
définir une chaîne de caractères	`mot="Python et Scilab"`	`mot="Python et Scilab"`
taille d'une chaîne	`len(mot)`	`length(mot)`
extraire des caractères	`mot[2:7]`	`part(mot,[11:16])`

boucle For	`for i in range(10):`
	`  print(i)`
		`for i=1:10`
		`  disp(i);`
		`end`

condition If	`if (i>3):`
	`  print(i)`
	`else`
	`  print("hello")`
		`if (i>3) then`
		`  disp(i)`
		`else`
		`  disp("hello")`
		`end`

définir une fonction qui possède un argument et renvoie 2 résultats	`def fonction(param):`
	`  res=param`
	`  res2=param*param`
	`  return res1,res2`
		`function [res,res2]=fonction(param)`
		`  res=param`
		`  res2=param*param`
		`endfunction`

V.2 Documentation de la fonction `match_template`

Compares a template against overlapped image regions.

Python : `result=cv2.matchTemplate(image, templ, method)`
Scilab : `result=matchTemplate(image, templ, method)`

Parameters :
- `image` : Image where the search is running. It must be 8-bit or 32-bit floating-point.
- `templ` : Searched template. It must be not greater than the source image and have the same data type. `result` : Map of comparison results. It must be single-channel 32-bit floating-point. If `image` is $W \times H$ and `templ` is $w \times h$, then `result` is $(W - w + 1) \times (H - h + 1)$.
- `method` : Parameter specifying the comparison method (see below).

The function slides through `image` , compares the overlapped patches of size $w \times h$ against `templ` using the specified method and stores the comparison results in `result` . Here are the formulae for the available comparison methods (I denotes `image`, T `template`, R `result`). The summation is done over template and/or the image patch : $x' = 0...w-1, y' = 0...h-1$

- method=CV_TM_SQDIFF

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

- method=CV_TM_SQDIFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- method=CV_TM_CCORR

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

- method=CV_TM_CCORR_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- method=CV_TM_CCOEFF

$$R(x,y) = \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))$$

where

$$T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$$
$$I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

- method=CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

After the function finishes the comparison, the best matches can be found as global minimums (when `CV_TM_SQDIFF` was used) or maximums (when `CV_TM_CCORR` or `CV_TM_CCOEFF` was used) using the `minMaxLoc` function. In case of a color image, template summation in the numerator and each sum in the denominator is done over all of the channels and separate mean values are used for each channel. That is, the function can take a color template and a color image. The result will still be a single-channel image, which is easier to analyze.

V.3 Définition de la fonction determine_fenetre

Définition Python

```
def determine_fenetre(bb0, pt0, pt1):
    nPts = len(pt0)
    ofx = []
    ofy = []
# A COMPLETER (com1)
    for i in range(nPts):
        ofx.append(pt1[i][0]-pt0[i][0])
        ofy.append(pt1[i][1]-pt0[i][1])
# A COMPLETER (com2)
    dx = mediane(ofx)
    dy = mediane(ofy)
# A COMPLETER (com3)
    dist0=[]
    for i in range(nPts):
        for j in range(i+1,nPts):
            temp0 = ((pt0[i][0] - pt0[j][0])**2 + (pt0[i][1] - pt0[j][1])**2)**0.5
            temp1 = ((pt1[i][0] - pt1[j][0])**2 + (pt1[i][1] - pt1[j][1])**2)**0.5
            dist0.append(temp1/temp0)
    shift = mediane(dist0)
# A COMPLETER (com4)
    s0 = 0.5 * (shift - 1) * bb0[2]
    s1 = 0.5 * (shift - 1) * bb0[3]
# A COMPLETER (com5)
    x1 = bb0[0] - s0 + dx
    y1 = bb0[1] - s1 + dy
    x2 = bb0[2] + s0 + dx
    y2 = bb0[3] + s1 + dy
    w = x2-x1
    h = y2-y1
# A COMPLETER (com6)
    bb1 = [int(x1),int(y1),int(w),int(h)]
    return [bb1, shift]
```

Définition Scilab

```
function [bb1, shift]=determine_fenetre(bb0, pt0, pt1)
    nPts = length(pt0)
    ofx = zeros(nPts)
    ofy = zeros(nPts)
// A COMPLETER (com1)
    for i = [1:nPts]
        ofx(i)=pt1(i,1)-pt0(i,1)
        ofy(i)=pt1(i,2)-pt0(i,2)
    end
// A COMPLETER (com2)
    dx = mediane(ofx)
    dy = mediane(ofy)
    dist0=zeros(nPts)
// A COMPLETER (com3)
    for i = [1:nPts]
        for j = [i+1,nPts]
            temp0 = ((pt0(i,1) - pt0(j,1))**2 + (pt0(i,2) - pt0(j,2))^2)^0.5
            temp1 = ((pt1(i,1) - pt1(j,1))**2 + (pt1(i,2) - pt1(j,2))^2)^0.5
            dist0(i)=temp1/temp0
        end
    end
    shift = mediane(dist0)
// A COMPLETER (com4)
    s0 = 0.5 * (shift - 1) * bb0(3)
    s1 = 0.5 * (shift - 1) * bb0(4)
// A COMPLETER (com5)
    x1 = bb0(1) - s0 + dx
    y1 = bb0(2) - s1 + dy
    x2 = bb0(3) + s0 + dx
    y2 = bb0(4) + s1 + dy
    w = x2-x1
    h = y2-y1
// A COMPLETER (com6)
    bb1 = [int(x1),int(y1),int(w),int(h)]
endfunction
```

CONCOURS COMMUNS POLYTECHNIQUES

EPREUVE SPECIFIQUE - FILIERE TSI

INFORMATIQUE

Durée : 3 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Les calculatrices sont interdites.

L'épreuve est composée de deux dossiers :
- un premier dossier de 20 pages contenant le sujet (pages numérotées de 1/20 à 16/20) et les annexes (pages numérotées de 17/20 à 20/20) ;
- un second dossier de 12 pages constituant le document réponse (DR) à rendre en fin d'épreuve (pages numérotées de DR – 1/12 à DR – 12/12).

Les consignes permettant de compléter le document réponse (DR) sont données à la page suivante.

Seul le document réponse est à rendre.

Remarques générales

Les réponses aux questions sont à rédiger sur le document réponse (DR) et ne doivent pas dépasser les dimensions des cadres proposés.

Si la réponse attendue est spécifique à un langage, la réponse peut être proposée en langage Python ou en langage Scilab. Vous devez alors clairement entourer le langage retenu pour répondre à la question. Il est possible de changer de langage au cours de l'épreuve.

On distingue deux cas de figure sur les documents réponse : un premier cas où le cadre de réponse propose deux langages. Un second où il existe deux cadres spécifiques.

Les exemples proposés ci-dessous issus respectivement de la question 5 (dans laquelle le langage retenu pour la réponse serait le langage Scilab) et de la question 8 (dans laquelle le langage retenu pour la réponse serait le langage Pyhton) illustrent le principe d'indication du langage support de la réponse.

Question 5

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

Python **Scilab**

Question 8

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

Python

```
def cherchePremiereOccurence(car,chaine ):
        ...
```

Scilab

```
function      cherchePremiereOccurence(car,chaine)
        ...
endfunction
```

Quel que soit le langage choisi, les structures algorithmiques doivent être clairement identifiables par des indentations visibles ou par des barres droites entre le début et la fin de la structure comme proposé ci-dessous :

Si *(Condition)*
  Alors
    | *Instructions*
  Sinon
    | *Instructions*
Fin Si

ACQUISITION ET EXPLOITATION D'UN DEPLACEMENT A PARTIR D'UN RECEPTEUR GPS

Le sujet consiste en l'analyse et l'exploitation de données GPS (Global Positioning System) acquises par un récepteur équipant un véhicule. Ces dernières sont obtenues via un dispositif de réception installé à bord et fonctionnant à une fréquence allant de 1 à 4 Hertz. Chacune des mesures reçues contient (entre autre) des informations de positionnement (latitude, longitude) et de vitesse.

Le décodeur de trames inclus dans le récepteur GPS envoie les informations au calculateur embarqué sur un bus de communication suivant un protocole particulier. Dans le cadre de ce sujet, le protocole retenu est le protocole NMEA 0183 (National Marine Electronics Association), utilisant des trames de type GPRMC (Recommended minimum specific GPS/Transit data) et utilisable avec la plupart des récepteurs GPS. Le format des trames GPRMC est défini en annexe 1 (page 17).

Problématique

Comment afficher sur une carte le parcours d'un véhicule ainsi que sa vitesse moyenne à partir de la latitude, de la longitude et de la vitesse extraites de trames GPS émises par un satellite ?

1 Acquisition des trames NMEA via une liaison série

L'objectif de cette première partie est d'enregistrer les trames GPS reçues par le décodeur dans un fichier texte nommé `tramesNMEA.txt`. Ce fichier sera utilisé en temps différé pour afficher le parcours réalisé.

Le décodeur GPS reçoit des trames provenant des satellites à la fréquence de 1 Hertz. Elles sont ensuite transmises sur un port série dans le but d'être traitées par le calculateur embarqué. La figure 1 ci-dessous présente le principe de fonctionnement du système étudié.

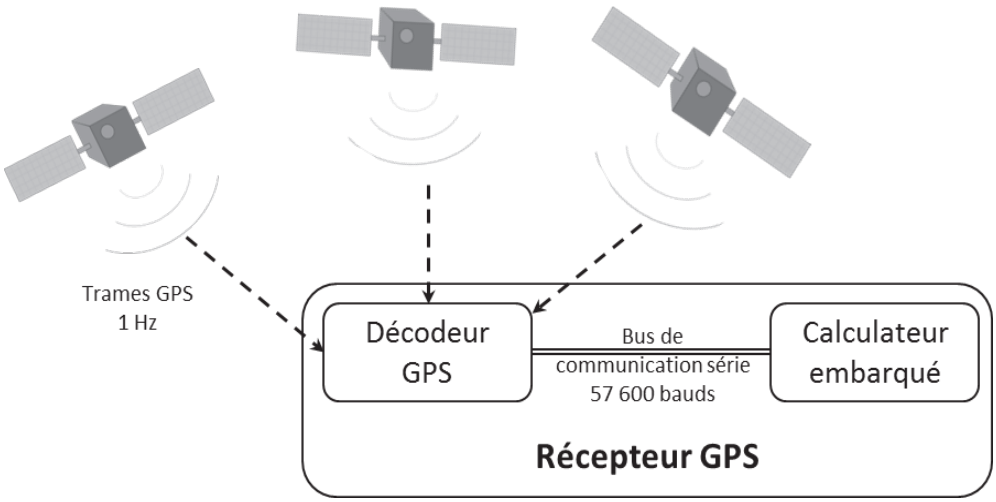


FIGURE 1 – Communication entre les différents composants du système

1.1 Liaison série

Le décodeur GPS envoie sur la liaison série de type RS-232C une trame GPRMC.

De manière générale, une liaison série permet de véhiculer, sur un nombre réduit de supports physiques, des informations élémentaires les unes à la suite des autres.

Dans le protocole RS-232C, pour chaque octet à transmettre, 1 bit de start, 8 bits de données (correspondant à l'octet à transmettre), puis 1 bit de stop sont envoyés sur le bus de communication. Ainsi, pour chacun des octets d'une trame GPRMC, 10 bits sont transmis. Le signal de communication est bivalent, chaque bit est codé par un signal électrique haut ou bas dont la durée dépend du débit choisi pour la transmission. Au débit de 57 600 bauds choisi ici, 57 600 bits sont donc transmis à chaque seconde.

Nous faisons l'hypothèse que la liaison est fiable, c'est-à-dire que l'on considère qu'il n'y a pas de panne du récepteur ou de déconnexion physique de la liaison série.

On considère une trame de 76 caractères codés en ASCII. On donne en annexe 1 (page 17) un exemple de trame et en annexe 2 (page 18) la table ASCII.

Question 1 *Au vu de la problématique du sujet, préciser combien d octets sont nécessaires dans une trame GPS de type GPRMC pour encoder la latitude et la longitude avec leurs précisions Nord – Sud ou Est – Ouest et la vitesse en nœuds. – Dans le dénombrement les séparateurs de champs (virgules) ne seront pas comptabilisés. –*

Question 2

1. *Quelle est la durée de transmission d une trame de 76 caractères sur le bus RS-232C ?*
2. *Le débit de transmission du bus série est-il suffisant pour traiter en temps réel les trames provenant des satellites ?*

1.2 Lecture des trames GPS

On souhaite utiliser une bibliothèque permettant la gestion des entrées/sorties sur une liaison série. Un extrait de la spécification de cette bibliothèque est donné ci-après en Python dans la table 1 (ci-dessous) et en Scilab dans la table 2 (page 5).

Python

```
def initSerie(port,bauds) :
"""
Initialise un port série et retourne un
 identifiant de port série idport :
 Entrées :
    * port : int, entier >=0
    * bauds : int, débit en bauds du port
 Sortie :
    * idport : int, l'identifiant du port
"""
```

Python

```
def lireSerie(idport) :
"""
 Attend l'arrivée d'un octet sur le port série
 et le retourne sous la forme d'un entier
 (code ASCII).
 Entrée :
    * idport : int, identifiant de port série
 Sortie :
    * car : int, valeur de l'octet lu sur le
                port série
"""
```

TABLE 1 – Extrait de la spécification d'une bibliothèque de gestion de liaison série en Python

Scilab

```
1 function [idport]= initSerie (port,bauds)
2 // Initialise un port serie et retourne un
3 // identifiant de port serie idport
4 // Entrées :
5 // * port : int, entier >=0
6 // * bauds : int , debit en bauds du port
7 // Sortie :
8 // * idport : int, l'identifiant du port
```

Scilab

```
1 function [car]= lireSerie (idport)
2 // Attend l'arrivée d'un octet sur le port série
3 // et le retourne sous la forme d'un entier
4 // (code ASCII).
5 // Entrée :
6 // * idport : int, identifiant de port série
7 // Sortie :
8 // * car : int, valeur de l'octet lu sur le
9 // port série
```

TABLE 2 – Extrait de la spécification d'une bibliothèque de gestion de liaison série en Scilab

Question 3 *Donner l instruction permettant d initialiser le port série numéroté « 0 » à une vitesse de 57 600 bauds. L identifiant du port sera stocké dans une variable nommée identifiant.*

Question 4

1. *En utilisant les annexes 1 et 2 (pages 17 et 18), déterminer le dernier caractère de la trame. Préciser quel est son code ASCII sous forme décimale.*
2. *A partir des informations contenues dans l annexe 3 (page 19), indiquer de quelle manière obtenir un caractère à partir de son code ASCII exprimé sous forme décimale.*

Définitions

On appelle **signature** d'une fonction le nom de la fonction ainsi que l'ensemble des paramètres reçus et éventuellement retournés par cette dernière. Cette signature doit être syntaxiquement correcte vis-à-vis du langage retenu (exemple : lignes 1 des tables 1 (page 4) et 2 (page 5) pour la fonction initSerie()).

On appelle **spécification** d'une fonction la description de l'objectif de la fonction ainsi que la description des paramètres d'entrée et de sortie. Pour les paramètres d'entrée, la description comprend également les contraintes liées à ces paramètres qui assurent le bon fonctionnement de la fonction (exemples : lignes 2 à 10 de la table 1 (page 4) pour la fonction initSerie() en langage Python et lignes 2 à 8 de la table 2 (page 5) pour la fonction initSerie() en langage Scilab).

La spécification apparaît en commentaire dans le début du corps d'une fonction juste après sa signature.

On donne dans la table 3 (page 6) l'algorithme en pseudo-code de la fonction lireTrame() permettant de lire et de retourner une trame à partir d'un port série préalablement ouvert.

On se propose d'écrire l'implémentation de cette fonction en langage Python ou en langage Scilab à partir des annexes 3, 4 ou 5 (page 19).

Question 5 *Suivant le langage retenu, traduire l algorithme proposé table 3 en précisant bien la signature et la spécification de la fonction.*

Algorithme : lecture et affichage d'une trame

Fonction lireTrame(*Paramètre à définir* **)**
- **Entrée :** *à définir*
- **Sortie :** *à définir*
- trame ← chaîne de caractères vide
- lu ← 0
- **Tant que** lu ≠ nombre entier correspondant au caractère de fin de trame en ASCII **faire**
  - lu ← lire octet série correspondant à un caractère
  - ajouter en fin de trame le caractère correspondant à lu
- **Fin Tant que**
- **Afficher** (trame)
- **Retourner** (trame)

Fin Fonction lireTrame

TABLE 3 – Algorithme en pseudo-code de la fonction lireTrame()

1.3 Enregistrement dans un fichier

Nous souhaitons maintenant enregistrer toutes les trames envoyées par le récepteur GPS dans un fichier texte nommé tramesNMEA.txt situé dans le dossier courant. Le fichier texte sera ouvert en début de programme de sorte à être vidé s'il existait déjà auparavant. Ici, le récepteur GPS envoyant des trames en continu, le programme à réaliser n'a pas de fin et ne se terminera que lorsqu'il sera explicitement fermé par l'utilisateur. Pour implémenter ce comportement, on pourra utiliser l'expression « while (True) :» en Python ou « while (%T) » en Scilab qui mettent en place, dans chacun des langages, une boucle infinie.

Les annexes 4 et 5 (page 19) présentent des fonctions en langages Python et Scilab permettant de manipuler les fichiers.

Question 6 *En utilisant les fonctions précédentes, écrire dans le langage de votre choix le programme principal qui, après initialisation du port de la liaison série, permet de lire les trames sur le port série et d enregistrer ces dernières, telles qu elles sont reçues, dans le fichier tramesNMEA.txt, au format texte.*

2 Exploitation du fichier de trames

L objectif de cette partie est d extraire et d enregistrer une trame dans une structure de données cohérente afin de pouvoir exploiter aisément les mesures.

On fait l hypothèse que les trames transmises ne contiennent pas d erreur.

2.1 Extraction d une trame

La description d'une trame GPMRC, donnée en annexe 1 (page 17), indique que les valeurs des différentes informations de la trame sont comprises entre les caractères « $ » et «*».

Nous souhaitons disposer d'une fonction cherchePremiereOccurrence() susceptible de retourner la position de la première occurrence d'un caractère ASCII (exemple d'utilisation de la fonction : rechercher la position de la première occurrence de « $ » dans une trame).

On donne, dans la table 4, le corps de la fonction isCharInString() en langages Python et Scilab. Cette fonction permet de déterminer si un caractère particulier, passé en paramètre, est présent dans une chaîne de caractères, elle aussi passée en paramètre.

Python

```
1  def isCharInString(car,chaine):
2      """
3      Recherche si un caractère est dans une
4      chaîne de caractères.
5      Entrées :
6        * car : str, caractère
7        * chaine : str, chaîne de caractères
8      Sortie :
9        * un booléen : True si le caractère est
10       présent, False sinon
11     """
12     n = len(chaine)
13     i=0
14     while i<n :
15         if chaine[i]==car:
16             return True
17     return False
```

Scilab

```
1  function [bool] = isCharInString(car,chaine)
2  // Recherche si un caractère est dans une
3  // chaîne de caractères.
4  // Entrées :
5  // * car : str, caractère
6  // * chaine : str, chaîne de caractères
7  // Sortie :
8  // * un booléen : True si le caractère est
9  // présent, False sinon
10     n = length(chaine);
11     i=1;
12     bool = %F;
13     while i<=n
14         if (part(chaine,[i])==car) then
15             bool = %T
16             break
17         end
18     end
19 endfunction
```

TABLE 4 – Code de la fonction isCharInString() de recherche d'un caractère dans une chaîne de caractères en langages Python et Scilab

Question 7

1. *Préciser pour quelle raison cette première version de la fonction isCharInString(), proposée table 4 (ci-dessus), ne fonctionne pas lors des tests de validation.*
2. *Proposer une modification du corps de la fonction en précisant les numéros de lignes de code modifiées ou créées.*

Question 8 *En vous appuyant sur l algorithme proposé table 4 (ci-dessus), compléter le corps de la fonction chercheePremiereOccurrence() qui retourne l indice de la première occurrence d un caractère passé en paramètre. Il convient de choisir et de justifier ce que la fonction retourne dans le cas où le caractère recherché n est pas présent dans la chaîne.*

On cherche à concevoir l'algorithme de la fonction extraireTrame().

Cette fonction doit permettre d'extraire et de retourner, au format chaîne de caractères, le contenu d'une trame (nommée NMEA) situé strictement entre les caractères «$» et «*» en s'assurant au préalable que ces caractères sont bien présents dans la chaîne. La trame extraite se nomme NMEA_extraite.

A titre d'exemple, le contenu de la trame NMEA_extraite issue de la fonction extraireTrame(), basée sur la trame donnée en exemple dans l'annexe 1 (page 17) est de la forme :

GPRMC,071139.988,A,4639.8232,N,00021.6810,E,8.95,184.14,141014,1.1,W,A

On précise que G est en première position de la trame extraite, c'est-à-dire en position 0 en langage Python et en position 1 langage en Scilab.

L'annexe 6 (page 20) indique la manière dont sont indexées les chaînes de caractères en langage Python et en langage Scilab. Cette annexe précise aussi comment extraire des sous-chaînes de caractères dans une chaîne de caractères.

Question 9

1. *Indiquer quelles sont les conditions nécessaires et suffisantes que doit respecter une trame quelconque afin de pouvoir être extraite grâce à la fonction extraireTrame().*
2. *A partir des fonctions proposées dans cette partie, établir, sous forme algorithmique, une implémentation de la fonction extraireTrame() dans laquelle la chaîne d entrée se nomme NMEA et la chaîne de sortie NMEA_extraite. Dans le cas où la chaîne ne peut pas être extraite, la fonction doit retourner une chaîne vide.*

2.2 Structure de données

Dans le cadre des traitements ultérieurs que nous souhaitons effectuer, il nous faut représenter une mesure GPS, que nous nommerons par la suite pointGPS, par sa latitude, sa longitude et sa vitesse.

Dans la trame GPS, les longitudes et latitudes sont données en degrés et minutes. Cette représentation ne facilitant pas les calculs, il est nécessaire de les convertir en minutes (1 degré étant égal à 60 minutes). Par ailleurs, on opte pour la convention suivante :
- pour la latitude, le signe positif est choisi pour le Nord ;
- pour la longitude, le signe positif est choisi pour l'Est.

Ainsi par exemple, si la trame contient la latitude «0314.2500» suivie de l'indicateur «S», la séquence «03» correspond à 3 degrés soit 180 minutes d'angle, la séquence «14.2500» correspond à 14,25 minutes d'angle et l'indicateur «S» implique que la latitude doit être précédée du signe moins. Ainsi, pour la latitude, on stockera -194.25 dans le pointGPS. Cette conversion est assurée par la fonction conversionLongLat2Min() dont la signature et les spécifications sont données dans la table 5 (ci-dessous).

Python

```python
def conversionLongLat2Min(ll,orientation) :
"""
Conversion de la la longitude ou de la latitude en minutes (nombre positif ou négatif)
  Entrées :
    * ll : str, latitude sous la forme ddmm.mmmm ou longitude sous la forme dddmm.mmmm
    * orientation : str, "N", "S", "E" ou "W".
  Sortie :
    * res : float : résultat en minutes d'angle
"""
```

Scilab

```scilab
function [res]= conversionLongLat2Min(ll,orientation)
// Conversion de la la longitude ou de la latitude en minutes (nombre positif ou négatif)
// Entrées :
// * ll : str, latitude sous la forme ddmm.mmmm ou longitude sous la forme dddmm.mmmm
// * orientation : str, "N", "S", "E" ou "W".
// Sortie :
// * res : float, résultat en minutes d'angle
```

TABLE 5 – Signature et spécification de la conversion de la latitude ou de la longitude en nombre exploitable en langages Python et Scilab

Afin de stocker un pointGPS on utilise une liste qui présente la structure interne suivante :

$$\textbf{pointGPS} = \textbf{[latitude, longitude, vitesse]}.$$

Cette liste sera renseignée par les latitudes, longitudes et les vitesses extraites des différentes trames envoyées par le décodeur et traitées par les fonctions précédentes.

Il existe deux principales méthodes pour extraire ces données, une première repose sur le comptage des séparateurs et peut s'adapter à des longueurs variables pour un même champ. La seconde, plus simple à mettre en œuvre, repose sur l'hypothèse que toutes les trames présentent exactement la même structure et que toutes les longueurs de champs restent identiques entre les différentes trames reçues.

Dans la suite du sujet, nous ferons l'hypothèse que toutes les longueurs de champs sont toujours identiques dans les différentes trames à traiter.

On rappelle qu'en appliquant la fonction extraireTrame() (définie dans la partie précédente) à la trame donnée en annexe 1 (page 17), on obtient :

GPRMC,071139.988,A,4639.8232,N,00021.6810,E,8.95,184.14,141014,1.1,W,A

Question 10

1. *A partir de la trame extraite issue de l annexe 1 (page 17) et rappelée ci-dessus, ainsi que des précisions apportées sur l annexe 6 (page 20), compléter le tableau donnant l index de début et l index de fin des informations nécessaires à l extraction de la latitude (y compris l orientation), de la longitude (y compris l orientation) et de la vitesse.*
2. *Compléter la fonction creationPointGPS() dont la spécification et la signature sont données dans la table 6 (ci-dessous). On utilisera pour cela la fonction conversionLongLat2Min() ainsi que les informations données précédemment.*

Python

```python
def creationPointGPS(NMEA_extraite) :
"""
Stocke les valeurs de latitude, longitude et vitesse dans une structure de type pointGPS
Entrée :
  * NMEA_extraite : chaîne de caractères, contenu de la trame (entre $ et *)
      (résultat de la fonction extraireTrame())
Sortie :
  * pointGPS : liste de la forme [latitude, longitude, vitesse]
"""
```

Scilab

```scilab
function [pointGPS]=creationPointGPS(NMEA_extraite)
// Stocke les valeurs de latitude, longitude et vitesse dans une structure de type pointGPS
// Entrée :
// * NMEA_extraite : chaîne de caractères, contenu de la trame (entre $ et *)
// (résultat de la fonction extraireTrame())
// Sortie :
// * pointGPS : liste de la forme [latitude, longitude, vitesse]
```

TABLE 6 – Spécification de la fonction creationPointGPS() en Python et Scilab

2.3 Affichage du fichier

Nous disposons maintenant de toutes les fonctions qui permettent d'exploiter le fichier de trames créé dans la première partie (tramesNMEA.txt). Nous souhaitons à présent afficher successivement chacun des pointGPS présents dans le fichier sur une console texte.

Pour la gestion des fichiers, on pourra utiliser, suivant le langage cible, les fonctions close(), open(), readline() et write() dont les descriptions sont proposées en annexe 4 (page 19) ou les fonctions file(), read() et write() dont les descriptions sont proposées en annexe 5 (page 19). On précise qu'il n'est pas indispensable d'utiliser toutes les fonctions.

Question 11 *Le fichier tramesNMEA.txt contient les éléments correspondant aux pointsGPS. En utilisant le langage de votre choix, établir le programme principal qui affiche sur la console chacun de ces éléments. Chaque ligne de la console devra être de la forme suivante :*

latitude_Pt_1 longitude_Pt_1 vitesse_Pt_1
latitude_Pt_2 longitude_Pt_2 vitesse_Pt_2

Chaque espace correspond à une tabulation.

3 Structure avancée de données

L objectif de cette partie est de stocker les points dans un arbre afin de pouvoir accéder aux pointsGPS ayant la plus grande vitesse de façon optimale.

Un tas binaire est une structure de données informatiques qui permet d'accéder au maximum (respectivement minimum) d'un ensemble de données en temps constant. On peut la représenter par un arbre binaire vérifiant deux contraintes :

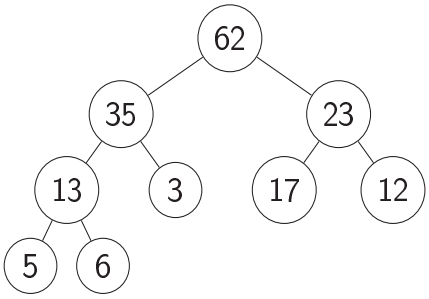
- c'est un arbre binaire parfait : tous les niveaux de l'arbre (excepté le niveau le plus bas) sont totalement remplis. Si le dernier n'est pas totalement rempli alors il doit l'être de gauche à droite ;
- c'est un tas : une clé est associée à chaque nœud de l'arbre. Cette dernière doit être supérieure ou égale aux clés de chacun de ses fils.

Ainsi, lorsque les clés sont des nombres (valeurs) et quand la relation d'ordre choisie est l'ordre naturel, on parle alors de tas-max (ou max-heap).

Un tas binaire étant un arbre binaire complet, on peut l'implémenter à l'aide d'un tableau tel que :

- la racine de l'arbre (niveau le plus haut) se trouve à l'index absolu 1 ;
- en considérant un nœud à l'index absolu i :
  - son fils gauche se trouve à l'index absolu $2i$;
  - son fils droit se trouve à l'index absolu $2i+1$;
- en considérant un nœud à l'index absolu $i > 1$:
  - son père se trouve à l'index absolu i 2, le symbole / désignant ici la division entière.

La figure 2 (page 11) illustre un arbre binaire (trié) et le tableau qu'on peut lui associer avec les index associés (index absolu, index en langage Python, index en langage Scilab).



Valeur dans l'arbre (clé)	62	35	23	13	3	17	12	5	6
Index absolu dans l'arbre	1	2	3	4	5	6	7	8	9
Index du tableau en Python	0	1	2	3	4	5	6	7	8
Index du tableau en Scilab	1	2	3	4	5	6	7	8	9

Le nœud ayant la valeur 62 est la racine de l arbre (niveau le plus haut). Les nœuds ayant les clefs 5 et 6 sont au niveau le plus bas.

FIGURE 2 – Arbre binaire et son implémentation en tableau

Dans un tas binaire, on insère l'élément à la prochaine position libre (la position libre la plus à gauche possible sur le dernier niveau) puis on effectue l'opération suivante (nommée percolateUp()) pour rétablir si nécessaire la propriété d'ordre du tas binaire.

Le fonctionnement de cette fonction peut être appréhendé de la manière suivante : tant que l'élément n'est pas la racine de l'arbre et que la clé de l'élément est strictement supérieure à son père, on échange les positions entre l'élément et son père. L'exemple de la figure 3 (ci-dessous) montre l'insertion d'un élément ayant une clé de valeur 72.

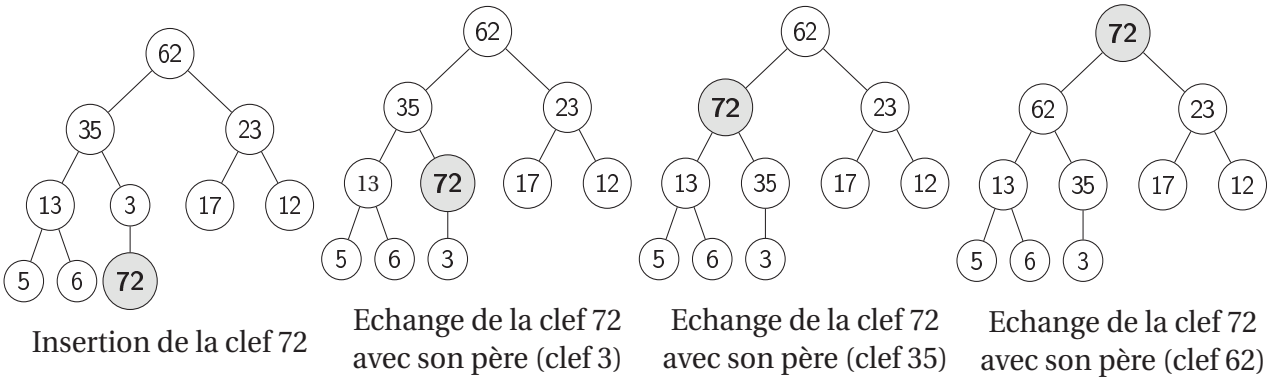


FIGURE 3 – Insertion d'un élément de clef 72

La table 7 (page 12) propose le code de la fonction percolateUp() en langages Python et Scilab.

On souhaite stocker les pointGPS dans un tas binaire. La liste (tas) sera donc de la forme suivante :

[[latitude_0, longitude_0, vitesse_0],[latitude_1, longitude_1, vitesse_1],...].

Quand un tas binaire est réalisé sur un ensemble structuré de variables comme c'est le cas sur les variables de type pointGPS, un des paramètres de cette variable structurée devient la clé. Les comparaisons présentes dans la fonction percolateUp() se font alors sur cette clé.

Python

```python
def percolateUp (tas) :
    index = len(tas)-1
    val = tas[index]
    while index>0 :
        index_pere = int((index-1)/2)
        pere = tas[index_pere]
        if pere > val :
            break
        else :
            tas[index] = pere
            tas[index_pere] = val
            index = index_pere
```

Scilab

```scilab
function [tas]=percolateUp(tas)
    index = size(tas);
    val=tas(index);
    index_pere = (index)/2;
    while index_pere>=1
        pere = tas(index_pere);
        if pere>val then
            break;
        else
            tas(index)=pere;
            tas(index_pere)=val;
            index=index_pere;
            index_pere = (index)/2;
        end
    end
endfunction
```

TABLE 7 – Fonction percolateUp() en langages Python et Scilab

Question 12 *Modifier la fonction percolateUp() de façon à stocker des variables de type pointGPS dans les nœuds de l arbre. Pour cela, on barrera la (ou les) ligne(s) modifiée(s) du document réponse et on écrira la modification en vis-à-vis (en précisant le (ou les) numéro(s) de ligne(s) modifiée(s)). Le paramètre vitesse, stocké dans un pointGPS (à la position 2 en langage Python et à la position 3 en langage Scilab), jouera le rôle de clé.*

Question 13 *Proposer une fonction insererPointGPS() qui agit sur le tas binaire passé en paramètre et qui permet d insérer un point GPS, passé en paramètre, dans ce dernier.*

Question 14

1. *Proposer l implémentation de la fonction extractMax() qui attend le tas passé en paramètre et qui permet d extraire de ce dernier, sans le modifier, le pointGPS possédant la plus grande vitesse.*
2. *Donner la complexité de cette opération.*

4 Interface graphique

L objectif de cette partie est d extraire les points contenus dans le fichier tramesNMEA.txt et de les tracer sur un fond de carte. La couleur de chaque segment du tracé doit dépendre de la vitesse du véhicule.

Tout d'abord, nous allons étudier un principe de lissage de la vitesse afin d'y associer un code couleur et dans un second temps nous étudierons la mise au point du tracé de chaque segment sur le fond de carte sélectionné.

4.1 Etude du principe de calcul des vitesses lissées et association d un code couleur à chaque vitesse lissée.

La vitesse d'un véhicule est soumise à des aléas fréquents. Elle peut varier fortement entre deux points successifs de mesure. Pour minimiser ces écarts, on souhaite réaliser un lissage sur 4

points. Pour cela, en un point considéré, on réalise la moyenne des vitesses entre les deux points mesurés précédemment, la vitesse du point en cours et la vitesse du point suivant.

Dans un premier temps, on considère que les vitesses sont stockées dans la liste présentée dans la table 8 (ci-dessous), dont on donne l'index des valeurs.

Valeur de vitesse	0.0	1.1	2.2	3.4	4.5	5.3	4.6	3.8	2.6	1.2
Index absolu dans le tableau	1	2	3	4	5	6	7	8	9	10
Index du tableau en langage Python	0	1	2	3	4	5	6	7	8	9
Index du tableau en langage Scilab	1	2	3	4	5	6	7	8	9	10

TABLE 8 – Exemple de tableau de vitesses

Question 15

1. *En utilisant la liste proposée table 8, préciser à partir de quel index absolu et jusqu à quel index absolu il est possible de déterminer une moyenne glissante.*
2. *Pour la liste proposée table 8, en déduire le nombre de moyennes glissantes différentes qu il est possible de déterminer.*
3. *Pour une liste de n vitesses, combien de moyennes glissantes serait-il possible de déterminer ?*

La table 9 donne l'algorithme partiel de la fonction moyenneGlissante() qui permet de réaliser une moyenne glissante sur 4 valeurs consécutives.

Cet algorithme est établi pour les index absolus (le codage de cet algorithme prend bien sûr en compte les spécificités de chaque langage).

Algorithme : calcul de la moyenne glissante

```
Fonction moyenneGlissante(vit)
    Entrée : vit : liste de vitesses instantanées
    Sortie : res : liste des vitesses lissées sur 4 valeurs
    res ← liste vide
    Pour i allant de index_i_absolu_début à index_i_absolu_fin inclus par pas de 1
        Som ← 0
        Moy ← 0
        Pour j allant de index_j_absolu_début à index_j_absolu_fin inclus par pas de 1
            A compléter
            A compléter
        Fin Pour
        Ajouter Moy dans la liste res
    Fin Pour
    Retouner res
Fin Fonction moyenneGlissante
```

TABLE 9 – Algorithme partiel en pseudo-code de la fonction moyenne glissante

Question 16

1. *Pour l algorithme proposé table 9 (page 13), déterminer les valeurs de* index_i_absolu_début *et* index_i_absolu_fin.
2. *Préciser quelles sont les valeurs de* index_j_absolu_début *et* index_j_absolu_fin *en fonction de* i.
3. *Compléter les lignes manquantes dans cet algorithme.*

Désormais, on dispose d'une liste de vitesses lissées (compte tenu de la méthode de calcul, par moyenne glissante, cette liste est toutefois de taille plus réduite que celle des vitesses initiales).

A chacune de ces vitesses (lissées), on souhaite associer un code couleur. Le code couleur est un entier variant de 0 à 255. Le code couleur 0 correspond à la vitesse la plus faible (dans la liste des vitesses lissées). Le code couleur 255 correspond à la vitesse la plus élevée (dans la liste des vitesses lissées). Le code couleur évolue linéairement en fonction de la vitesse. On cherche à obtenir une liste constituée des couples *vitesse_lissee, code_couleur)*.

Question 17 *Dans la liste des vitesses lissées, indiquer quelles opérations de recherche sont nécessaires pour associer respectivement le code couleur 0 et le code couleur 255.*

Question 18 *En déduire la fonction mathématique à établir pour associer le code couleur à n importe quelle vitesse lissée présente dans la liste.*

4.2 Application : affichage du tracé

On souhaite maintenant représenter le déplacement réel de manière graphique, sur une carte. On dispose pour cela d'une bibliothèque graphique détaillée en annexe 7 (page 20). Le principe de fonctionnement consiste à initialiser une carte (un fichier image de type JPEG ou PNG) en lui associant une échelle et une origine. Cette origine est saisie par l'utilisateur du programme. L'initialisation est réalisée par l'appel de la fonction initCarte() (voir annexe 7 page 20).

Afin de faciliter le tracé du déplacement, nous souhaitons implémenter une fonction ajoutePoint() qui trace un nouveau point sur la carte et le relie au précédent par un segment si ce dernier n'est pas le premier point tracé. S'il s'agit du premier point du déplacement, la fonction se contente de tracer ce point sur la carte.

On précise que cette fonction est en mesure de s'adapter aux deux cas de figure énoncés ci-dessus grâce à l'utilisation de paramètres par défaut. Ces derniers apparaissent directement dans la signature de la fonction en Python ou sont traités dans le corps de la fonction en Scilab.

Les spécifications de la fonction ajoutePoint() sont données table 10 (page 15) pour les deux langages (pour la version en langage Scilab, les premières lignes de code gérant les valeurs par défaut sont également présentes).

On suppose que la surface de la carte à afficher est réduite (quelques centaines de mètres carrés). La détermination de l'affichage des points GPS sur cette carte relève alors d'une opération de conversion. A partir de la circonférence de la Terre, on montre qu'un degré de latitude correspond à 111,199233 km soit 111 199,233 mètres et qu'un degré de longitude correspond à $111\,199{,}233 \cdot \cos(\text{Latitude})$ mètres.

Ces opérations de conversions sont réalisées dans le corps des fonctions tracePoint() et traceSegment() présentées en annexe 7 (page 20).

La couleur du tracé du segment dépend uniquement du code couleur associé au second point.

Python

```python
def ajoutePoint (id_carte,lat_Pt_1,long_Pt_1,lat_Pt_2=0,long_Pt_2=0,coul=255) :
"""
Trace le point Pt_1 de latitude lat_Pt_1 et de longitude long_Pt_1 sur la carte correspondant à
l'identifiant id_carte. Trace un segment entre le point Pt_1 et le point Pt_2 si ce dernier existe.
Entrées :
  * id_carte : int, identifiant de la carte
  * lat_Pt_1, long_Pt_1 : flt, latitude et longitude du point Pt_1
  * lat_Pt_2, long_Pt_2 : optionnel, flt, latitude et longitude du point Pt_2
  * coul : optionnel, int, code couleur entre 0 et 255
"""
```

Scilab

```scilab
function ajoutePoint (id_carte, lat_Pt_1, long_Pt_1, lat_Pt_2, long_Pt_2, coul)
// Trace le point Pt_1 de latitude lat_Pt_1 et de longitude long_Pt_1 sur la carte correspondant à
// l'identifiant id_carte. Trace un segment entre le point Pt_1 et le point Pt_2 si ce dernier existe.
// Entrées :
// * id_carte : int, identifiant de la carte
// * lat_Pt_1, long_Pt_1 : double, latitude et longitude du point Pt_1
// * lat_Pt_2, long_Pt_2 : optionnel, double, latitude et longitude du point Pt_2
// * coul : optionnel, int, code couleur entre 0 et 255
    if exists("lat_Pt_2","local")==0 then
        lat_Pt_2 = 0 ; end // Valeur par défaut
    if exists("long_Pt_2","local")==0 then
        long_Pt_2 = 0 ; end // Valeur par défaut
    if exists("coul","local")==0 then
        coul = 255 ; end // Valeur par défaut
```

TABLE 10 – Spécification de la fonction ajoutePoint() en Python et Scilab

Question 19 *Sachant que le point de départ sera affiché en rouge, proposer une implémentation de la fonction ajoutePoint().*

On dispose de la fonction trameGPS2Liste() (dont les spécifications sont données dans la table 11, page 16). Elle permet d'extraire les informations à partir du fichier tramesNMEA.txt et de retourner une liste de points de type pointGPSLisse (lire point GPS lissé) dans une liste en ayant calculé la vitesse lissée ainsi que le code couleur correspondant.

On précise que le calcul de la vitesse lissée par moyenne glissante est un peu plus complexe que le principe étudié dans la partie précédente car chaque point dispose d'une vitesse lissée et d'un code couleur même si le calcul direct n'a pas été possible. Le premier point (origine du déplacement) possède une vitesse lissée nulle. La vitesse du second point est la moyenne entre celle du premier point et celle du troisième point. La vitesse du dernier point est égale à celle de l'avant dernier point.

Ainsi un pointGPSLisse a la structure suivante :

$$\text{pointGPSLisse} = [\text{latitude}, \text{longitude}, \text{vitesseLissee}, \text{codeCouleur}].$$

La fonction trameGPS2Liste() retourne donc une liste de la forme suivante :

$$[\text{pointGPSLisse\_0}, \text{pointGPSLisse\_1}, \ldots].$$

Question 20 *Compléter le programme principal qui affiche sur une carte le déplacement correspondant à la suite des trames GPS contenues dans le fichier tramesNMEA.txt. Le nom du fichier contenant la carte ainsi que l échelle et l orientation seront saisis à la console par l utilisateur.*

Python

```python
def trameGPS2Liste (fichier) :
"""
Ouvre un fichier de trames et retourne une liste de points GPS lissés
[pointGPSLisse_0,pointGPSLisse_1,...].
Entrée :
  * fichier : str, nom du fichier contenant les trames
Sortie :
  * res : list, liste de quadruplets
"""
```

Scilab

```scilab
function [res] = trameGPS2Liste (fichier)
// Ouvre un fichier de trames et retourne une liste de points GPS lissés
// [pointGPSLisse_0,pointGPSLisse_1,...].
// Entrée :
// * fichier : str, nom du fichier contenant les trames
// Sortie :
// * res : list, liste de quadruplets
```

TABLE 11 – Spécification de la fonction trameGPS2Liste() en Python et Scilab

5 Synthèse des activités

Question 21 *Proposer une synthèse des activités permettant d expliquer comment la problématique initiale a été résolue. Préciser éventuellement, quels peuvent être les avantages d organiser les données sous la forme d un tas binaire, comme présenté dans la partie 3 (page 10).*

Fin de l énoncé

ANNEXES

Annexe 1 – Le protocole NMEA 0183, type GPRMC

Dans le protocole NMEA 0183, chaque trame est constituée de chaînes de caractères, terminées par un retour chariot et un saut de ligne. Les caractères constituant la trame ont un code ASCII compris entre 20 et 127 mais sont codés sur un octet. Toute trame commence par les caractères $GP, suivis de trois caractères définissant le type de trame. Chaque trame de type GPRMC est donc préfixée par $GPRMC.

On donne ci-dessous un exemple de trame :

$GPRMC,071139.988,A,4639.8232,N,00021.6810,E,8.95,184.14,141014,1.1,W,A*65

Les trames (cf exemple ci-dessus), sont constituées de 13 champs texte, séparés par des virgules et terminés par une somme de contrôle de parité, résultat d'un « ou exclusif » de chaque octet compris strictement entre le caractère initial "$" et le caractère "*". L'interprétation de chaque champ dépend alors de sa position.

Position	Exemple	Interprétation
0	$GPRMC	Type de trame. Nous considérons uniquement le type RMC.
1	071139.988	Heure UTC de la mesure sous la forme hhmmss.sss (avec hh heures, mm minutes dans l'heure, ss.sss secondes et secondes décimales)
2	A	"A" pour mesure valide, "V" pour mesure invalide
3	4639.8232	Latitude donnée sous la forme ddmm.mmmm (avec dd degrés, mm.mmmm minutes et minutes décimales)
4	N	Complète le champ 3 pour préciser s'il s'agit de la latitude Nord ("N") ou Sud ("S")
5	00021.6810	Longitude donnée sous la forme dddmm.mmmm
6	E	Complète le champ 5 pour préciser s'il s'agit de la longitude Est ("E") ou Ouest ("W")
7	8.95	Vitesse en nœuds
8	184.14	Cap du déplacement par rapport au pôle Nord magnétique en degrés décimaux
9	141014	Date universelle donnée sous la forme jjmmaa (avec jj jour du mois, mm mois, aa année)
10	1.1	Variation magnétique sous la forme d.d en degrés décimaux.
11	W	Sens de la variation magnétique ("E" = Est, "W" = Ouest)
12	A *65	Mode de fonctionnement du récepteur Somme de contrôle exprimée en base hexadécimale sur deux chiffres
fin	<CR><LF>	La trame se termine par deux caractères : un retour chariot (code ASCII 13) puis un saut de ligne (code ASCII 10).

Annexe 2 – Table ASCII

Code décimal	Caractère ASCII	Description	Décimal	Caractère	Décimal	Caractère	Décimal	Caractère
0	NUL	Null	32	Space	64	@	96	`
1	SOH	Start of heading	33	!	65	A	97	a
2	STX	Start of text	34	”	66	B	98	b
3	ETX	End of text	35	#	67	C	99	c
4	EOT	End of transmission	36	$	68	D	100	d
5	ENQ	Enquiry	37	%	69	E	101	e
6	ACQ	Acknowledge	38	&	70	F	102	f
7	BEL	Bell	39	’	71	G	103	g
8	BS	Backsapce	40	(	72	H	104	h
9	TAB	horizontal tab	41	)	73	I	105	i
10	LF	New line feed, new line	42	*	74	J	106	j
11	VT	Vertical tab	43	+	75	K	107	k
12	FF	NP form feed, new page	44	‘	76	L	108	l
13	CR	Carriage return	45	-	77	M	109	m
14	SO	Shift out	46	.	78	N	110	n
15	SI	Shift in	47	/	79	O	111	o
16	DLE	Data link espace	48	0	80	P	112	p
17	DC1	Device control 1	49	1	81	Q	113	q
18	DC2	Device control 2	50	2	82	R	114	r
19	DC3	Device control 3	51	3	83	S	115	s
20	DC4	Device control 4	52	4	84	T	116	t
21	NAK	Negative acknowledge	53	5	85	U	117	u
22	SYN	Synchronous idle	54	6	86	V	118	v
23	ETB	End of trans. block	55	7	87	W	119	w
24	CAN	Cancel	56	8	88	X	120	x
25	EM	End of medium	57	9	89	Y	121	y
26	SUB	Substitute	58	:	90	Z	122	z
27	ESC	Escape	59	;	91	[	123	{
28	FS	File separator	60	<	92	\	124	|
29	GS	Group separator	61	=	93	]	125	}
30	RS	Record separator	62	>	94	^	126	~
31	US	Unit separator	63	?	95	_	127	DEL

Annexe 3 – Fonctions communes aux deux langages

On considère que les fonctions suivantes sont communes au langage Python et au langage Scilab.

chr(i) : Return the string representing a character whose Unicode codepoint is the integer i. For example, chr(97) returns the string 'a'. This is the inverse of ord(). The valid range for the argument is from 0 through 1,114,111 (0x10FFFF in base 16). ValueError will be raised if i is outside that range.
Remarque : pour i compris entre 0 et 127, la fonction retourne le caractère correspondant au code ASCII.

ord(c) : Given a string representing one Unicode character, return an integer representing the Unicode code point of that character. For example, ord('a') returns the integer 97 and ord('\u2020') returns 8224. This is the inverse of chr().

Annexe 4 – Fonctions Python de traitement des fichiers

close() : Flush and close this stream. This method has no effect if the file is already closed. Once the file is closed, any operation on the file (e.g. reading or writing) will raise a ValueError. As a convenience, it is allowed to call this method more than once ; only the first call, however, will have an effect.

open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None) : Open file and return a corresponding file object. If the file cannot be opened, an OSError is raised.
file is either a string or bytes object giving the pathname.
mode is an optional string that specifies the mode in which the file is opened. It defaults to 'r' which means open for reading in text mode. Other common values are 'w' for writing (truncating the file if it already exists), 'x' for exclusive creation and 'a' for appending (which on some Unix systems, means that all writes append to the end of the file regardless of the current seek position).

readline(size=-1) : Read until newline or EOF and return a single str. If the stream is already at EOF, an empty string is returned. If size is specified, at most size characters will be read.

write(s) : Write the string s to the stream and return the number of characters written.

Annexe 5 – Fonctions Scilab de traitement des fichiers

[unit [,err]]=file('open', file-name [,status] [,access [,recl]] [,format]) allows to open a file with specified properties and to get the associated unit number unit.
- file-name : string, file name of the file to be opened ;
- status : string, the status of the file to be opened :
  - "new" : file must not exist new file (default) ;
  - "old" : file must already exists ;
  - "unknown" : unknown status ;
  - "scratch" : file is to be deleted at end of session.

file(action,unit) : allows to close the file , or move the current file pointer. action : is one of the following strings :
- "close" : closes the file(s) given by the logical unit descriptors given in units ;
- "rewind" : puts the pointer at beginning of file ;
- "backspace" : puts the pointer at beginning of last record ;
- "last" : puts the pointer after last record.

[x] = read(file-desc,m,n,[format]) :reads row after row the mxn matrix x (n=1 for character chain) in the file file-desc (string or integer). Each row of the matrix x begin in a new line of file-desc file. Depending on format, a given row of the x matrix may be read from more than one line of file-desc file.

write(file-desc,a,[format]) : writes row-by-row a real matrix or a column vector of character strings in a formatted file. Each row of the a argument begin in a new line of file-desc file. Depending on format a given row of the a argument may be written in more than one line of file-desc file.

Annexe 6 – Précisions sur les chaînes de caractères en langage Python et en langage Scilab

Le stockage des chaînes de caractères n'est pas strictement identique dans les deux langages et les fonctions de traitement de chaînes ne sont pas non plus rigoureusement identiques. Pour lever toute ambiguïté, le tableau ci-contre précise les index de début et de fin de la chaîne "abcde" dans les langages Python et Scilab.

	Python	Scilab
Index de début	0	1
Index de fin	4	5

La table 12 précise comment extraire la sous-chaîne "bcd" de la chaîne "abcde" dans les deux langages.

Python

```
1 >>> chaine = "abcde"
2 >>> sous_chaine = chaine[1:4]
3 >>> print(sous_chaine)
4         'bcd'
```

Scilab

```
1 --> chaine = "abcde"
2 --> sous_chaine = part(chaine,2:4)
3 --> printf(sous_chaine)
4         bcd
```

TABLE 12 – Extraction de chaîne de caractère en langages Python et Scilab

Annexe 7 : API d une bibliothèque graphique

Nous donnons ci-dessous la description d'une bibliothèque graphique utilisable dans les langages Python et Scilab.

Fonction	Description
initCarte(carte,echelle,Lat,long)	Initialise une carte en fixant l'échelle de celle-ci ainsi que son orientation. Entrées : – carte : str, chemin du fichier image ; – echelle : flt en Python, double en Scilab, échelle en pixel/m ; – Lat : flt en Python, double en Scilab, latitude de l'origine ; – long : flt en Python, double en Scilab, longitude de l'origine. Sortie : – idcarte : int, identifiant de la carte.
afficheCarte(idcarte)	Affiche la carte ayant l'identifiant idcarte à l'écran. Entrée : – idcarte : int, identifiant de la carte à afficher à l'écran.
tracePoint(idcarte,X1,Y1)	Calcule la projection et trace un point de coordonnées (X1,Y1) et de couleur rouge sur la carte d'identifiant idcarte. Entrées : – idcarte : int, identifiant de la carte ; – X1 : flt en Python, double en Scilab, abscisse du point ; – Y1 : flt en Python, double en Scilab, ordonnée du point.
traceSegment(idcarte,X1,Y1,X2,Y2,coul)	Calcule les projections et trace un segment de couleur coul entre les points de coordonnées (X1,Y1) et (X2,Y2) sur la carte d'identifiant idcarte. Suivant la vitesse du véhicule, la couleur peut varier de gris clair à noir. Entrées : – idcarte : int, identifiant de la carte ; – X1 : flt en Python, double en Scilab, abscisse du point 1 ; – Y1 : flt en Python, double en Scilab, ordonnée du point 1 ; – X2 : optionnel, flt en Python, double en Scilab, abscisse du point 2 ; – Y2 : optionnel, flt en Python, double en Scilab, ordonnée du point 2 ; – coul : optionnel, int, couleur du segment entre 0 et 255 (255 par défaut).

TSIIN07

Acquisition et exploitation d un deplacement a partir d un recepteur GPS

Document Reponse

Question 1

.....

.....

.....

Question 2

1. .....

.....

.....

2. .....

.....

.....

Question 3

.....

Question 4

1. .....

.....

2. .....

.....

Question 5

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

Python **Scilab**

Question 6

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

Python **Scilab**

Question 7

1. .....

   .....

   .....

2.

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

Python **Scilab**

Question 8

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

Python

```
def cherchePremiereOccurence(car,chaine):
    """ Retourne l'indice de la première occurrence d'un caractère.
    Entrées :
      * car : str, caractère
      * chaine : str, chaine de caractères
    Sortie(s) :

    """
    n = len(chaine)

    while
        if chaine[ ]==car:
```

Scilab

```
function cherchePremiereOccurence(car,chaine)
//  Retourne l'indice de la première occurrence d'un caractère.
// Entrées :
// * car : str, caractère
// * chaine : str, chaine de caractères
//  Sortie :
//
    n=length(chaine)


    while
        if part( )==car then





endfunction
```

.....

.....

Question 9

1. .....

.....

.....

2.

Question 10

1. GPRMC,071139.988,A,4639.8232,N,00021.6810,E,8.95,184.14,141014,1.1,W,A

Python **Scilab**

	Latitude	**Orientation**	**Longitude**	**Orientation**	**Vitesse**
Index début					
Index fin					

2. **Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.**

Python

```
def creationPointGPS(NMEA_extraite) :
    latitude =
    longitude =
    vitesse =
    pointGPS =
    return pointGPS
```

Scilab

```
function [pointGPS] = creationPointGPS(NMEA_extraite)
    latitude =
    longitude =
    vitesse =
    pointGPS =
endfunction
```

Question 11

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

Python **Scilab**

Question 12

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

Python

```
def percolateUp (tas) :
    index = len(tas)-1
    val = tas[index]
    while index>0 :
        index_pere = int((index-1)/2)
        pere = tas[index_pere]
        if pere > val :
            break
        else :
            tas[index] = pere
            tas[index_pere] = val
            index = index_pere
```

Scilab

```
function [tas]=percolateUp(tas)
    index = size(tas);
    val=tas(index);
    index_pere = (index)/2;
    while index_pere>=1
        pere = tas(index_pere);
        if pere>val then
            break;
        else
            tas(index)=pere;
            tas(index_pere)=val;
            index=index_pere;
            index_pere = (index)/2;
        end
    end
endfunction
```

Question 13

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

```
def insererPointGPS( ) :
```

Python

```
function [tas]=insererPointGPS( )
```

Scilab

Question 14

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

1.

```
def extractMax( ) :
```

Python

```
function [ ]=extractMax( )
```

Scilab

2. Complexité de l'opération d'extraction : .....

Question 15

1. Index absolu de début : .....

   Index absolu de fin : .....

2. Nombre de moyennes glissantes : .....

3. Nombre de moyennes glissantes en fonction de n : .....

Question 16

1. *index_i_absolu_début* = ...

   index_i_absolu_fin = ...

2. *index_j_absolu_début* = ...

   index_j_absolu_fin = ...

3.

```
Fonction moyenneGlissante(vit)
    Entrée : vit : liste de vitesses instantanées
    Sortie : res : liste des vitesses lissées sur 4 valeurs
    res ← liste vide
    Pour i allant de index_i_absolu_début à index_i_absolu_fin inclus par pas de 1
        Som ← 0
        Moy ← 0
        Pour j allant de index_j_absolu_début à index_j_absolu_fin inclus par pas de 1
            .....
            .....
        Fin Pour
        Ajouter Moy dans la liste res
    Fin Pour
    Retouner res
Fin Fonction moyenneGlissante
```

Question 17

.....

.....

.....

.....

Question 18

.....

.....

.....

.....

.....

.....

.....

.....

Question 19

Réponse en langage Python OU en langage Scilab. Entourer le langage retenu.

Python

```
def ajoutePoint(id_carte,lat_Pt_1,long_Pt_1,lat_Pt_2=0,long_Pt_2=0,coul=255) :
```

Scilab

```
function ajoutePoint (id_carte, lat_Pt_1, long_Pt_1, lat_Pt_2, long_Pt_2, coul)
    if exists("lat_Pt_2","local")==0 then
        lat_Pt_2 = 0 ; end // Valeur par défaut
    if exists("long_Pt_2","local")==0 then
        long_Pt_2 = 0 ; end // Valeur par défaut
    if exists("coul","local")==0 then
        coul = 255 ; end // Valeur par défaut
```

Question 20

Python

```
# Saisie du nom de fichier


# Saisie du nom de la carte
carte = input("Saisir le nom de la carte : ")
# Saisie de l'échelle (stockée sous forme de chaîne de caractères)
echelle = input("Saisir l'échelle : ")
echelle = flt(echelle) # Conversion de la chaîne de caractères en flottant
# Saisie de la latitude de l'origine de la carte
lat0 = flt(input("Saisir la latitude de l'origine : "))
# Saisie de la longitude de l'origine de la carte
long0 = flt(input("Saisir la longitude de l'origne : "))
# Récupération des points àpartir du fichier de trame tramesNMEA.txt


#Initialisation de la carte


# Affichage de la carte


# Ajout de points sur la carte àpartir de la liste de points pts
for i
```

Encadré pour répondre en langage Scilab à la page suivante.

Question 20 – Suite

Scilab

```
// Saisie du nom de fichier


// Saisie du nom de la carte
file = input("Saisir le nom de la carte : ")
// Saisie de l'échelle (le type de la variable correspond au type saisi)
echelle = input("Saisir l'échelle : ")
// Saisie de la latitude de l'origine de la carte
lat0 = input("Saisir la latitude de l'origine : ")
// Saisie de la longitude de l'origine de la carte
long0 = input("Saisir la longitude de l'origine : ")
// Récupération des points àpartir du fichier de trame tramesNMEA.txt


//Initialisation de la carte


// Affichage de la carte


// Ajout de points sur la carte àpartir de la liste de points pts
for i
```

Question 21

.....

.....

.....

.....

.....

.....

.....

043

Epreuve d'Informatique et Modélisation

Durée 4 h

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, d'une part il le signale au chef de salle, d'autre part il le signale sur sa copie et poursuit sa composition en indiquant les raisons des initiatives qu'il est amené à prendre.

L'usage de calculatrices est interdit.

AVERTISSEMENT

La **présentation**, la lisibilité, l'orthographe, la qualité de la **rédaction, la clarté et la précision** des raisonnements entreront pour une **part importante** dans **l'appréciation des copies**. En particulier, les résultats non justifiés ne seront pas pris en compte. Les candidats sont invités à encadrer les résultats de leurs calculs.

L'épreuve comporte deux parties qui peuvent être traitées indépendamment l'une de l'autre. Une première partie modélisation (durée conseillée 1H30) et une seconde informatique (durée conseillée 2 H30)

Remarques préliminaires importantes : il est rappelé aux candidat(e)s que

Les explications des phénomènes étudiés interviennent dans la notation au même titre que les développements analytiques et les applications numériques ; les résultats exprimés sans unité ne seront pas comptabilisés.

Tout au long de l'énoncé, les paragraphes en italiques ont pour objet d'aider à la compréhension du problème.

Tout résultat fourni dans l'énoncé peut être admis et utilisé par la suite, même s'il n'a pas été démontré par le (la) candidat(e).

Les applications numériques, effectuées sans calculatrice, pourront supporter des arrondis ou simplifications judicieux.

*Ce problème traite de systèmes d'identification par radio fréquence (RFID). Il est constitué de deux parties totalement indépendantes : **l'étude d'un système de type navigo** pour la 1ère partie. Aucune connaissance particulière sur les antennes n'est demandée.*

En 1948, Harry Stockman décrivit dans un article un moyen de communiquer grâce à la réflexion des ondes, annonçant ainsi les futurs systèmes autonomes communiquant par ondes électromagnétiques. Leur but est, par exemple, d'identifier des objets à distance, d'où le nom d'identification radio-fréquence (RFID : Radio Frequency Identification). Quelques années plus tard, les premiers systèmes d'identification utilisant les ondes réfléchies étaient développés.
Les progrès réalisés dans le domaine de l'électronique ont permis son développement et son intégration dans de nombreux domaines. Aujourd'hui, la RFID (du fait de son prix et de la taille des étiquettes) prend une place de plus en plus importante dans la vie courante, et d'un simple fonctionnement en mode tout-ou-rien, au stockage et au traitement d'informations, les applications couvrent des domaines allant de la télédétection (identification d'animaux, antivols, localisation...) aux transactions de la vie courante: par exemple dans les systèmes de contrôle d'accès aux transports en commun, type passe Navigo de la RATP pour le métro parisien. Les puces RFID tentent aujourd'hui de supplanter les codes à barres en jouant de leurs avantages, à savoir qu'il est possible d'écrire, d'effacer et de réécrire les données stockées dans une puce un grand nombre de fois, que leur portée peut être supérieure aux lecteurs optiques utilisés pour les codes à barres, et que la communication peut se faire à travers certains obstacles contrairement aux systèmes à lecture optique.

Un système RFID passif est composé de deux entités qui communiquent entre elles (Fig.1) :

- *Un TAG passif (dénommé TAG par la suite) ou étiquette intelligente (aussi appelé transpondeur), associé à l'élément à identifier. Il est capable de répondre à une demande venant d'un lecteur. Le TAG n'a pas d'alimentation de type batterie ou pile mais est autoalimenté par l'onde électromagnétique reçue.*
- *Une station de base ou lecteur RFID qui a pour mission d'identifier le TAG. Le lecteur envoie une onde électromagnétique en direction de l'élément à identifier, cette onde alimente le TAG qui peut alors communiquer avec le lecteur grâce à sa puce électronique interne. En retour, le lecteur reçoit l'information renvoyée par le TAG.*

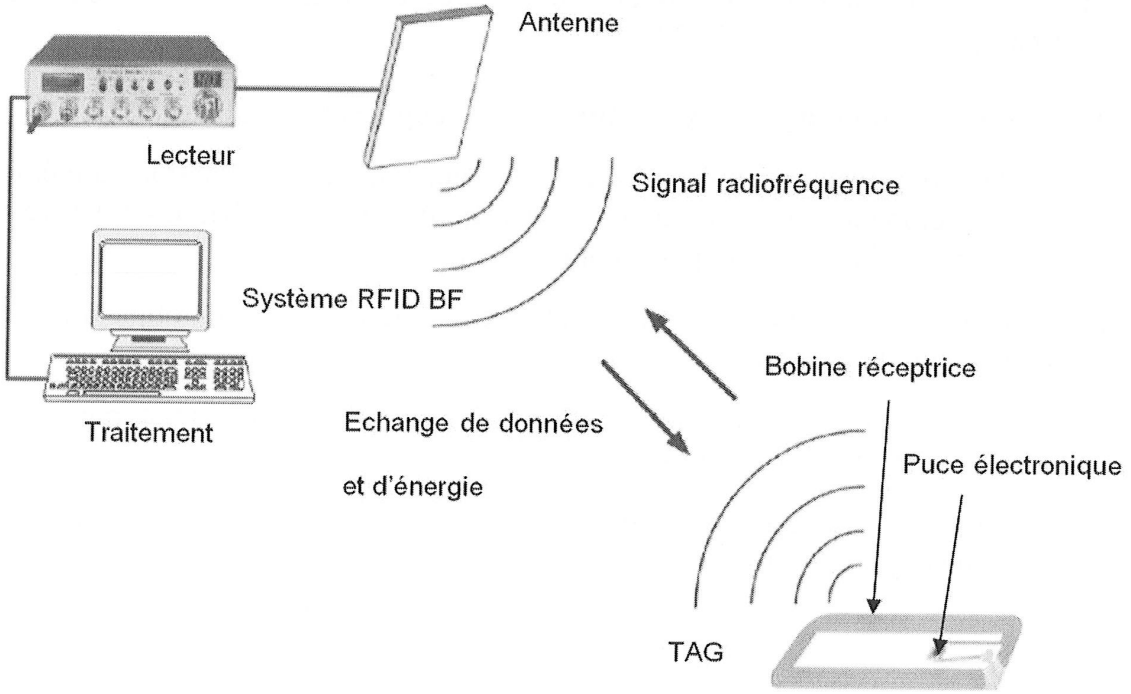


Figure 1

La figure 1 présente le fonctionnement général d'un système RFID. Le lecteur relié à une antenne émettrice agit en maître par rapport au TAG : si le TAG est dans la zone de lecture du lecteur, ce dernier l'active en lui envoyant une onde électromagnétique et entame la communication. Le TAG est quant à lui, constitué d'une antenne et d'une puce électronique qui module l'onde réémise vers le lecteur. En démodulant le signal reçu, le lecteur relié à une application interne récupère l'information pour la traiter, il est chargé de l'interface et de la gestion de l'identification des TAGs qui se présentent à lui.
Il existe plusieurs familles de systèmes RFID dont le principal critère de différentiation est la fréquence de fonctionnement. Les système RFID utilisent des bandes de fréquence à 125 kHz (bande BF), 13,56 MHz (bande HF), 860-960 MHz (bande UHF) et 2,45 GHz. On précise que la puissance rayonnée par l'antenne d'émission est une fonction croissante de la fréquence est varie en $\frac{1}{\lambda^4}$.

Bande	*Fréquence*	*Portée*	*Pouvoir de Pénétration dans un conducteur*
BF	*125 kHz*	+	++++
HF	*13.56 MHz*	++	+++
UHF	*860-960 MHz*	+++	++
UHF	*2,45 GHz*	++	+

+ : faible , ++ : Assez bon , +++ : Bon , ++++ : Excellent

En fonction des différentes fréquences, les principes physiques mis en œuvre ne sont pas les mêmes et le problème aborde certains aspects de la communication.

PREMIERE PARTIE

Système RFID à 13.56 MHz en couplage magnétique

I.A Les fréquences utilisées en RFID

I.A1. Pour quelle expérience majeure le physicien Hertz (1857-1894) est-il resté célèbre ?

I.A2. Que signifient bande HF et bande UHF ? Pourquoi dit-on que la fréquence 2,45 GHz appartient aux microondes?

I.A3. Pourquoi les ondes UHF portent plus loin que les ondes HF et les ondes BF ?

I.A4. Pourquoi le pouvoir de pénétration des ondes dans un conducteur augmente quand la fréquence baisse ?

I.A5. Pourquoi tous les systèmes RFID HF utilisent-ils la même fréquence de 13,56 MHz ?

I.B Modélisation d'un système RFID à 13.56 MHz : la carte à puce sans contact.

On considère une bobine b_1 circulaire d'axe (Oz) située dans le plan (Oxy), de rayon a, comprenant N_1 spires identiques et supposées confondues (toutes de même diamètre) réalisées en fil conducteur de diamètre négligeable et parcourue par le courant i_1 sinusoïdal de fréquence f : $i_1 = I_1\sqrt{2}\sin(\omega t)$. *Les dimensions caractéristiques (a, OM,...) sont de l'ordre de 10 cm. On rappelle que :* $\mu_0 = 4\pi \cdot 10^{-7}\,\mathrm{H \cdot m^{-1}}$.

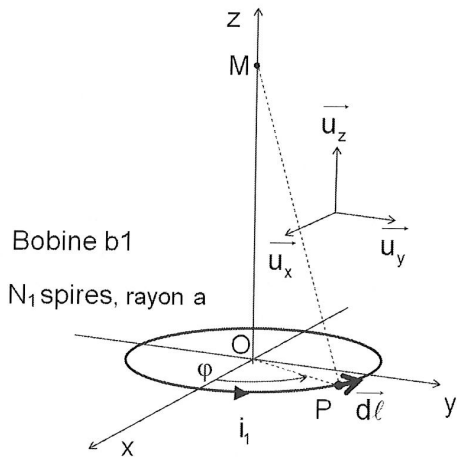


Figure 2

I.B1. Rappeler les conditions d'application de l'approximation des régimes quasi stationnaires (ARQS). Cette approximation s'applique-t-elle ici ?

Soit M un point de l'axe (Oz) tel que $\overrightarrow{OM} = z \cdot \vec{u_z}$ *et* $z \geq 0$.

I.B2. Quelle est la direction du champ magnétique $\vec{B_1}(M,t)$ en M? On note $B_{1eff}(a,z)$ la valeur efficace de la norme de $\vec{B_1}(M,t)$. Pour $I_1 = 20$ mA, on a représenté sur la figure 3 les courbes de niveau de $B_{1eff}(a,z)$ pour $0 \leq a \leq 0.2$ m et pour $0 \leq z \leq 0.2$ m. A partir de ces courbes de niveau et sans calcul supplémentaire représenter B_{1eff} en fonction de a pour $z = z_0$. Préciser les valeurs remarquables.

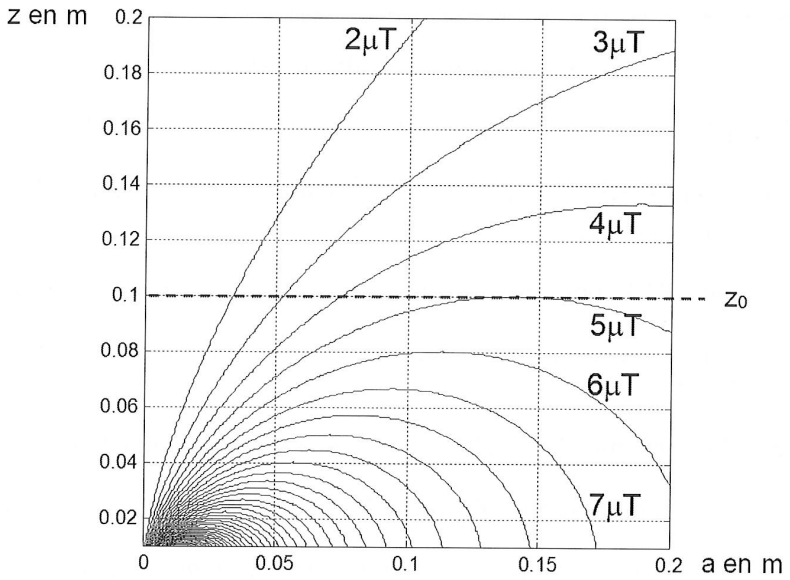


Figure 3

Application numérique : pour N_1= 100 spires, I_1 = 20 mA, $z_0 = 10$ cm déterminer la valeur maximale de B_{1eff}.

I.B3. Donner une expression du flux propre Φ_1 de la bobine b_1 faisant intervenir une intégrale simple. On ne cherchera pas à calculer Φ_1.

I.B4. Rappeler la définition de l'inductance propre L_1 de b_1.

On considère maintenant une bobine b_2 circulaire d'axe (Oz) située dans un plan parallèle au plan (Oxy) à $z = d$, de rayon b et de surface $S_2 = \pi \cdot b^2$, comprenant N_2 spires identiques et confondues (toutes de même diamètre) réalisées en fil conducteur de diamètre négligeable. Les deux bobines b_1 et b_2 sont couplées magnétiquement et forment un transformateur à air. On note L_2 l'inductance propre de b_2 et M l'inductance mutuelle entre b_1 et b_2 ($M = k\sqrt{L_1 L_2} > 0$, où k est le coefficient de couplage entre les deux bobines). La bobine b_2 est connectée à un circuit électrique comportant une résistance R_L en série : c'est le modèle du transpondeur (TAG). La bobine b_1 est alimentée par une tension u_1 et constitue l'antenne associée au lecteur RFID, R_1 et R_2 modélisent les résistances des bobines b_1 et b_2 et R_L modélise la résistance d'entrée de la puce électronique du TAG. En modulant R_L, le TAG communique avec le lecteur. (Le TAG est donc modélisé par L_2, R_2 et R_L qui est variable).

Pour les applications numériques prendre : d = 10 cm, N_1 = 100 spires, N_2 = 4 spires, S_2 = 30 cm^2, L_1 = 40 μH, L_2 = 6.3 μH, R_2 = 10 Ω, R_L = 30 kΩ ; k = 0,63% (soit M = 0,1 μH); f_0 = 13,56 MHz, I_1 = 20 mA.

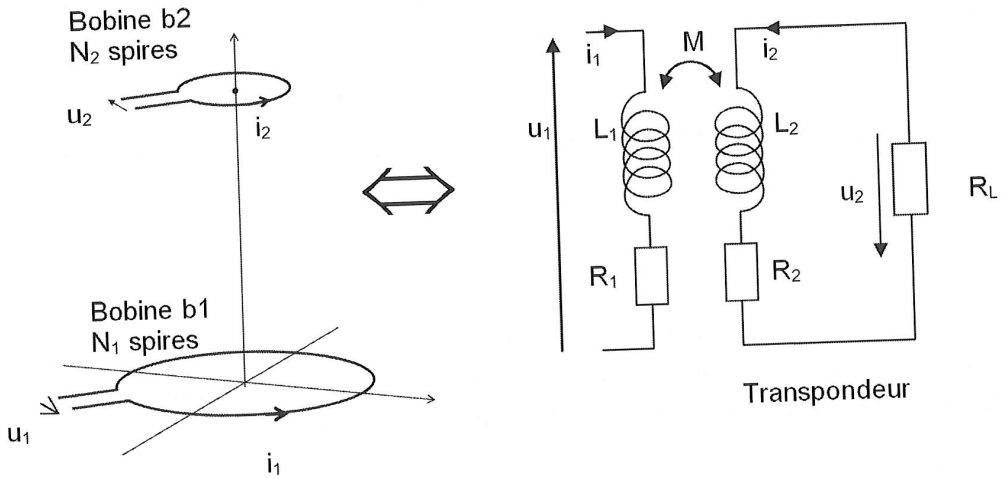


Figure 4

I.B5. Donner une approximation de M en considérant que $b \ll a$ et que $b \ll d$. En exploitant le résultat de la question I.B2, et pour des nombres de spires N_1 et N_2 et une distance d entre les deux bobines fixés, il existe une valeur optimale du rayon a de la bobine b_1 maximisant la mutuelle M. Application numérique : déterminer la valeur optimale de a.

I.B6. Exprimer les tensions u_1 et u_2 en fonction de R_1, R_2, L_1, L_2, M, i_1, i_2, $\frac{di_1}{dt}$ et $\frac{di_2}{dt}$.

Aux grandeurs temporelles, u_1, u_2, i_1 *et* i_2 *on associe les grandeurs complexes* $\underline{U_1}$, $\underline{U_2}$, $\underline{I_1}$ *et* $\underline{I_2}$. *La seule entrée du montage est la tension* u_1 (*ou* $\underline{U_1}$) ; u_2, i_1 *et* i_2 *sont donc des inconnues* (ou $\underline{U_2}$, $\underline{I_1}$ *et* $\underline{I_2}$).

I.B7. Ecrire trois équations liant les tensions complexes $\underline{U_1}$ et $\underline{U_2}$ et les courants $\underline{I_1}$ et $\underline{I_2}$.

I.B8. Mettre $\underline{U_1}$ et $\underline{U_2}$ sous la forme : $\underline{U_1} = \underline{Z_{11}} \cdot \underline{I_1}$ et $\underline{U_2} = \underline{Z_{21}} \cdot \underline{I_1}$. Montrer en particulier que $\underline{Z_{21}} = \frac{-jMR_L\omega}{R_L + R_2 + jL_2\omega}$ et que $\underline{Z_{11}} = a + jb + \frac{(M\omega)^2}{R_L + R_2 + jL_2\omega}$. Préciser les expressions des coefficients réels a et b.
Application numérique : donner l'ordre de grandeur de la tension efficace de u_2.

On insère un condensateur C_2 *en parallèle à* R_L *tel que* $L_2C_2\omega_0^2 = 1$ *et* $\omega_0 = 2\pi f_0$. *On note* $Q_2 = 1/(R_2C_2\omega_0)$ *et* $Q_L = R_L/(L_2\omega_0)$ *; les valeurs numériques correspondantes sont* $C_2 = 22$ pF, $Q_2 = 53{,}3$ *et* $Q_L = 55{,}9$.

I.B9. En négligeant R_2 devant R_L, déterminer la nouvelle expression de l'impédance $\underline{Z'_{21}}$. Mettre sous la forme :

$$\underline{Z'_{21}} \approx \frac{-R_{21}}{1 + jQ_T \cdot \left(\frac{\omega}{\omega_0} - \frac{\omega_0}{\omega}\right)}$$

Exprimer la résistance R_{21} ainsi que le facteur de qualité Q_T équivalent du transpondeur en fonction de Q_L et de Q_2. Conclure sur l'intérêt du condensateur C_2. Application numérique : déterminer les ordres de grandeur des valeurs efficaces de i_2 et de u_2.

Si le champ électromagnétique a une amplitude trop faible, l'énergie reçue par le transpondeur n'est pas suffisante pour un fonctionnement normal. Ainsi, l'alimentation de la puce ne se fait correctement que si la tension vérifie $U_{2eff} > 4{,}6\,V$.

I.B10. En appliquant la loi de Faraday et avec les approximations adéquates, indiquer quel est l'ordre de grandeur de la valeur numérique du champ efficace B_{min} nécessaire au niveau de la bobine b_2.

I.B11. Faire un bilan de puissance sur le schéma de la figure 4 en modifiant R_L par une impédance $\underline{Z_L} = \frac{R_L}{1 + jQ_L\frac{\omega}{\omega_0}}$.

La bobine b_1 est en fait reliée à un condensateur, C_1 tel que $L_1C_1\omega_0^2 = 1$. *L'ensemble est alimenté par une source de tension de fem e_G et de résistance interne R_G (Fig. 5). Pour les applications numériques prendre* $R_G = 50\,\Omega$.

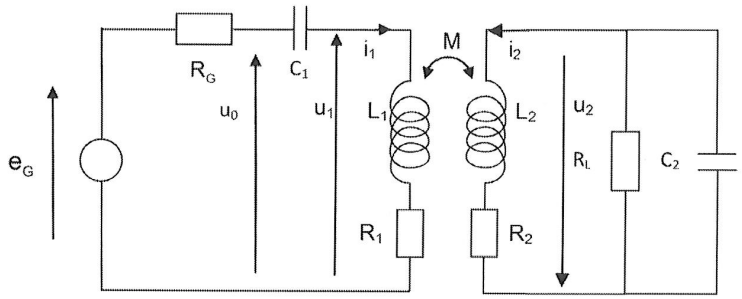


Figure 5

En plus de la valeur 30 kΩ, R_L *peut prendre les deux autres valeurs particulières suivantes :* 0 Ω *et l'infini.*

I.B12. Montrer que pour $\omega = \omega_0$ la tension $\underline{U_0}$ peut s'écrire $\underline{U_0} = \left(R_1 + \underline{Z}(\omega_0)\right)\cdot \underline{I_1}$ où l'impédance $\underline{Z}(\omega_0)$ s'écrit :

$$\underline{Z}(\omega_0) = \frac{(M\omega_0)^2}{\dfrac{R_L}{1 + j\omega_0 R_L C_2} + R_2 + jL_2\omega_0}$$

Application numérique : déterminer l'ordre de grandeur de $\underline{Z}$ (module et argument) lorsque $R_L = 0$ puis lorsque $R_L = \infty$. Montrer comment réaliser en pratique ces deux conditions avec des interrupteurs commandables.

Le transpondeur communique avec la base émettrice en modulant la résistance R_L entre les deux valeurs $R_L = 0$ et $R_L = \infty$.

I.B13. Montrer alors que l'amplitude de la tension u_0 contient l'information délivrée par la puce. Conclusion.

Deuxième partie : Informatique. Modélisation d'un système de titres de transport sans contact

La première partie a permis de montrer comment la puce RFID peut être alimentée à distance par le lecteur, et comment un signal peut être transmis de l'une à l'autre. L'objectif des parties suivantes est de mettre en place quelques algorithmes visant à :

- simuler la récupération d'un message binaire à partir de la tension captée par le lecteur (Partie 1) ;
- contrôler l'intégrité du message récupéré et corriger les erreurs éventuelles (Partie 2),
- déterminer si un voyageur est autorisé ou non à franchir un point de contrôle (Partie 3),
- traiter les informations recueillies par le système afin d'améliorer le service (Partie 4).

Ces quatre parties sont indépendantes.

Les algorithmes demandés seront réalisés, au choix, dans le langage Python ou le langage Scilab. **Le candidat doit précsier en tête de partie le langage choisi et s'y tenir.**On supposera que tout module nécessaire à l'utilisation des fonctions usuelles (pi, sin...) a été importé.

Simulation numérique de la démodulation d'amplitude par le lecteur

La première partie a montré que la puce et le lecteur peuvent tous deux recevoir des messages contenus dans des tensions alternatives d'amplitudes variables. Nous allons maintenant modéliser une telle tension puis simuler numériquement le comportement d'un dispositif permettant d'en extraire une information binaire.

La tension $e(t)$ reçue par le lecteur peut être modélisée par une sinusoïde de fréquence $f = 13{,}56$ MHz dont l'amplitude E_0 est modulée par le signal binaire $b(t)$ transmis par la puce :

$$e(t) = E_0(t)\sin(2\pi f t) \text{ avec } \begin{cases} E_0(t) = E_{\min} = 1{,}3 \text{ V lorsque } b(t) = 0 \\ E_0(t) = E_{\max} = 1{,}5 \text{ V lorsque } b(t) = 1 \end{cases}$$

La puce transmet un nouveau bit toutes les 16 périodes de la porteuse. A titre d'illustration, la **Figure 1** représente la tension $e(t)$ correspondant à la séquence de bits (0, 1, 0). La tension est en volts et le temps en secondes.

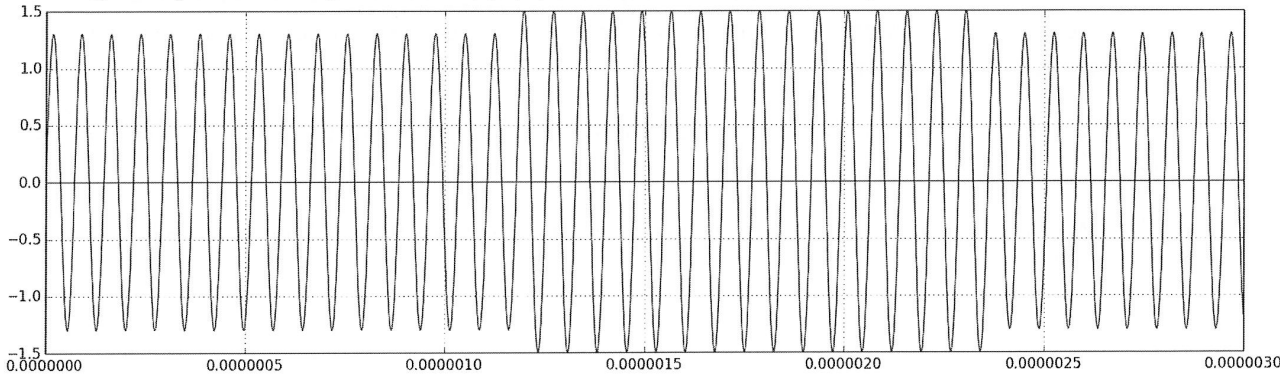


Figure 1 : tension d'entrée du démodulateur (sinusoïde modulée en amplitude par un signal binaire)

La simulation porte sur un intervalle de temps $[0, T_{\max}]$. Le temps sera représenté numériquement par une liste de N instants régulièrement espacés que l'on écrira, compte tenu des conventions de numérotation propres à chaque langage, T = $[t_0=0, t_1, ..., t_{N-1}=T_{\max}]$ sous Python ou T = $[t_1=0, ...t_N=T_{\max}]$ sous Scilab.

Q1. Écrire une fonction `init_T(Tmax,dt)` prenant pour arguments la durée `Tmax` de la simulation et le pas de temps `dt` et retournant la liste `T`. Si `Tmax` n'est pas multiple de `dt`, on arrondira la durée de la simulation au multiple entier de dt immédiatement supérieur.

La tension d'entrée $e(t)$ sera de même représentée par une liste `E` telle que `E[i]`$=e(t_i)$.

Q2. Écrire une fonction `init_E(T,f)` prenant pour arguments la liste `T` des instants de la simulation et la fréquence `f` de la porteuse et retournant la liste des valeurs $e(t_i)$ de la tension e aux instants `T[i]`, d'après la définition de $e(t)$ et pour le message binaire (0,1,0) (on considèrera que la simulation ne se poursuit pas au-delà).

Pour récupérer l'information binaire contenue dans une telle tension, il faut en extraire l'amplitude. On utilise pour cela le dispositif de la **Figure 2**, appelé **détecteur d'enveloppe**.

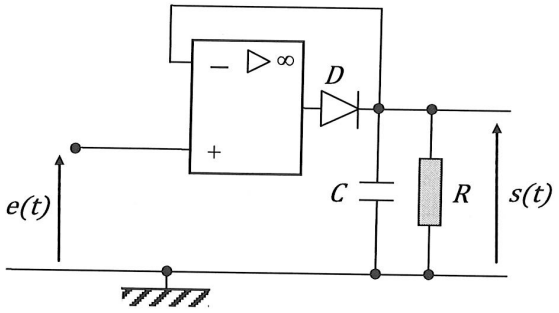


Figure 2 : détecteur d'enveloppe

La modélisation du comportement de ce montage permet d'exprimer la tension de sortie $s(t)$ en fonction de la tension d'entrée $e(t)$ de la façon suivante (τ est la constante de temps du circuit RC) :

1. Si la diode est *bloquée*, alors $\frac{\mathrm{d}s}{\mathrm{d}t}(t) + \frac{1}{\tau}s(t) = 0$ et il faut vérifier que $s(t) > e(t)$
2. Si la diode est *passante*, alors $s(t) = e(t)$ et il faut vérifier que $\frac{\mathrm{d}s}{\mathrm{d}t}(t) + \frac{1}{\tau}s(t) > 0$

Autrement dit, la diode peut prendre deux états (bloquée ou passante), chacun soumis à une condition (inégalité) et muni d'une équation d'évolution (égalité) ; lorsque la condition cesse d'être vérifiée, la diode change d'état. Nous allons utiliser ces équations pour simuler numériquement l'évolution de la tension $s(t)$. Pour cela, on propose d'utiliser une variable (par exemple booléenne) indiquant l'état de la diode et, à chaque pas de temps t_i :

- de calculer $s(t_{i+1})$ en utilisant l'équation correspondant à la valeur de la variable à l'instant t_i,
- *puis* de tester la condition correspondante à partir de la valeur de $s(t_{i+1})$ et, si elle n'est plus vérifiée, de mettre à jour la variable.

Le résultat est stocké dans une liste `S` avec `S[i]` $= s(t_i)$.

Q3. Donner une approximation de $\frac{\mathrm{d}s}{\mathrm{d}t}(t_i)$ en fonction de $s(t_i)$, $s(t_{i+1})$ et $\Delta t = t_{i+1} - t_i$ en utilisant la formule d'Euler explicite. En déduire, dans le cas où la diode est *bloquée* à l'instant t_i, la relation de récurrence donnant $s(t_{i+1})$ en fonction de $s(t_i)$, τ et Δt.

Dans le cas où la diode est *passante* à l'instant t_i, on ne peut pas utiliser cette formule pour tester l'état de la diode à t_{i+1}. On utilise donc l'approximation *d'Euler arrière* ou *Euler implicite*, qui consiste à utiliser la même formule mais pour approcher $\frac{\mathrm{d}s}{\mathrm{d}t}(t_{i+1})$ au lieu de $\frac{\mathrm{d}s}{\mathrm{d}t}(t_i)$.

Q4. Pourquoi ne peut-on pas utiliser la formule d'Euler explicite pour effectuer le test ici ? Donner, en utilisant la démarche proposée, une condition portant sur $s(t_{i+1})$, $s(t_i)$, τ et Δt permettant de déterminer si la diode se bloque ou non à l'instant t_{i+1}.

Q5. Écrire alors une fonction `solve(T,E,tau)` prenant pour arguments la liste `T` des instants de la simulation, la liste `E` des tensions d'entrée et la constante de temps `tau` et retournant la liste `S` des tensions de sortie. Les conditions initiales seront prises nulles et, si nécessaire, l'état initial de la diode sera supposé passant.

Les **Figures 3, 4 et 5** donnent les résultats (entrées et sorties numériques) obtenus pour trois pas de temps différents choisis parmi 1 ns, 10 ns et 100 ns. Sur ces trois graphes, la tension d'entrée *avant discrétisation* est celle de la **Figure 1** et la constante de temps du circuit est $\tau = 1$ µs. Seul le pas de temps change d'une simulation à l'autre.

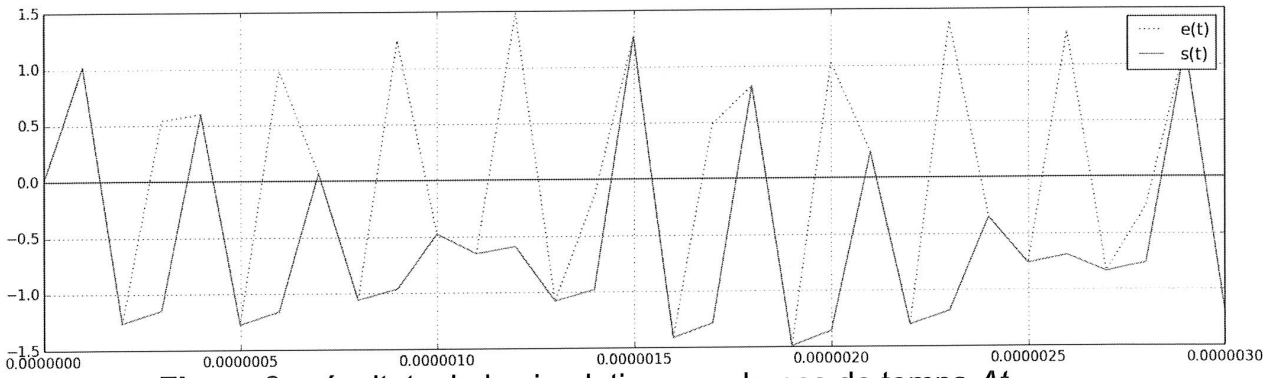


Figure 3 : résultats de la simulation pour le pas de temps Δt_1

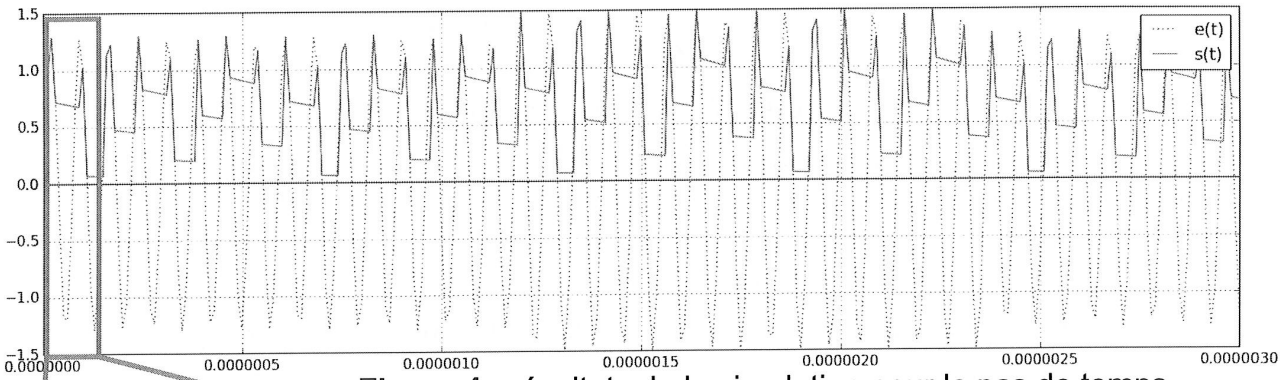
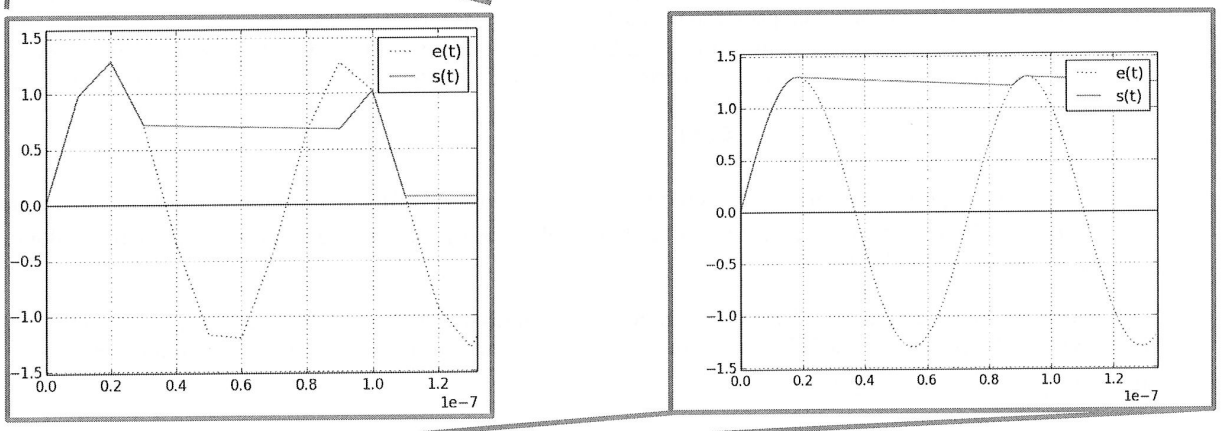


Figure 4 : résultats de la simulation pour le pas de temps Δt_2 (avec zoom à l'origine)



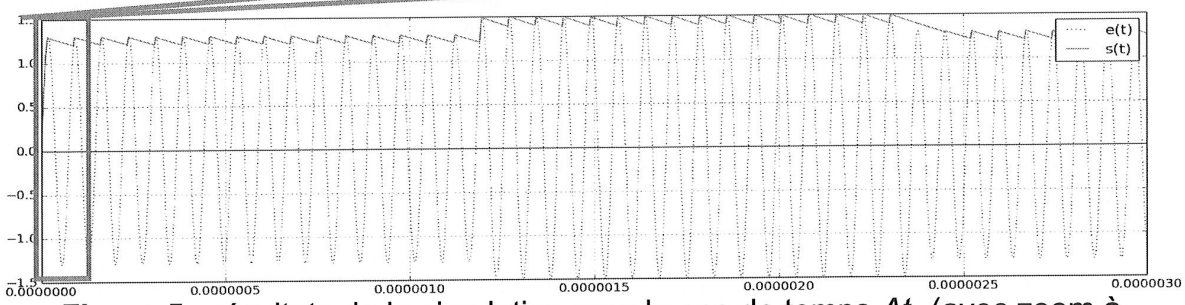


Figure 5 : résultats de la simulation pour le pas de temps Δt_3 (avec zoom à l'origine)

Q6. **Indiquer la valeur du pas de temps (1, 10 ou 100 ns) correspondant à chacune de ces trois simulations, en justifiant vos réponses.**

Q7. Expliquer en quelques phrases les causes des différences obtenues entre les trois résultats $s(t)$. Repérer en particulier les instants auxquels la diode change d'état. Que constatez-vous ?

Pour distinguer l'état "haut" de l'état "bas" de la tension $s(t)$ et ainsi extraire les 0 et les 1 du message binaire transmis par modulation, il est nécessaire de déterminer un seuil séparant les deux niveaux.

Q8. Indiquer, pour chacun des trois résultats, s'il est possible d'identifier un tel seuil, et donc si la récupération du message binaire semble réalisable *si l'on se base uniquement sur ce résultat*. Conclure sur le critère que doit respecter le pas de temps d'une simulation temporelle pour que les résultats de celle-ci aient une chance d'être pertinents.

Vérification de l'intégrité des données et correction des erreurs

Le signal transmis par une liaison RFID 13,56 MHz peut être perturbé par toutes sortes de facteurs pouvant provoquer des erreurs dans les données : autres signaux électromagnétiques, masses métalliques, imperfections du matériel électronique... En pratique, il est donc indispensable de pouvoir détecter ces erreurs et, dans la mesure du possible, les corriger sans que cela ne nécessite une nouvelle transmission ; l'objet de cette partie est de mettre en place quelques algorithmes dans ce but.

2.1. Bit de parité

Une technique simple et très répandue pour s'assurer qu'une donnée binaire sera lue correctement par son récepteur est de lui adjoindre un **bit de parité**, égal par définition à :

- 0 si la donnée contient un nombre pair de 1 (et, donc, si ses bits sont de somme paire),
- 1 si la donnée contient un nombre impair de 1 (et, donc, si ses bits sont de somme impaire).

Après réception de la donnée, le récepteur recalcule le bit de parité et le compare à celui que l'émetteur lui a adressé. Si la donnée n'a pas été altérée lors de la transmission, alors les deux bits de parité sont forcément identiques.

Q9. Donner les bits de parité associés aux représentations binaires des entiers 5, 16 et 37.

Q10. Écrire une fonction `parite(bits)` prenant pour argument une liste `bits` constituée d'entiers valant 0 ou 1 et retournant l'entier 0 ou 1 correspondant à son bit de parité.

Les techniques de vérification les plus simples consistent à découper la donnée en blocs et à joindre un bit de parité à chaque bloc. Par exemple, certains protocoles transmettent sept bits de données pour un bit de parité.

Q11. Donner un exemple d'erreur n'étant pas détectable par cette technique. Si une erreur a été détectée, est-il possible de la corriger sans retransmettre la donnée ?

2.2. Code de Hamming

Le code de Hamming est un exemple d'utilisation des bits de parité pour détecter et corriger des erreurs. Nous nous intéressons ici au code dit (7,4), ainsi appelé car il consiste à joindre trois bits de parité à quatre bits de données, ce qui donne un message d'une longueur totale de sept bits. Ces trois bits de parité sont définis ainsi : si la donnée s'écrit (d_1, d_2, d_3, d_4) avec $d_i = 0$ ou 1, alors :

- p_1 est le bit de parité du triplet (d_1, d_2, d_4),
- p_2 est le bit de parité du triplet (d_1, d_3, d_4),
- p_3 est le bit de parité du triplet (d_2, d_3, d_4).

Le message encodé, que l'on transmet, s'écrit alors comme suit : (p_1, p_2, d_1, p_3, d_2, d_3, d_4).

Q12. Écrire une fonction `encode_hamming(donnee)` prenant pour argument une liste `donnee` de quatre bits (représentés par des entiers valant 0 ou 1) et retournant une liste de bits contenant le message encodé. On pourra appeler la fonction `parite(bits)` précédemment définie.

Le contrôle après réception d'un message ainsi encodé est relativement simple. On pourrait naturellement recalculer les trois bits de parité de la donnée et les comparer aux valeurs transmises, mais la technique proposée par Hamming est de calculer les trois *bits de contrôle* suivants, notés (c_1, c_2, c_3), à partir du *message complet* (données et bits supplémentaires), noté (m_1, ..., m_7) :

- c_1 est le bit de parité de l'ensemble (m_4, m_5, m_6, m_7),
- c_2 est le bit de parité de l'ensemble (m_2, m_3, m_6, m_7),
- c_3 est le bit de parité de l'ensemble (m_1, m_3, m_5, m_7).

On montre que si le message a bien été encodé selon les règles précédentes et n'a pas été altéré, alors les trois bits de contrôle doivent être à 0. Si ce n'est pas le cas, alors il y a eu une erreur ; l'intérêt de la technique de Hamming est que

dans le cas particulier où l'erreur est unique, le mot de contrôle donne la représentation binaire de la position de cette erreur en numérotant à partir de 1. Par exemple, si $(c_1, c_2, c_3) = (0,1,1)$, alors l'erreur porte sur le troisième bit du message. Il suffit ainsi d'inverser ce bit (le mettre à 1 s'il est à 0, et inversement) pour corriger l'erreur.

La donnée décodée est alors constituée des quatre bits (d_1, d_2, d_3, d_4) qui se trouvent respectivement en positions 3, 5, 6 et 7 (toujours en numérotant à partir de 1) conformément à la description de l'encodage donnée ci-dessus.

Q13. Écrire une fonction `decode_hamming(message)` prenant pour argument une liste de sept bits et retournant une liste de quatre bits contenant la donnée décodée. En cas d'erreur, on affichera à l'écran un avertissement indiquant la position du bit affecté et on effectuera la correction. On supposera dans cette question que s'il y a une erreur, alors elle est unique.

Q14. Déterminer le codage de Hamming de la donnée `1011`, puis la donnée décodée par l'algorithme dans l'hypothèse où les deux premiers bits du message codé ont été incorrectement transmis. Quel a été l'effet de la "correction" sur la donnée dans ce cas ?

Q15. Sans coder, proposer un moyen simple de différencier une double erreur d'une erreur unique au moyen d'un bit de parité supplémentaire et expliquer comment cela permet d'éviter le problème mis en évidence à la question précédente. On s'appuiera sur les techniques introduites dans cette partie. On ne demande pas d'essayer de corriger la double erreur.

Utilisation des données de la puce pour autoriser ou non le passage

Lorsqu'un utilisateur présente son titre de transport face au lecteur d'un point de contrôle, la puce et le lecteur s'identifient mutuellement, puis le lecteur récupère les données de la puce permettant de déterminer si le passage est autorisé ou non. A l'issue de cette récupération, l'ordinateur auquel est relié le lecteur dispose d'un fichier texte `0001.txt` semblable à l'exemple ci-dessous :

```
49987654
1, 3, 2015-08-31
2014-10-29, 08:34:15, 4568
2014-10-28, 20:21:48, 365
2014-10-28, 18:47:54, 987
```

(identifiant du titre de transport)
(première et dernière zone de validité, date de fin de validité)
(dates, heures et identifiants des lieux des trois derniers passages)

En d'autres termes, ce fichier possède toujours *exactement* la structure suivante :

- la première ligne contient un entier servant à identifier le titre de transport,
- le réseau de transport est divisé en plusieurs zones numérotées, et le titre n'est valide que dans un ensemble de zones contigües ; la seconde ligne contient les bornes de l'intervalle dans lequel le titre est valide (*ici, il s'agit des zones 1 à 3 incluses*) ainsi que la date de fin de validité du titre au format *aaaa-mm-jj* (*ici, il s'agit du 31 août 2015*),
- les trois lignes suivantes contiennent des données relatives aux trois derniers passages effectués à l'aide du titre : date, heure (au format *hh:mm:ss* sur 24 heures) et identifiant entier du point de passage (gare, arrêt...).

On donne le bloc d'instructions utilisé pour lire les deux premières lignes de ce fichier. On précise qu'il n'est pas nécessaire de connaître toutes les syntaxes relatives à la manipulation des chaînes pour traiter les questions suivantes.

sous Python

```python
fichier = open('0001.txt')
lignes = fichier.readlines()
fichier.close()

# Ligne 1: recuperation de l'identifiant du titre
id_titre = int(lignes[0])

# Ligne 2 : recuperation des donnees du titre de transport
donnees_titre = lignes[1].rstrip('\n').split(',')
zones = [ int(donnees_titre[0]), int(donnees_titre[1]) ]
ch_date_fin = donnees_titre[2].split('-')
```

```
date_fin = [ int(ch_date_fin[0]), int(ch_date_fin[1]), int(ch_date_fin[2]) ]
```

Sous Scilab

```
fichier = mopen('0001.txt')
lignes = mgetl(fichier)
mclose(fichier)

// Ligne 1: recuperation de l'identifiant du titre
id_titre = strtod(lignes(1))

// Ligne 2 : recuperation des donnees du titre de transport
donnees_titre = strsplit(lignes(2),',')
zones = [ strtod(donnees_titre(1)), strtod(donnees_titre(2)) ]
ch_date_fin = strsplit(donnees_titre(3),'-')
date_fin = [ strtod(ch_date_fin(1)), strtod(ch_date_fin(2)), strtod(ch_date_fin(3)) ]
```

Q16. Donner les types et les valeurs des variables `id_titre`, `zones` et `date_fin` à l'issue de ces instructions pour le fichier `0001.txt` donné ci-dessus.

On souhaite placer le contenu des trois dernières lignes dans un tableau d'entiers nommé **`passages`**, dont chaque ligne corresponde à un passage (dans l'ordre dans lequel ils apparaissent dans le fichier) et dont les colonnes soient définies comme suit (le premier chiffre donné est l'indice sous Python, le deuxième est l'indice sous Scilab) :

Indice	0/1	1/2	2/3	3/4	4/5	5/6	6/7
Contenu	Année	Mois	Jour	Heures	Minutes	Secondes	Point de passage

Q17. Écrire le bloc d'instructions à exécuter à la suite des opérations précédentes pour construire le tableau `passages` à partir des lignes 3, 4 et 5 contenues dans la liste `lignes`.

Les données relatives au lecteur sont décrites par les variables suivantes (*on utilise une initiale majuscule pour bien différencier les données du lecteur de celles du titre ; le type est indiqué entre parenthèses*) :

- **`Zone`** (*entier*) : indique la zone dans laquelle se trouve le lecteur,
- **`Id_Point`** (*entier*) : indique l'identifiant du point de passage où se trouve le lecteur,
- **`Liste_noire`** (*liste d'entiers*) : contient les identifiants des titres ayant été déclarés perdus, volés ou détériorés par leurs propriétaires, et devant donc être refusés,
- **`Maintenant`** (*liste de six entiers*) : contient la date et l'heure au format ci-dessus [année, mois, jour, heures, minutes, secondes].

Le passage doit être autorisé si les conditions suivantes sont *toutes* vérifiées :

- l'identifiant du titre n'est pas dans la liste noire du lecteur,
- la zone du lecteur appartient à l'intervalle de validité du titre,
- la date du jour est antérieure à la date de fin de validité du titre,
- si l'une des trois dernières validations a été effectuée au même point de passage que celui où est installé le lecteur, elle doit avoir été effectuée il y a plus de 450 secondes (ceci afin de décourager l'utilisation frauduleuse d'un même titre par plusieurs voyageurs).

Enfin, lorsqu'un passage est refusé, un message apparaît sur un afficheur LCD pour donner la raison du refus. Cet afficheur possède une seule ligne et il faut donc définir des priorités au cas où plusieurs des conditions ci-dessus ne seraient pas remplies. L'ordre de priorité et les messages correspondants sont donnés ci-dessous :

1. `"Titre refusé"` si l'identifiant est dans la liste noire,
2. `"Non valide dans cette zone"` si le lecteur est hors des zones de validité du titre,
3. `"Titre expiré"` si la date de fin de validité du titre est dépassée,
4. `"Titre déjà validé"` si le titre a déjà été validé dans le même lieu il y a moins de 450 secondes.

Q18. Écrire une fonction `estAvant(date1,date2)` prenant pour arguments deux dates au format `[année, mois, jour]` (donc sous forme de listes de trois entiers chacune) et retournant `True` si `date1` est antérieure ou égale à `date2`, et `False` sinon.

Q19. Écrire une fonction `nbSecondesEntre(heure1,heure2)` prenant pour arguments deux horaires au format `[heures, minutes, secondes]` (donc sous forme de listes de trois entiers chacun) et retournant le nombre de secondes séparant les deux instants. Le résultat devra être positif si heure1 est *postérieure* à heure2.

Q20. Écrire alors une fonction `testPassage`, dont les arguments sont à préciser, retournant la valeur `True` si le passage est autorisé et la valeur `False` sinon et, dans ce dernier cas, affichant à l'écran le message correspondant aux règles de priorité ci-dessus. On pourra utiliser toutes les variables définies dans cette partie et appeler les fonctions définies dans les deux questions précédentes.

Exploitation des données enregistrées par le système

A chaque fois qu'un point de contrôle est franchi, le système collecte des données relatives au passage : lieu, date, heure... Ces données sont ensuite enregistrées dans la base de données du transporteur, qui comporte trois tables :

- la table **`passages`** dédiée aux passages des voyageurs, constituée des champs :
  - **`date`** qui contient la date du passage au format *aaaa-mm-jj*,
  - **`heure`** qui contient l'heure du passage au format *hh:mm:ss*,
  - **`id_point`** qui est l'identifiant (entier) du point de passage,
  - **`id_titre`** qui est l'identifiant (entier) du titre de transport utilisé ;

- la table **`points`** dédiée aux points de passage, constituée des champs (entiers) :
  - **`id`** qui est l'identifiant du point de passage (clé primaire),
  - **`zone`** qui est le numéro de la zone où se trouve le point de passage,
  - **`ligne`** qui est le numéro de la ligne sur laquelle se trouve le point de passage ;

- et la table **`titres`** dédiée aux titres de transport, constituée des champs (entiers) :
  - **`id`** qui est l'identifiant du titre de transport (clé primaire),
  - **`zone_min`** qui est la plus petite zone couverte par le titre,
  - **`zone_max`** qui est la plus grande zone couverte par le titre.

Ces informations sont utilisées par le transporteur pour effectuer des études statistiques sur la fréquentation de ses lignes, en vue d'améliorer le service.
Par exemple, certaines lignes desservant majoritairement des zones d'activités ou des établissements d'enseignement connaissent une forte baisse de leur fréquentation en été, ce qui permet d'alléger le service ; pour choisir la période concernée, il faut connaître précisément l'évolution de la fréquentation au cours de l'été, ainsi que sa répartition au cours de la journée.

Q21. Donner la requête SQL permettant de récupérer les dates et les heures de tous les passages ayant eu lieu sur la ligne numérotée 1 entre le 1er juillet et le 31 août 2014 (inclus).

Un autre exemple de problématique est celui de l'efficacité des dézonages : à certaines périodes de l'année, les voyageurs sont autorisés à emprunter l'ensemble du réseau de transport quelles que soient les zones de validité de leur abonnement. Pour évaluer l'efficacité d'une telle mesure, il faut connaître le nombre de trajets en ayant bénéficié.

Q22. Donner la requête SQL permettant de compter le nombre de passages dézonés, c'est-à-dire ayant eu lieu hors de l'intervalle de validité du titre utilisé, effectués le 31 décembre 2014.

Question 1 :
Parmi les affirmations suivantes lesquelles sont vraies :

A) En informatique la mémoire est un dispositif électronique qui sert à stocker des informations.

B) La mémoire du disque dur doit pouvoir fonctionner en mode lecture mais pas en mode écriture pour ne pas être détériorée.

C) La mémoire du disque dur doit pouvoir fonctionner en mode écriture mais pas en mode lecture pour des raisons de sécurité informatique.

D) Dans la mémoire vive d'un ordinateur sont stockées définitivement des données importantes.

Question 2 :
Parmi les affirmations suivantes lesquelles sont vraies :

A) Une machine à calculer peut stocker une infinité de données puisqu'elle peut par exemple tracer la courbe représentative d'une fonction.

B) Une machine à calculer, aussi performante soit-elle, ne peut contenir qu'un nombre fini de données.

C) Tout nombre entier peut être représenté par une suite finie de 0 et de 1.

D) Tout nombre réel peut être représenté par une suite finie de 0 et de 1.

Question 3 :
Parmi les affirmations suivantes lesquelles sont vraies :

A) Un nombre entier naturel qui est représenté en binaire par une suite de 0 et de 1 et qui se termine par 1 est pair.

B) Le nombre décimal 0,1 possède une représentation binaire finie.

C) Sur un octet de mémoire le plus grand entier naturel représentable par une suite de 0 ou de 1 est 255.

D) 110 en binaire représente 10 en base dix.

Question 4 :
Parmi les scripts suivant lesquels échangent les valeurs de a et b ?

A)	B)	C)	D)
c=a	c=a	a=a+b	a=a+b
b=c	b=c	b=a-b	b=a-b
c=a	a=b	a=a-b	a=a+b

Question 5 :

On définit par le script suivant une fonction ORDONNE :

```
def ORDONNE(L):
    n=len(L)
    for i in range(0,n-1):
        if L[i]>L[i+1]:
            c=L[i]
            L[i]=L[i+1]
            L[i+1]=c
    return L
```

Parmi les assertions suivantes lesquelles sont vraies :

A) ORDONNE([1,3,2,5,4]) renvoie [1,2,3,4,5].

B) Si L est une liste de nombres ORDONNE(L) renvoie une liste de nombres réordonnée dans l'ordre croissant.

C) ORDONNE([1,3,5,2,4]) renvoie [1,2,4,3,5].

D) ORDONNE(ORDONNE([3,2,1])) renvoie [1,2,3].

Question 6 :

L est une liste de n entiers quelconques. Parmi les assertions suivantes lesquelles sont vraies :

A) Pour ordonner L, il suffit d'utiliser n-1 fois la fonction ORDONNE.

B) Pour ordonner L, il suffit d'utiliser n fois la fonction ORDONNE.

C) Une liste L ordonnée dans le sens décroissant n'est pas modifiée par la fonction ORDONNE.

D) Une liste L ordonnée dans le sens croissant est un invariant pour la fonction ORDONNE.

NB : La fonction ORDONNE est sauvegardée dans un fichier nommé ORDONNE.py

Question 7 :

On définit par le script suivant une fonction Ordonnum :

```
def Ordonnum(L):
    from ORDONNE import ORDONNE
    n=len(L)
    for i in range(0,n-1):
        L=ORDONNE(L)
    return L
```

L est une liste de n entiers quelconques. Parmi les assertions suivantes lesquelles sont vraies :

A) Ordonnum est une fonction dont l'argument est un réel.

B) Ordonnum est une fonction dont la valeur de sortie est un réel.

C) Ordonnum(L) renvoie True si L est ordonnée dans le sens croissant.

D) Ordonnum(L) renvoie les éléments de la liste L ordonnés dans le sens croissant.

Question 8 :

On définit par le script suivant une fonction Ordonnam :

```
def Ordonnam(L):
    from ORDONNE import ORDONNE
    n=len(L)
    P=list(L)
    for i in range(0,n-1):
        L=ORDONNE(L)
    test=True
    for i in range(0,n-1):
        if L[i]-P[i]!=0:
                test=False
    return test
```

L est une liste de n entiers quelconques. Parmi les assertions suivantes lesquelles sont vraies :

A) Ordonnam est une fonction dont l'argument est un booléen.

B) Ordonnam est une fonction dont la valeur de sortie est un booléen.

C) Ordonnam est une fonction qui dit si oui ou non une liste est ordonnée dans le sens croissant.

D) Ordonnam(L) renvoie les éléments de la liste L ordonnés dans le sens croissant.

Question 9 :

On définit par le script suivant une fonction Ordonnim :

```
def Ordonnim(L):
    from ORDONNE import ORDONNE
    n=len(L)
    P=L
    for i in range(0,n-1):
        L=ORDONNE(L)
    test=True
    for i in range(0,n-1):
        if L[i]-P[i]!=0:
                test=False
    return test
```

L est une liste de n entiers quelconques. Parmi les assertions suivantes lesquelles sont vraies :

A) Ordonnim est une fonction qui a la même action que Ordonnam.

B) Ordonnim renvoie toujours True.

C) Dans Ordonnim la variable locale P est constamment égale à L lors de l'exécution de la fonction.

D) Ordonnim(L) renvoie les éléments de la liste L ordonnés dans le sens croissant.

Question 10 :

Parmi les assertions suivantes lesquelles sont vraies :

A) La complexité d'un algorithme est une notion qui permet de quantifier la difficulté à concevoir cet algorithme.

B) La complexité d'une algorithme est une notion qui permet de quantifier le coût de cet algorithme tant en terme de temps d'exécution qu'en terme de place mémoire utilisée pendant l'exécution en fonction du nombre et de la taille des données du problème qu'on veut traiter.

C) Deux algorithmes de complexités différentes aboutissent nécessairement à deux résultats différents.

D) Pour un même résultat il existe des algorithmes de complexités différentes. Celui qui est le plus efficace est celui dont la complexité est la plus élevée pour un nombre de données fixé.

Question 11 :

On définit par le script suivant une fonction Ordonnom :
Parmi les assertions suivantes lesquelles sont vraies :

```
def Ordonnom(L):
    for i,s in enumerate(L):
        j=i
        while 0<j and s<L[j-1]:
            L[j]=L[j-1]
            j=j-1
        L[j]=s
    return L
```

A) Ordonnom et Ordonnum sont deux fonctions qui ont le même résultat de sortie si on les applique à une même liste.

B) Ordonnom et Ordonnum sont deux algorithmes qui n'ont pas toujours le même résultat si on les applique à une même liste.

C) La complexité dans le cas le pire de Ordonnom est en $O(n^3)$ où n désigne la taille de la liste L.

D) La complexité dans le cas le pire de Ordonnom est en $O(n)$ où n désigne la taille de la liste L.

Question 12 :

Richard joue avec un dé. Il lance le dé jusqu'à obtenir un 6. On souhaite conserver tous les résultats des différents lancés dans une variable, y compris le 6 qui clôt l'expérience.
Parmi les assertions suivantes lesquelles sont vraies ?

A) Je vais utiliser une structure de pile car je ne connais pas le nombre de données à mémoriser avant d'avoir fini.

B) Je vais choisir une structure de tableau car je vais pouvoir accéder en une instruction à toutes valeurs stockées et si je choisis pour ce tableau une taille suffisament grande je suis certain que ça suffira.

C) En pseudo code le script suivant répond à ma demande

```
Creer RESULTATS
  Lancer=0
  Tant que RESULTATS vide ou Lancer<>6 faire
    Lire(Lancer)  #LA machine attend la saisie du resultat du de
    Empiler(Lancer, RESULTATS)
  fin Tant que
  Empiler(Lancer,RESULTATS)
```

D) En pseudo code le script suivant répond à ma demande

```
Creer RESULTATS
  Lancer=0
  Tant que RESULTATS vide et Lancer<>6 faire
    Lire(Lancer)  #La machine attend la saisie du resultat du de
    Empiler(Lancer, RESULTATS)
  fin Tant que
  Empliler(Lancer, RESULTATS)
```

Question 13 :

Richard parie avec Julie que la somme des lancers qu'il va obtenir avant de faire 6 va être au moins de 100.
Parmi les différentes versions de la fonction gain suivantes lesquelles pourront-ils utiliser pour savoir qui a gagné ?

A)
```
def gain(resultat):
    S=0
    while resultat!=[]:
        S=S+resultat.pop()
    if S<100:
        return('Richard gagne')
    else:
        return('Julie gagne')
```

B)
```
def gain(resultat):
    S=0
    while resultat!=[]:
        S=S+resultat.pop()
    if S>=100:
        return('Richard gagne')
    else:
        return('Julie gagne')
```

C)
```
def gain(resultat):
    S=0
    while resultat!=[]:
        S=S+resultat.append()
    if S>=100:
        return('Richard gagne')
    else:
        return('Julie gagne')
```

D)
```
def gain(resultat):
    S=0
    while resultat!=[]:
        S=S+resultat.append()
    if S<100:
        return('Richard gagne')
    else:
        return('Julie gagne')
```

Question 14 :

Antoine propose de parier avec Richard en utilisant la fonction suivante :

```
def gagne(resultat):
    Score1=0
    Score2=0
    while resultat!=[]:
        p=resultat.pop()
        if p==1:
            score1=score1+1
        else
            score2=score2+1

    if score1>score2:
        return('Richard gagne')
    else:
        return('Antoine gagne')
```

En utilisant cette fonction gagne, parmi les assertions suivantes lesquelles sont justes :

A) Antoine gagne si le nombre de 1 obtenus par Richard est strictement supérieur au nombre de 2.

B) Antoine gagne si le nombre de 1 obtenus par Richard est strictement inférieur au nombre de 2.

C) Antoine gagne si Richard ne fait pas de 1.

D) Antoine gagne si Richard obtient une suite de 1 au moins de longueur 2.

Question 15 :

Pour les questions 15 à 18 on considère une base de données utilisée dans l'agence de location immobilière CHEZ MOI à Toulouse. Cette base de données contient les deux tables suivantes :

APPARTEMENTS(<u>APT_ID</u>,APT_PRO,APT_VILLE,APT_TARIF,APT_SURF)

PROPRIETAIRES(<u>PRO_ID</u>,PRO_NOM,PRO_PRENOM,PRO_ADRESSE,PRO_T

La première regroupe les données sur les appartements : leur identifiant, l´ identifiant de leur propriétaire, la ville, le montant du loyer et la surface.
La deuxième regroupe les données sur les propriétaires des appartements : leur identifiant, le nom, le prénom, leur adresse et le numéro de téléphone.
Les clés primaires sont soulignées.
A quoi servent ces clés ?

A) A enregistrer toutes les données.

B) A effectuer les mises à jour.

C) A distinguer chaque enregistrement de manière unique.

D) A dupliquer plusieurs lignes d'une table.

Question 16 :

Que fait la requête SQL suivante :
SELECT * FROM APPARTEMENTS WHERE APT_SURF>40 ;

A) Elle sélectionne tous les champs de la table APPARTEMENTS.

B) Elle sélectionne les données de la table APPARTEMENTS concernant les appartements dont la surface est strictement supérieure à 40.

C) Elle compte le nombre d'appartement de surface supérieure à 40 dans la table APPARTEMENTS.

D) Elle n'est pas valide.

Question 17 :

Lesquelles des requêtes suivantes permettent d'obtenir l'identifiant des propriétaires des appartements dont le loyer est le plus élevé ?

A) SELECT APT_PRO FROM APPARTEMENTS WHERE APT_TARIF = (SELECT MAX(APT_TARIF) FROM APPARTEMENTS) ;

B) SELECT APT_PRO FROM APPARTEMENTS WHERE MAX(APT_TARIF) ;

C) SELECT APT_PRO FROM APPARTEMENTS WHERE APT_TARIF = (SELECT MAX(APT_TARIF) FROM PROPIETAIRES) ;

D) SELECT APT_PRO FROM PROPRIETAIRES WITH MAX(APT_TARIF) ;

Question 18 :

Quel opérateur est le plus adapté pour obtenir le numéro de téléphone des propriétaires des appartements dont le loyer est le plus élevé.

A) Une jointure.

B) Une division cartésienne.

C) Une intersection.

D) Il est impossible d'obtenir ces données car les tables contenant les numéros de téléphones et le montant des loyers sont distinctes.

Question 19 :

On considère le problème de cauchy : $\begin{cases} \frac{dy}{dt}(t) = y^3(t) \\ y(0) = 1 \end{cases}$ pour $t \in [0,1]$. On décide de calculer de manière approchée par une méthode d'Euler la solution. Soit n un entier non nul on pose $y_k = y(\frac{k}{n})$. Parmi les assertions suivantes, lesquelles correspondent bien à un schéma d'Euler explicite pour le problème de Cauchy posé :

A) $y_0 = 1$ et $y_{k+1} = y_k + n.y_k^3, \forall k \in [|0, n-1|]$.
B) $y_0 = 1$ et $y_{k+1} = ny_k + y_k^3, \forall k \in [|0, n-1|]$.
C) $y_0 = 1$ et $y_{k+1} = y_k + \frac{1}{n}.y_k^3, \forall k \in [|0, n-1|]$.
D) $y_0 = 1$ et $y_{k+1} = y_k + n.y_{k+1}^3, \forall k \in [|0, n-1|]$.

Question 20 :

On reprend le problème posé à la question précédente. On implémente sur une machine une fonction permettant de calculer pour n donné les valeurs $(y_k)_{0 \leq k \leq n}$ donné par le schéma d'Euler explicite précédent. Parmi les assertions suivantes lesquelles sont vraies.

A) Plus n est grand plus le temps de calcul sera élevé.
B) Plus n est grand, plus l'erreur commise entre la solution calculée et la solution réelle est forte.
C) Le choix de n n'influe pas sur la qualité de la solution calculée.
D) Choisir un n correctement c'est faire un compromis entre le temps de calcul et la qualité de la solution obtenue.

Banque commune École Polytechnique - interENS

PSI

Session 2015

Épreuve de Modélisation

Durée: 5 heures

Aucun document n'est autorisé

Aucune calculatrice n'est autorisée

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Système de rééducation musculaire par frein magnétique

Figure 1 : exemple de système de rééducation musculaire par frein magnétique : la rééducation des muscles liés au genou.

Lorsqu'une personne se blesse gravement et qu'elle doit subir une opération chirurgicale, il est indispensable de lui redonner une excellente motricité une fois l'opération réalisée. Cela passe par de la rééducation musculaire, qui consiste à redonner aux muscles leur tonicité initiale. Dans la plupart des cas, cette rééducation se fait en utilisant des systèmes très simples, comme par exemple des haltères ou des pinces à ressorts. Cependant, lorsque les accidents sont graves, il est utile d'utiliser des machines aux fonctions complexes, comme celle représentée sur la figure 1.

Pour rééduquer les muscles d'une articulation, il est nécessaire de les faire travailler alternativement selon trois modes :

- le mode isométrique : cela consiste à bloquer le mouvement articulaire du patient et à lui demander d'exercer avec son muscle l'effort maximum qu'il peut développer jusqu'à ce que la douleur devienne trop grande ;
- le mode isocinématique : cela consiste à demander au patient de commander ses muscles pour bouger ses membres à vitesse constante ;
- le mode isotonique : cela consiste à exercer sur les membres du patient un couple résistant constant, contre lequel les muscles doivent lutter au cours du mouvement qu'ils imposent.

La figure 2 montre une machine de rééducation classique, et son modèle mécanique associé. Le patient est assis sur un fauteuil. Il doit soulever des masses à l'aide de la partie inférieure de ses jambes, en agissant sur les muscles liés à son genou, en poussant sur un coussin d'appui. Cette machine est dite « passive », car elle ne contient aucun système de commande pour exercer des efforts résistants sur les muscles. Seule la gravité en exerce.

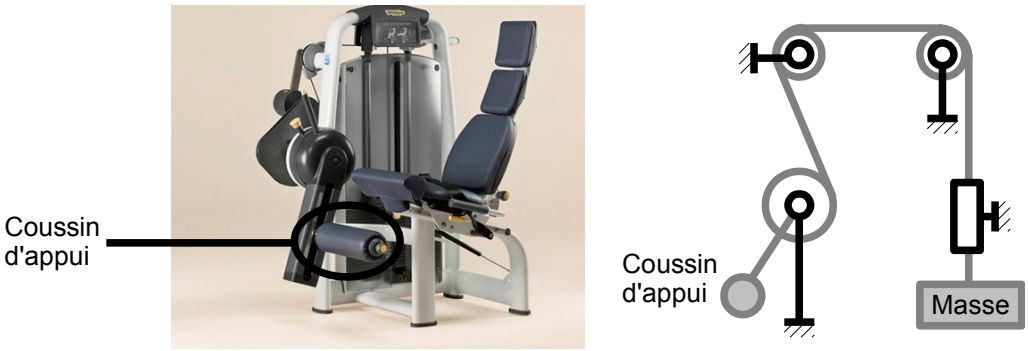


Figure 2 : machine de rééducation musculaire passive : photographie et modèle mécanique associé.

Q1 : Indiquer quel(s) mode(s) de travail une machine passive, telle que celle représentée sur la figure 2, permet de réaliser.

Afin de permettre au patient de rééduquer ses muscles selon les trois modes, il est nécessaire de concevoir une machine dite « active », c'est-à-dire possédant un système de commande permettant de réguler les efforts résistants sur les muscles. Les performances attendues par une telle machine sont résumées dans le cahier des charges suivant.

Exigence	**Niveau**	
Modes de fonctionnement	Isométrique, isocinématique et isotonique au choix	
Stabilité	Mode isotonique	Marge de gain de la FTBO : 10 dB minimum Marge de phase de la FTBO : 45°
Précision	Mode isométrique	5° de précision en position angulaire
	Mode isocinématique	Vitesse de rotation insensible aux perturbations en couple dues à l'homme
		10% de précision en vitesse angulaire, en régime permanent
	Mode isotonique	Précision totale en couple

L'objectif de ce sujet est de concevoir une telle machine à partir d'un frein magnétique. L'épreuve est structurée ainsi : la partie 1 consiste à étudier un fluide particulier aux propriétés très adaptées à la problématique ; la partie 2 porte sur la conception d'un frein qui est alors intégré dans le système de rééducation, dont l'étude est menée dans la partie 3.

Même s'il est conseillé de traiter les parties dans l'ordre du sujet (pour comprendre la logique d'enchaînement des questions et les différentes notations), le candidat pourra, s'il le souhaite, tirer profit de l'indépendance des parties.

Partie 1 : étude du fluide magnéto-rhéologique

La rhéologie est une branche de la mécanique étudiant les déformations et l'écoulement de la matière sous l'effet d'une contrainte appliquée. Certaines catégories de fluides, découverts en 1949, possèdent la propriété de voir leur comportement rhéologique, principalement leur viscosité, modifié lorsqu'un champ magnétique ou électrique leur est appliqué. On les appelle fluides magnéto-rhéologique (MR) ou électro-rhéologique (ER) et ce sont en général des huiles dotées de particules aux propriétés permettant de tels comportements. Dans le cas des fluides MR par exemple, la viscosité, faible en l'absence de champ magnétique, augmente lorsqu'on lui en applique un jusqu'à pouvoir se comporter quasiment comme un solide. L'utilisation d'électroaimants dans une chaîne asservie permet d'envisager de contrôler la « phase » du fluide et ainsi un grand nombre d'applications.

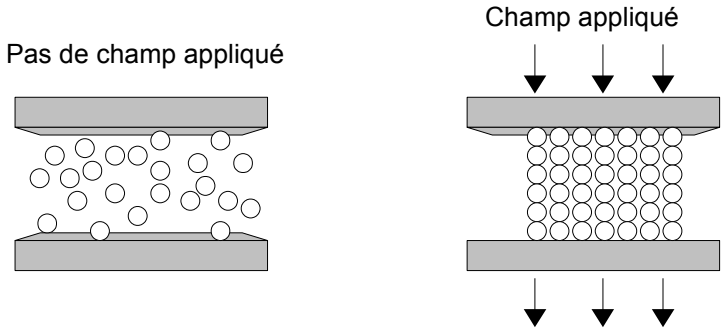


Figure 3 : principe de fonctionnement d'un fluide contrôlable.

En l'absence de champ, les particules sont dispersées de façon aléatoire dans le fluide. Lorsqu'un champ est appliqué, les particules se disposent selon des chaînes de direction colinéaire au champ (comme le montre la figure 3). Sur le dessin, on suppose que le déplacement du fluide s'effectue parallèlement aux plaques extérieures et que ces dernières sont fixes.

Q2 : À partir de la figure 3, expliquer physiquement pourquoi la viscosité du fluide augmente lorsqu'un champ règne dans le fluide. On parle sur cet exemple de mode *valve* ou *clapet* : expliquer. Dans le mode de cisaillement direct, seule une des plaques est fixe, la seconde se déplaçant de façon parallèle à la première sous l'action d'une force. Expliquer ce qu'il se passe et le terme de « cisaillement direct ».

À la différence des ferrofluides, autre catégorie de matériau intelligent, les fluides MR sont composés de particules plus grosses, de l'ordre du micromètre, là où les ferrofluides sont sensiblement des nanoparticules. Cela conduit à un comportement microscopique différent, puisque les ferrofluides peuvent rester en suspension sous l'effet du mouvement brownien alors que les fluides MR sont trop denses pour cela. De plus, aucun changement rhéologique n'apparaît lorsqu'un champ est appliqué à un ferrofluide. La figure 4 montre par exemple l'allure des particules magnétisables trouvées dans un fluide MR.

Q3 : Évaluer, à partir de la figure 4 et en supposant que celle-ci est typique d'un fluide MR, l'ordre de grandeur de la fraction (en pourcentage) du volume occupé par les

particules. Préciser également le volume typique d'une particule. Expliquer quelle grandeur risque de varier, et comment, si l'on modifie la densité des particules.

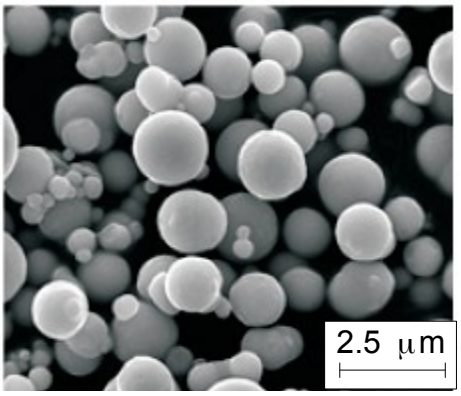


Figure 4 : particules magnétisables d'un fluide MR.

Fluides ER et MR : comparaison

Les développements technologiques récents sur les fluides MR permettent à ceux-ci de « rattraper » les performances des ER, alors qu'il étaient longtemps de qualités inférieures. Le tableau suivant résume les performances comparatives de ces deux types.

Fluide	ER	MR
Contrainte de rupture (kPa)	2 à 5	50 à 100
Viscosité (Pa.s)	0,1 à 10	0,1 à 10
Températures de fonctionnement en °C	- 25 à + 125	- 40 à + 150
Temps de réponse	Inférieur à 1 ms	Inférieur à 1 ms
Masse volumique (g.cm^{-3})	1 à 2	3 à 4
Type de champ d'excitation et valeur maximale	Électrique, ~ 4 000 000	Excitation magnétique, ~ 250 000
Exigences électriques	2 à 5 kV pour 1 à 10 mA	2 à 25 V pour 1 à 2 A

Q4 : Comparer les fluides MR aux ER. Comparer en particulier les plages de puissances électriques. Conclure sur le type de fluide le plus intéressant d'emploi.

Q5 : Préciser pourquoi on considère l'excitation magnétique plutôt que le champ magnétique. Préciser les unités usuelles puis de base du Système International pour les deux champs maximaux.

Contrainte de cisaillement et équations du modèle

On se restreint dans tout le reste du sujet à un fluide MR. Le paramètre fondamental que contrôle l'intensité du champ magnétique est la contrainte de cisaillement du fluide notée T. En mécanique, la contrainte de cisaillement T est une contrainte appliquée tangentiellement (de manière parallèle) à la face d'un matériau, ce qui entraîne en général sa déformation. Plus particulièrement en mécanique des fluides, c'est cette contrainte qui permet d'expliquer les déplacements relatifs entre différentes

« couches » de fluide : en effet, la « couche » au contact des parois possède une vitesse nulle, la « couche » en contact avec la précédente « glisse » sur celle-ci un peu plus facilement, et ainsi de suite. Ce modèle permet de démontrer que les vitesses dans une conduite à symétrie cylindrique suivent un profil parabolique. À contrainte fixée, plus le fluide sera visqueux et plus les variations de vitesses dans le fluide seront faibles. On donne pour la suite la relation simplifiée entre la contrainte de cisaillement T, la viscosité μ et le gradient de vitesse par rapport à la direction d'écoulement (par exemple selon $\vec{e_z}$) $\lambda = \frac{\partial v}{\partial z}$ pour un type de fluide appelé *fluide newtonien* :

$$T = \mu \lambda = \mu \frac{\partial v}{\partial z}$$

Dans une conduite, cette contrainte de cisaillement peut par exemple être créée par une différence de pression entre l'entrée et la sortie du tube. L'augmentation de cette contrainte est proportionnelle à celle du débit volumique Q du fluide considéré.

Le père fondateur de la science rhéologique, Eugene Bingham, étudia au début des années 1920 l'écoulement des peintures, fluide non newtonien. La loi de Bingham est la première loi de comportement d'un « fluide à seuil », autrement dit un fluide ne se déformant qu'au-delà d'une certaine contrainte. Dans ce modèle théorique, il faut atteindre une certaine contrainte de cisaillement, appelée « contrainte d'écoulement » T_0 pour que le débit commence à être non nul et varie par la suite proportionnellement à T avec le même coefficient de proportionnalité que celui du modèle newtonien.

Q6 : Tracer, à partir des indications du sujet et sans calculs, les allures des courbes donnant le débit d'une conduite donnée en fonction de la contrainte de cisaillement T, pour une fluide newtonien puis un fluide de Bingham. On indiquera en particulier la contrainte d'écoulement T_0.

Le débit dépendant de nombreux paramètres, principalement les données géométriques de la conduite, on préfère aujourd'hui utiliser le modèle plastique de Bingham donnant T en fonction de λ. Dans le cas d'un fluide MR, on peut même tracer les courbes en tenant compte de la contribution de l'excitation magnétique appliquée. On obtient ainsi une famille de courbes, comme sur la figure suivante :

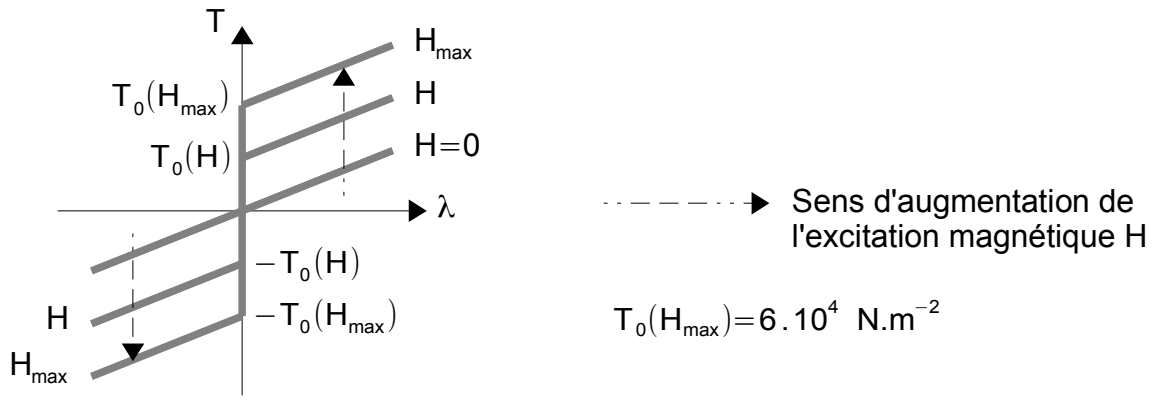


Figure 5 : relation entre la contrainte de cisaillement et le gradient de vitesse pour différentes excitations magnétiques.

Q7 : Expliquer comment lire sur cette figure la valeur de la viscosité et comment celle-ci varie en fonction de l'excitation magnétique.

Sur la base de résultats expérimentaux, Stanway et *al*. ont proposé en 1987 un modèle de la force totale F consistant en un amortisseur visqueux de coefficient α en parallèle avec un système à frottement solide f_C constant de type Coulomb, représenté par la figure 6.

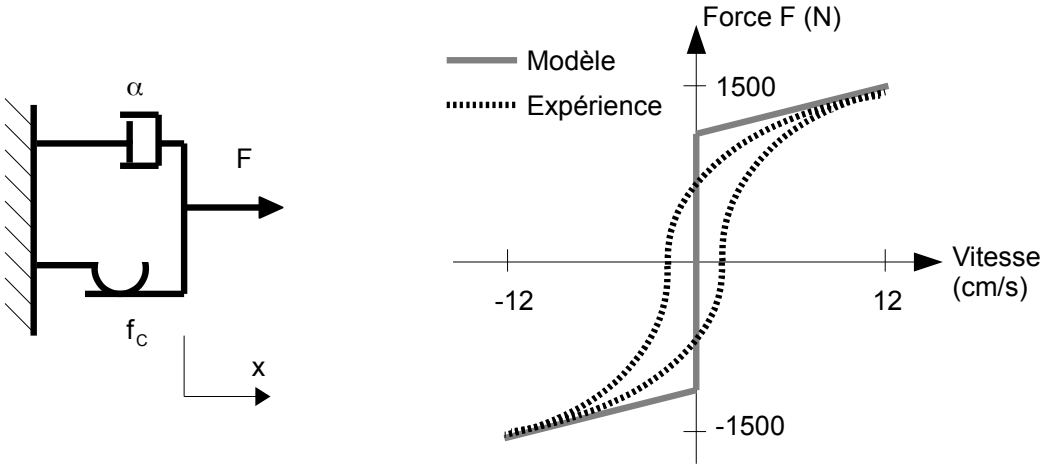


Figure 6 : modèle de Stanway et comparaison avec l'expérience.

Q8 : Justifier la pertinence de ce modèle. Donner, pour un paramétrage unidimentionnel et selon le signe de la vitesse, l'expression de la force. Évaluer les valeurs des paramètres du modèle.

Q9 : Indiquer le phénomène qui n'est pas encore pris en compte, et qui semble apparaître avec l'expérience. Commenter la qualité du modèle de Bingham.

Q10 : Un second modèle a été proposé par Gamota et *al*. selon la figure 7. Commenter. Ce modèle conduit toutefois à des équations non linéaires difficiles à résoudre numériquement car nécessitant un pas d'intégration numérique très petit. Expliquer cette phrase.

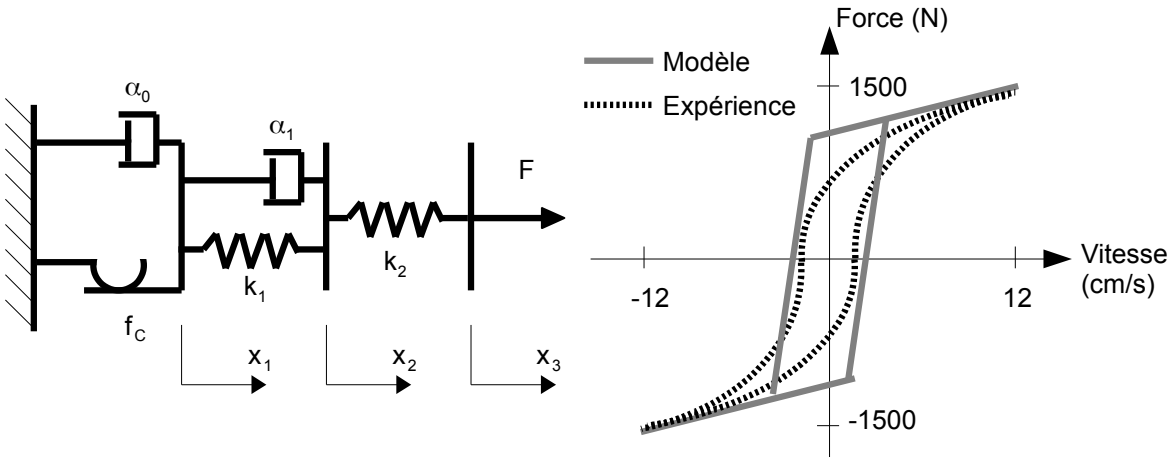


Figure 7 : modèle de Gamota et comparaison avec l'expérience.

Nombres de Hedström et Bingham

Le nombre de Hedström H_e est un nombre adimensionné utilisé en rhéologie pour caractériser le comportement laminaire ou turbulent d'un fluide de Bingham. Il s'écrit sous la forme $H_e = B_m R_e$ où B_m est le nombre de Bingham, lui aussi sans dimension, caractérisant de son côté le rapport entre les contraintes élastiques et les contraintes visqueuses (les premières étant divisées par les secondes).

Q11 : Le nombre R_e désigne un troisième nombre adimensionné au programme de PSI. Nommer ce nombre et rappeler son expression en fonction de la vitesse caractéristique du fluide V, la dimension caractéristique L, la masse volumique du fluide ρ et la viscosité (dynamique) de celui-ci μ. Expliquer brièvement son intérêt.

Q12 : Le nombre de Bingham s'écrit sous la forme $B_m = K T^a L^b \mu^c V^d$ où L, μ et V sont définis à la question précédente, T est la contrainte de cisaillement, K est une constante adimensionnée valant 1 et (a,b,c,d) est un quadruplet d'entiers relatifs constants, les plus petits possibles en valeur absolue. Déterminer le quadruplet. Interpréter physiquement ce nombre.

Q13 : En déduire l'expression de H_e. Pour une longueur caractéristique L de 1 cm, proposer un ordre de grandeur numérique de ces trois nombres à partir des documents précédents. Conclure sur la nature turbulente ou laminaire de l'écoulement, dans le cas où celui-ci serait celui d'un fluide newtonien puis de Bingham.

Conclusion

Cette partie nous a permis d'exhiber quelques propriétés et modèles de comportement d'un fluide MR. Cela a été nécessaire pour envisager son utilisation dans la conception d'un frein magnétique.

Partie 2 : conception du frein magnétique

La première partie nous a permis d'établir un modèle pour les fluides MR reliant le cisaillement T en fonction de la contrainte d'écoulement T_0, qui dépend de l'excitation magnétique H appliquée, de la viscosité μ et du gradient de vitesse par rapport à la direction d'écoulement λ :

$$T = T_0(H) + \mu\lambda$$

Cette partie s'intéresse à une géométrie particulière de frein utilisant les fluides MR, et à son dimensionnement en vue de son utilisation pour la machine de rééducation musculaire.

Paramétrage de la géométrie du frein et hypothèses

La structure étudiée est celle du frein disque, une des plus faciles à fabriquer et qui offre un bon compromis entre ses performances et sa compacité. Dans ce frein, la partie rotorique est un disque lié à un arbre cylindrique, en rotation à l'intérieur d'une cavité statorique. Entre le stator et le rotor, on trouve le fluide MR sur les surfaces du disque rotorique, et face aux parois latérales de ce dernier, une bobine torique assurant la création de l'excitation magnétique, l'ensemble des matériaux étant supposés linéaires. Les paramètres géométriques sont indiqués sur la figure 8.

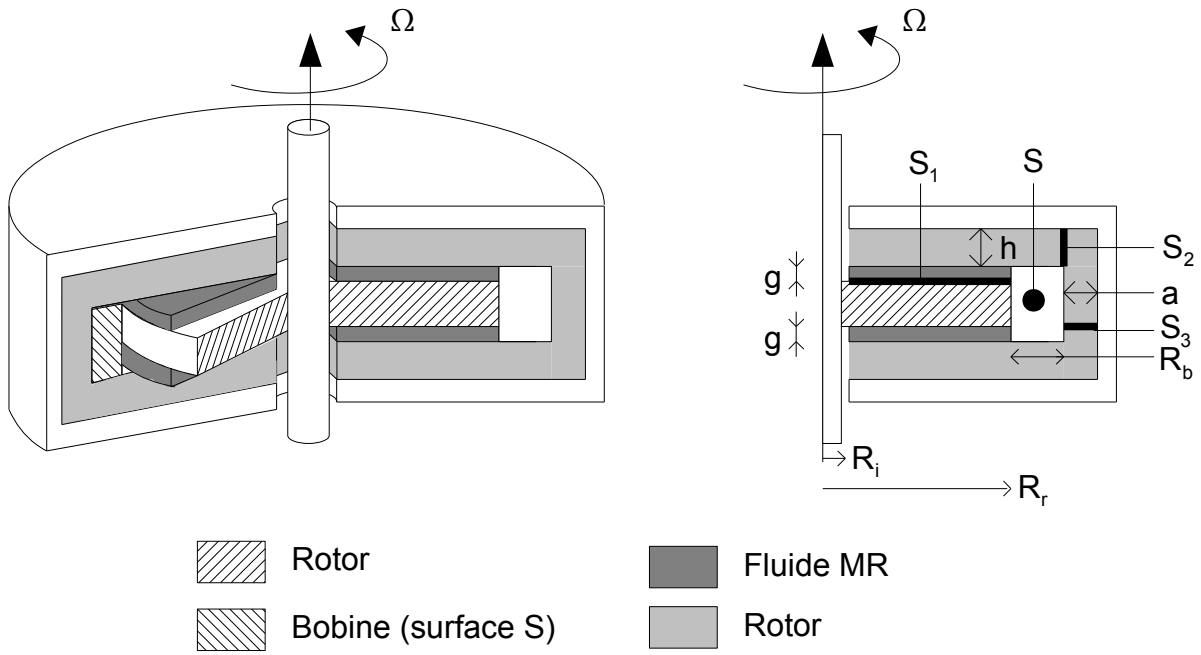


Figure 8 : paramétrage de la géométrie du frein.

La longueur h désigne l'épaisseur statorique de matériau magnétique, g celle des couches de fluide MR, S_1, S_2 et S_3 sont les surfaces désignées sur le dessin de la figure 8 (surfaces cylindriques de révolution autour de l'arbre) et S la section de la bobine, constituée de N spires parcourue par une intensité I et d'une épaisseur radiale R_b. Pour simplifier, les fils seront considérés comme tous distants de R_r de l'axe. Le rotor tourne à une vitesse angulaire Ω supposée constante (régime permanent). On suppose dans toute la suite que R_b et R_i sont négligeables devant R_r et que l'excitation magnétique

est uniforme pour tout point de la surface S_1 correspondant à la surface de contact entre le fluide MR et le rotor.

Q14 : Reproduire sur la copie le dessin de la figure 9 en indiquant et en justifiant le sens du courant électrique nécessaire pour avoir le champ proposé, ainsi que l'allure de la disposition des particules du fluide MR.

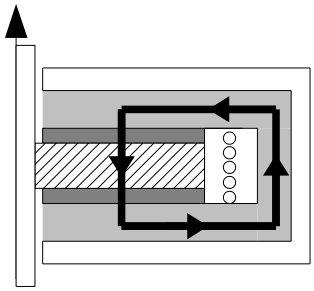


Figure 9 ***à reproduire sur la copie*** : quelques orientations des phénomènes physiques en jeu (le trait noir représente la ligne de champ).

Contrainte de cisaillement et calculs des couples

La forme de la contrainte de cisaillement proposée plus haut nous conduit à envisager le calcul de deux couples différents. Le premier, appelé « couple magnétique » et noté C_m, en référence au fait qu'il est lié aux phénomènes magnétiques, sera le couple dû à T_0. On prend pour expression de celui-ci l'intégrale :

$$C_m = \int_A r\, T_0\, dA$$

où dA désigne un élément élémentaire de la surface active traversée par le champ d'excitation magnétique, c'est-à-dire S_1. Le second, noté C_v, est celui calculé par rapport à la contribution de la partie visqueuse du cisaillement, autrement dit et avec les mêmes notations :

$$C_v = \int_A r\, \mu\, \lambda(r)\, dA$$

On rappelle que l'on suppose la contrainte d'écoulement T_0 uniforme sur la surface S_1.

Q15 : Justifier brièvement la forme commune des deux intégrales proposées.

Q16 : Calculer C_m pour l'ensemble des surfaces actives et montrer que celui-ci peut s'écrire $C_m = C_1 T_0 R_r^3$. Préciser la valeur de la constante C_1. Expliquer alors comment choisir l'excitation pour optimiser ce couple.

Le calcul du second couple nécessite de trouver une forme d'évolution du gradient de vitesse λ. À cette fin, on modélise le champ de vitesses du fluide MR entre le stator et le rotor, dans une coupe orthoradiale, selon la représentation de la figure 10. On fait l'hypothèse d'un écoulement laminaire et d'une vitesse de déplacement constante entre stator et rotor.

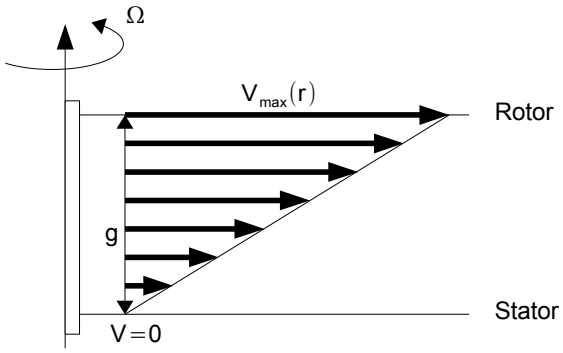


Figure 10 : modèle du champ de vitesses.

Q17 : Justifier cette double hypothèse par des raisons physiques ainsi que l'utilisation d'une coupe orthoradiale.

Q18 : Démontrer qu'en un point de S situé à une distance r de l'axe, on a $\lambda = r\Omega/g$. En déduire que C_V s'exprime sous la forme $C_2 \mu R_r^4$. On précisera la valeur de C_2 en fonction de Ω et g.

Critères de performances

On s'intéresse à trois critères de performances pouvant caractériser le frein :

- le rapport du couple magnétique sur le volume du stator, noté T_V ;
- le rayon du rotor, à couple fixé, déjà noté R_r auparavant ;
- la consommation électrique W_e, correspondant aux pertes Joule dans la bobine.

On suppose que les champs et excitations magnétiques dans le fer et dans le fluide MR sont juste à saturation, ce qui assure d'après la figure 5 la plus grande valeur de T_0 possible. On note ces grandeurs respectivement B_{fer}, H_{fer}, B_{MR} et H_{MR}.

Q19 : Expliquer physiquement l'intérêt de chacun de ces paramètres et s'ils doivent être maximisés ou minimisés.

Q20 : Rappeler la propriété fondamentale du flux magnétique et utiliser celle-ci pour relier B_{MR}, B_{fer}, R_r et h. Montrer également que a et h sont quasi égaux.

Q21 : En négligeant g devant h, exprimer T_V lorsque $B_{fer} = 2\,B_{MR}$. Conclure sur la pertinence du choix de champs à quasi saturation.

Q22 : Grâce à une hypothèse sur les perméabilités magnétiques que l'on justifiera, montrer que l'on a $H_{MR} \simeq \frac{2g}{NI}$.

Q23 : En supposant que la bobine est constituée d'un seul fil cylindrique de cuivre de conductivité σ, montrer que $W_e = C_3 . R_r . (2g . H_{MR})^2$ en précisant la valeur de la constante C_3 en fonction de S et σ.

Q24 : Décrire les influences de T_0 et de g sur W_e. Conclure.

Dimensionnement du frein pour son usage en rééducation

On s'intéresse désormais à quelques ordres de grandeur nécessaires pour le système de rééducation que nous étudierons plus complètement plus loin. On veut obtenir un couple magnétique de 2 N.m. On considère que l'on est tout juste à saturation du champ d'excitation sur la surface S_1 et que la perméabilité magnétique relative μ_{MR} du fluide MR vaut 2. On rappelle que certaines valeurs numériques se trouvent à la page 5 et on prend pour la conductivité du cuivre $\sigma = 6.10^7$ S.m^{-1}. Comme précédemment, on suppose que $B_{fer} = 2 B_{MR}$. Enfin, on choisit g = 0,1 mm et S = 1 mm^2.

Q25 : Calculer les champs magnétiques au niveau de l'interface rotor/fluide.

Q26 : Calculer T_V et en déduire le volume V du stator. La figure 5 est ici utile.

Q27 : Déterminer alors le rayon R_r puis h et enfin le rayon du stator. Commenter.

Q28 : Calculer la puissance électrique consommée. Commenter.

Vérification de l'hypothèse

On veut vérifier par simulation numérique la validité de l'hypothèse d'excitation (ou de champ) uniforme à l'interface rotor/fluide. Pour cela, on utilise la méthode des volumes finis, qui consiste à découper les volumes considérés (stator, rotor, bobine) en volumes élémentaires de très petite taille et à calculer le champ créé par et pour chacun puis à sommer leurs contributions en un point pour obtenir le champ total en ce point. Pour simplifier, on prend des volumes élémentaires cubiques de côté c.

Q29 : Donner la propriété des équations électromagnétiques qui permet d'assurer que le champ total est la somme des contributions élémentaires. Préciser comment varient la précision des simulations et la complexité spatiale (c'est-à-dire en mémoire) d'un tel algorithme en fonction de c. Expliquer pourquoi l'utilisation de la symétrie de révolution permet d'améliorer les temps de calculs.

Q30 : On assimile grossièrement le frein à un cylindre de 10 cm de hauteur et d'un diamètre de 6 cm. En considérant que c = 1 mm et que la mémoire nécessaire pour stocker la valeur du champ en chaque volume élémentaire est de 32 bits, donner en méga-octets la taille de la mémoire nécessaire aux calculs du champ en tout point du frein. On précisera les éventuelles approximations ou hypothèses. Expliquer ce qu'il se passe alors si on décide d'améliorer la précision en passant à c = 0,1 mm puis c = 0,01 mm. Conclure.

Les valeurs du champ sont ensuite calculées pour une section radiale du frein et stockées dans un fichier texte nommé `donnees_champ.txt`. Chaque ligne de ce fichier correspond aux valeurs d'une rangée horizontale de volumes élémentaires. Dans chaque ligne, les valeurs du champ sont séparées par une tabulation et la position de la valeur dans cette ligne correspond à la rangée verticale du volume élémentaire considéré. Le script Python suivant est alors utilisé, les numéros sur la gauche correspondant aux numéros de lignes du programme.

```
def recup_champ():
        fich=open('donnees_champ.txt','r')
        champ=[ ]
        for x in fich:
                interm = x.strip().split('\t')
                champ += [interm]
        return champ
```

La méthode `strip` appliquée à une chaîne de caractères renvoie une copie de la chaîne initiale dans laquelle les caractères codant un retour à la ligne (ainsi que les espaces) présents en début et en fin de chaîne sont supprimés. La méthode `split` décompose la chaîne initiale en une liste de sous-chaînes, le découpage s'effectuant à chaque occurrence de la chaîne passée en argument. Enfin, `\t` correspond au codage en type `string` d'une tabulation.

Q31 : Expliquer ce que contient la variable `champ` en sortie de fonction. On précisera en particulier le type des éléments de la variable `champ`.

Q32 : On suppose donnés les entiers `interface_hor` correspondant au numéro de la ligne (numérotées à partir de zéro en haut) indiquant la position horizontale de l'interface rotor/fluide et `interface_ver` donnant le numéro de la colonne (numérotées à partir de zéro à gauche) délimitant la frontière rotor/bobine. Proposer et commenter un programme Python ou Scilab affichant la moyenne et la variance des valeurs du champ sur la surface S_1. Ce programme devra renvoyer une seule grandeur de votre choix, permettant de fournir un indicateur de la quasi-constance ou non du champ sur cette surface. On indiquera par des traits verticaux en début de ligne les indentations pour éviter toute ambiguïté dans celles-ci.

Q33 : Les résultats de simulation sont les suivants, en général et plus particulièrement pour les points situés sur la surface S_1. Commenter et conclure sur la pertinence de l'hypothèse effectuée précédemment.

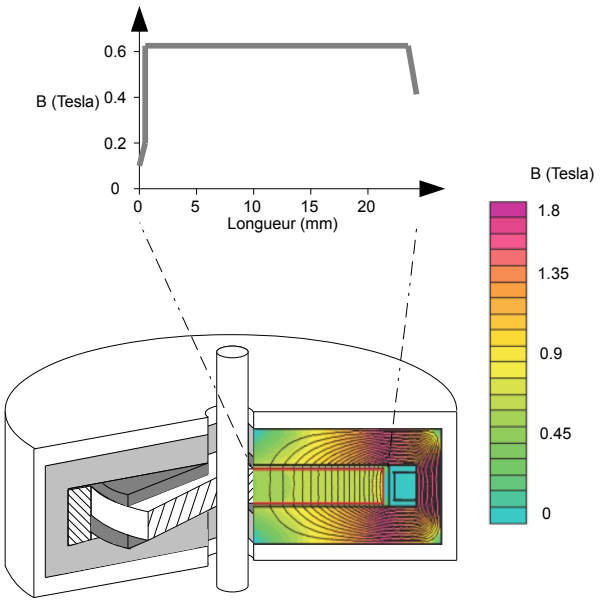


Figure 11 : résultats des simulations pour le calcul du champ magnétique B régnant dans le frein.

Partie 3 : étude du système de rééducation

Les parties précédentes ont permis de proposer un modèle pour le frein magnétique. L'objectif de cette partie est de montrer que ce frein, une fois inséré dans un système de rééducation musculaire, permet d'offrir au patient les différents modes de fonctionnement qui sont utiles d'un point de vue médical. Les performances attendues par un tel système de rééducation sont résumées dans le cahier des charges présent dans l'introduction du sujet.

Modélisation du système de rééducation musculaire

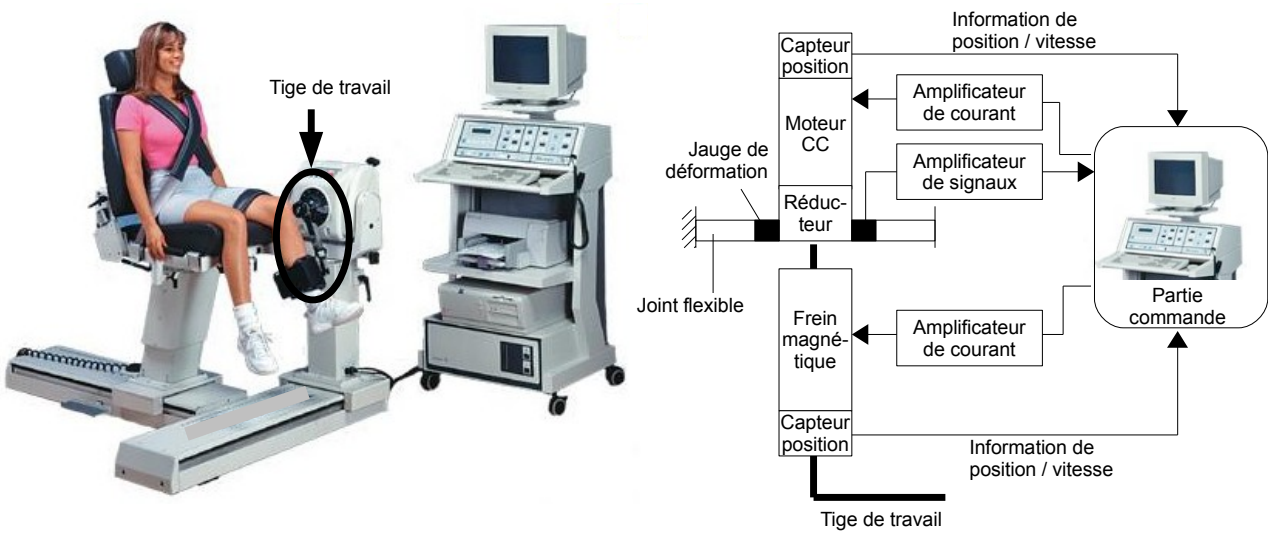


Figure 12 : solution technologique du système de rééducation par frein magnétique.

Le système de rééducation musculaire qui utilise le frein magnétique est décrit sur la figure 12. La photographie qui y est représentée montre la rééducation des muscles d'un genou. Le patient positionne son pied sur la tige de travail. Elle est mise en mouvement par un moteur à courant continu, dont la vitesse de rotation est réduite par un réducteur à engrenages. Le frein magnétique exerce un couple résistant sur l'axe en sortie du réducteur, donc sur l'axe de la tige de travail. Un capteur par jauge de déformation, associé à un joint flexible, permet de mesurer le couple exercé par le moteur. Des capteurs permettent de mesurer les positions et les vitesses angulaires du moteur et de la tige de travail, et des amplificateurs permettent d'amplifier les signaux. L'ensemble est piloté par une partie commande spécifiquement développée pour les applications médicales.

Les équations qui régissent le comportement du moteur à courant continu sont :

- $u_m(t)=e_m(t)+R_m i_m(t)+L_m \dfrac{d i_m(t)}{dt}$
- $e_m(t)=k_e \omega_m(t)$
- $C_m(t)=k_m i_m(t)$

où u_m est la tension électrique d'alimentation du moteur, i_m l'intensité qui traverse son bobinage, e_m la force contre électromotrice, ω_m la vitesse angulaire de son rotor, C_m le couple qu'il délivre, R_m sa résistance électrique interne et L_m l'inductance de son induit. $k_m=k_e$ est une constante de proportionnalité.

Le réducteur utilisé est un réducteur à engrenages. Il est basé sur la technologie des trains épicycloïdaux. Il y a deux étages de réduction, identiques, en série. Pour le premier étage de réduction, le pignon, positionné sur l'axe du moteur, possède 12 dents. Il engrène par un contact extérieur à un satellite de 7 dents, qui lui-même engrène par un contact intérieur à une couronne dentée de 26 dents, liée au bâti. Le porte satellite du second train est la sortie du réducteur, située dans le prolongement de l'axe du moteur.

Q34 : Modéliser le réducteur à engrenages, en proposant un schéma cinématique montrant toutes les pièces qui le constituent. Respecter les dimensions relatives des pièces sur le schéma. Déterminer le rapport de réduction de ce réducteur.

Le modèle retenu pour le pilotage du système de rééducation est représenté sur la figure 13. θ est l'angle de rotation de la tige de travail. θ_{ref} est la position angulaire souhaitée. C_f est le couple fourni par le frein magnétique. C_h est le couple fourni par l'homme qui subit une rééducation. C_r est le couple fourni par le moteur, via le réducteur.

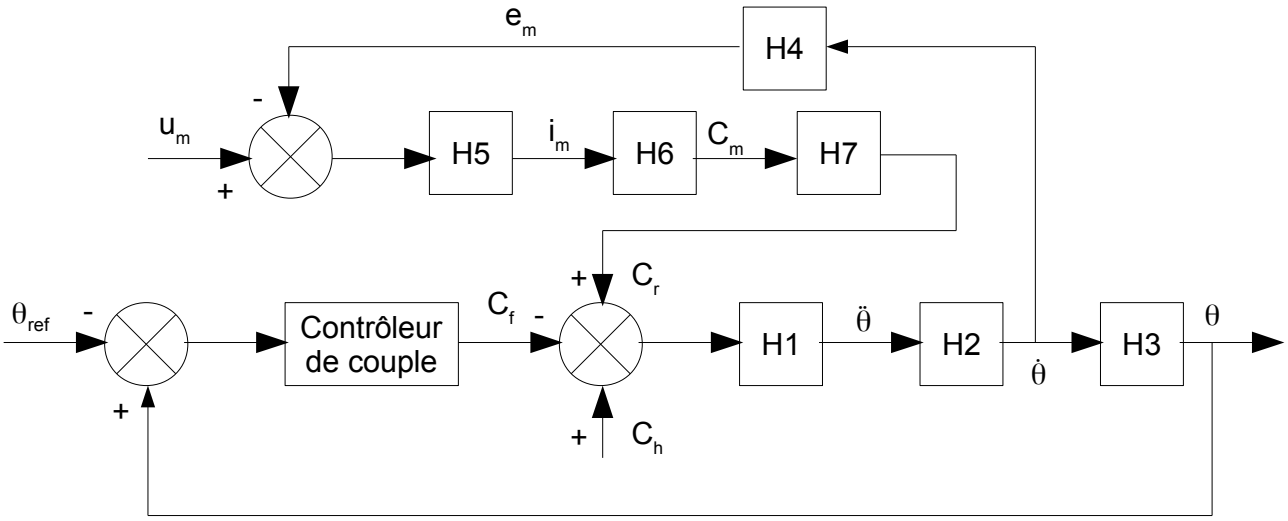


Figure 13 : modèle retenu pour pilotage du système de rééducation.

Q35 : Déterminer les expressions analytiques des fonctions H5 et H6.

Q36 : En notant J l'inertie des pièces en rotation ramenées sur la tige de travail, déterminer les expression analytiques des fonctions H1, H2 et H3.

Q37 : On a $k_m = 0{,}04\,\mathrm{N.m.A^{-1}}$. Déterminer les valeurs numériques des fonctions H4 et H7.

Etude du mode isométrique

On étudie dans un premier temps la capacité du système à rééduquer le patient dans le mode isométrique. Pour ce mode, le fonctionnement est le suivant : le moteur positionne la tige de travail dans la position souhaitée. À partir de cette position, le moteur n'agit plus, mais le contrôleur de couple délivre un couple de freinage proportionnel à la différence entre θ_{ref} et θ : $C_f = k_f(\theta - \theta_{ref})$, avec $k_f = 1\,\mathrm{N.m/°}$.

Q38 : Déterminer l'équation différentielle qui gère le mouvement de la tige de travail, et donc du pied du patient.

Q39 : On ne demande jamais à un patient d'exercer avec son genou plus de 2 N.m. En négligeant les effets dynamiques dans l'équation précédent, indiquer si le système est capable d'atteindre la précision angulaire souhaitée pour le mode isométrique.

Etude du mode isocinématique

On étudie maintenant la capacité du système à rééduquer le patient dans le mode isocinématique. Dans ce mode, le moteur est alimenté en permanence pour imposer une vitesse de rotation à la tige de travail, à vitesse constante. Le patient exerce un couple résistant avec sa jambe, plus ou moins intensément, en fonction de sa capacité musculaire et de l'évolution de sa rééducation.

Q40 : Dans ce mode, l'évolution angulaire de la tige de travail s'exprime sous le format suivant : $\theta = H_\theta . \theta_{ref} + H_h . C_h + H_m . U_m$. Déterminer les expressions de H_θ, H_h et H_m en fonction de H1, H2, H3, H4, H5, H6, H7 et H8. H8 correspond à la fonction de transfert du contrôleur de couple.

Q41 : On suppose que la fonction de transfert du contrôleur de couple est une constante de proportionnalité. Indiquer si, dans ce mode de fonctionnement, le système peut satisfaire l'exigence d'insensibilité de la vitesse de rotation aux perturbations de couple dues à l'homme.

Pour ce mode de fonctionnement, on décide d'utiliser une loi de contrôle du couple qui est représentée sur la figure 14. Un petit couple constant C_0 est toujours ajouté au frein. Le bloc dénommé PID fournit en sortie un signal qui tient compte de ϵ, de sa dérivée et de sa primitive, selon la loi $C_c(t) = k_P \epsilon(t) + k_I \int_0^t \epsilon(u) du + k_D \frac{d\epsilon(t)}{dt}$. La saturation limite la valeur de C_c à la valeur $C_{S\max}$ si elle la dépasse.

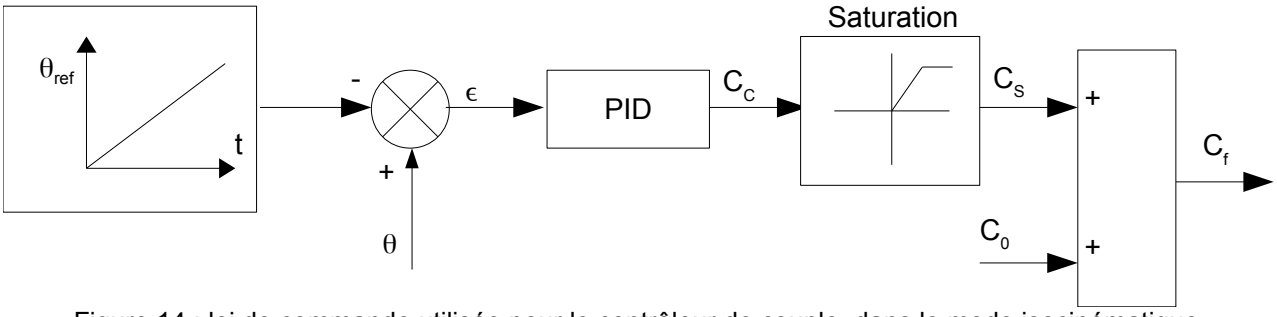


Figure 14 : loi de commande utilisée pour le contrôleur de couple, dans le mode isocinématique.

Q42 : On appelle `epsilon_0` la valeur prise par ϵ à l'instant $t - dt$ et `epsilon_1` la valeur prise par ϵ à l'instant t, où dt est un nombre de type flottant désignant le temps d'échantillonnage. On appelle par ailleurs `int_epsilon_tmdt` la valeur de l'intégrale de ϵ de l'instant 0 à l'instant $t - dt$. Écrire et commenter une fonction

```
calc_Cf(epsilon_1,epsilon_0,dt,C_Smax,C_0,...
        int_epsilon_tmdt,k_P,k_I,k_D)
```

qui retourne la valeur prise par C_f à l'instant t. On calculera l'intégrale par la méthode des trapèzes. Le langage utilisé sera Python ou Scilab.

La figure 15 montre l'évolution des grandeurs physiques lors de l'utilisation du système dans son mode isocinématique.

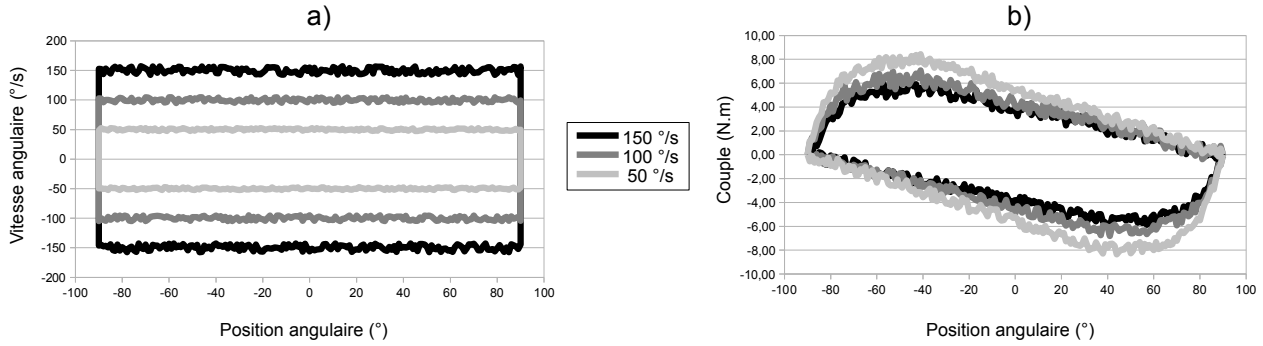


Figure 15 : mode isocinématique. Gauche : évolution de la vitesse angulaire en fonction de la position angulaire de la tige de travail. Droite : évolution du couple délivré par le patient en fonction de la position angulaire de la tige de travail.

Q43 : Indiquer la raison pour laquelle la mesure de l'angle ne se fait pas sur une plage de position angulaire plus grande que la plage [-90° ; 90°]. Expliquer pourquoi les vitesses sont mesurées positivement et négativement.

Q44 : Indiquer si le système est capable de répondre au cahier des charges, concernant la précision de 10% sur la vitesse angulaire en régime permanent.

Etude du mode isotonique

Pour faire travailler le patient en mode isotonique, il suffit de permettre au système de générer le couple souhaité. La figure 16 montre le modèle de la loi de commande retenue pour ce mode. On réalise un asservissement du couple. La fonction de transfert du frein est, en première approximation, modélisée par une fonction du premier ordre de gain $k_F = 2\,\text{N.m.A}^{-1}$ et de constante de temps $T_F = 0{,}05\,\text{s}$. Le correcteur utilisé est $C(p) = k_C\left(1 + \frac{1}{T_C p}\right)$.

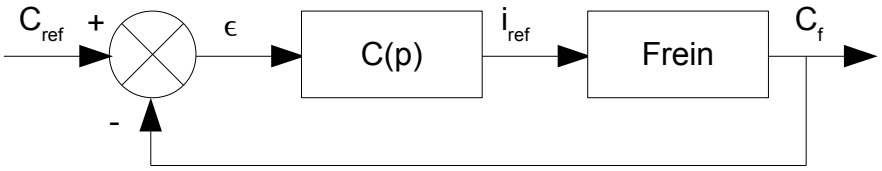


Figure 16 : modèle de la loi de commande retenue pour le mode isotonique.

Q45 : Indiquer si le système satisfait au cahier des charges en terme de précision.

Q46 : Déterminer les valeurs de k_C et T_C pour obtenir la marge de phase souhaitée dans le cahier des charges, à la pulsation $\omega = 1/(5T_C)$.

Q47 : Indiquer si, pour ce réglage du correcteur, la marge de gain satisfait le cahier des charges.

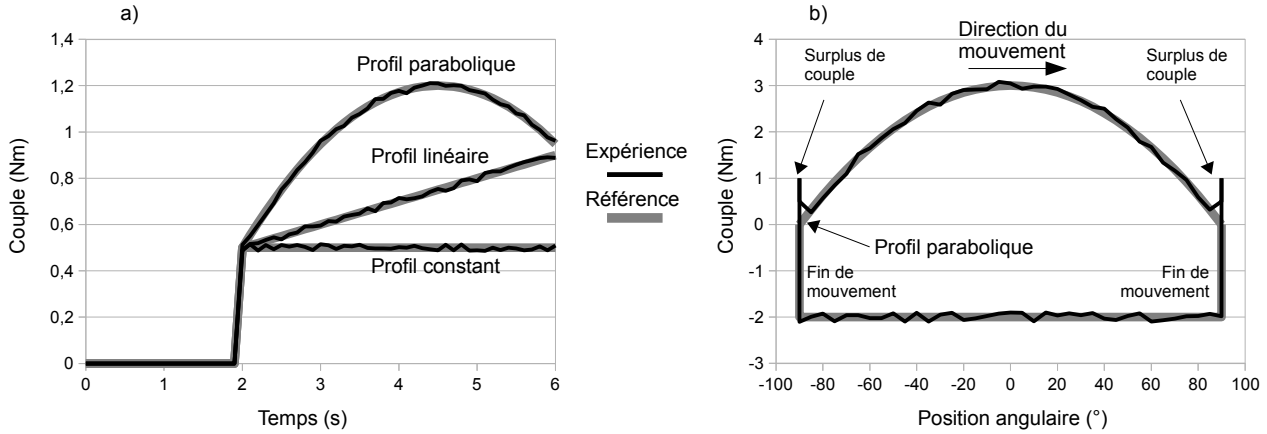


Figure 17 : évolution du couple en mode isotonique. Gauche : évolution du couple au cours du temps, pour différents profils d'évolution souhaités. Droite : évolution du couple en fonction de la position angulaire, pour un profil d'évolution souhaite.

L'évolution du couple, en fonction de différents profils d'évolution souhaités, est représentée sur la figure 17.

Q48 : Indiquer si l'évolution du couple au cours du temps, en fonction des différents profils, est cohérente par rapport au choix du correcteur retenu. Expliquer une origine possible pour le surplus de couple visible sur la figure 17 (b).

Conclusion

Cette étude, qui prend fin ici, a permis de montrer que le système de rééducation musculaire basé sur un frein magnétique permet de rééduquer le patient selon les différents modes isométrique, isocinématique et isotonique. Il représente ainsi un système technologique intéressant pour le monde de la médecine.

Remerciements

Les éléments de cette étude ont été extraits du travail du docteur More Thomas AVRAAM, à l'Université libre de Bruxelles.

EXERCICE I. INFORMATIQUE

Les algorithmes demandés doivent être écrits en Python. On sera très attentif à la rédaction et notamment à l'indentation du code.

Voici, par exemple, un code Python attendu si l'on demande d'écrire une fonction nommée `maxi` qui calcule le plus grand élément d'un tableau d'entiers :

```
def maxi(t):
    """Données: t un tableau d'entiers non vide
       Résultat: le maximum des éléments de t"""
    n =len(t) # la longueur du tableau t
    maximum = t[0]
    for k in range(1,n):
        if t[k] >  maximum:
            maximum = t[k]
    return maximum
```

L'instruction `maxi([4,5,6,2])` renverra alors 6.

I.1. Donner la décomposition binaire (en base 2) de l'entier 21.

On considère la fonction `mystere` suivante :

```
def mystere(n, b):
    """Données: n > 0 un entier et b > 0 un entier
       Résultat: ......."""
    t = [] # tableau vide
    while n > 0:
        c = n % b
        t.append(c)
        n = n // b
    return t
```

On rappelle que la méthode `append` rajoute un élément en fin de liste. Si l'on choisit par exemple `t = [4,5,6]`, alors, après avoir exécuté `t.append(12)`, la liste `t` a pour valeur `[4,5,6,12]`.

Pour $k \in \mathbb{N}^*$, on note c_k, t_k et n_k les valeurs prises par les variables `c`, `t` et `n` à la sortie de la k-ème itération de la boucle "while".

I.2. Quelle valeur est renvoyée lorsque l'on exécute `mystere(256,10)` ?

On recopiera et complétera le tableau suivant, en ajoutant les éventuelles colonnes nécessaires pour tracer entièrement l'exécution.

k	1	2	$\cdots$
c_k			$\cdots$
t_k			$\cdots$
n_k			$\cdots$

I.3. Soit $n>0$ un entier. On exécute `mystere(n,10)`. On pose $n_0=n$.

I.3.a. Justifier la terminaison de la boucle `while`.

I.3.b. On note p le nombre d'itérations lors de l'exécution de `mystere(n,10)`. Justifier que pour tout $k\in[\![0,p]\!]$, on a $n_k\leq\frac{n}{10^k}$. En déduire, une majoration de p en fonction de n.

I.4. En s'aidant du script de la fonction `mystere`, écrire une fonction `somme_chiffres` qui prend en argument un entier naturel et renvoie la somme de ses chiffres. Par exemple, `somme_chiffres(256)` devra renvoyer 13.

I.5. Ecrire une version récursive de la fonction `somme_chiffres`, on la nommera `somme_rec`.

EXERCICE II. PROJECTION ORTHOGONALE

On considère $\mathscr{M}_2(\mathbb{R})$ l'espace vectoriel euclidien des matrices carrées d'ordre 2 à coefficients réels muni du produit scalaire canonique défini pour A et B matrices de $\mathscr{M}_2(\mathbb{R})$ par : $(A|B)=\text{trace}({}^tAB)$.

II.1. Si $A=\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ et $A'=\begin{pmatrix} a' & b' \\ c' & d' \end{pmatrix}$ sont deux matrices de $\mathscr{M}_2(\mathbb{R})$, que vaut le réel $\left(A\middle|A'\right)$?

II.2. On note $\mathscr{T}$ le sous-espace vectoriel formé des matrices triangulaires supérieures de $\mathscr{M}_2(\mathbb{R})$.

Donner, pour le produit scalaire canonique, une base orthonormée de $\mathscr{T}$ et de son orthogonal $\mathscr{T}^\perp$.

II.3. Si $A=\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$, déterminer le projeté orthogonal de la matrice A sur $\mathscr{T}$, ainsi que la distance de la matrice A à $\mathscr{T}$.

PROBLEME III. SURJECTIVITE DE L'APPLICATION EXPONENTIELLE DE $\mathscr{M}_n(\mathbb{C})$ VERS $\mathrm{GL}_n(\mathbb{R})$

On note, pour n entier $n \geq 2$, $\mathscr{M}_n(\mathbb{C})$ l'espace vectoriel des matrices carrées d'ordre n à coefficients complexes.

On notera $1 \leq i, j \leq n$ pour indiquer que : $1 \leq i \leq n$ et $1 \leq j \leq n$.

Partie préliminaire

Une norme $\|.\|$ sur l'espace vectoriel $\mathscr{M}_n(\mathbb{C})$ est une norme d'algèbre si elle vérifie la propriété :
pour tout couple de matrices (A,B) de $\mathscr{M}_n(\mathbb{C})$, $\|AB\| \leq \|A\|\|B\|$.

III.1. On note pour $A=(a_{i,j})$ élément de $\mathscr{M}_n(\mathbb{C})$, $\|A\|_\infty = \sup\limits_{1\leq i,j\leq n} \left|a_{i,j}\right|$ et $\|A\| = n\|A\|_\infty$.

L'application $\|.\|$ est une norme sur l'espace vectoriel $\mathscr{M}_n(\mathbb{C})$. Démontrer que c'est une norme d'algèbre.

Dans la suite de cette partie préliminaire, on munit $\mathscr{M}_n(\mathbb{C})$ de cette norme d'algèbre.

III.2. Justifier simplement qu'une série de vecteurs de $\mathscr{M}_n(\mathbb{C})$ absolument convergente est convergente.

III.3. Si M est une matrice de $\mathscr{M}_n(\mathbb{C})$, établir que la série de réels positifs $\sum \left\|\frac{1}{k!}M^k\right\|$ converge et en déduire que la série de matrices $\sum \frac{1}{k!}M^k$ converge.

Si M est une matrice de $\mathscr{M}_n(\mathbb{C})$, on notera $\exp(M) = \sum\limits_{k=0}^{+\infty} \frac{1}{k!}M^k$, exponentielle de la matrice M.

Première partie

On pourra utiliser librement le résultat suivant :

si T est une matrice triangulaire de $\mathscr{M}_n(\mathbb{C})$ dont les éléments diagonaux sont $\lambda_1, \lambda_2, \ldots, \lambda_n$, alors la matrice $\exp(T)$ est une matrice triangulaire dont les éléments diagonaux sont $e^{\lambda_1}, e^{\lambda_2}, \ldots, e^{\lambda_n}$.

III.4. Si M est une matrice de $\mathscr{M}_n(\mathbb{C})$, rappeler pourquoi la matrice M est trigonalisable et déterminer une relation entre $\det(\exp(M))$ et $e^{\mathrm{tr}(M)}$.

III.5. Soit la matrice $A = \begin{pmatrix} 3 & 6 & -6 \\ -1 & -9 & 11 \\ 0 & -5 & 7 \end{pmatrix}$.

Donner le déterminant de la matrice A. En déduire qu'il n'existe aucune matrice B à coefficients réels vérifiant $B^2 = A$ et qu'il n'existe aucune matrice M à coefficients réels vérifiant $\exp(M) = A$.

Objectifs et exemple

Ce paragraphe ne comporte aucune question, il permet de se familiariser avec les objectifs du problème.

Si A est une matrice carrée inversible à coefficients réels, nous allons démontrer dans ce problème :

- que pour tout entier naturel non nul p, il existe une matrice B de $\mathscr{M}_n(\mathbb{C})$ vérifiant $B^p = A$,
- qu'il existe une matrice M de $\mathscr{M}_n(\mathbb{C})$ vérifiant $\exp(M) = A$.

On se limitera dans ce sujet aux matrices carrées de taille 3.

Le problème a pour objectif de prouver l'existence de ces matrices et de les expliciter.

On commence par un exemple développé dont le candidat pourra s'inspirer notamment pour la troisième partie.

On utilise toujours la matrice $A = \begin{pmatrix} 3 & 6 & -6 \\ -1 & -9 & 11 \\ 0 & -5 & 7 \end{pmatrix}$.

Le polynôme caractéristique de la matrice A est $\chi_A = (X-2)^2(X+3)$.

On cherche le reste dans la division euclidienne du polynôme X^n par le polynôme χ_A de la forme $aX^2 + bX + c$ où a, b et c vont dépendre de n.

Pour cela on remplace dans la relation $X^n = \chi_A Q + aX^2 + bX + c$, X par -3, puis par 2. Ensuite, on dérive cette expression et on remplace à nouveau X par 2 (Q est le quotient).

On obtient le système suivant $\begin{cases} 9a - 3b + c = (-3)^n \\ 4a + 2b + c = 2^n \\ 4a + b = n2^{n-1} \end{cases}$

admettant pour unique solution $\begin{cases} a = \frac{1}{25}\left((-3)^n - 2^n + 5n2^{n-1}\right) \\ b = \frac{1}{25}\left(-4.(-3)^n + 4.2^n + 5n2^{n-1}\right) \\ c = \frac{1}{25}\left(4.(-3)^n + 21.2^n - 30n2^{n-1}\right) \end{cases}$.

On déduit du théorème de Cayley-Hamilton que pour tout entier naturel n,

$$A^n = aA^2 + bA + cI_3$$

$$= \frac{1}{5}\begin{pmatrix} -(-3)^n + 6.2^n & -6.(-3)^n + 6.2^n & 6.(-3)^n - 6.2^n \\ 2.(-3)^n - 2.2^n + 5n2^{n-1} & 12.(-3)^n - 7.2^n + 5n2^{n-1} & -12.(-3)^n + 12.2^n - 5n2^{n-1} \\ (-3)^n - 2^n + 5n2^{n-1} & 6.(-3)^n - 6.2^n + 5n2^{n-1} & -6.(-3)^n + 11.2^n - 5n2^{n-1} \end{pmatrix}.$$

On pose alors, pour tout réel t, la matrice $\gamma(t) \in \mathscr{M}_3(\mathbb{C})$:

$$\gamma(t) = \frac{1}{5}\begin{pmatrix} -3^t e^{i\pi t} + 6.2^t & -6.3^t e^{i\pi t} + 6.2^t & 6.3^t e^{i\pi t} - 6.2^t \\ 2.3^t e^{i\pi t} - 2.2^t + 5t2^{t-1} & 12.3^t e^{i\pi t} - 7.2^t + 5t2^{t-1} & -12.3^t e^{i\pi t} + 12.2^t - 5t2^{t-1} \\ 3^t e^{i\pi t} - 2^t + 5t2^{t-1} & 6.3^t e^{i\pi t} - 6.2^t + 5t2^{t-1} & -6.3^t e^{i\pi t} + 11.2^t - 5t2^{t-1} \end{pmatrix}.$$

On a les résultats suivants :

- $\gamma(-1)=A^{-1}$,
- pour tout entier naturel p non nul : $\left(\gamma(\frac{1}{p})\right)^p=A$,
- $\exp(\gamma'(0))=A$.

Par exemple,

$$B=\gamma(\tfrac{1}{2})=\tfrac{1}{5}\begin{pmatrix} -i\sqrt{3}+6\sqrt{2} & -6i\sqrt{3}+6\sqrt{2} & 6i\sqrt{3}-6\sqrt{2} \\ 2i\sqrt{3}-2\sqrt{2}+5\frac{\sqrt{2}}{4} & 12i\sqrt{3}-7\sqrt{2}+5\frac{\sqrt{2}}{4} & -12i\sqrt{3}+12\sqrt{2}-5\frac{\sqrt{2}}{4} \\ i\sqrt{3}-\sqrt{2}+5\frac{\sqrt{2}}{4} & 6i\sqrt{3}-6\sqrt{2}+5\frac{\sqrt{2}}{4} & -6i\sqrt{3}+11\sqrt{2}-5\frac{\sqrt{2}}{4} \end{pmatrix}$$

vérifie $B^2=A$.

Deuxième partie

On notera F l'espace vectoriel sur le corps $\mathbb{C}$ des applications de $\mathbb{R}$ dans $\mathbb{C}$ combinaisons linéaires d'applications du type $x\longmapsto x^k\rho^x e^{i\theta x}$ où $k\in\{0,1,2\}$, $\rho\in]0,+\infty[$ et $\theta\in]0,2\pi]$.

(Rappel : pour $\rho\in]0,+\infty[$, $\rho^x=e^{x\ln\rho}$.)

III.6.

III.6.a. Déterminer un élément f de F vérifiant pour tout entier naturel n, $f(n)=\alpha(-3)^n+\beta n^2 2^n$, si α et β sont deux constantes complexes.

III.6.b. Si f est un élément de F et si x_0 est un réel, expliquer pourquoi $x\longmapsto f(x+x_0)$ est encore un élément de F.

III.7.

III.7.a. Soit θ un réel. Démontrer que la suite de nombres complexes $\left(n^2\left(\frac{2}{3}\right)^n e^{i\theta n}\right)$ converge vers 0.

III.7.b. Soit $k_1\in\{0,1,2\}$, $\rho_1\in]0,+\infty[$, $\theta_1\in]0,2\pi]$, $k_2\in\{0,1,2\}$, $\rho_2\in]0,+\infty[$ et $\theta_2\in]0,2\pi]$, $\theta_1\neq\theta_2$.

Démontrer que si α et β sont deux constantes complexes vérifiant, pour tout entier naturel n,

$\alpha n^{k_1}\left(\rho_1\right)^n e^{i\theta_1 n}+\beta n^{k_2}\left(\rho_2\right)^n e^{i\theta_2 n}=0$, alors $\alpha=\beta=0$.

On pourra, par exemple, supposer $\rho_1\leq\rho_2$ et commencer par examiner les cas $\rho_1<\rho_2$ et $\rho_1=\rho_2$.

III.7.c. On admet alors que si f est un élément de F vérifiant pour tout entier naturel n, $f(n)=0$, alors f est l'application nulle.

Que peut-on dire de deux applications f et g de F vérifiant pour tout entier naturel n, $f(n)=g(n)$?

III.8. Dans la suite de cette partie, A est une matrice inversible de $\mathscr{M}_3(\mathbb{R})$.

Expliquer pourquoi on peut trouver 9 applications $\omega_{i,j}$ éléments de F telles que, pour tout entier naturel n, $A^n = \left(\omega_{i,j}(n)\right)_{1\leq i,j\leq 3}$.

Discuter en fonction du nombre de racines du polynôme caractéristique de la matrice A.

On ne demande pas de résoudre des systèmes, une explication de la méthode pourra suffire.

III.9. On pose pour tout réel t, la matrice $\gamma(t) = \left(\omega_{i,j}(t)\right)_{1\leq i,j\leq 3} \in \mathscr{M}_3(\mathbb{C})$.

III.9.a. Quelles sont les matrices $\gamma(0)$ et $\gamma(1)$?

III.9.b. Justifier que, pour tout couple d'entiers naturels (n,m), on a la relation :

$$\gamma(n+m) = \gamma(n)\gamma(m).$$

III.9.c. Pour x réel et m entier naturel, on pose $f(x) = \omega_{i,j}(x+m)$ et $g(x) = \sum_{k=1}^{3} \omega_{i,k}(x)\omega_{k,j}(m)$.

Démontrer que l'on a $f = g$ et en déduire, pour tout entier naturel m, la relation $\gamma(x+m) = \gamma(x)\gamma(m)$.

III.9.d. En déduire que, pour tout couple (x,y) de réels, $\gamma(x+y) = \gamma(x)\gamma(y)$.

III.10. Démontrer que $\gamma(-1) = A^{-1}$ et que, pour tout entier naturel p non nul, $\left(\gamma(\frac{1}{p})\right)^p = A$.

III.11. Justifier que l'application γ définie pour tout réel t par $\gamma(t) = \left(\omega_{i,j}(t)\right)_{1\leq i,j\leq 3}$ est dérivable sur $\mathbb{R}$ et que la fonction γ est une solution de l'équation différentielle

$$u'(t) = \gamma'(0)u(t) \quad \text{vérifiant} \quad u(0) = I_3$$

où la fonction inconnue u vérifie, pour tout réel t, $u(t) \in \mathscr{M}_3(\mathbb{C})$.

Trouver la solution sur $\mathbb{R}$ de l'équation différentielle $u'(t) = \gamma'(0)u(t)$ vérifiant $u(0) = I_3$ et en déduire que l'on a : $\exp(\gamma'(0)) = A$.

Troisième partie : exemple

Soit la matrice $A = \begin{pmatrix} 3 & 0 & 1 \\ 1 & -1 & -2 \\ -1 & 0 & 1 \end{pmatrix}$.

III.12. Donner le polynôme caractéristique de la matrice A.

La matrice A est-elle diagonalisable ?

III.13. Déterminer, par la méthode développée dans ce problème, les éléments suivants :

III.13.a. La matrice A^{-1}.

III.13.b. Une matrice B de $\mathscr{M}_3(\mathbb{C})$ vérifiant $B^2 = A$.

III.13.c. Une matrice M de $\mathscr{M}_3(\mathbb{C})$ vérifiant $\exp(M) = A$.

Fin de l'énoncé

Automate d'exploration de l'hémostase

Partie 1 Présentation du système

1.1 Le support

La société Stago est un laboratoire pharmaceutique de l'industrie du Diagnostic In Vitro (DIV) entièrement dédiée à l'exploration de l'hémostase et de la thrombose. L'hémostase est le processus physiologique qui permet d'interrompre le saignement pour éviter l'hémorragie.

L'objet de cette étude, le STA Compact (figure 1), est un automate de laboratoire destiné à l'analyse de l'hémostase.

Figure 1 - STA compact

Les figures 2 et 3 situent le STA Compact dans son environnement et précisent ses fonctions.

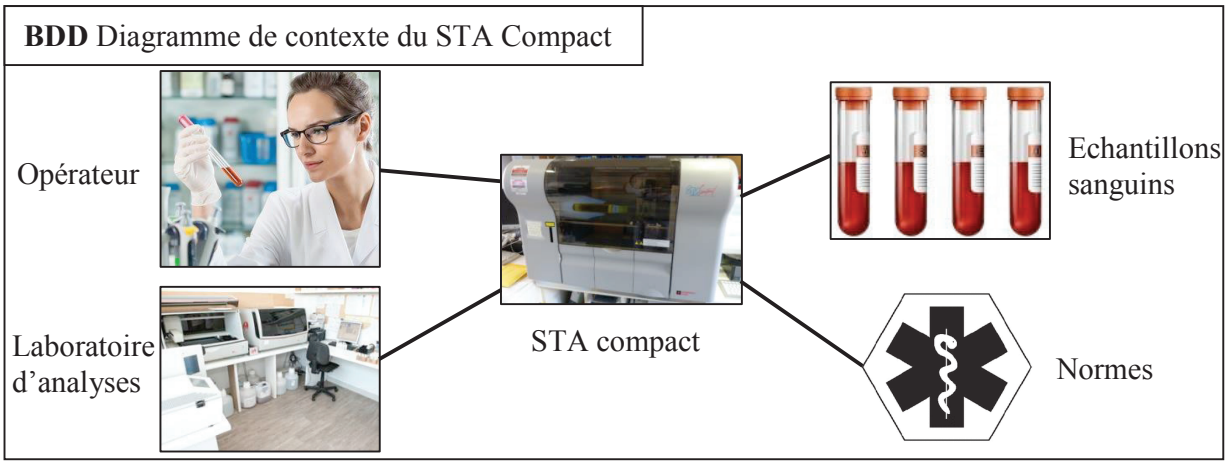


Figure 2 - Diagramme de contexte du STA Compact

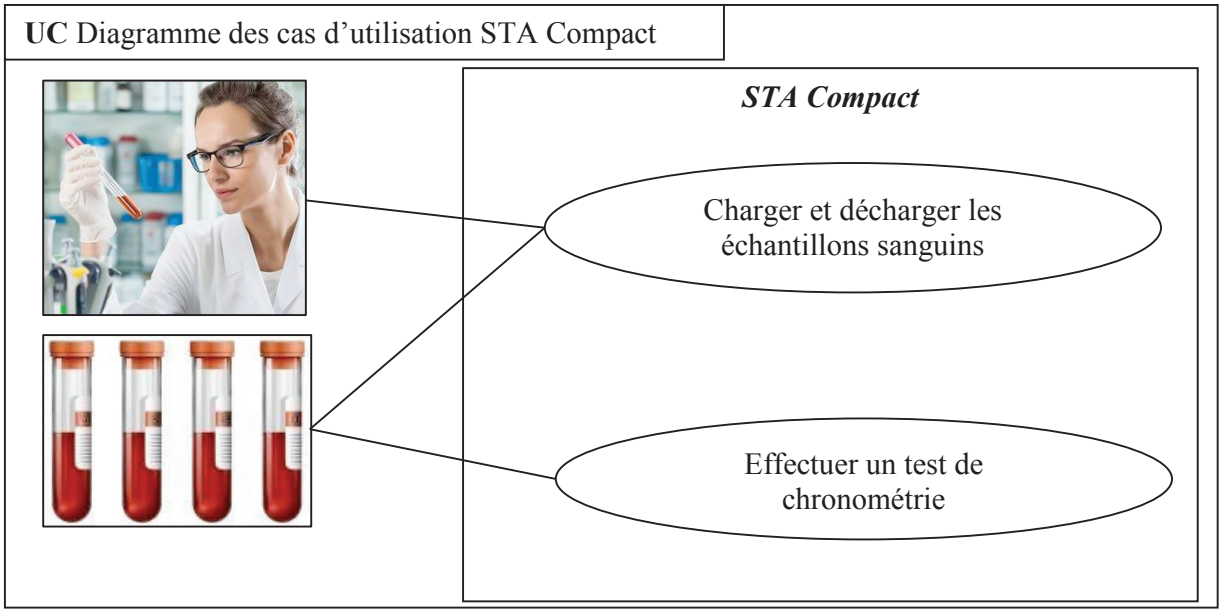


Figure 3 - Diagramme des cas d'utilisation du STA Compact

1.2 La chronométrie

Le STA Compact permet de réaliser, entre autre, des tests de chronométrie afin de mesurer un temps de coagulation.

1.2.1 Principe

Le principe du test de chronométrie est le suivant :

- une dose de réactif est mélangée à une dose de plasma sanguin précédemment étuvée dans une cuvette contenant une bille ;
- l'ensemble est chauffé alors que la bille est mise en oscillation dans le mélange par un champ magnétique ;
- on mesure l'amplitude de l'oscillation qui diminue sensiblement lors d'une variation de viscosité du mélange sang-réactif ;
- le temps écoulé jusqu'à la diminution des oscillations donne le temps de coagulation.

1.2.2 Déroulement d'un test de chronométrie (voir dessin figure 4 et les diagrammes des exigences figures 5 et 6, pages 4 et 5)

Etape 1 : préparation
Les flacons de plasma sanguin à analyser sont placés dans un tiroir, de même que les flacons de réactifs. Afin de garantir la stérilité, les flacons sont fermés par un opercule qui sera percé par l'aiguille de prélèvement.
Des cuvettes, contenant une bille, sont au préalable clipsées sur une bande, elle-même enroulée sur une bobine de stockage.
Pour chaque test, une cuvette (et sa bille) est déclipsée et placée dans la zone d'étuvage à l'aide d'un vérin (non étudié ici).

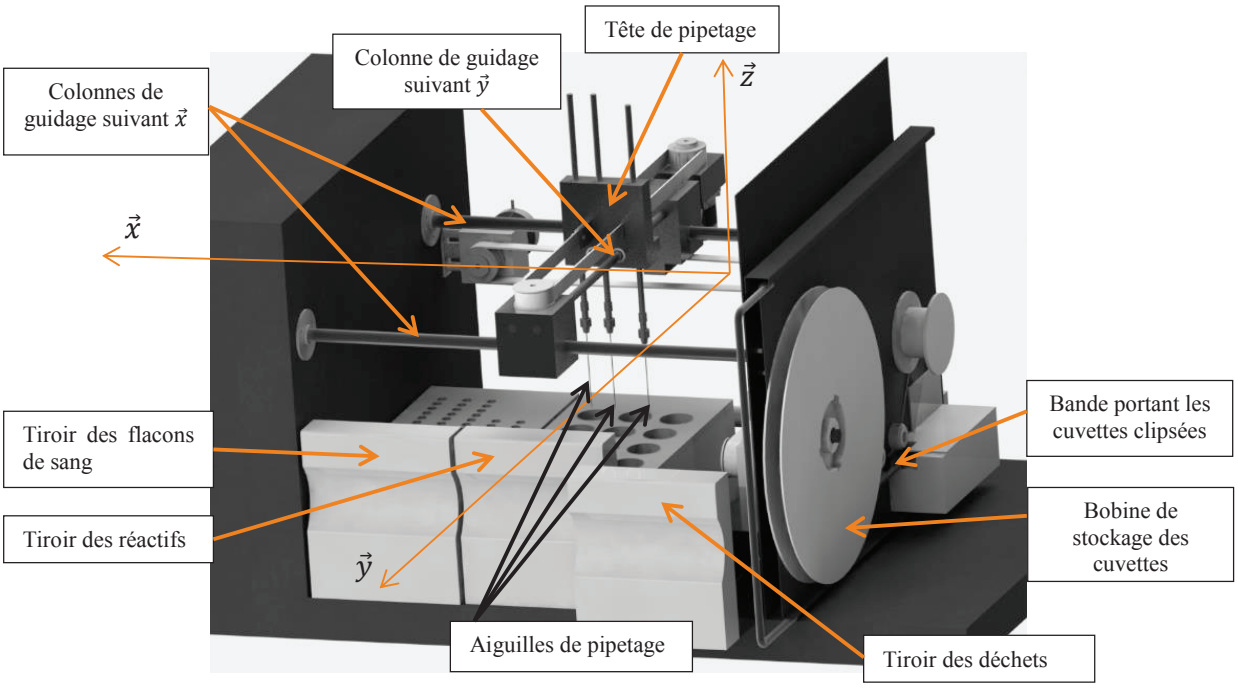


Figure 4 - Structure interne du STA Compact

Etape 2 : prélèvement des produits
Les aiguilles de prélèvement des doses de plasma et de réactifs sont reliées à la tête de pipetage. Elles peuvent avoir un mouvement de translation verticale (selon la direction $\vec{z}$) par rapport à cette tête. Deux types de réactifs sont utilisés. La tête de pipetage possède donc trois aiguilles : une pour le sang et une par type de réactif.

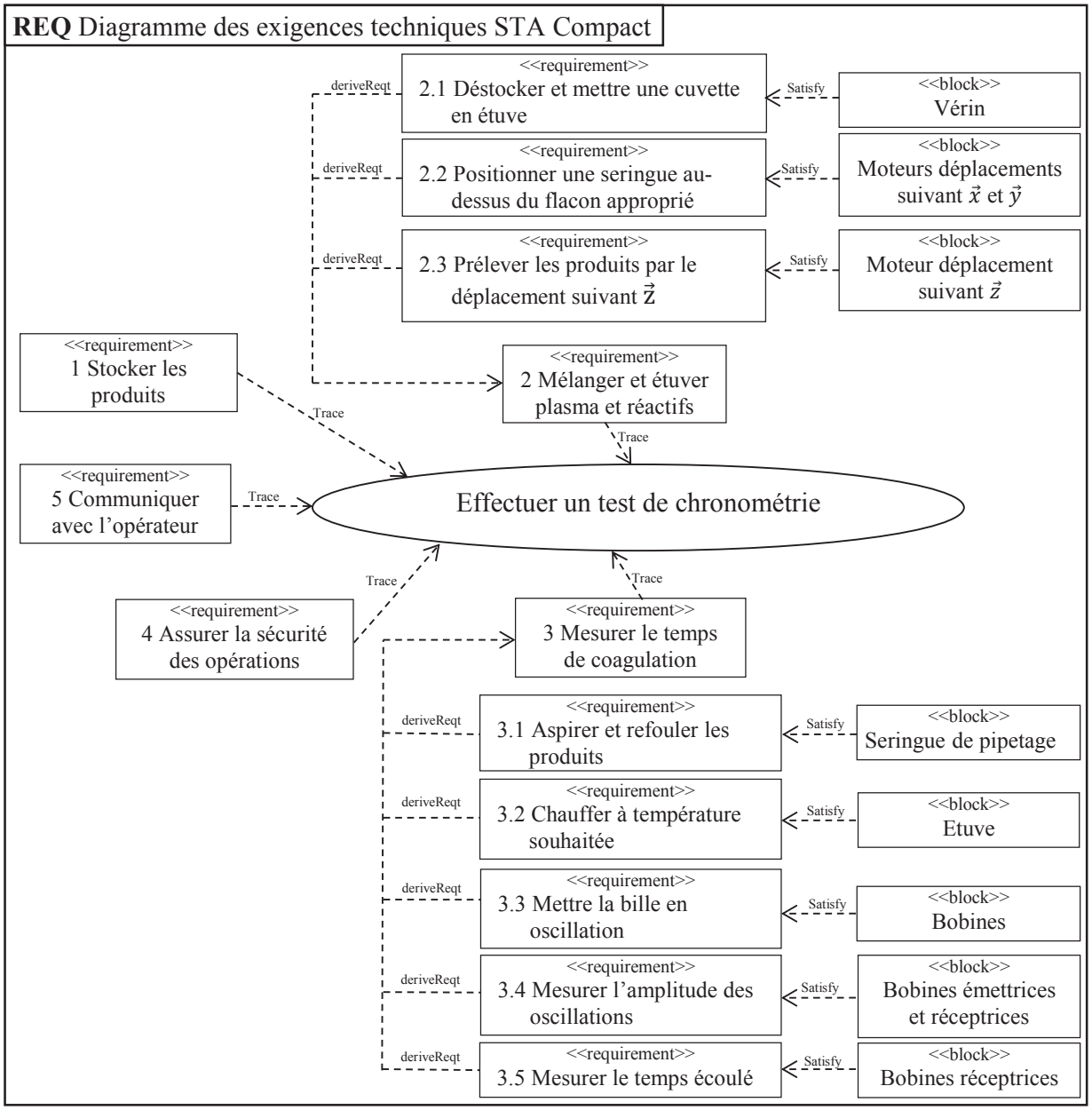


Figure 5 - Diagramme des exigences techniques avec les solutions technologiques

Successivement, pour chaque produit (plasma puis réactifs), la tête de pipetage est positionnée au-dessus du flacon approprié, l'aiguille correspondante prélève la quantité nécessaire, puis l'ensemble tête de pipetage/aiguilles vient déposer le produit dans la cuvette d'analyse. Les aiguilles sont ensuite plongées dans un flacon de nettoyage. L'aspiration et le refoulement des liquides (plasma et réactifs) se font à l'aide d'une même seringue de pipetage motorisée (non représentée).

Etape 3 : mesure
Lorsque le mélange est réalisé dans la cuvette, celle-ci est placée dans la zone de mesure. Deux bobines motrices provoquent l'oscillation de la bille. Deux autres bobines, l'une émettrice et l'autre réceptrice, effectuent la mesure des amplitudes d'oscillation.
Le test terminé, la cuvette est placée dans un tiroir à déchets.
Le système est connecté à un poste informatique permettant les échanges d'ordres et de comptes-rendus avec le manipulateur.

Le diagramme de la figure 5 (page 4) présente un extrait du diagramme des exigences techniques assorti des éléments qui permettent de satisfaire ces exigences.
La figure 6 présente un extrait du diagramme des exigences, se limitant à celles pour lesquelles nous allons étudier certains critères.

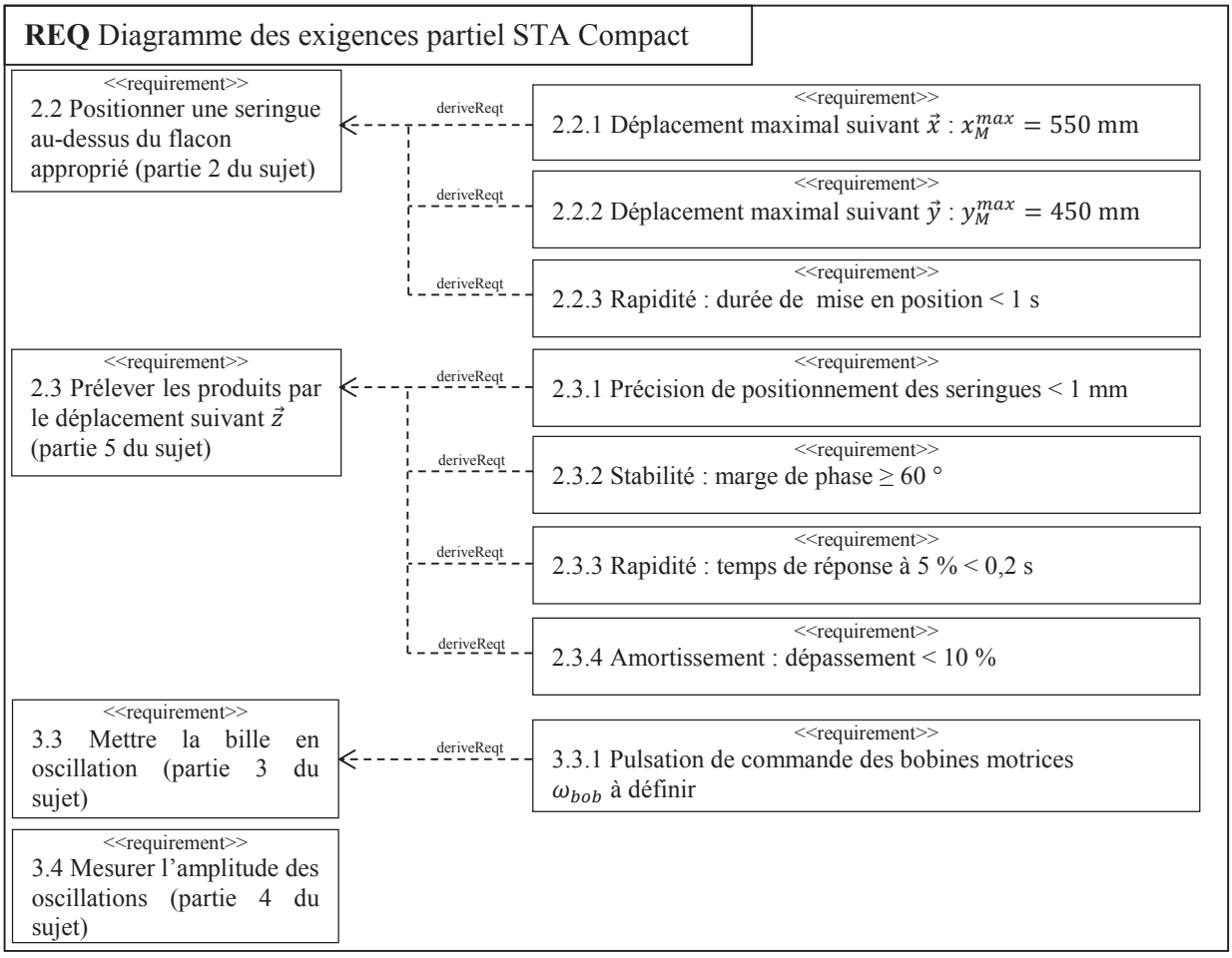


Figure 6 - Diagramme partiel des exigences

L'étude comporte 4 parties :

- la partie 2 analyse les solutions techniques mises en place pour satisfaire à l'exigence 2.2 « Positionner une seringue au-dessus du flacon approprié » et plus particulièrement à la loi de commande des moteurs ;
- les parties 3 et 4 analysent les solutions techniques et informatiques pour satisfaire à l'exigence 3 « Mesurer le temps de coagulation » ;
- la partie 5 analyse la capacité du système à satisfaire à l'exigence 2.3 « Prélever les produits » et plus particulièrement l'exigence relative à la précision du volume prélevé.

Partie 2 Analyse de l'exigence 2.2 « Positionner une seringue au-dessus d'un flacon »

Objectif : déterminer la loi de commande en vitesse optimale des axes de translation $\vec{x}$ et $\vec{y}$.

2.1 Mise en situation

La tête de pipetage, dont le diagramme de bloc interne est fourni figure 8 (page 6), est guidée en translation suivant $\vec{y}$ par rapport à une traverse intermédiaire, elle-même guidée en translation suivant $\vec{x}$ par rapport au bâti (figure 7, page 6).

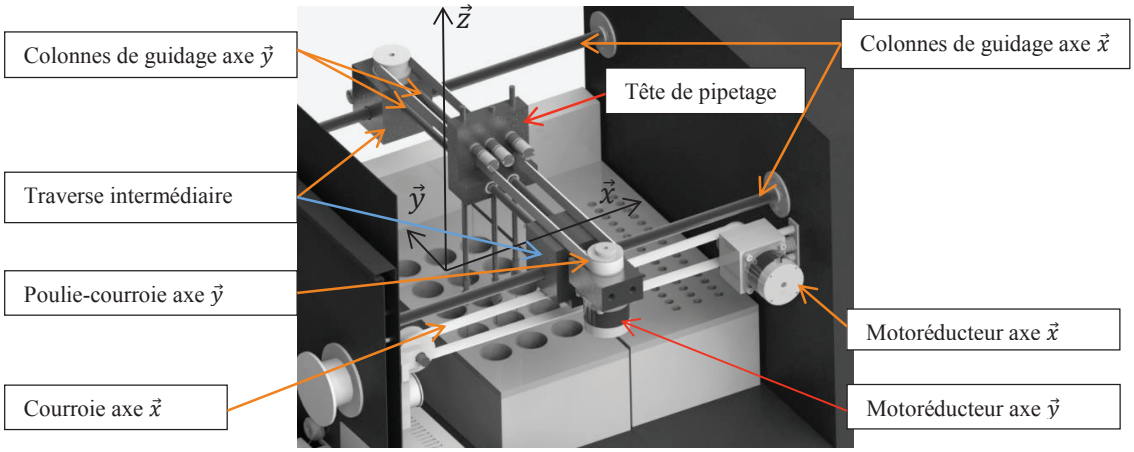


Figure 7 - Système de guidage suivant $\vec{x}$ et $\vec{y}$

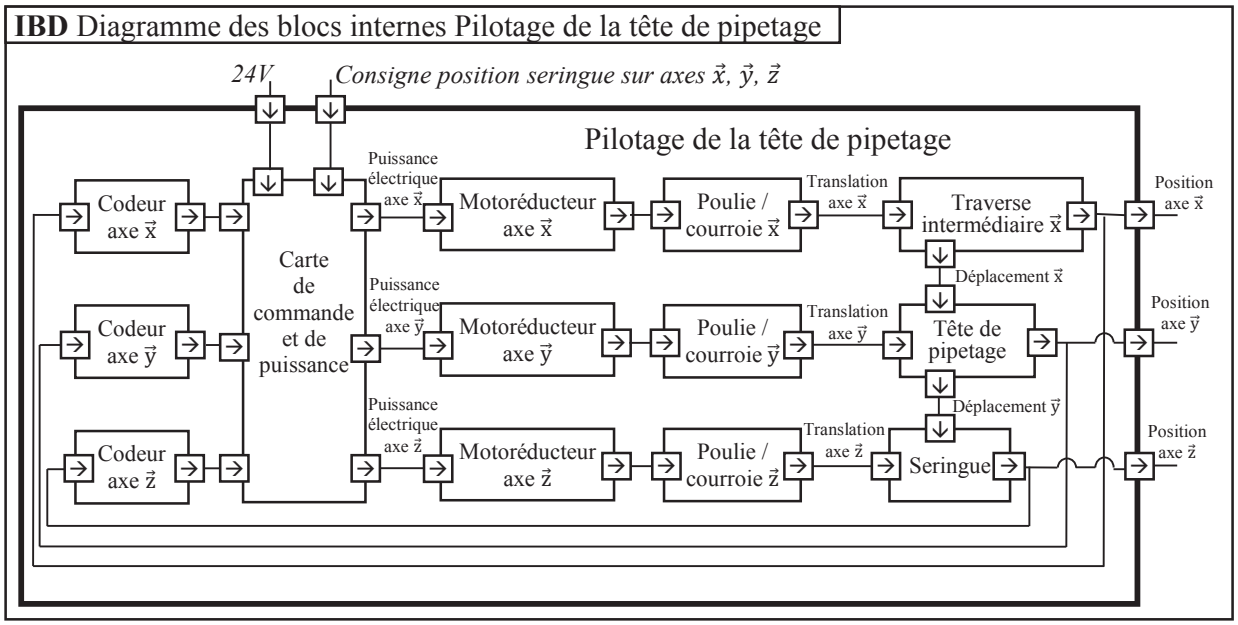


Figure 8 - Diagramme de bloc interne de la tête de pipetage

A la mise en service de l'appareil, la tête de pipetage est placée à une position de référence.
Après chaque déplacement, la tête revient à cette position de référence qui permet le nettoyage des aiguilles.
On se propose d'étudier le protocole de déplacement de cette tête pour atteindre une position définie par un point M de coordonnées $(x_M, y_M, 0)$ par rapport à la position de référence.
Chaque axe est mis en mouvement par un motoréducteur et un système poulie-courroie.
Les vitesses de translation maximum sur chaque axe (V_{max}^x et V_{max}^y) sont obtenues pour les vitesses de rotation maximum des moteurs. Les chaînes cinématiques sont identiques : nous avons donc $V_{max}^x = V_{max}^y = V_{max}$.
Soient V_M^x et V_M^y (V_M^x et $V_M^y \leq V_{max}$) les vitesses maximum suivant chaque axe lors du déplacement pour atteindre le point M de coordonnées $(x_M, y_M, 0)$.

2.2 Optimisation de la commande

Afin de valider le choix des moteurs, on étudie le déplacement sur l'axe $\vec{x}$ qui est le plus grand. On note V^x la vitesse de la tête de pipetage selon cet axe.
On rappelle que la distance maximum à parcourir est $x_M^{max} = 550$ mm en 1 seconde.

La loi de commande sur chaque axe est définie par un trapèze de vitesse (figure 9) avec les temps d'accélération et de décélération (T_a) identiques. De plus, les moteurs se mettent en route et s'arrêtent en même temps. T est la durée totale du déplacement. Nous allons chercher à optimiser cette loi de commande de sorte que le moteur fournisse une puissance instantanée minimale.

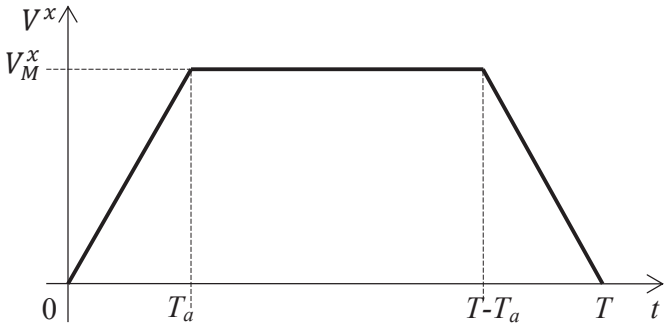


Figure 9 - Loi de commande de vitesse en trapèze

Le modèle de calcul pour cette commande d'axe est le suivant :
- le mouvement de rotation du moteur (vitesse ω_m^x) est transformé en mouvement de translation (vitesse V^x) ;
- le rapport de transmission de la chaîne cinématique est $\lambda = \frac{V^x}{\omega_m^x}$;
- la distance à parcourir est x_M^{max} ;
- l'inertie équivalente de l'ensemble des pièces en mouvement ramenée à l'arbre moteur est J_e ;
- pour cette question, les frottements et la pesanteur sont négligés, il n'y a donc pas de couple résistant.

Question 1. Exprimer la vitesse maximale V_M^x en fonction de x_M^{max}, T et T_a.

Question 2.
- Par application du théorème de l'énergie cinétique sur l'ensemble des pièces en mouvement, exprimer le couple moteur C_m en fonction de V^x, T_a, J_e et λ durant les trois phases du mouvement.
- Préciser à quel(s) instant(s) t la puissance fournie par le moteur est maximale (P_{max}).
- Exprimer cette puissance P_{max} en fonction de V_M^x, λ, J_e et T_a.
- Donner alors l'expression de P_{max} en fonction de x_M^{max}, λ, J_e, T et T_a.

Question 3. A partir de cette expression, montrer que P_{max} est minimale pour un réglage du temps d'accélération T_a tel que $T_a = \frac{T}{3}$.

Pour cette nouvelle commande avec $T_a = \frac{T}{3}$, on cherche à valider le choix du moteur en étudiant le déplacement maximum suivant $\vec{x}$.
Les caractéristiques de la chaîne cinématique sont :
- vitesse maximale du moteur : $N_{max}^{mot} = 4\,150$ tr · min^{-1} ;
- rapport de réduction du réducteur $k = 1/10$;
- rayon de poulie $R_p = 20$ mm.

Question 4. Déterminer la vitesse de rotation maximum ω_{max}^x que doit atteindre le moteur. Le choix de celui-ci est-il validé ?

Partie 3 Analyse de l'exigence 3.3 « Mettre la bille en oscillation »

Objectif : déterminer la pulsation optimale des bobines motrices.

3.1 Mise en situation

Le principe de la chronométrie consiste à mesurer la variation de l'amplitude d'oscillation d'une bille placée dans la cuvette de mesure (figures 10 et 11).
La bille, roulant sans glisser sur le fond cylindrique de la cuvette, est mise en mouvement par un champ magnétique variable induit par deux bobines motrices placées de part et d'autre de la tête de mesure.
L'amplitude des oscillations est mesurée par deux autres bobines, l'une émettrice, l'autre réceptrice. Après amplification du signal mesuré, on obtient un signal quasi-sinusoïdal, reflet de l'oscillation de la bille. A viscosité constante, on obtient un balancement pendulaire constant de la bille. Quand la viscosité augmente (phénomène de coagulation), l'amplitude d'oscillation de la bille varie.

Pour chaque mesure, le champ magnétique est ajusté en fonction de la viscosité initiale du milieu et du type de test.

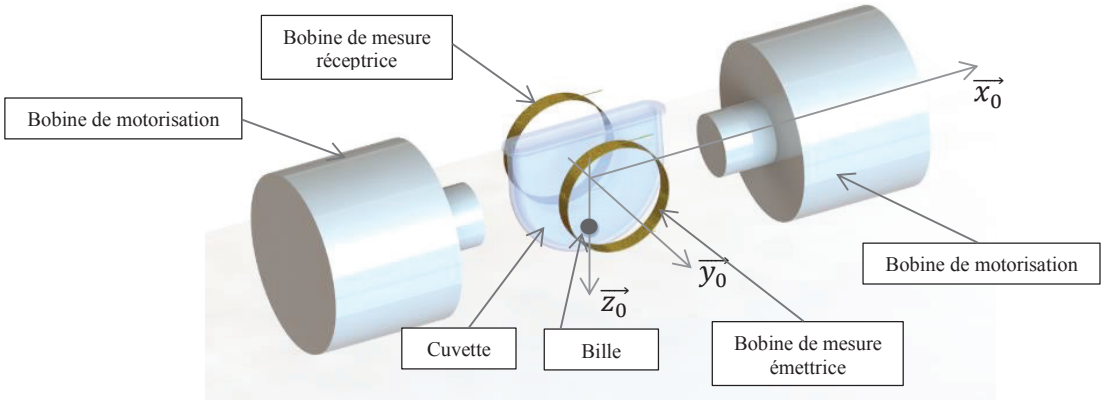


Figure 10 - Ensemble cuvette + bille avec bobines motrices et bobines de mesure

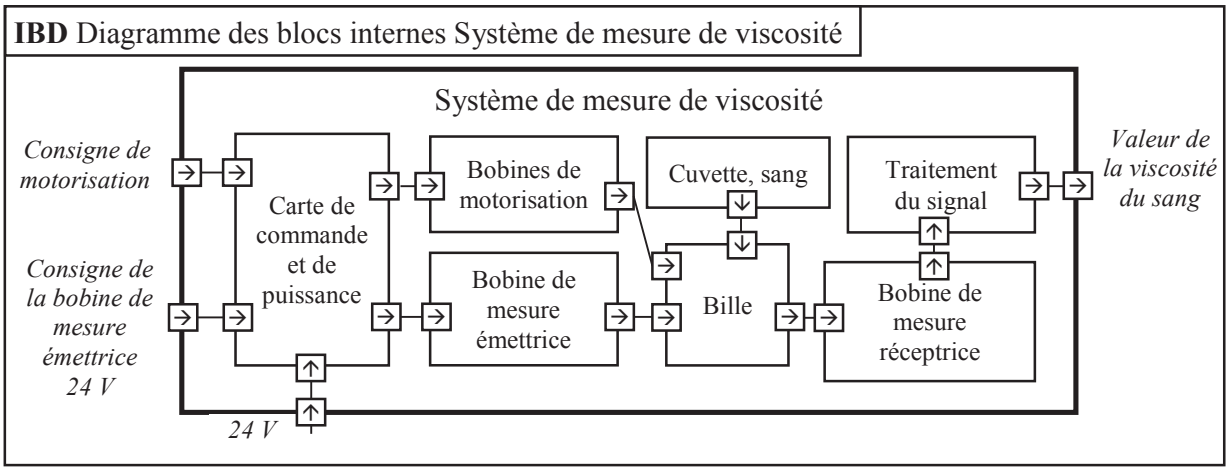


Figure 11 - Diagramme de bloc interne du système de mesure

Dans un premier temps, on se propose de modéliser le comportement de la bille pour en déduire le réglage de la commande des bobines motrices (partie 3.3, page 9).
Dans un second temps, un algorithme détermine le temps de coagulation à partir de l'amplitude des oscillations. Celui-ci sera défini paragraphe 4.1, page 14 (partie informatique).

3.2 Hypothèses de calculs

Le schéma de calcul est donné figure 12.
Hypothèses :

- la bille de masse m, de centre de masse G, de rayon r, roule sans glisser sur un rail circulaire de rayon R dans le plan $(O, \overrightarrow{x_0}, \overrightarrow{z_0})$. Cette hypothèse de roulement sans glissement sera vérifiée paragraphe 3.4, page 12 (partie informatique) ;
- I est le point de contact entre la bille et le rail circulaire ;
- la position de la bille sur le rail est repérée par : $\theta = (\overrightarrow{z_0}, \overrightarrow{z_1}) = (\overrightarrow{x_0}, \overrightarrow{x_1})$.

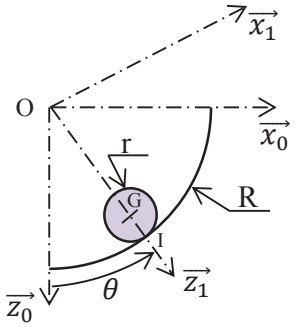


Figure 12 - Bille en contact avec le rail de la cuvette

On note :

- $\{T(rail \to bille)\} = \begin{Bmatrix} -N_I\overrightarrow{z_1} + T_I\overrightarrow{x_1} \\ \vec{0} \end{Bmatrix}_I$, le torseur associé à l'action mécanique du rail sur la bille ;
- f le coefficient d'adhérence au contact bille/cuvette : $f = 0{,}1$;
- $\{T(bob \to bille)\} = \begin{Bmatrix} \vec{F}(bob \to bille) = F(t)\overrightarrow{x_0} \\ \vec{0} \end{Bmatrix}_G$, le torseur associé à l'effort résultant des deux bobines de motorisation sur la bille, avec $F(t) = F_0 \sin(\omega_{bob} t)$;
- $\{T(fluide \to bille)\} = \begin{Bmatrix} \overrightarrow{F_v}(fluide \to bille) = -f_v\,\vec{V}(G \in \text{bille}/0) \\ \vec{0} \end{Bmatrix}_G$, le torseur associé à l'action du fluide sur la bille induite par la viscosité. On se place dans l'hypothèse simplificatrice d'un écoulement laminaire pour lequel le modèle de Stokes est applicable : le coefficient de frottement visqueux vaut alors $f_v = 6\pi.r.\eta$ où η est la viscosité du sang qui varie lors de la coagulation ;
- $\{T(g \to bille)\} = \begin{Bmatrix} m \cdot g\,\overrightarrow{z_0} \\ \vec{0} \end{Bmatrix}_G$, le torseur associé à l'action de la pesanteur sur la bille ;
- $\{V(\text{bille}/0)\} = \begin{Bmatrix} \vec{\Omega}(\text{bille}/0) = \omega_b\overrightarrow{y_0} \\ \vec{V}(G \in \text{bille}/0) = v\overrightarrow{x_1} \end{Bmatrix}_G$, le torseur cinématique de la bille par rapport au rail 0 ;
- $J = \frac{2}{5} m \cdot r^2$, le moment d'inertie de la bille autour de l'axe $(G, \overrightarrow{y_0})$;
- $R = \|\overrightarrow{OI}\|$, le rayon du rail, $r = \|\overrightarrow{GI}\|$, le rayon de la bille.

On notera $F(p)$ la transformée de Laplace de la fonction $f(t)$ où p représente la variable de Laplace.

3.3 Analyse

Question 5. En exprimant la condition de roulement sans glissement en I, déterminer ω_b et v, les composantes du torseur cinématique en G de la bille par rapport au rail 0, en fonction de $\dot{\theta}$, r et R.

Question 6. En justifiant clairement la démarche et les théorèmes utilisés :

- Montrer que les efforts normal N_I et tangentiel T_I du rail sur la bille sont liés à l'angle θ par les équations suivantes :

$$N_I = F(t)\sin\theta + mg\cos\theta + m(R - r)\dot{\theta}^2 \qquad (1)$$

$$T_I = \frac{2}{5} m(r - R)\ddot{\theta} \qquad (2)$$

- Justifier l'équation du mouvement de la bille :

$$\frac{7}{5}m(R-r)\ddot{\theta} + f_v(R-r)\dot{\theta} + mg\sin\theta = F(t)\cos\theta \qquad (3)$$

Question 7. θ étant petit, linéariser l'équation du mouvement puis en déduire la fonction de transfert $H(p) = \frac{\Theta(p)}{F(p)}$. Mettre $H(p)$ sous la forme canonique d'un système du second ordre dont on donnera les expressions du gain statique K_s, de la pulsation propre non amortie ω_0 et du coefficient d'amortissement ξ en fonction de f_v, R, r, m et g.

Question 8. On prendra les valeurs numériques suivantes pour cette question : $\omega_0 = 21{,}8\ \text{rad}\cdot s^{-1}$; $K_s = 25\ \text{N}^{-1}$; $\xi = 4 \cdot f_v$.
Tracer, sur le document réponse, le diagramme asymptotique de Bode en gain, ainsi que l'allure du diagramme réel pour les valeurs suivantes du coefficient de frottement visqueux f_v :
$f_v = 0{,}005$; $f_v = 0{,}05$; $f_v = 0{,}2$.

Question 9. La sollicitation des bobines est sinusoïdale : $F(t) = F_0 \sin(\omega_{bob} t)$. Préciser, en justifiant votre réponse, la valeur à laquelle il faut régler la pulsation ω_{bob} pour pouvoir observer, au mieux, l'évolution du coefficient de frottement f_v.

Question 10.

- Exprimer, pour un système du second ordre, en fonction de ξ, le rapport des amplitudes de sortie à $\omega \to 0$ et $\omega = \omega_0$ pour une même amplitude du signal d'entrée.
- Les figures 13 a, b, c, d (pages 10 et 11) représentent, avec f_v constant, l'évolution de la position de la bille θ [°] en fonction du temps t [s] pour différentes valeurs de pulsation ω_{bob}. A partir de ces courbes et des résultats précédents, déterminer la valeur du coefficient d'amortissement ξ.
- En déduire la valeur numérique du coefficient de viscosité η du sang correspondant.
- A partir de ces analyses, en justifiant votre réponse, donner l'allure de la courbe θ en fonction de t obtenue à la pulsation ω_0 lorsque la viscosité du sang varie au fur et à mesure de la coagulation (si l'on suppose que f_v augmente avec la coagulation).

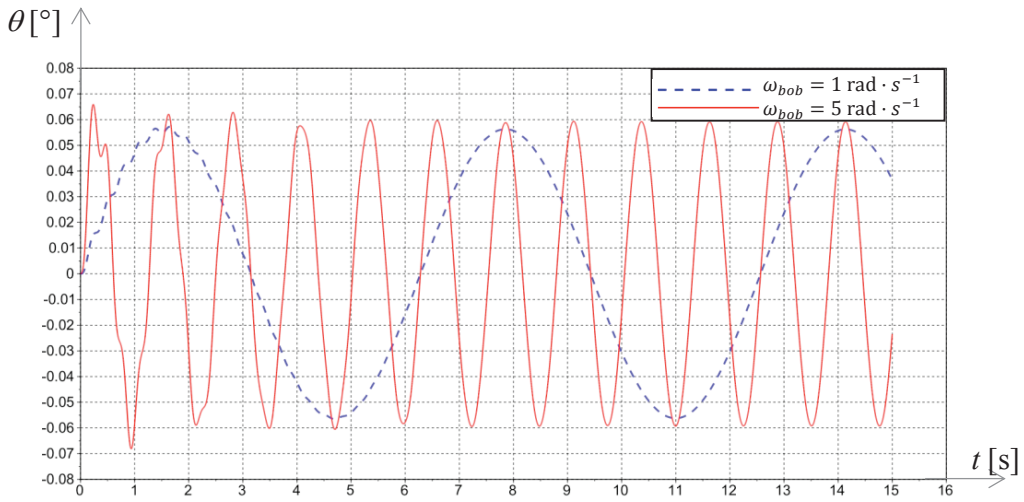


Figure 13 a - θ[°] en fonction de t [s] pour $\omega_{bob} = 1$ et 5 rad $\cdot$ s^{-1}

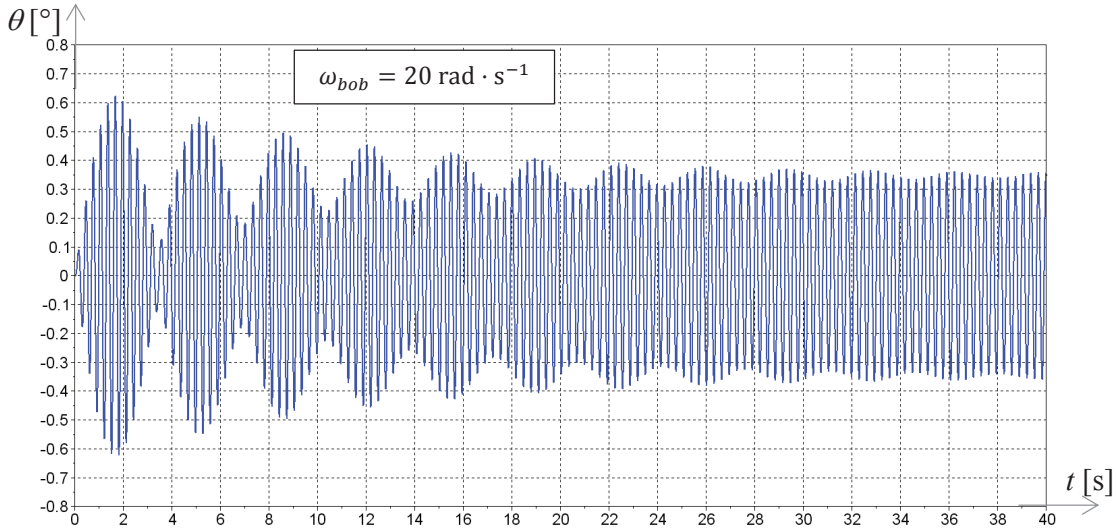


Figure 13 b - θ [°] en fonction de t [s] pour $\omega_{bob} = 20\ \mathrm{rad \cdot s^{-1}}$

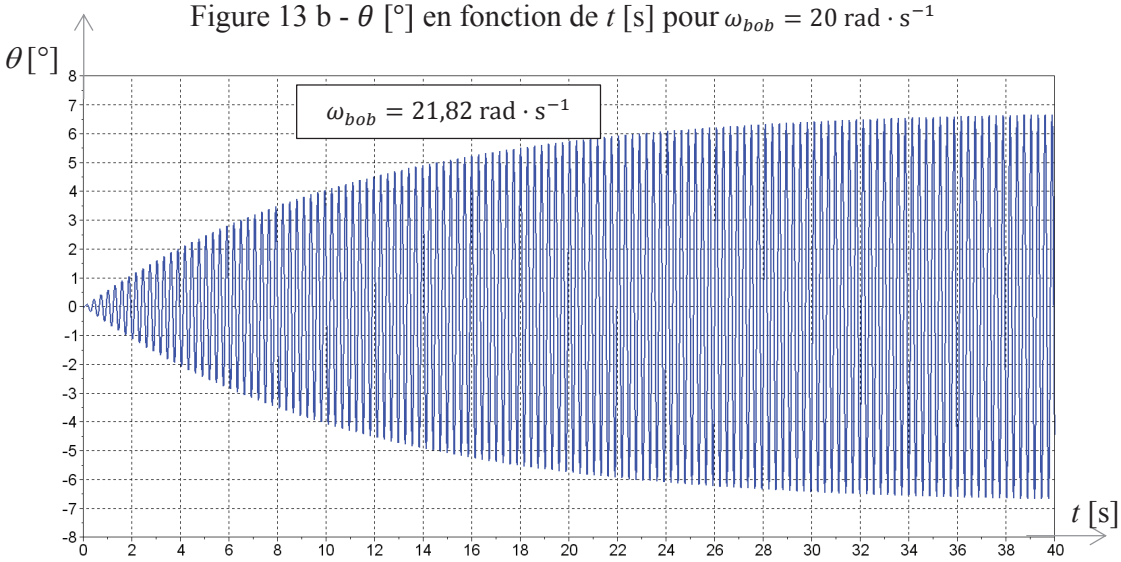


Figure 13 c - θ[°] en fonction de t [s] pour $\omega_{bob} = 21{,}82\ \mathrm{rad \cdot s^{-1}}$

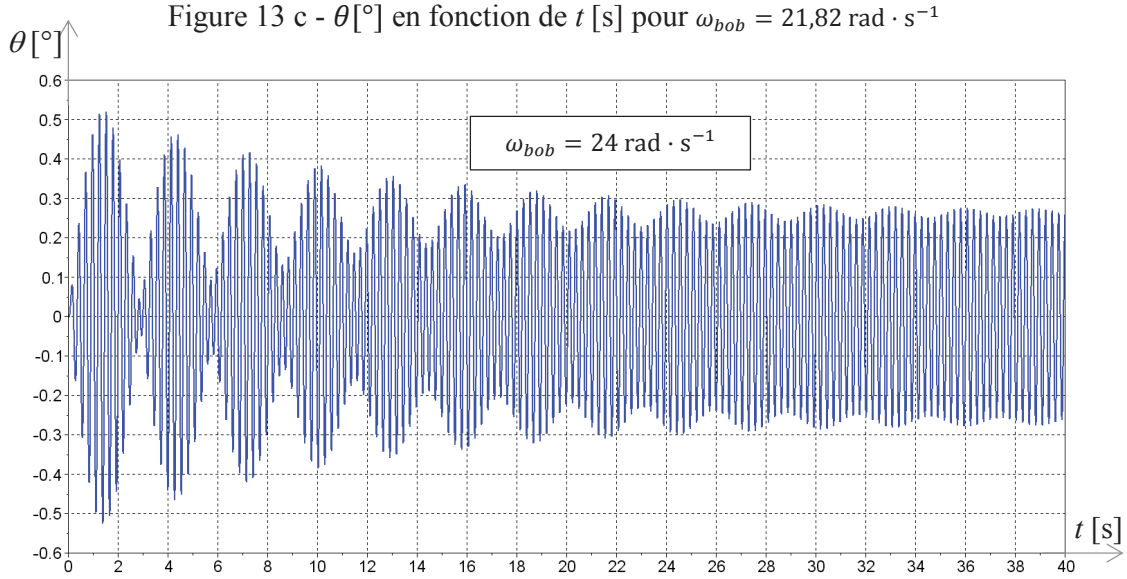


Figure 13 d - θ[°] en fonction de t [s] pour $\omega_{bob} = 24\ \mathrm{rad \cdot s^{-1}}$

3.4 Validation des hypothèses (partie informatique)

Objectif : on se propose de vérifier la validation de l'hypothèse de linéarisation faite Question 7 (page 10), ainsi que l'hypothèse de roulement sans glissement faite Question 5 (page 9).

Ce paragraphe fait plus spécifiquement appel aux compétences acquises dans le cadre du programme d'informatique. Le candidat est libre de choisir entre les langages Python et Scilab pour l'écriture des fonctions demandées.
On désigne par tableaux les éléments qui pourraient être de type list ou ndarray sous Python ou matrice sous Scilab.

Cette étude se fera en 3 temps :
- résolution de l'équation différentielle non linéaire ;
- validation de la linéarisation ;
- validation de l'hypothèse de roulement sans glissement.

L'étude dynamique a permis de déterminer :
- les équations liant N_I et T_I en fonction de θ et de ses dérivées :

$$N_I = F_0 \sin(\omega_{bob} t) \sin\theta + mg\cos\theta + m(R-r)\dot{\theta}^2 \quad (1)$$

$$T_I = \frac{2}{5} m(r-R)\ddot{\theta} \; ; \quad (2)$$

- l'équation différentielle non linéaire régissant la position de la bille :

$$\frac{7}{5} m(R-r)\ddot{\theta} + f_v(R-r)\dot{\theta} + mg\sin\theta = F_0 \sin(\omega_{bob} t)\cos\theta \quad (3)$$

avec : $\dot{\theta}(0) = 0$ et $\theta(0) = 0$.

En posant Y tel que $Y(t) = \begin{pmatrix} \theta(t) \\ \dot{\theta}(t) \end{pmatrix}$, l'équation différentielle (3) peut se mettre sous la forme du problème de Cauchy : $\frac{dY(t)}{dt} = F(t, Y(t))$ avec

$$F(t, Y(t)) = \begin{pmatrix} \dot{\theta}(t) \\ \frac{5}{7m(R-r)} \left(F_0 \sin(\omega_{bob} t)\cos\theta(t) - f_v(R-r)\dot{\theta}(t) - mg\sin\theta(t)\right) \end{pmatrix}$$

et $Y(0) = Y_0 = \begin{pmatrix} \theta(0) \\ \dot{\theta}(0) \end{pmatrix}$.

La réponse $\theta(t)$ recherchée sur l'intervalle $[0, T_{maxi}]$ sera obtenue par la méthode d'Euler explicite.
Le pas de calcul, noté h, sera choisi constant.
L'intervalle de temps discrétisé est alors représenté par le tableau :
$T = [t_0 = 0, t_1, \dots, t_{N-1} = T_{maxi}]$.
Pour chaque instant t_i, une valeur approchée Y_i de la solution $Y(t_i)$ de l'équation différentielle est recherchée. L'ensemble des Y_i et des temps t_i représente N vecteurs de dimensions 3, qui seront stockés en mémoire sous la forme :

- soit d'un tableau : $SY = \begin{bmatrix} 0 & \theta(0) & \dot{\theta}(0) \\ t_1 & \theta(t_1) & \dot{\theta}(t_1) \\ \vdots & \vdots & \vdots \\ T_{maxi} & \theta(T_{maxi}) & \dot{\theta}(T_{maxi}) \end{bmatrix}$;
- soit d'une liste de listes :

$SY = \left[[0, \theta(0), \dot{\theta}(0)], [t_1, \theta(t_1), \dot{\theta}(t_1)], \dots, [T_{maxi}, \theta(T_{maxi}), \dot{\theta}(T_{maxi})]\right]$.

Les grandeurs physiques m, R, r, f_v, g, F_0 et ω_{bob} sont appelées m, R, r, fv, g, F0, wbob et ont été initialisées en début de programme.

Question 11. Donner la relation de récurrence qui lie Y_{i+1} à Y_i , $F(t_i, Y_i)$ et le pas de calcul h.

Question 12. Ecrire une fonction **fi**(t_i, Y_i), qui prend pour arguments t_i la valeur du temps discrétisé et Y_i la valeur du vecteur Y au temps discrétisé t_i et qui retourne la valeur de $F\big(t_i, Y(t_i)\big)$.

Question 13. Ecrire une fonction **Euler** (Y_{ini}, h, T_{maxi}, F) qui prend en paramètres Y_{ini} un tableau à 2 dimensions (ou une liste de listes) contenant la condition initiale Y_0, h le pas de calcul, T_{maxi} le temps final de calcul et F la fonction du problème de Cauchy. Cette fonction renverra le tableau SY. L'appel à cette fonction dans le programme se fera par la commande $SY =$ **Euler**(Y_0, h, T_{maxi}, fi).

Question 14. Donner la complexité de la fonction **Euler** pour T_{maxi} fixée. En déduire comment évolue le temps de calcul quand le pas h est divisé par 10.

Afin de vérifier l'hypothèse du roulement sans glissement de la bille sur le rail, on doit vérifier qu'en chaque instant t_i on a $\left|\frac{T_I}{N_I}\right|_i \leq f$ ($f = 0{,}1$ coefficient d'adhérence de la bille sur le rail).

N_I et T_I sont définis par les équations (1) et (2).

Question 15. Ecrire une fonction **VerifRSG**(SY, f) qui prend pour argument le tableau SY et f (coefficient d'adhérence de la bille sur le rail) et qui renvoie **True** si le critère d'adhérence est vérifié (c'est-à-dire si $\left|\frac{T_I}{N_I}\right| \leq f, \forall t_i$), **False** sinon.

Les figures 14 et 15 ci-dessous sont issues de la résolution numérique.

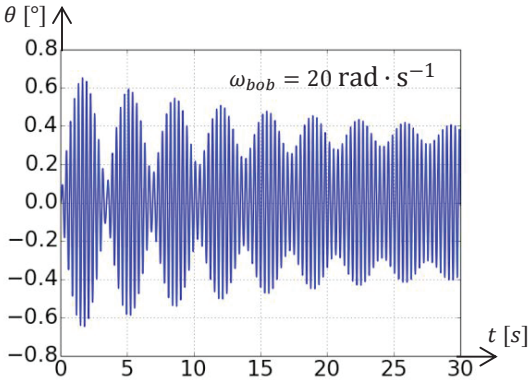


Figure 14 a

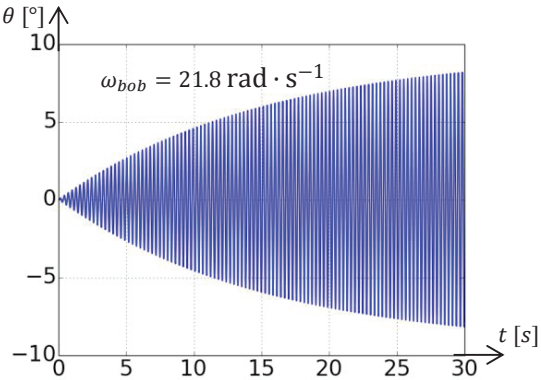


Figure 14 b

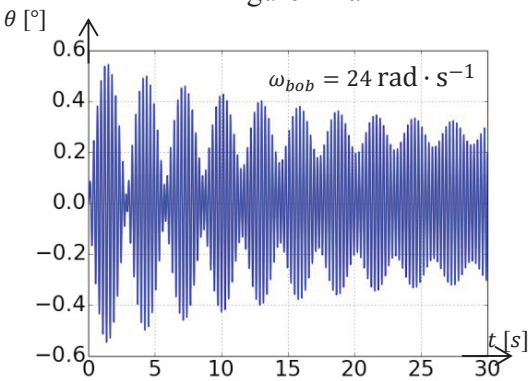


Figure 14 c

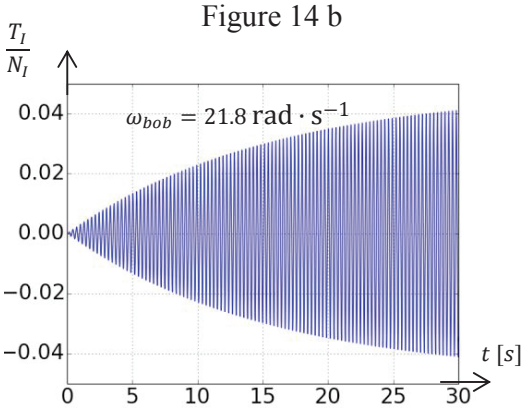


Figure 15

Figures 14 a, b, c - θ[°] en fonction de t [s] pour différentes valeurs de ω_{bob}
Figure 15 - T_I/N_I en fonction de t [s]

Question 16. A partir de ces résultats, conclure quant aux hypothèses précédentes (validation de la linéarisation, hypothèse de roulement sans glissement).

Partie 4 Analyse de l'exigence 3.4 « Mesurer l'amplitude des oscillations » (partie informatique)

Objectif : écrire un algorithme permettant de mesurer le temps de coagulation à partir des mesures réalisées par les deux bobines émettrice et réceptrice.

Cette partie fait plus spécifiquement appel aux compétences acquises dans le cadre du programme d'informatique. Le candidat est libre de choisir entre les langages Python et Scilab pour l'écriture des fonctions demandées.
Dans la suite, on désigne par tableaux les éléments qui pourront être définis de type list ou ndarray sous Python ou matrice sous Scilab (au choix du candidat).

L'étude se fait en deux temps :
- calcul du temps de coagulation à partir des valeurs stockées dans un tableau de mesures ;
- étude de la gestion du tableau de mesures.

4.1 Calcul du temps de coagulation

Afin de mesurer le temps de coagulation, les signaux issus des bobines sont amplifiés, filtrés, échantillonnés puis stockés dans un tableau nommé *M*. La période d'échantillonnage est notée *Te*.

On considèrera que la mesure s'effectue sur une durée fixe : la taille de *M* est fixe et égale à N_{maxi}.

Le temps de coagulation se détermine à partir de la valeur efficace du signal sur une durée donnée.

La **valeur efficace** (appelée aussi Root Mean Square, ou moyenne quadratique) est définie par : $X_{eff}(i) = \sqrt{\frac{1}{N}\sum_{k=i}^{N-1+i} {x_k}^2}$ où $\{x_k\}$ sont les valeurs des *N* mesures successives considérées. Elle est donc l'image de l'amplitude des oscillations à l'instant considéré.

Question 17. Ecrire la fonction **Valeur_efficace**(*T, a, b*), qui prend pour argument un tableau *T*, les entiers *a* et *b* et renvoie la valeur efficace des *b* éléments consécutifs du tableau à partir de l'indice *a*. On supposera pour cette question que *a* et *b* sont tels qu'il n'y a pas de débordement d'indice possible. Donner la complexité de la fonction **Valeur_efficace**.

Dans la suite, on supposera $N \ll N_{maxi}$.

Le **temps de coagulation** est défini comme le temps au bout duquel la valeur efficace des signaux a atteint 70 % de la valeur efficace initiale.

Si cette valeur n'est pas atteinte sur la totalité du tableau M, on considèrera que la mesure a échoué.

L'algorithme de calcul est le suivant :

Données : *M*, *Te*, *N* : le tableau de mesures, la période d'échantillonnage (en millisecondes), le nombre de mesures à considérer.
Résultat : *Tc* le temps de coagulation (en secondes).

```
1   Temps_coagulation(M, Te, N)
2       N_maxi ←taille(M)
3       X_eff_0 ←Valeur_efficace(M, 0, N)
4       i ← 0
5       X_eff ← X_eff_0
6       Tant que X_eff > 0,7 · X_eff_0 et que i < N_maxi − N faire
7           X_eff ←Valeur_efficace(M, i, N)
```

8 $i \leftarrow i + 1$
9 **Fin Tant que**
10 **Si** $i = N_{maxi} - N$
11 **Retourner** « la mesure a échoué »
12 **Sinon**
13 $Tc \leftarrow i \cdot Te/1000$
14 **Retourner** *Tc*

Question 18.

- Quelle est la complexité de la fonction **Temps_coagulation**(*M, Te, N*) ?
- On propose de modifier la ligne 7 de la façon suivante :

7' $X_{eff} \leftarrow \sqrt{X_{eff} \cdot X_{eff} + M[i+N] \cdot M[i+N] - M[i] \cdot M[i]}$

Comparer les deux solutions en terme de temps d'exécution. Justifier.

4.2 Gestion du tableau de mesures

Afin d'optimiser la durée de l'analyse, le calcul de la valeur efficace des oscillations s'effectue en temps réel.
On admet l'existence d'une fonction **Mesure**, de paramètre Te (période d'échantillonnage), qui fait l'acquisition d'une valeur issue des bobines après amplification et filtrage. Celle-ci renvoie la valeur de la mesure au prochain instant $k \times Te,\ k \in \mathbb{N}$. Il n'est pas demandé d'écrire cette fonction et elle pourra être directement appelée par la commande **Mesure**(Te). On peut considérer que la durée du calcul du temps de coagulation est négligeable devant Te.

Les mesures sont stockées dans une zone mémoire appelée tampon, de la façon suivante :

- les N premières mesures sont mémorisées dans le tampon ;
- à partir de la $(N+1)^{ième}$ mesure, chaque nouvelle mesure est stockée en mémoire, alors que la plus ancienne est déstockée.

Question 19. Pourquoi ne peut-on pas utiliser de structure de type pile pour gérer le tampon ?

On utilise une structure de type file pour gérer le tampon. Les fonctions relatives aux files sont voisines des fonctions relatives aux piles. Un tableau présentant ces deux structures est donné ci-dessous.

Piles	Files
creer_pile( ) : renvoie une pile vide	creer_file( ) : renvoie une file vide
empiler(*p, a*) : ajoute un élément *a* en tête de la pile *p*	enfiler(*f, a*) : ajoute un élément *a* en queue de la file *f*
depiler(*p*) : renvoie et supprime l'élément en tête de la pile *p*	defiler(*f*) : renvoie et supprime l'élément en tête de la file *f*

Toutes ces opérations s'effectuent en temps constant.

Question 20. Ecrire, en n'utilisant que la fonction **Mesure**(Te) et les fonctions spécifiques aux files, la fonction **Init_tampon**(*N, Te*) qui crée une file d'attente *T*, réalise *N* mesures successives à la période *Te* et stocke celles-ci dans *T*. Elle renvoie *T*.

L'algorithme de détermination du temps de coagulation est modifié de la façon suivante :

Données : *M, Te,* N_{maxi} : le tableau de mesures, la période d'échantillonnage (en millisecondes), le nombre maximal de mesures réalisées
Résultat : *Tc* le temps de coagulation (en secondes)

```
1   Temps_coagulation(M, Te, N_maxi)
2       T ←Init_tampon(N, Te)
3       X_eff_0 ←Valeur_efficace(T)
4       i ← 0
5       X_eff ← X_eff_0
6       Tant que X_eff > 0,7 · X_eff_0 et que i < N_maxi − N faire
7           ---- à compléter ---


11          i ← i + 1
12      Fin Tant que
13      Si i = N_maxi − N
14          Retourner « la mesure a échoué »
15      Sinon
16          Tc← i · Te/1000
17      Retourner Tc
```

Question 21. En n'utilisant que la fonction **Mesure**(Te) et les fonctions spécifiques aux files, compléter, sur votre copie, les lignes 7 et suivantes de l'algorithme de mesure modifié.

Partie 5 Analyse de l'exigence 2.3 « Prélever les produits par le déplacement suivant $\vec{z}$ de la tête de pipetage »

Objectif : régler la commande du moteur afin de respecter le cahier des charges.

5.1 Modélisation de la motorisation

Les déplacements verticaux des aiguilles de la tête de pipetage (axe $\vec{z}$) sont assurés par un ensemble motoréducteur à courant continu et système pignon-crémaillère.
On note :

- $\theta_m(t)$ et $\omega_m(t)$ l'angle et la vitesse angulaire du moteur ;
- $\omega_r(t)$ la vitesse angulaire en sortie de réducteur ;
- $k_r = \frac{\omega_r}{\omega_m} = \frac{1}{19{,}2}$ le rapport de réduction du réducteur ;
- $c_m(t)$ le couple moteur ;
- J_m l'inertie du moteur et J_r l'inertie du réducteur ramenée à l'arbre moteur ;
- m_p = 0,2 kg la masse en translation ;
- $F_r(t) = 1$ N l'effort de l'opercule sur l'aiguille ;
- c_{res} le couple résistant ramené à l'arbre moteur modélisant l'ensemble des frottements, y compris les frottements internes au réducteur ($c_{res} \leq 0$) ;
- $R_p = 10$ mm le rayon du pignon du système pignon crémaillère ;
- $\omega_{mn} = 4\,150 \text{ tr} \cdot \text{min}^{-1}$ la vitesse de rotation nominale du moteur ;
- $c_{mn} = 5 \cdot 10^{-3} \text{ N} \cdot \text{m}$ le couple moteur nominal ;
- $E_c(\text{S}/\text{R}_\text{g})$ l'énergie cinétique de l'ensemble S par rapport au référentiel R_g.

Question 22. Déterminer l'expression de l'inertie équivalente J_{eq}, ramenée à l'arbre moteur, de S, l'ensemble des pièces en mouvement par rapport au référentiel galiléen R_g, définie par : $E_c(\text{S}/\text{R}_\text{g}) = \frac{1}{2} J_{eq} \cdot {\omega_m}^2$.

Dorénavant on se place dans la situation la plus défavorable, à savoir le sens de la remontée de l'aiguille. La pesanteur est prise en compte.

Question 23.

- A partir d'un théorème d'énergie-puissance, écrire l'équation du mouvement liant $c_m(t)$ et les efforts extérieurs.
- Mettre celle-ci sous la forme : $c_m(t) = c_r(t) + J_{eq} \cdot \dot{\omega}_m$. (4)
- Donner l'expression de $c_r(t)$ en fonction de $c_{res}, F_r(t), m_p, g, R_p$ et k_r.

La tête de pipetage est asservie en position. Le schéma-bloc de cet asservissement est ébauché sur le document réponse (Question 24). Un codeur mesure l'angle de rotation moteur et un hacheur module la tension aux bornes du moteur.
On note :

- $u(t)$ la tension aux bornes du moteur, $i(t)$ l'intensité, $e(t)$ la force électromotrice ;
- R la résistance de l'induit, L son inductance, K la constante de force électromotrice ;
- $K_{cod} = 2\,000$ points/tr le gain du codeur tel que $m_\theta(t) = K_{cod} \cdot \theta_m(t)$;
- K_{adap} le gain permettant d'adapter la consigne $z_c(t)$ à l'image de la position $m_\theta(t)$;
- $H_{cor}(p)$ la fonction de transfert du correcteur ;
- $K_h = 0{,}094\ \mathrm{V} \cdot \mathrm{point}^{-1}$ le gain du hacheur.

Les équations du moteur à courant continu sont les suivantes :

$$u(t) = R \cdot i(t) + L \cdot \frac{di(t)}{dt} + e(t)\ ; \qquad (5)$$

$$e(t) = K \cdot \omega_m(t)\ ; \qquad (6)$$

$$c_m(t) = K \cdot i(t). \qquad (7)$$

Question 24. En tenant compte des notations précédentes, compléter sous forme littérale, sur le document réponse, le schéma-bloc de l'asservissement en position.

Question 25. Déterminer l'expression de K_{adap} pour que l'écart calculé $\varepsilon(t)$ soit proportionnel à l'erreur $z_c(t) - z(t)$.

On note :

- i_0 l'intensité initiale ;
- i_∞ et ω_∞ l'intensité et la vitesse du moteur en régime permanent ;
- c_{r0} le couple résistant $c_r(t)$ supposé constant.

Question 26. Déterminer les expressions de $\left(\frac{\Omega_m(p)}{U(p)}\right)_{c_{r0}=0}$ et de $\left(\frac{I(p)}{U(p)}\right)_{c_{r0}=0}$. Mettre celles-ci sous forme canonique.

Afin de déterminer les caractéristiques du moteur, on applique à celui-ci un échelon de tension $(u_0(t))$ d'amplitude 24 V. On mesure la vitesse $\omega_m(t)$ et l'intensité $i(t)$. Les résultats obtenus sont donnés sur le document réponse.

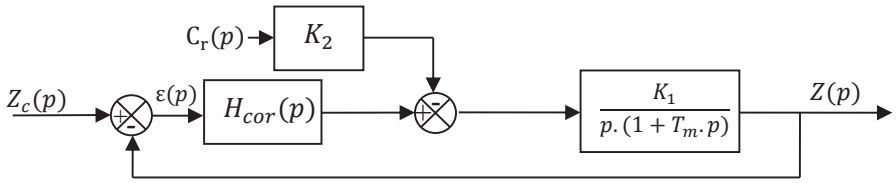
Question 27. Pour cette question, vous justifierez vos résultats à l'aide des tracés nécessaires sur le document réponse.

- A partir de ces courbes et des résultats de la question 26, indiquer si l'hypothèse d'une inductance négligeable est pertinente. Justifier la réponse.

- Dans cette hypothèse d'une inductance négligeable et à partir des équations (4), (5), (6) et (7), déterminer les expressions de i_0, i_∞ et ω_∞ en fonction de u_0, c_{r0}, R et K.
- Déduire de cette étude les valeurs numériques de K et R.
- Déterminer la valeur numérique du couple résistant ramené à l'arbre moteur c_{r0} et de l'inertie équivalente ramenée à l'arbre moteur J_{eq}.

5.2 Réglage de l'asservissement

Les résultats précédents ont permis de modéliser l'asservissement de position par le schéma-bloc ci-dessous :



avec $K_2 = 2{,}78 \cdot 10^{-2}$ N^{-1}, $K_1 = 856$ s^{-1}, $T_m = 3 \cdot 10^{-2}$ s.
Le couple résistant C_r est constant et vaut $C_{r0} = 2{,}7 \cdot 10^{-3}$ N·m.
On suppose le correcteur proportionnel : $H_{cor}(p) = K_p$.

Le diagramme des exigences est donné figure 6 (page 5).

Question 28. Déterminer l'expression de la fonction de transfert en boucle ouverte $H_{bo}(p) = \frac{Z(p)}{\varepsilon(p)}$ ainsi que la fonction de transfert $H_{cr}(p) = \left(\frac{Z(p)}{C_r(p)}\right)_{Z_c=0}$.

Question 29. Déterminer l'erreur statique pour une entrée de type échelon d'amplitude Z_{c0} dans l'hypothèse d'une perturbation nulle ($C_{r0} = 0$). Déterminer ensuite l'erreur due à une perturbation constante C_{r0}, définie comme la valeur finale de la position $z(t)$ dans le cas d'une consigne de position nulle ($z_c = 0$). En déduire la valeur de K_p pour satisfaire le critère de précision du cahier des charges.

Question 30. Les diagrammes de Bode en gain et en phase de $H_{bo}(p)$ sont donnés sur le document réponse pour $K_p = 1$. Pour la valeur de K_p déterminée précédemment, indiquer si le critère de stabilité est satisfait en justifiant votre démarche par les tracés nécessaires sur le document réponse.

Afin d'améliorer le comportement, on implante un correcteur Proportionnel Intégral ayant pour fonction de transfert : $H_{cor}(p) = \frac{K_p(1+T_i \cdot p)}{T_i \cdot p}$ avec $K_p = 1$ et $T_i = 1$ s.
Les diagrammes de Bode de la fonction de transfert en boucle ouverte avec ce correcteur sont donnés sur le document réponse.
On souhaite une marge de phase d'au moins 60°.

Question 31. Justifier le choix de ce correcteur. Déterminer le coefficient K_p pour satisfaire au cahier des charges. Justifier vos calculs par les tracés nécessaires sur le document réponse.

Question 32. La figure 16 (page 19) donne la réponse à un échelon de position de 50 mm avec le correcteur précédemment réglé. Vérifier qu'elle est conforme au cahier des charges. Justifier clairement vos réponses en donnant les valeurs numériques pour chaque critère.

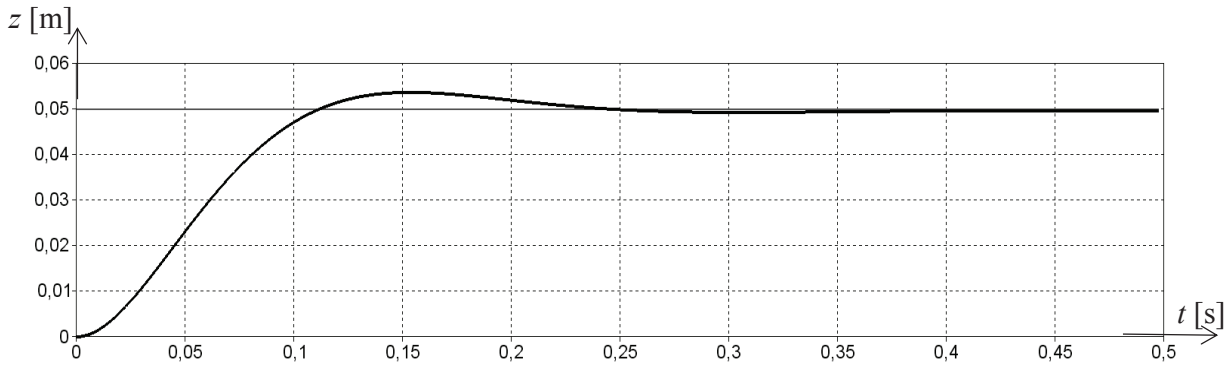


Figure 16 - Réponse à un échelon de position de 50 mm avec un correcteur PI

5.3 Précision du volume prélevé

Lorsque la tête de pipetage a atteint la position souhaitée, définie par les coordonnées X_M et Y_M, l'aiguille de la seringue est plongée dans le liquide à prélever. La hauteur immergée de l'aiguille, Z_v, définit la quantité de produit qui sera aspirée (figure 17).

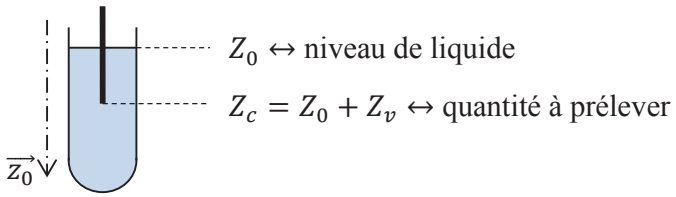


Figure 17 - Flacon et seringue

Chaque flacon n'étant pas rempli de la même façon, le niveau n'est donc pas connu précisément.
Un capteur capacitif détecte au fur et à mesure de la descente de l'aiguille la présence de liquide. Il délivre une information binaire (notée Niv). Lorsque l'aiguille est en contact avec le liquide, le capteur change d'état et inversement.
Un codeur incrémental, donnant 2 000 impulsions par tour de l'axe moteur, indique la position de l'aiguille par la variable Z_{cap} (position mesurée en mm).
La période d'échantillonnage de cette information est $Te = 10$ ms.
Le moteur est mis en marche dès qu'une valeur est affectée à la consigne de position Z_c ; l'ordre de mise en marche est noté V_d.
En début de descente, le niveau est inconnu, la consigne de position initiale Z_m est imposée : $Z_c = Z_m$. La seringue descend (V_d est assigné à 1).
Au passage du niveau détecté par le capteur inductif, correspondant au contact de l'aiguille avec le liquide, la position Z_0, donnée par le codeur incrémental, est mémorisée.
La consigne de position est alors modifiée à une valeur $Z_c = Z_0 + Z_v$.
L'aspiration débute lorsque la consigne est atteinte et s'arrête quand le détecteur de niveau ne perçoit plus de liquide en contact avec la seringue.
Le cycle de cette opération est décrit par le diagramme d'état figure 18.

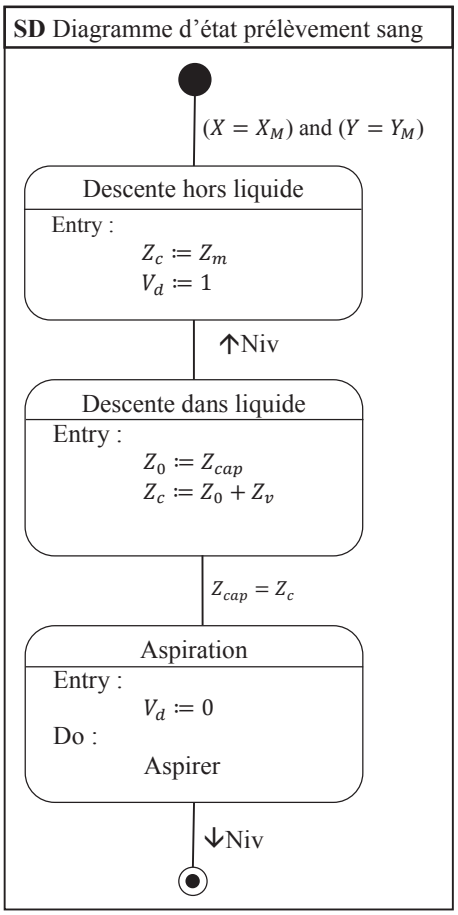


Figure 18 - Diagramme d'état

On rappelle les données suivantes :

- la vitesse maximale de rotation du moteur supposée égale à la vitesse nominale est : $N_{maxi} = 4\,150 \text{ tr} \cdot \text{min}^{-1}$;
- le rapport de réduction du réducteur est $k_r = \frac{\omega_r}{\omega_m} = \frac{1}{19{,}2}$ où ω_m et ω_r sont les vitesses en sortie moteur et réducteur ;
- le rayon du pignon (système pignon-crémaillère) est R_p= 10 mm.

Question 33. Calculer les erreurs de mesure de Z_0 dues à l'échantillonnage d'une part et à la conversion analogique numérique du codeur incrémental d'autre part. En déduire l'erreur maximale de position notée ΔZ_{mes}. Cette erreur est-elle compatible avec le cahier des charges ?

Pour pallier ce défaut de mesure, le constructeur met en place une nouvelle procédure explicitée par le diagramme de la figure 19 :

- le moteur peut avoir deux vitesses de rotation qui correspondent à une vitesse lente de la tête de pipetage ($-V_l$ en montée, $+V_l$ en descente) et une vitesse rapide ($-V_r$ en montée, $+V_r$ en descente) ; ces valeurs seront affectées à la variable V ;
- la consigne initiale de position $Z_c = Z_m$ reste inchangée ;
- lors de la détection de niveau, la position correspondante Z_{cap} est détectée et mémorisée, puis la tête remonte. La consigne est réglée à $Z_c = Z_0 - \Delta Z_{mes}$;
- lorsque le niveau est de nouveau détecté, l'aiguille s'arrête et le système mémorise la nouvelle position Z_0 donnée par le codeur, la consigne de position est alors modifiée à la valeur $Z_c = Z_0 + Z_v$ où Z_v est la hauteur définie précédemment correspondant au volume à prélever.

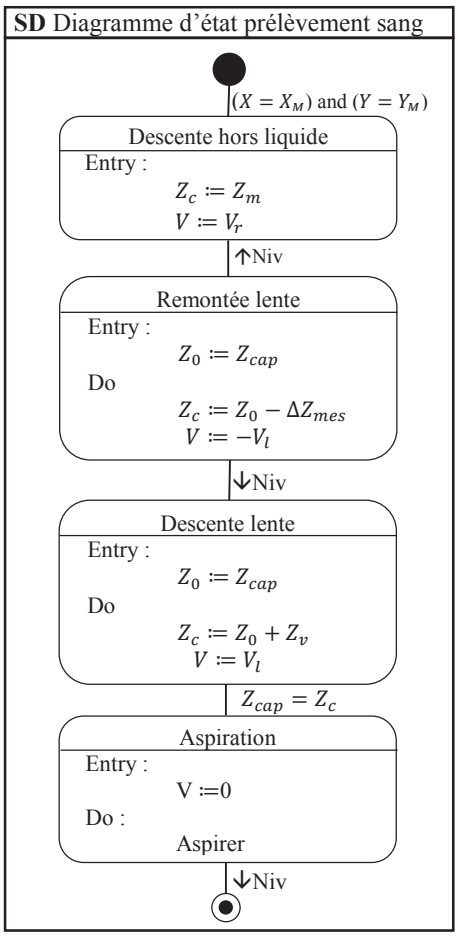


Figure 19 - Diagramme d'état corrigé

A vitesse lente, la vitesse de rotation du moteur est $N_{lent} = 1\,500 \text{ tr} \cdot \text{min}^{-1}$. Le diamètre du flacon est $D_f = 15$ mm.

Question 34. Calculer la nouvelle erreur maximale de position $\Delta Z'_{mes}$ avec l'application de cette nouvelle procédure. Donner l'erreur de volume correspondante.

Synthèse

Question 35. Faire une synthèse des choix qui ont été faits pour satisfaire le critère de précision du volume prélevé.

Fin de l'énoncé

CONCOURS COMMUNS POLYTECHNIQUES

EPREUVE SPECIFIQUE - FILIERE PC

MODELISATION DE SYSTEMES PHYSIQUES OU CHIMIQUES

Durée : 4 heures

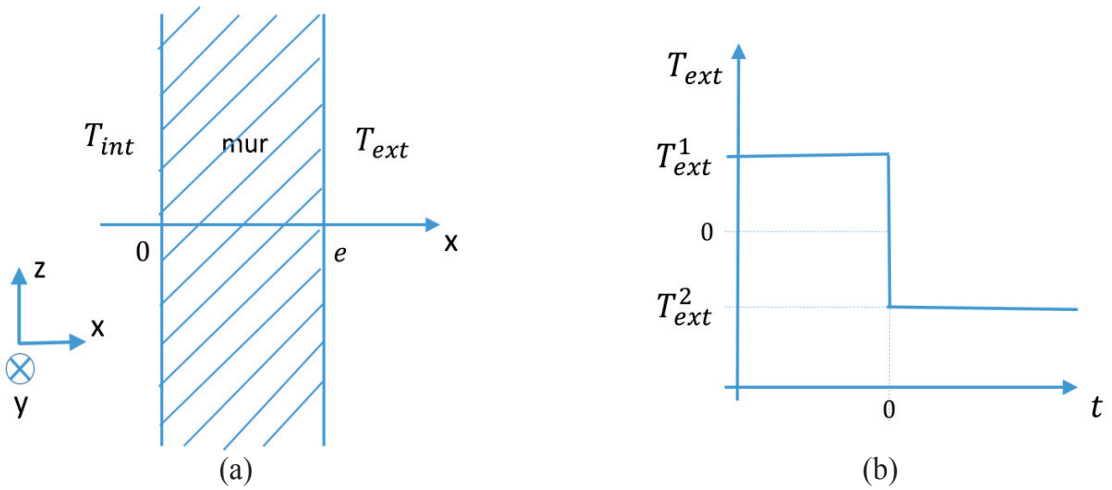
N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Les calculatrices sont interdites

Le sujet comporte deux parties indépendantes. Le candidat précisera au début de sa copie le langage de programmation (Python ou Scilab) qu'il a choisi et toutes les questions seront traitées dans le même langage. Un bonus sera accordé aux copies soignées avec des programmes bien commentés. Plusieurs fonctions du langage Scilab sont rappelées en annexe A. Les candidats choisissant le langage Python pourront utiliser les bibliothèques numpy et matplotlib.pyplot. Une documentation simplifiée de plusieurs fonctions de ces bibliothèques est présente en annexes B et C.

SIMULATION NUMERIQUE DU TRANSFERT THERMIQUE DANS UN MUR EN REGIME TRANSITOIRE

On étudie les transferts thermiques dans le mur d'une maison, figure 1(a). La température à l'intérieur de la maison est constante dans le temps et égale à $T_{int} = 20$ °C. Aux temps négatifs ($t < 0$), la température extérieure est égale à $T_{ext1} = 10$ °C. A $t = 0$, elle chute brusquement à $T_{ext2} = -10$ °C et elle reste égale à cette valeur aux temps positifs ($t > 0$), figure 1(b). On souhaite étudier l'évolution du profil de température dans le mur au cours du temps.



Figures 1 (a) - Schéma du mur étudié. (b) - Evolution de la température extérieure au cours du temps.

Le mur a une épaisseur $e = 40$ cm. Les propriétés physiques du mur sont constantes : conductivité thermique $\lambda = 1{,}65\ \mathrm{W.m^{-1}.K^{-1}}$, capacité thermique massique $c_p = 1\,000\ \mathrm{J.kg^{-1}.K^{-1}}$, masse volumique $\rho = 2\,150\ \mathrm{kg.m^{-3}}$.

PARTIE I : ETUDE PRELIMINAIRE

Dans cette partie, on établit l'équation gouvernant les variations de la température et on la résout en régime permanent.

I.A. Equation gouvernant la température

On suppose que la température dans le mur T ne dépend que du temps t et de la coordonnée x.

I.A.1. A quelle condition peut-on supposer que la température ne dépend pas des coordonnées y et z ?

I.A.2. Donner l'équation générale qui décrit le transport de chaleur dans un solide en l'absence de source d'énergie. Comment cette équation se simplifie-t-elle sous les hypothèses de la question I.A.1 ?

I.B. Conditions aux limites

On envisage plusieurs types de conditions aux limites.

(i) La température est imposée aux limites du système.

(ii) La paroi extérieure est isolée par un matériau de très faible conductivité.

I.B.1. Traduire chacune de ces conditions aux limites sur la fonction $T(x,t)$ et/ou sa dérivée.

Dans toute la suite, on adoptera des conditions aux limites de type température imposée.

I.C. Solutions en régime permanent

I.C.1. Résoudre l'équation obtenue à la question I.A.2. en régime permanent, avec les conditions aux limites de type températures imposées (question I.B.(i)) :

- pour un instant particulier négatif $t_1 < 0$,
- pour un instant particulier positif $t_2 > 0$, très longtemps après la variation de température extérieure, quand le régime permanent est de nouveau établi dans le mur.

I.C.2. Quelle est la nature des profils $T(x)$ obtenus (en régime permanent) à ces deux instants ? Tracer à la main les deux profils sur un même graphique sur la copie.

I.C.3. Sur le même graphique, tracer à la main qualitativement les profils intermédiaires à différents instants entre la variation brutale de la température extérieure ($t = 0$) et l'instant t_2 où le régime est de nouveau permanent.

PARTIE II : RESOLUTION NUMERIQUE

II.A. Equation à résoudre

On cherche à résoudre numériquement l'équation aux dérivées partielles :

$$\alpha \frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} \qquad (1)$$

où α est une constante. A l'équation (1) sont associées les conditions :

$$\begin{array}{ll} T(0,t) = T_{int} & \text{pour tout } \ t > 0 \\ T(e,t) = T_{ext2} & \text{pour tout } \ t > 0 \\ T(x,0) = ax + b & \text{pour tout } \ x \in [0,e] \end{array}$$

II.A.1. Quelle est l'expression de α en fonction des paramètres physiques du mur ?

II.A.2. Exprimer a et b en fonction de T_{int}, T_{ext1} et e.

Pour effectuer la résolution de l'équation (1), nous utiliserons la méthode des différences finies présentée dans la partie II.B.

II.B. Méthode des différences finies

II.B.1. Discrétisation dans l'espace et dans le temps

On divise l'intervalle $[0, e]$, représentant l'épaisseur du mur, en $N + 2$ points, numérotés de 0 à $N + 1$, régulièrement espacés de Δx (figure 2, page suivante). Cette division est appelée « discrétisation ». La distance Δx est appelée le « pas d'espace ». A l'intérieur du mur (frontières intérieure et extérieure exclues) se trouvent donc N points. On cherche à obtenir la température en ces points particuliers à chaque instant.

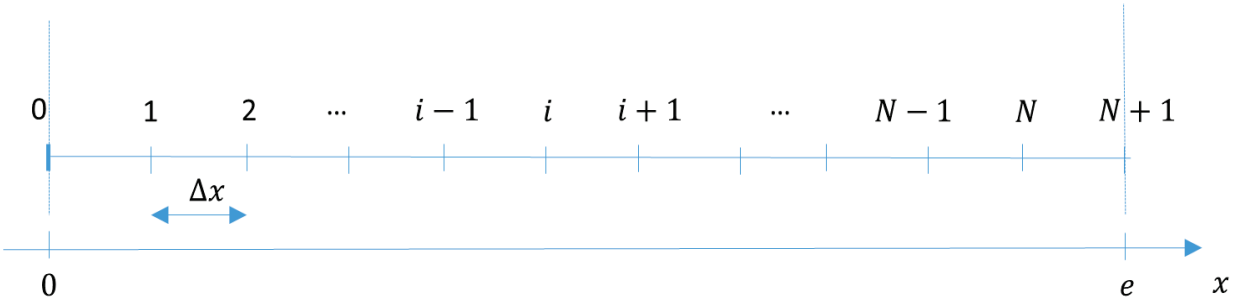


Figure 2 - Discrétisation spatiale dans la direction x.

II.B.1.a. Donner l'expression de Δx en fonction de N et de l'épaisseur du mur e.

II.B.1.b. Donner l'abscisse x_i du i^e point en fonction de i et Δx, sachant que $x_0 = 0$ et $x_{N+1} = e$.

Le temps est discrétisé en $ItMax$ intervalles de durée Δt et on ne s'intéresse au profil de température qu'aux instants particuliers $t_k = k.\Delta t$. L'intervalle élémentaire de temps Δt est appelé le « pas de temps ».

Pour résoudre l'équation (1), deux méthodes sont proposées :

- méthode utilisant un schéma explicite,
- méthode utilisant un schéma implicite.

II.B.2. Méthode utilisant un schéma explicite

II.B.2.a. A l'aide d'un développement limité de la fonction $x \longmapsto T(x,t)$, donner une expression de $T(x+\Delta x, t)$ à l'ordre 3 ($o(\Delta x^3)$) en fonction de T et de ses dérivées partielles par rapport à x évaluées en (x,t). De même, donner une expression de $T(x-\Delta x, t)$ à l'ordre 3.

II.B.2.b. En déduire une expression approchée à l'ordre 1 ($o(\Delta x)$) de $\left.\frac{\partial^2 T}{\partial x^2}\right|_{x,t}$ (dérivée partielle spatiale seconde de T évaluée au point x à l'instant t) en fonction de $T(x+\Delta x, t)$, $T(x-\Delta x, t)$ et $T(x,t)$ et Δx.

On note T_i^k la température $T(x_i, t_k)$, évaluée au point d'abscisse x_i à l'instant t_k. De même, on note $T_{i+1}^k = T(x_i + \Delta x, t_k)$ et $T_{i-1}^k = T(x_i - \Delta x, t_k)$.

II.B.2.c. Déduire de la question précédente une expression approchée de $\left.\frac{\partial^2 T}{\partial x^2}\right|_{x_i,\, t_k}$ (dérivée partielle spatiale seconde de T évaluée en x_i à l'instant t_k) en fonction de T_i^k, T_{i+1}^k et T_{i-1}^k et Δx.

La dérivée partielle temporelle de l'équation (1) est maintenant approchée grâce à un développement limité.

II.B.2.d. A l'aide d'un développement limité de la fonction $t \longmapsto T(x,t)$, donner une expression de $T(x, t+\Delta t)$ à l'ordre 1 ($o(\Delta t)$) en fonction de T et de sa dérivée partielle par rapport à t évaluées en (x,t).

II.B.2.e. En déduire une valeur approchée de $\left.\frac{\partial T}{\partial t}\right|_{x,t}$ (dérivée partielle par rapport au temps de T évaluée au point x à l'instant t) à l'ordre 0 ($o(1)$) en fonction de $T(x, t+\Delta t)$, $T(x,t)$ et Δt.

II.B.2.f. Donner une expression de $\left.\frac{\partial T}{\partial t}\right|_{x_i,\, t_k}$ (dérivée partielle par rapport au temps de T évaluée en x_i à l'instant t_k) en fonction de Δt, T_i^k et T_i^{k+1}, avec $T_i^{k+1} = T(x_i, t_k + \Delta t)$.

L'équation (1) est valable en chaque point d'abscisse x_i et à chaque instant t_k.

II.B.2.g. Ecrire la forme approchée de cette équation au point i et à l'instant k en approchant $\left.\frac{\partial^2 T}{\partial x^2}\right|_{x,t}$ avec la formule obtenue à la question II.B.2.c. et en approchant $\left.\frac{\partial T}{\partial t}\right|_{x,t}$ avec la formule obtenue à la question II.B.2.f.

II.B.2.h. Montrer que l'équation obtenue à la question II.B.2.g peut s'écrire sous la forme :

$$T_i^{k+1} = rT_{i-1}^k + (1-2r)T_i^k + rT_{i+1}^k \qquad (2)$$

en précisant la valeur du paramètre r en fonction de $\Delta x, \Delta t$ et α.

L'équation (2) est appelée schéma numérique explicite. Si on connaît la température en tous les points $x_1, x_2, \dots, x_{N-1}, x_N$ à l'instant t_k, on peut calculer grâce à elle la température en tous les points à l'instant ultérieur t_{k+1}.

II.B.2.i. L'équation (2) est-elle valable dans tout le domaine, c'est-à-dire pour toute valeur de i, $0 \le i \le N+1$? Que valent T_0^k et T_{N+1}^k ?

II.B.2.j. Dans cette question, on élabore une fonction `schema_explicite` permettant de calculer la température en chaque point au cours du temps selon la formule (2). Parmi les variables d'entrée se trouvera un vecteur `T0` de dimension N, défini en dehors de la fonction, contenant les valeurs de la température aux points de discrétisation à l'instant initial. Au sein de la fonction, un algorithme calculera itérativement la température avec un nombre maximal d'itérations `ItMax`. En sortie de la fonction, on récupérera le nombre d'itérations réellement effectuées, `nbIter` et une matrice `T_tous_k`, de dimensions $N \times ItMax$. Chaque colonne de cette matrice contient le vecteur $\boldsymbol{T^k}$ dont les éléments sont les valeurs de la température aux N points $x_1, \dots, x_N$ (points à l'intérieur du mur) à l'instant k :

$$\boldsymbol{T}^k = \begin{pmatrix} T_1^k \\ T_2^k \\ \dots \\ T_{N-1}^k \\ T_N^k \end{pmatrix} \qquad \text{et} \qquad \texttt{T\_tous\_k} = \begin{pmatrix} T_1^1 & T_1^2 & \dots & T_1^k & \dots & T_1^{k-1} & T_1^k \\ T_2^1 & T_2^2 & & T_2^k & & T_2^{k-1} & T_2^k \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ T_{N-1}^1 & T_{N-1}^2 & \dots & T_{N-1}^k & \dots & T_{N-1}^{k-1} & T_{N-1}^k \\ T_N^1 & T_N^2 & & T_N^k & & T_N^{k-1} & T_N^k \end{pmatrix}.$$

On souhaite arrêter le calcul lorsque la température ne varie presque plus dans le temps. Dans ce but, on évaluera la norme 2 de $\boldsymbol{T}^k - \boldsymbol{T}^{k-1}$ à chaque itération. La définition de la norme 2 est rappelée à la question II.B.2.j.*(vi)*.

II.B.2.j.*(i)* Ecrire l'en-tête de la fonction en précisant bien les paramètres d'entrée et de sortie.

II.B.2.j.*(ii)* Le schéma numérique (2) permet d'approcher avec succès la solution à la condition $r < 1/2$. Programmer un test qui avertit l'utilisateur si cette condition n'est pas respectée.

II.B.2.j.*(iii)* Affecter la valeur 2 000 à `ItMax`. Créer la matrice `T_tous_k` de dimensions $N \times ItMax$ en la remplissant de zéros.

II.B.2.j.*(iv)* Remplacer la première colonne de `T_tous_k` par le vecteur des valeurs initiales `T0`.

II.B.2.j.*(v)* Calculer le profil de température à l'instant $k = 1$ $(t = \Delta t)$, en distinguant le cas $i = 1$, le cas $2 \leq i \leq N - 1$ et le cas $i = N$. Affecter ces valeurs à la deuxième colonne de `T_tous_k`.

II.B.2.j.*(vi)* Ecrire une fonction `calc_norme` qui calcule la norme 2 d'un vecteur. On rappelle que la norme 2 d'un vecteur $\boldsymbol{V}$ s'écrit :

$$\|\boldsymbol{V}\|_2 = \sqrt{\sum_{i=1}^{N} V_i^2} \qquad \text{avec} \qquad \boldsymbol{V} = \begin{pmatrix} V_1 \\ \vdots \\ V_i \\ \vdots \\ V_N \end{pmatrix}.$$

II.B.2.j.*(vii)* Elaborer une boucle permettant de calculer itérativement le profil de température aux instants $t_k = k.\Delta t$ avec $k \geq 2$. Cette boucle sera interrompue lorsque la norme 2 du vecteur $\boldsymbol{T^k - T^{k-1}}$ deviendra inférieure à 10^{-2} ou lorsque le nombre d'itérations atteindra la valeur $ItMax$ (prévoir les deux cas). Utiliser, pour cela, la fonction `calc_norme` définie à la question II.B.2.j.*(vi)*.

II.B.2.j.*(viii)* Ecrire la fin de la fonction afin de renvoyer tous les arguments de sortie définis au début de la question II.B.2.j.

II.B.3. Méthode utilisant un schéma implicite

Le schéma explicite (2) ne converge que si le pas de temps Δt est suffisamment faible par rapport au pas d'espace Δx. Si l'on souhaite effectuer un calcul pour un temps physique long, beaucoup d'itérations seront nécessaires et le temps de calcul sera très long. C'est pourquoi on préfère d'autres types de schémas appelés schémas implicites.

Dans cette partie, la dérivée partielle seconde par rapport à x de la température apparaissant dans l'équation (1) est évaluée au point d'abscisse x_i et à l'instant $k + 1$:

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{x,t} \approx \left.\frac{\partial^2 T}{\partial x^2}\right|_{x_i, t_{k+1}}$$

et la dérivée partielle par rapport à t est évaluée au point d'abscisse x_i et à l'instant k :

$$\left.\frac{\partial T}{\partial t}\right|_{x,t} \approx \left.\frac{\partial T}{\partial t}\right|_{x_i, t_k}.$$

II.B.3.a. Donner la nouvelle expression approchée de l'équation (1) définie en page 3.

II.B.3.b. Montrer que l'équation obtenue à la question II.B.3.a. peut être mise sous la forme

$$T_i^k = -rT_{i-1}^{k+1} + (1 + 2r)T_i^{k+1} - rT_{i+1}^{k+1} \quad . \qquad (3)$$

L'équation (3) est appelée schéma implicite car la température à l'instant t_k est exprimée en fonction de la température à l'instant ultérieur t_{k+1}.

Le système d'équations ainsi obtenu peut être écrit sous la forme :

$$M\boldsymbol{T}^{k+1} = \boldsymbol{T}^{k} + r\,\boldsymbol{v} \qquad (4)$$

où M est une matrice carrée $N \times N$ et $\boldsymbol{T}^{k}$ et $\boldsymbol{T}^{k+1}$ sont les vecteurs de dimension N définis par :

$$\boldsymbol{T}^{k} = \begin{pmatrix} T_1^k \\ T_2^k \\ \dots \\ T_{N-1}^k \\ T_N^k \end{pmatrix} \quad \text{et} \quad \boldsymbol{T}^{k+1} = \begin{pmatrix} T_1^{k+1} \\ T_2^{k+1} \\ \dots \\ T_{N-1}^{k+1} \\ T_N^{k+1} \end{pmatrix}$$

et $\boldsymbol{v}$ est un vecteur de taille N faisant intervenir les conditions aux limites.

II.B.3.c. Préciser l'expression de la matrice M et l'expression du vecteur $\boldsymbol{v}$.

A chaque pas de temps, il faut inverser le système matriciel :

$$M\boldsymbol{T}^{k+1} = \boldsymbol{T}^{k} + r\,\boldsymbol{v}$$

pour obtenir $\boldsymbol{T}^{k+1}$à partir de $\boldsymbol{T}^{k}$.

II.B.3.d. Le but de cette question est d'écrire une fonction `CalcTkp1` qui permet de résoudre un système matriciel tridiagonal en utilisant l'algorithme de Thomas présenté ci-dessous.

Algorithme de Thomas :
On cherche à résoudre un système matriciel tridiagonal de la forme :

$$M\,\boldsymbol{u} = \boldsymbol{d} \qquad (5)$$

où M est une matrice de dimensions $N \times N$ tridiagonale, c'est-à-dire une matrice dont tous les éléments sont nuls, sauf sur la diagonale principale, la diagonale supérieure et la diagonale inférieure

$$M = \begin{pmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & 0 & \\ & a_3 & b_3 & c_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & 0 & & & & & \\ & & & & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & & & a_N & b_N \end{pmatrix}$$

et où les vecteurs $\boldsymbol{u}$ et $\boldsymbol{d}$, de dimension N, s'écrivent **:**

$$\boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} \quad \text{et} \quad \boldsymbol{d} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{N-1} \\ d_N \end{pmatrix}$$

Dans cet algorithme, on calcule d'abord les coefficients suivants :

$$c'_1 = \frac{c_1}{b_1}$$
$$c'_i = \frac{c_i}{b_i - a_i c'_{i-1}} \quad \text{pour} \quad i = 2, 3, \dots, N-1$$

et

$$d'_1 = \frac{d_1}{b_1}$$
$$d'_i = \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} \quad \text{pour} \quad i = 2, 3, \dots, N\,.$$

Les inconnues $u_1, u_2, \dots, u_N$ sont alors obtenues par les formules :

$$u_N = d'_N$$
$$u_i = d'_i - c'_i u_{i+1} \quad \text{pour} \quad i = N-1, N-2, \dots, 2, 1\ .$$

II.B.3.d.*(i)* En utilisant l'algorithme de Thomas, écrire une fonction `CalcTkp1` qui permet de calculer le vecteur $\boldsymbol{u}$, solution du système matriciel (5), à partir de la matrice M et du vecteur $\boldsymbol{d}$.

II.B.3.e. Dans cette question, une fonction `schema_implicite` est élaborée avec les mêmes arguments d'entrée et de sortie que la fonction `schema_explicite` (définis à la question II.B.2.j.) et qui utilise les mêmes critères d'arrêt (définis à la question II.B.2.j.*(vii)*).

II.B.3.e.*(i)* Ecrire l'en-tête de la fonction en précisant les paramètres d'entrée et de sortie.

II.B.3.e.*(ii)* Affecter la valeur 2 000 à `ItMax`. Créer la matrice `T_tous_k` dont les dimensions sont $N \times ItMax$ en la remplissant de zéros.

II.B.3.e.*(iii)* Remplacer la 1re colonne de `T_tous_k` par le vecteur des valeurs initiales `T0`.

II.B.3.e.*(iv)* Définir la matrice `M` et le vecteur `v` qui interviennent dans l'équation (4).

II.B.3.e.*(v)* Calculer le profil de température à l'instant $k = 1$ $(t = \Delta t)$. Affecter ces valeurs à la deuxième colonne de `T_tous_k`.

II.B.3.e.*(vi)* Ecrire une boucle permettant de calculer itérativement le profil de température aux instants ultérieurs $t_k = k \times \Delta t$ avec $k \geq 2$, en prévoyant un arrêt lorsque la norme 2 du vecteur $\boldsymbol{T^k - T^{k-1}}$ devient inférieure à 10^{-2} ou lorsque le nombre d'itérations atteint la valeur `ItMax` (prévoir les deux cas). Utiliser pour cela la fonction `calc_norme` définie à la question II.B.2.j.*(vi)*.

II.B.3.e.*(vii)* Ecrire la fin de la fonction afin de renvoyer tous les arguments de sortie définis au début de la question II.B.2.j.

II.C. Programme principal

II.C.1. Début du programme

II.C.1.a. Définir les variables `epais` (épaisseur du mur), `conduc` (conductivité thermique), `rho` (masse volumique), `Cp` (capacité thermique massique), `Tint` (température intérieure), `Text1`

(température extérieure pour les instants $t < 0$), `Text2` (température extérieure pour les instants $t > 0$), `N` (nombre de points de calcul **à l'intérieur du mur**) et `Dt` (intervalle de temps élémentaire) et leur affecter les valeurs correspondant au problème physique défini au début de l'énoncé. On prendra un nombre de points de discrétisation $N = 60$ et un pas de temps Δt de 25 secondes.

II.C.1.b. Calculer les coefficients `a` et `b` avec la formule trouvée à la question II.A.2.

II.C.1.c. Créer un vecteur `x` dont les éléments $x_1, x_2, \ldots, x_N$ sont définis à la question II.B.1.b.

II.C.1.d. Calculer le vecteur des températures initiales `T0`.

II.C.1.e. Calculer `alpha` selon la formule trouvée à la question II.A.1. Calculer `r` en utilisant la formule calculée à la question II.B.2.h.

II.C.2. Calcul des températures

II.C.2.a. Ecrire un morceau de programme qui demande à l'utilisateur quel schéma (explicite ou implicite) il souhaite utiliser et qui appelle la fonction correspondante.

II.C.3. Analyse du résultat

II.C.3.a. Ecrire un morceau de programme permettant de tracer sur un même graphique le profil de température en fonction de `x` tous les 100 pas de temps.

II.C.3.b. Faire afficher le temps en heures au bout duquel le régime permanent est établi.

Fin de l'énoncé

ANNEXE A : COMMANDES ET FONCTIONS USUELLES DE SCILAB

A=[a b c d;e f g h;i j k l]

Description : commande permettant de créer une matrice dont la première ligne contient les éléments a, b, c, d, la seconde ligne contient les éléments e, f, g, h et la troisième, les éléments i, j, k, l.

Exemple : A=[1 2 3 4 5;3 10 11 12 20;0 1 0 0 2]

```
⇨   1.  2.   3.   4.   5.
    3.  10.  11.  12.  20.
    0.  1.   0.   0.   2.
```

A(i,j)

Arguments d'entrée : les coordonnées de l'élément dans le tableau A.

Argument de sortie : l'élément (i, j) de la matrice A.

Description : fonction qui retourne l'élément (i, j) de la matrice A. Pour obtenir toute la colonne j de la matrice A, on utilise la syntaxe $A(:, j)$. De même, pour accéder à l'intégralité de la ligne i de la matrice A, on écrit $A(i, :)$.

Exemple :

```
A=[1 2 3 4 5;3 10 11 12 20;0 1 0 0 2]
A(2,4)
⇨ 12
A(:,3)
⇨   3.
    11.
    0.
A(2,:)
⇨ 3.  10.  11.  12.  20.
```

x=[x1:Dx:x2]

Description : commande permettant de créer un vecteur dont les éléments sont espacés de Dx et dont le premier élément est x_1 et le dernier élément est le plus grand multiple de Dx inférieur ou égal à x_2.

ATTENTION : le vecteur ainsi créé est un vecteur ligne. Pour convertir un vecteur ligne en un vecteur colonne, on le transpose en utilisant l'apostrophe « ' » : x_trans=x'.

Exemple :

```
x=[2:0.5:6.3]
⇨ 2.  2.5  3.  3.5  4.  4.5  5.  5.5  6.
x_trans=x'
⇨ 2.
  2.5
  3.
  3.5
  4.
  4.5
  5.
  5.5
  6.
```

zeros(n,m)

Arguments d'entrée : deux entiers n et m correspondant aux dimensions de la matrice à créer.

Argument de sortie : un tableau (matrice) d'éléments nuls.

Description : fonction créant une matrice (tableau) de dimensions $n \times m$ dont tous les éléments sont nuls.

Exemple : zeros(3,4)

⇨ 0. 0. 0. 0.
0. 0. 0. 0.
0. 0. 0. 0.

plot(x,y)

Arguments d'entrée : un vecteur d'abscisses x (tableau de dimension n) et un vecteur d'ordonnées y (tableau de dimension n).

Description : fonction permettant de tracer sur un graphique n points dont les abscisses sont contenues dans le vecteur x et les ordonnées dans le vecteur y.

Exemple :
```
x= [3:0.1:5]
y=sin(x)
plot(x,y)
```

ANNEXE B : BIBLIOTHEQUE NUMPY DE PYTHON

Dans les exemples ci-dessous, la bibliothèque numpy a préalablement été importée à l'aide de la commande : **import numpy as np**

On peut alors utiliser les fonctions de la bibliothèque, dont voici quelques exemples :

np.array(liste)

Argument d'entrée : une liste définissant un tableau à 1 dimension (vecteur) ou 2 dimensions (matrice).

Argument de sortie : un tableau (matrice).

Description : fonction permettant de créer une matrice (de type tableau) à partir d'une liste.

Exemple :
```
np.array([4,3,2])
⇨ [4 3 2]
np.array([[5],[7],[1]])
⇨ [[5]
   [7]
   [1]]
np.array([[3,4,10],[1,8,7]])
⇨ [[3 4 10]
   [1 8 7]]
```

$\boldsymbol{A[i,j]}$.

Arguments d'entrée : un tuple contenant les coordonnées de l'élément dans le tableau A.

Argument de sortie : l'élément $(\mathrm{i}+1, \mathrm{j}+1)$ de la matrice A.

Description : fonction qui retourne l'élément $(i+1, j+1)$ de la matrice A. Pour obtenir toute la colonne j+1 de la matrice A, on utilise la syntaxe $A[:,j]$. De même, pour accéder à l'intégralité de la ligne i+1 de la matrice A, on écrit $A[i,:]$.

ATTENTION : en langage Python, les lignes d'un $\boldsymbol{A}$ de dimension $\boldsymbol{n \times m}$ sont numérotées de 0 à $\boldsymbol{n-1}$ et les colonnes sont numérotées de 0 à $\boldsymbol{m-1}$

Exemple :
```
A=np.array([[3,4,10],[1,8,7]])
A[0,2]
⇨ 10
A[:,2]
⇨ [10 7]
A[1,:]
⇨ [1 8 7]
```

np.zeros((n,m))

Arguments d'entrée : un tuple de deux entiers correspondant aux dimensions de la matrice à créer.
Argument de sortie : un tableau (matrice) d'éléments nuls.
Description : fonction créant une matrice (tableau) de dimensions $n \times m$ dont tous les éléments sont nuls.
Exemple : np.zeros((3,4))

⇨ [[0 0 0 0]
[0 0 0 0]
[0 0 0 0]]

np.linspace(Min,Max,nbElements)

Arguments d'entrée : un tuple de 3 entiers.
Argument de sortie : un tableau (vecteur).
Description : fonction créant un vecteur (tableau) de $nbElements$ nombres espacés régulièrement entre Min et Max. Le 1er élément est égal à Min, le dernier est égal à Max et les éléments sont espacés de $(Max - Min)/(nbElements - 1)$:
Exemple : np.linspace(3,25,5)

⇨ [3 8.5 14 19.5 25]

ANNEXE C : BIBLIOTHEQUE MATPLOTLIB.PYPLOT DE PYTHON

Cette bibliothèque permet de tracer des graphiques. Dans les exemples ci-dessous, la bibliothèque matplotlib.pyplot a préalablement été importée à l'aide de la commande :
import matplotlib.pyplot as plt

On peut alors utiliser les fonctions de la bibliothèque, dont voici quelques exemples :

plt.plot(x,y)

Arguments d'entrée : un vecteur d'abscisses x (tableau de dimension n) et un vecteur d'ordonnées y (tableau de dimension n).
Description : fonction permettant de tracer sur un graphique de n points dont les abscisses sont contenues dans le vecteur x et les ordonnées dans le vecteur y. Cette fonction doit être suivie de la fonction **plt.show()** pour que le graphique soit affiché.
Exemple :

```
x= np.linspace(3,25,5)
y=sin(x)
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

plt.xlabel(nom)

Argument d'entrée : une chaîne de caractères.
Description : fonction permettant d'afficher le contenu de nom en abscisse d'un graphique.

plt.ylabel(nom)

Argument d'entrée : une chaîne de caractères.
Description : fonction permettant d'afficher le contenu de nom en ordonnée d'un graphique.

plt.show()

Description : fonction réalisant l'affichage d'un graphe préalablement créé par la commande **plt.plot(x,y)**. Elle doit être appelée après la fonction plt.plot et après les fonctions plt.xlabel et plt.ylabel.

CONCOURS COMMUNS POLYTECHNIQUES

EPREUVE SPECIFIQUE - FILIERE PSI

MODELISATION ET INGENIERIE NUMERIQUE

Durée : 4 heures

N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Les calculatrices sont autorisées

Le sujet comporte 16 pages dont :
— 14 pages de texte de présentation et énoncé du sujet ;
— 2 pages d'annexes.

Toute documentation autre que celle fournie est interdite.

REMARQUES PRELIMINAIRES

Les développements mathématiques, les schémas, les graphes et les courbes seront rendus dans leur forme définitive sur la copie (les brouillons ne seront pas acceptés).

Il est demandé au candidat de bien vouloir inscrire les résultats et les développements nécessaires aux différentes questions sur sa copie, **en précisant bien le numéro de la question traitée et, si possible, dans l'ordre des questions.** Les résultats attendus seront obligatoirement entourés.

Dispositif Médical d'Injection

I Présentation du système

Les problèmes de contamination ont engendré le développement de systèmes pour protéger les aiguilles d'injection et ainsi limiter les risques d'accidents.

De nombreux dispositifs sont actuellement développés afin d'améliorer la qualité des injections ou l'ergonomie pour les patients pour améliorer l'adhésion au traitement.

Actuellement, il existe deux types de contenant pour les solutions médicamenteuses : le vial et la cartouche (figures 1).

(a) Vial (b) Cartouche

Figures 1 – Différents contenants actuels

I.1 Dispositifs Médicaux d'Injection innovants (DMI)

Eveon, jeune start-up dans le monde des dispositifs médicaux d'injection, a l'ambition de réaliser un DMI sécurisé, automatisé et facile d'utilisation. Le dispositif réalisé est un DMI monodose, miniaturisé, automatique et adapté à tout type d'injection. Ce dispositif est présenté schématiquement sur les figures 2.

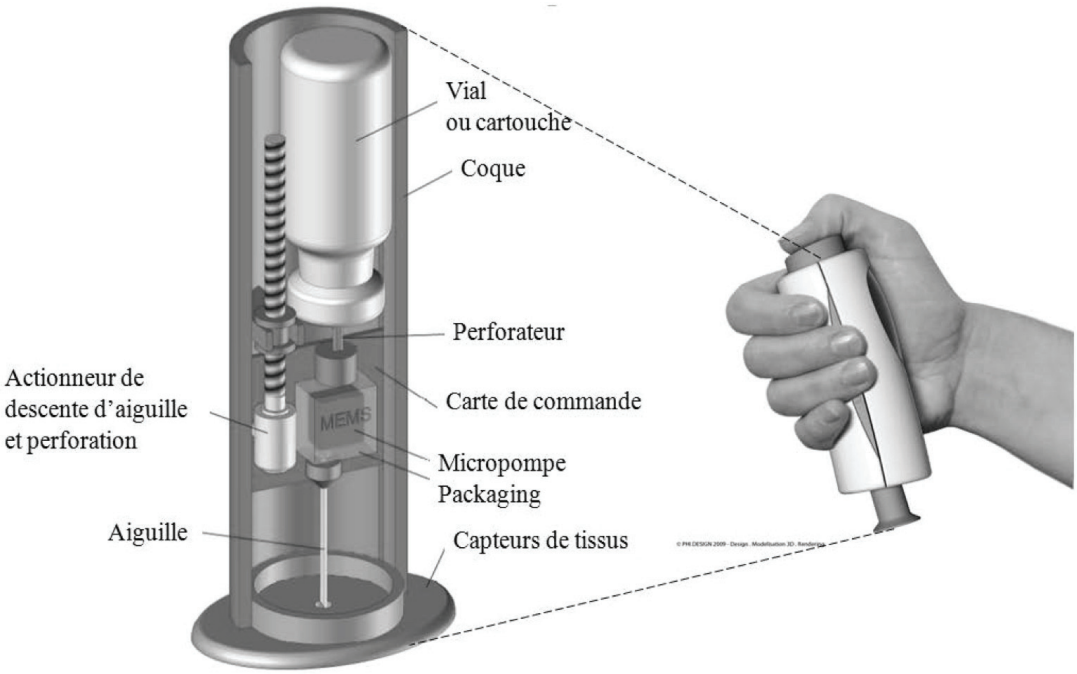
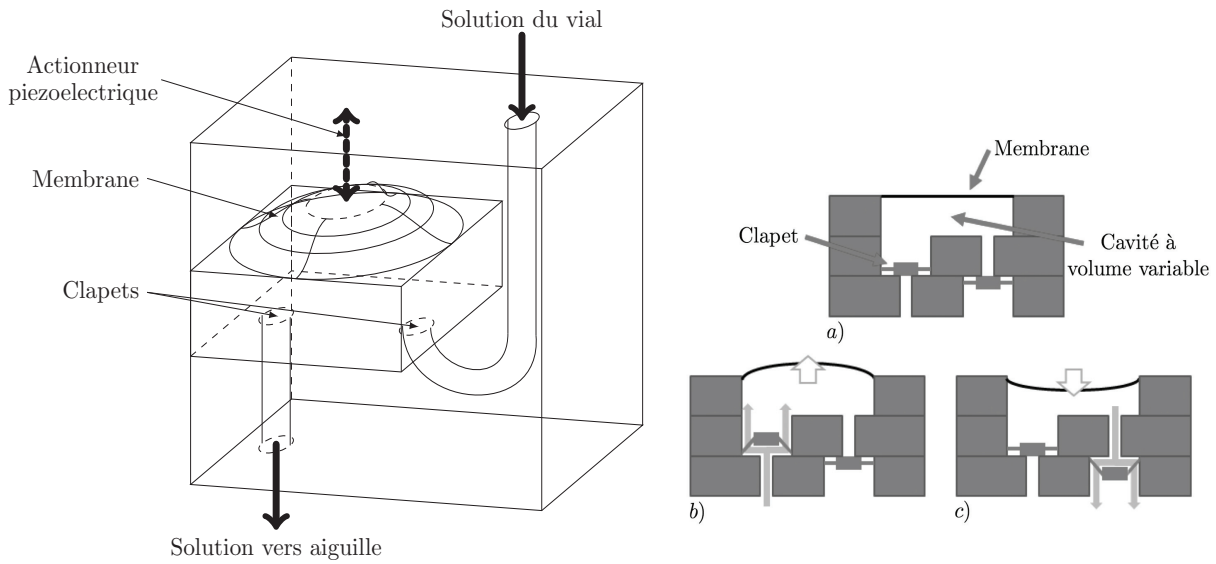


Figure 2 – Dispositif médicalisé d'injection de l'entreprise Eveon

La solution retenue pour la pompe est l'utilisation d'une micropompe à membrane MEMS, permettant de résoudre les problèmes liés à l'utilisation d'un piston.

Le principe est simple : il s'agit de déformer une membrane délimitant un volume donné afin de créer successivement dans celui-ci des phases de dépression et de sur-pression, permettant via un système de clapet de respectivement aspirer le liquide puis de le refouler. Le principe est décrit sur les figures 3.



(a) Structure

(b) Fonctionnement des clapets : a) membrane au repos, b) aspiration du liquide, c) refoulement du liquide

Figures 3 – Micropompe à membrane

La déformation de la membrane est assurée par un actionneur thermique collé sur la face supérieure de la membrane, mais également par un actionneur piézoélectrique permettant d'assurer une action mécanique suffisante sur la membrane en cas de pression élevée dans la cavité de la pompe. En effet, l'actionneur thermique étant peu puissant et la température d'échauffement limitée pour ne pas dégrader le liquide médicamenteux, il est nécessaire de prévoir un actionneur de secours pour assurer les mouvements de la membrane.

Le système d'injection ainsi asservi par un système de capteurs de débit (débimètre) est ainsi modélisé par le schéma-bloc fonctionnel simplifié donné sur la figure 15 - annexe 1.

I.2 Exigences du système

Un extrait des exigences du système est donné sur la figure 4 :

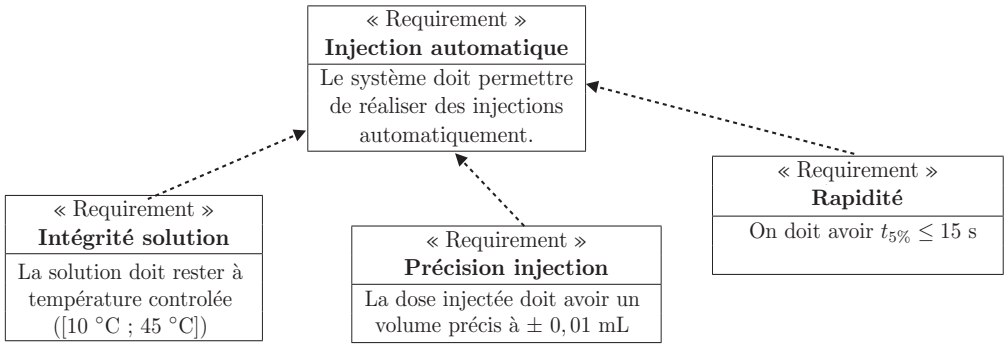


Figure 4 – Diagramme partiel des exigences

Objectif
L'objectif de ce sujet est de modéliser le système médicalisé d'injection afin de vérifier les performances du cahier des charges concernant le contrôle de la quantité de solution injectée.

II Modélisation de l'asservissement du volume injecté

Objectif
L'objectif de cette partie est de modéliser l'ensemble de l'asservissement du volume injecté afin de vérifier que le système respecte l'exigence du cahier des charges.

Dans la suite du sujet, la transformée de Laplace d'une fonction f(t) sera notée F(p).

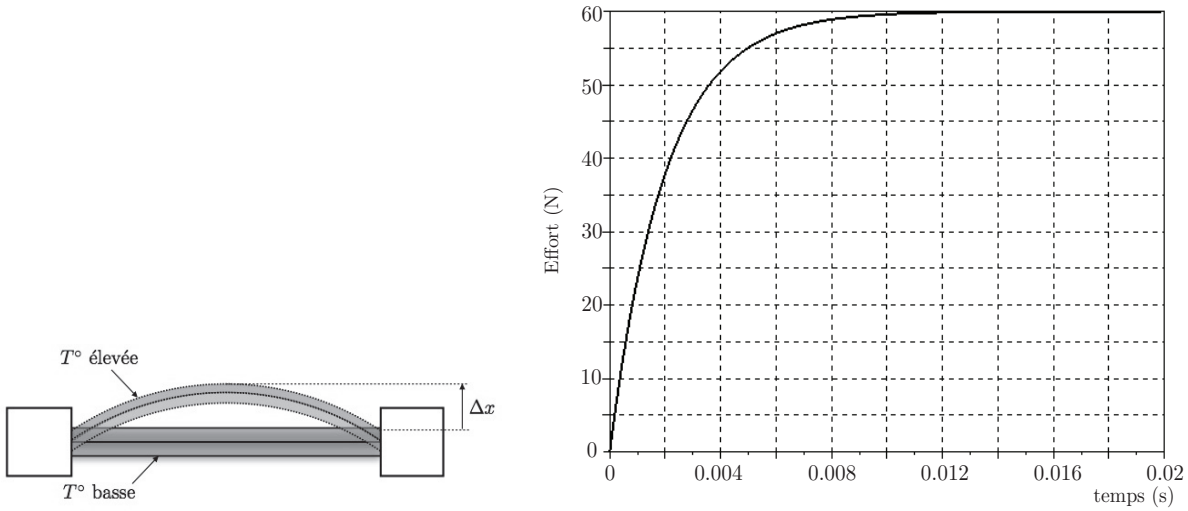
II.1 Actionneur thermique

Objectif
L'objectif de cette partie est de modéliser le fonctionnement de l'actionneur thermique.

L'actionneur thermique est une membrane bimétallique qui, sous l'effet thermique, se déforme. Elle fonctionne sur le principe suivant : si on accole deux matériaux possédant des coefficients de dilatation thermique différents, une élévation de la température va provoquer la déformation de la membrane du côté du matériau possédant le coefficient de dilatation le plus élevé (figures 5).

Afin d'identifier le comportement de l'actionneur thermique, on réalise un essai en alimentant l'actionneur sous un échelon de tension $u_{th} = 12$ V.

Q1. A l'aide de la courbe de réponse obtenue (figure 5b), déterminer la fonction de transfert de l'actionneur thermique (supposée du premier ordre) : $H_{\text{ath}}(p) = \dfrac{F_{th}(p)}{U_{th}(p)} = \dfrac{K_{th}}{1+\tau_{th}p}$.



(a) Actionneur thermique (b) Réponse à un échelon de tension

Figures 5 – Actionneur thermique et réponse à un échelon de tension

En réalité, l'actionneur thermique est alimenté par une tension hachée, puisqu'il est successivement chargé et déchargé pour obtenir l'effet de pompage.

II.2 Écoulement dans le canal de l'aiguille

Objectif
L'objectif de cette partie est de modéliser l'écoulement dans une conduite afin de déterminer la relation entre le débit dans la conduite et la pression en entrée de conduite.

Le canal de l'aiguille est modélisé par un tube cylindrique d'axe (Oz), de longueur L et de section circulaire de rayon R. Le fluide médicamenteux est assimilé à de l'eau de masse volumique ρ et de viscosité dynamique η. L'écoulement unidirectionnel et stationnaire se fait dans le sens des z croissants. On note $P(0) = P(z = 0)$ la pression à l'entrée du canal et $P(L) = P(z = L)$ la pression à la sortie du canal.

Q2. Rappeler les hypothèses à vérifier pour pouvoir appliquer le théorème de Bernoulli. Si le canal est horizontal, comment s'exprime le théorème de Bernoulli entre l'entrée et la sortie en fonction de la perte de charge Δh homogène à une longueur ?

Q3. Définir et évaluer le nombre de Reynolds correspondant à l'écoulement dans l'aiguille de longueur 5 cm et de diamètre intérieur 200 µm dans le cas d'un débit de 5 mL·min^{-1}. La viscosité du fluide est $\eta = 1{,}0.10^{-3}$ Pa·s.

Q4. Les pertes de charge régulières sont données par la relation $\Delta h = \Lambda \dfrac{\Delta x}{2R} \dfrac{< v^2 >}{2g}$ où g est l'accélération de la pesanteur et Λ le paramètre dit de frottements, sans dimension. Que représente $< v^2 >$? Quelle est la dimension de Δx ? Que peut représenter Δx ? En déduire la perte de charge entre l'entrée et la sortie de l'aiguille.

Q5. Un calcul non demandé ici montre que la vitesse moyenne de l'écoulement dans la seringue est donnée par la relation $< v > = \dfrac{P(0) - P(L)}{8\eta L} R^2$. En déduire l'expression du débit volumique du fluide noté Q pour cet écoulement.

Q6. Pour une aiguille de longueur 50,0 mm et de diamètre intérieur 0,2 mm, calculer, toujours en négligeant la pesanteur, la valeur numérique de la pression à exercer au sommet de la colonne de liquide pour assurer un débit de 5,0 mL·min^{-1} à la sortie de l'aiguille. On prendra $\eta = 1{,}0 \cdot 10^{-3}$ Pa·s .

Pour la suite du sujet, on supposera que la fonction de transfert représentative du comportement de la conduite s'écrit : $H_{\text{conduite}}(p) = \dfrac{Q(p)}{F_{charges}(p)} = K_d = 9{,}6.10^4$ m^3.s^{-1}.N^{-1}.

II.3 Membrane de la micropompe

Objectif
L'objectif de cette partie est de modéliser le fonctionnement de la membrane de la micropompe afin de lier les efforts exercés sur la membrane avec le débit effectif de la pompe.

II.3.1 Modélisation générale

Le comportement de la membrane est modélisé par un assemblage de segments reliés entre eux par des liaisons pivots comme décrit sur la figure 6 page 6.

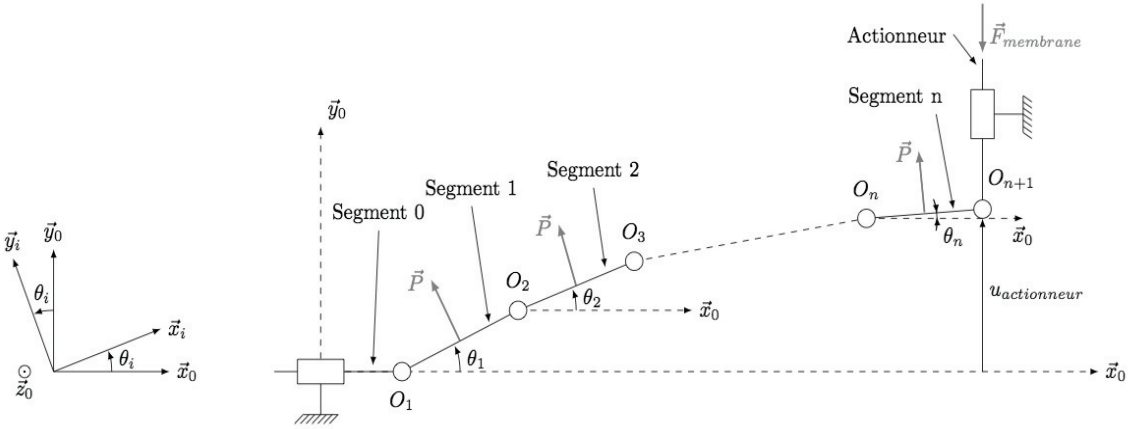


Figure 6 – Modélisation de la membrane

Les hypothèses réalisées sont :

— chaque segment i est supposé indéformable, d'axe directeur noté $\vec{x}_i$, de longueur L_0, de sorte que tout déplacement selon l'axe $\vec{x}_0$ est négligé ;

— chaque segment i est relié par des liaisons pivots supposées parfaites aux segments $i+1$ et $i-1$, le torseur modélisant les actions mécaniques de la liaison entre le segment i et le segment $i-1$ est au point O_i de la forme : $\{\mathcal{T}_{i-1\to i}\} = \begin{Bmatrix} x_i\vec{x}_0 + y_i\vec{y}_0 \\ \vec{0} \end{Bmatrix}_{O_i}$;

— il existe un effort de torsion entre chaque segment i et $i-1$, défini par un torseur exprimé au point O_i de la forme : $\{\mathcal{T}^{torsion}_{i-1\to i}\} = \begin{Bmatrix} \vec{0} \\ -k_t(\theta_i - \theta_{i-1})\vec{z}_0 \end{Bmatrix}_{O_i}$;

— le déplacement de l'actionneur est donné selon l'axe $\vec{y}_0$ par la distance $u_{actionneur}$;

— sur chaque segment i s'applique un glisseur de résultante $\vec{P}$ au point M_i centre du segment i tel que $\{\mathcal{T}_{\mathrm{P}\to\mathrm{i}}\} = \begin{Bmatrix} P\vec{y}_i \\ \vec{0} \end{Bmatrix}_{M_i}$ avec $\overrightarrow{O_iM_i} = \dfrac{L_0}{2}\vec{x}_i$;

— l'effort exercé par l'actionneur est modélisé par un glisseur (figure 6) tel que $\{\mathcal{T}_{\mathrm{actionneur}\to\mathrm{membrane}}\} = \begin{Bmatrix} -F\vec{y}_0 \\ \vec{0} \end{Bmatrix}_{O_{n+1}}$;

— le segment 0 possède un degré de liberté selon l'axe $\vec{x}_0$ grâce à une liaison glissière avec le bâti d'axe $\vec{x}_0$.

Q7. Appliquer le principe fondamental de la statique au segment i au point O_i dans la base $\mathcal{B}_0 = (\vec{x}_0, \vec{y}_0, \vec{z}_0)$. En déduire trois équations scalaires par projection dans le repère.

Q8. En faisant l'hypothèse $\sin\theta_i \simeq \theta_i$ et $\cos\theta_i \simeq 1$, linéariser les 3 équations ainsi obtenues.

II.3.2 Application dans le cas de deux segments

Afin de mettre en place une méthode de résolution, nous utilisons le cas d'un système comportant simplement 2 segments (figure 7 page 7). On a alors les valeurs numériques suivantes :

— la longueur d'un segment i est de 1,25 mm ;

— l'effort $\vec{P}$ exercé par le fluide sur chaque segment est considéré constant et de norme $P = 10$ N ;

— la raideur de torsion k_t est identique pour chaque nœud avec $k_t = 5$ N/rad ;
— la norme de l'effort exercé par l'actionneur $\vec{F}_{\text{membrane}}$ est : $F = 8$ N ;
— on suppose par la suite que θ_3 est nul ($\theta_3 = 0$).

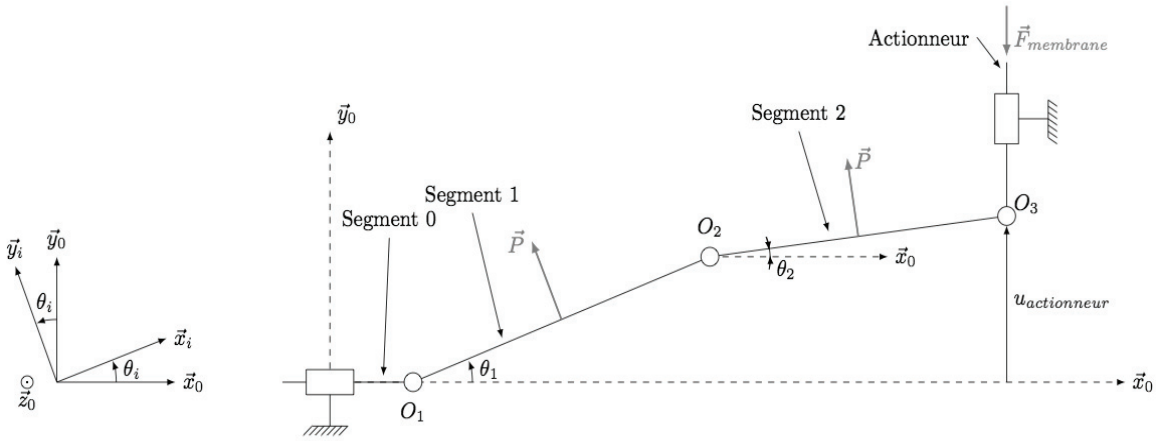


Figure 7 – Modélisation de la membrane avec 2 segments

Les 6 équations issues du Principe Fondamental de la Statique appliqué successivement au segment 1 au point O_1 puis au segment 2 au point O_2, ainsi que les deux conditions limites portant sur x_1 et y_3 permettent d'obtenir le système matriciel ci-dessous :

$$\begin{bmatrix} 1 & 0 & -P & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -2k_t & 0 & -L_0 & k_t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -P & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & k_t & 0 & 0 & -2k_t & 0 & -L_0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ \theta_1 \\ x_2 \\ y_2 \\ \theta_2 \\ x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -P \\ -\frac{PL_0}{2} \\ 0 \\ -P \\ -\frac{PL_0}{2} \\ 0 \\ F \end{bmatrix}.$$

Le problème discret mis en équation peut ainsi être traité numériquement. Pour déterminer les déplacements de la membrane en fonction des actions mécaniques, il est nécessaire d'inverser le système matriciel précédent.

Pour cela, il est décidé d'utiliser un algorithme basé sur la méthode du pivot de Gauss, permettant de déduire les inconnues de liaisons x_1,y_1, θ_1....

Une version de l'algorithme du pivot de Gauss est donnée dans l'annexe 2.

On appelle pour la résolution le programme **Gauss** et on entre la ligne de commande $sol = Gauss(H,G)$ avec les matrices $[H]$ et $[G]$ suivantes :

$$[H] = \begin{bmatrix} 1 & 0 & -10 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -10 & 0 & -0{,}00125 & 5 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -10 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 5 & 0 & 0 & -10 & 0 & -0{,}00125 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad [G] = \begin{bmatrix} 0 \\ -10 \\ -0{,}00625 \\ 0 \\ -10 \\ -0{,}00625 \\ 0 \\ 8 \end{bmatrix}.$$

Q9. Donner la sortie affichée par la ligne de commande *Affichage 1* (ligne 37 pour le code Python et 42 pour le code Scilab) à la **première itération (i=1)** .

Remarque : la fonction NP.CONCATENATE(A,B.T), ligne 5 du code python, permet d'assembler A avec la transposée de B par la droite.

Le résultat donné par l'application du programme est le vecteur solution suivant :

$$sol = \begin{bmatrix} x_1 & y_1 & \theta_1 & x_2 & y_2 & \theta_2 & x_3 & y_3 \end{bmatrix}^T = \begin{bmatrix} 0 & -12 & 0{,}000916 & 0 & -2 & 0{,}0000833 & 0 & 8 \end{bmatrix}^T.$$

Q10. En déduire le déplacement de la membrane noté $u_{\text{actionneur}}$ sur la figure 7.

II.3.3 Modélisation dynamique

La modélisation permet de connaître le comportement statique de la membrane et donc son comportement sous charge. Néanmoins, il est nécessaire d'étudier son comportement dynamique pour évaluer la fonction de transfert de la membrane. Ainsi, la modélisation dynamique revient à écrire le système matriciel suivant :

$$\begin{bmatrix} 0 & -J \\ -J - mL^2 & -mL^2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} k_t & -2k_t \\ -2k_t & k_t \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} LF - P\frac{L}{2} \\ -LF + P\frac{L}{2} \end{bmatrix}.$$

Or, il a été vu à la **Q6** que le débit est directement proportionnel à P (via les pertes de charge dans la conduite). Ainsi, le système matriciel précédent peut s'écrire sous la forme :

$$[A][\ddot{\Theta}] + [B][\dot{\Theta}] + [C][\Theta] = [F] \quad (1)$$

où $[\Theta] = [\theta_1 \theta_2]$ et $[F]$ est une matrice 2×1 exprimée en fonction de F et L.

On cherche à résoudre l'équation différentielle (1) à l'aide de la méthode d'Euler à deux pas. Pour simplifier la programmation, nous nous intéressons dans la suite à la résolution d'une équation différentielle d'ordre 2 non matricielle, de la forme :

$$a.\ddot{y} + b.\dot{y} + c.y = f$$

où a, b, c et f sont des scalaires et y la solution recherchée. Une partie du programme de résolution de cette équation par la méthode d'Euler est donnée ci-après, la déclaration des variables

a, b, c et f étant supposée déjà effectuée. De plus, *dt* et *nbdt* correspondent respectivement au pas de temps et au nombre de pas de temps de la simulation.

Version Python

```
import numpy as np
def derivee2(y,dy)
        ddy = (f-c*y-b*dy)/a
        return ddy

def Euler_ordre2(dt,nbdt):
    y=[0]   #condition init. nulle
    dy=[0]  #condition init. nulle
    for i in range(nbdt):
        dy=dy+...  #a completer
        y=y+[y[i]+dt*dy[i]]
  return y
```

Version Scilab

```
function ddy=derivee2(y,dy)
        ddy = (f-c*y-b*dy)/a
endfunction

function y=Euler_ordre2(dt,nbdt)
        y=[0]    //condition init .nulle
        dy=[0]   //condition init. nulle
        for i=1:nbdt
                dy(i+1)=  //a completer
                y(i+1)=y(i)+dt*dy(i)
        end
endfunction
```

Q11. Ecrire la ligne à compléter dans le code précédent permettant d'effectuer une résolution de l'équation différentielle précédente par la méthode d'Euler.

Le résultat donne l'évolution de Θ en fonction du temps. Il est alors possible de déterminer le déplacement de la membrane en fonction du temps et donc le débit effectif de fluide.

On prendra par la suite comme fonction de transfert de la membrane :

$$H_{membrane}(p) = \frac{Q(p)}{F_m(p)} = \frac{K_m p}{\omega_0^2 p^2 + 1} = \frac{0{,}00023p}{0{,}00015p^2 + 1}.$$

II.4 Commande de l'actionneur piézoélectrique

Objectif

L'objectif de cette partie est de décrire la commande de l'actionneur piézoélectrique.

L'actionneur piézoélectrique (figure 8) est actionné en cas d'insuffisance de l'actionneur thermique à respecter le débit imposé.

Cette condition est vérifiée par le capteur de débit étudié ci-après et des capteurs de température disposés sur la membrane. L'objectif de ces capteurs est de vérifier que la température de la membrane ne dépasse pas la valeur imposée par le cahier des charges (45 °C).

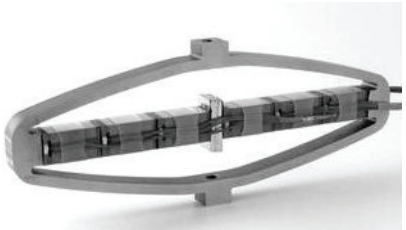


Figure 8 – Actionneur piézoélectrique utilisé

Q12. En supposant que le capteur de débit fournit la valeur du débit notée $q_{\text{réel}}$ et que le capteur de température fournit la valeur de la température notée $t_{membrane}$, écrire une fonction qui s'appelle ACTIONNER(TMEMB) qui retourne la valeur True si $t_{membrane} \leq 45$ et False si $t_{membrane} > 45$ selon la température.

Dans la suite du sujet, nous supposerons que l'actionneur piézoélectrique n'est pas activé.

II.5 Débitmètre à fil chaud

Objectif
L'objectif de cette partie est de modéliser le fonctionnement du débimètre à fil chaud utilisé comme capteur de débit dans l'asservissement.

II.5.1 Principe de base d'un anémomètre à fil chaud : loi de King

Un anémomètre à fil chaud (non miniaturisé) est constitué d'un fil d'environ $\ell = 1$ mm de long et de diamètre d de l'ordre de quelques µm. Les mesures sont le plus souvent effectuées dans des souffleries (écoulement d'air allant de 0,1 m/s à plusieurs centaines de m/s). Le principe de l'anémométrie à fil chaud est basé sur le refroidissement éolien et consiste à mesurer la puissance thermique transférée depuis un fil chauffé par effet Joule et refroidi par le passage du fluide. La puissance emportée donne une mesure indirecte de la vitesse d'écoulement V (figure 9).

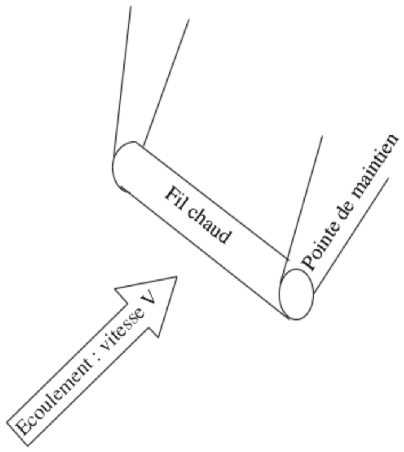


Figure 9 – Fil chaud dans l'écoulement

On note m la masse du fil, c la capacité thermique massique du matériau formant le fil, T_w la température du fil, R_w la résistance électrique du fil, T_0 la température supposée uniforme du fluide loin du fil. L'intensité du courant électrique traversant le fil est I et la puissance thermique transférée du fil vers l'extérieur est noté P_{th}.

Q13. Si le fil est plus chaud que l'extérieur, quel est le signe de P_{th} si le système considéré est le morceau de fil ?

Q14. Effectuer un bilan d'enthalpie sur le fil et en déduire que T_w satisfait à l'équation différentielle

$$\alpha \frac{\mathrm{d}T_w}{\mathrm{d}t} = R_w I^2 - P_{th}$$

où l'on déterminera α en fonction des données du problème.

Q15. La puissance thermique évacuée par le fil peut être transférée selon 4 possibilités différentes. Lesquelles ?

Q16. Parmi les 4 possibilités de transfert thermique, nous ne retiendrons que la conduction et la convection vers le fluide. Ce transfert se fait par la surface latérale $\mathcal{A}$ du fil. Exprimer $\mathcal{A}$ en fonction des dimensions du fil. On appelle $\vec{j_q}$ le vecteur densité volumique de courants thermiques à la surface du fil. En supposant le fil assez long pour négliger les effets de bord, comment est orienté le vecteur $\vec{j_q}$? De quelles variables dépend-il ? Si on suppose de plus $\vec{j_q}$ uniforme en norme à la surface du fil, que vaut P_{th} ?

Q17. $\vec{j_q}$ à la surface est donné par la relation $\| \vec{j_q} \| = h \mid T_w - T_0 \mid$. Quelle est l'unité de h ?

Q18. Rappeler, en explicitant chacun des termes, l'expression de la loi de Fourier. On introduira la conductivité thermique λ_f du fluide environnant. Le nombre de Nusselt $\mathcal{N}_u$ permet de comparer le transfert thermique avec ou sans écoulement du fluide environnant ($\mathcal{N}_u$ est d'autant

plus grand que la vitesse V de l'écoulement est grande). On montre que ce nombre vaut dans le cas du fil chaud $\mathcal{N}_u = \dfrac{hd}{\lambda_f}$. Vérifier que cette quantité est bien sans dimension.

Q19. Montrer alors qu'en régime permanent, l'équation différentielle obtenue **Q14** se simplifie en $R_w I^2 = \pi \ell \lambda_f (T_w - T_0)\mathcal{N}_u$.

Q20. Application numérique : le fil chaud dissipe une puissance de 0,25 W. La conductivité thermique de l'air est de 0,02 $\mathrm{W{\cdot}K^{-1}{\cdot}m^{-1}}$ et la différence de température est de l'ordre de 200 °C. Que vaut le nombre $\mathcal{N}_u$? Que dire du transfert par convection par rapport au transfert par conduction ?

Q21. En 1914, King a proposé la loi suivante : $\mathcal{N}_u = a + b\sqrt{\mathcal{R}_e}$ où $\mathcal{R}_e$ est le nombre de Reynolds et a et b deux coefficients qui ne dépendent pas de la vitesse de l'écoulement V. Quelle est l'expression du nombre de Reynolds dans ce problème ? On introduira le coefficient de viscosité dynamique du fluide noté η et la masse volumique du fluide notée ρ. Comment varie alors $\mathcal{N}_u$ avec la vitesse V ?

Le fil chaud est fait d'un matériau dont la résistivité électrique dépend de la température de manière affine. La résistance R_w du fil s'écrit alors $R_w(T_w) = R_0(1+\alpha(T_w - T_0))$ avec α une constante dépendant du matériau.

Q22. Montrer alors que $\dfrac{R_w I^2}{(R_w - R_0)} = a_1 + b_1\sqrt{V}$ où l'on exprimera a_1 et b_1 en fonction de ℓ, λ_f, α, R_0, d, a, b, η et de ρ.

II.5.2 Electronique d'asservissement : anémométrie à température constante (CTA)

On vient de voir que la résistance du fil dépend directement de la vitesse V de l'écoulement. L'anémométrie à température constante (CTA) consiste à garder la résistance R_w constante et donc la température T_w du fil constante. On mesurera donc V à travers les fluctuations de l'intensité I qui traverse le fil chaud.

Q23. La résistance du fil chaud est insérée dans un circuit type « pont de Wheatstone ». Ce circuit comporte deux résistances égales à R_1, une résistance R_x que l'on peut faire varier et le fil chaud représenté par la résistance R_w. En utilisant deux diviseurs de tension bien choisis, montrer que la tension e est égale à $e = E\left(\dfrac{1}{1+\delta} - \dfrac{1}{1+\beta}\right)$ où l'on précisera les expressions de β et δ. Quelle est la condition sur R_w et R_x pour que le pont soit équilibré, c'est-à-dire $e = 0$?

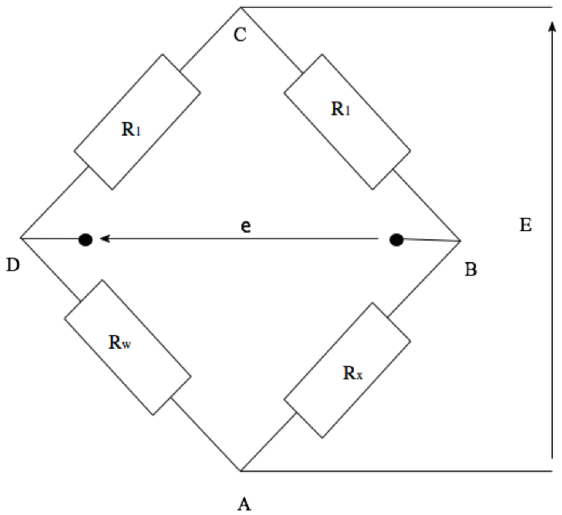


Figure 10 – Pont de Wheatstone

On peut choisir d'équilibrer le pont ($e = 0$) en jouant sur la valeur de R_x, ce qui va fixer la température de travail du fil chaud.

Q24. Si on augmente la valeur de R_x, est-ce qu'on sélectionne une température de travail plus élevée ou plus faible ? Dans le cas d'un fluide médicamenteux, pourquoi vaut-il mieux choisir R_x de telle façon que la température du fil chaud n'excède pas 100 °C ?

Lorsque la vitesse V de l'écoulement varie, le pont sera déséquilibré car R_w va varier. Afin de maintenir la température constante, le circuit électrique doit comporter une boucle de rétroaction. Le circuit est donné figure 11. L'Amplificateur Linéaire Idéal (ALI) fonctionne en régime linéaire et est idéal.

Q25. Que valent les courants d'entrée i_+ et i_- respectivement dans les bornes d'entrée + et − de l'ALI ? Que vaut la tension entre ces deux bornes d'entrée dans le cas d'un fonctionnement linéaire ?

Q26. Montrer que ce circuit va permettre d'ajuster le courant I pour que le fil chaud soit maintenu à température constante lorsque la vitesse V de l'écoulement varie.

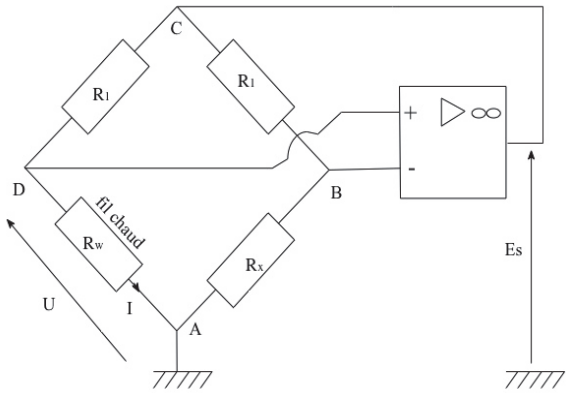


Figure 11 – Circuit avec rétroaction

Q27. Montrer que la tension de sortie de l'ALI E_s vaut γU où l'on exprimera γ en fonction de R_1 et R_w. En vous appuyant sur les questions **Q22** et **Q26**, montrer que la tension de sortie de l'ALI vérifie la relation ${E_s}^2 = A + B\sqrt{V}$ où l'on ne cherchera pas à établir les expressions de A et de B. Cette relation s'appelle la loi de King.

II.5.3 Validation expérimentale de la modélisation

Comme il est difficile de contrôler tous les paramètres qui interviennent dans la loi de King, les coefficients A et B sont déterminés par un étalonnage empirique. Pour chaque vitesse V de l'écoulement, on relève la tension de sortie E_s. Les résultats sont résumés dans le tableau ci-dessous.

V (m·s^{-1})	0	0,5	1,0	1,5	2,0	2,5	3,0	4,0	5,0
E_s (V)	4,0	5,0	5,4	5,7	6,0	6,2	6,4	6,7	7,0

Q28. Que vaut le coefficient A ?

Q29. La loi de King est un modèle trop fort. On observe qu'il est plus facile d'ajuster les valeurs au modèle suivant : ${E_s}^2 = A + BV^n$ où l'exposant n est compris entre 0,4 et 0,6. Quelle courbe doit-on tracer pour trouver l'exposant n ? Déterminer alors cet exposant à l'aide du tableau fourni.

II.5.4 Le débitmètre à fil chaud MEMS

La structure miniaturisée est la suivante :

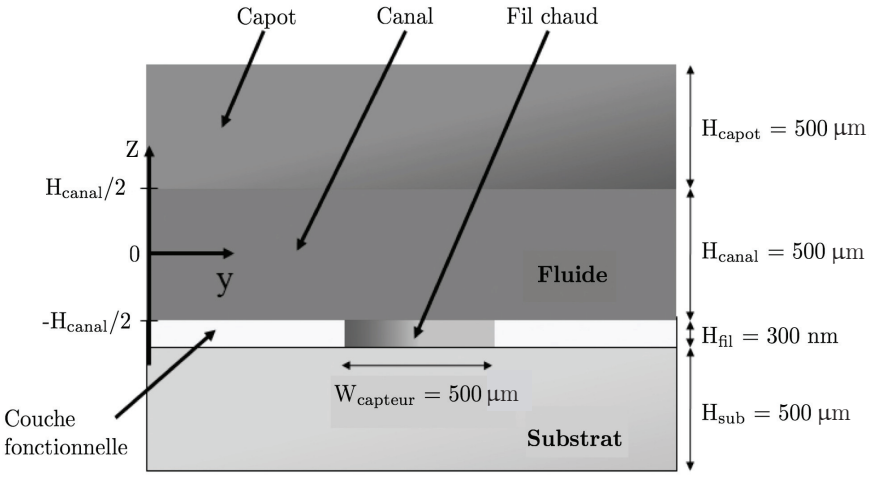


Figure 12 – Structure MEMS du débimètre

On trouve les données (en unités SI) suivantes pour quelques matériaux :

Matériaux	Masse volumique	Capacité thermique massique	Conductivité thermique
Silicium	2 330	700	130
Verre	1 650	730	1,4
Plastique	130	1 180	0,2
Air	1,19	1 006	0,023

Q30. Quel matériau préconiseriez-vous pour le substrat ? Justifiez !

On supposera par la suite que le débimètre, étudié dans cette partie et utilisé comme capteur de débit dans l'asservissement du système, est modélisable par un gain pur de coefficient : $H_{\text{capt}}(p) = \dfrac{K_{capt}}{p}$ avec $K_{capt} = 3{,}6.10^3$ V.m^{-3}.s^{-1}.

II.6 Simulation de l'asservissement en volume injecté

Objectif
L'objectif de cette partie est de mettre en œuvre le modèle théorique de l'asservissement et d'effectuer la simulation permettant de vérifier les performances de l'asservissement.

Le schéma-bloc retenu pour cette partie est donné ci-dessous. Il correspond à un modèle simplifié du schéma-bloc fonctionnel donné figure 15 - annexe 1.

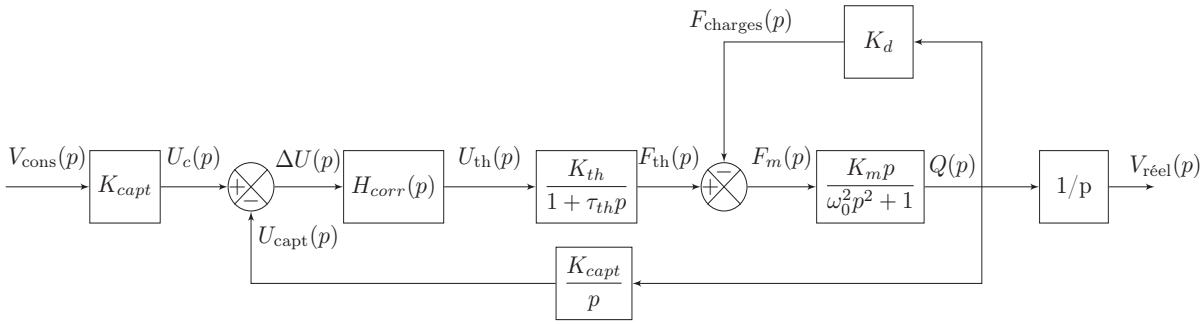


Figure 13 – Schéma-bloc fonctionnel simulé

On suppose tout d'abord que le correcteur 1 (figure 15 - annexe 1) est un gain pur de la forme $H_{\text{corr}}(p) = K_{\text{corr}}$.

Q31. Déterminer la fonction de transfert de l'asservissement du volume délivré $H(p) = \dfrac{V_{\text{réel}}(p)}{V_{\text{cons}}(p)}$.

Q32. Le système est-il précis pour une entrée de type échelon de volume ? Justifier.

Une simulation effectuée avec un correcteur de gain $K_{\text{corr}} = 1$ et une entrée échelon d'amplitude 5 mL donne le résultat suivant :

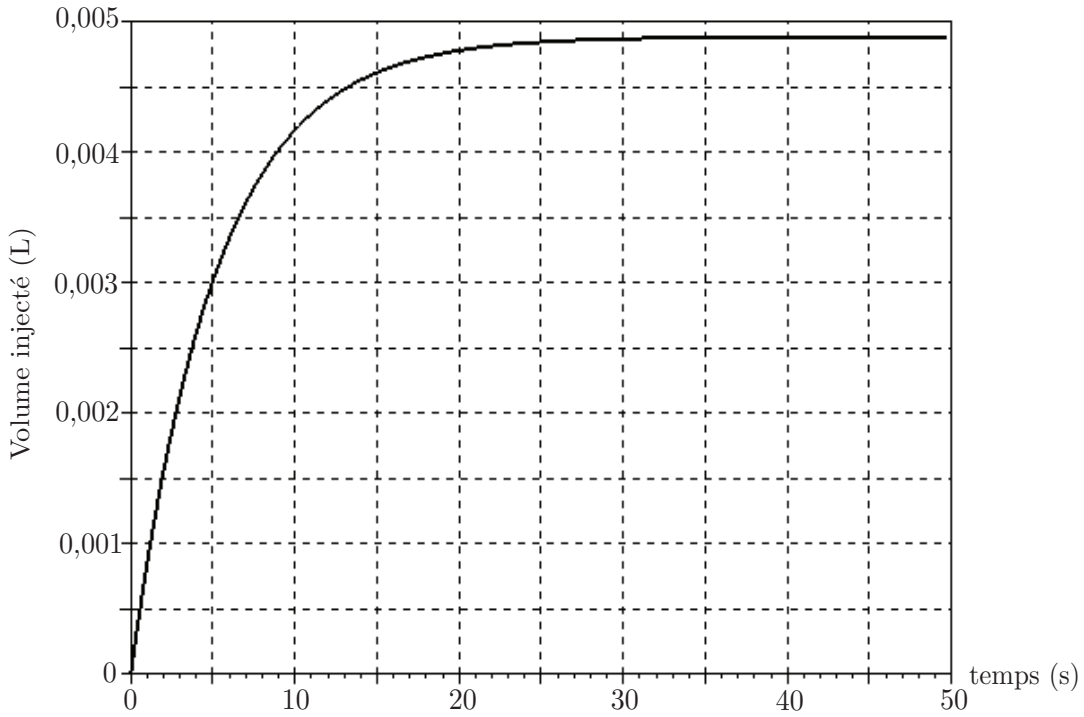


Figure 14 – Simulation d'injection d'un volume de 5 mL avec $K_{\text{corr}} = 1$

Q33. Le système ainsi réalisé respecte-t-il les exigences définies par le constructeur ?

Annexe 1 - Schéma-bloc fonctionnel simplifié

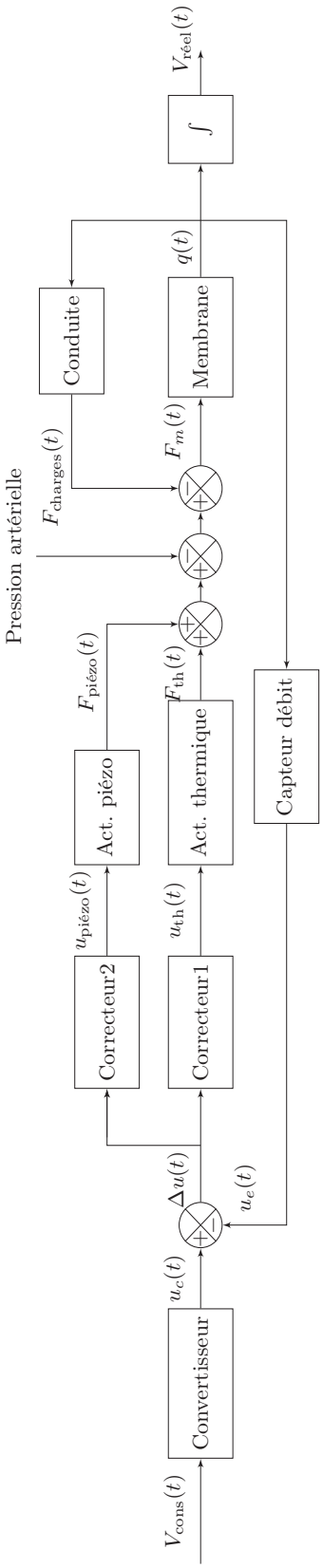


Figure 15 – Schéma-bloc fonctionnel simplifié

On retrouve ainsi sur ce schéma-bloc fonctionnel les grandeurs :

- $V_{\text{cons}}(t)$ le volume de la solution à injecter ;
- $V_{\text{réel}}(t)$ le volume réel déjà injecté de la solution ;
- $u_c(t)$ la tension consigne proportionnelle au volume à injecter ;
- $u_{\text{piézo}}$ et u_{th} les tensions de commande après correction envoyées aux actionneurs piézoélectrique et thermique ;
- F_{th}, $F_{\text{piézo}}$, F_{charges} les actions mécaniques exercées sur la membrane par respectivement l'actionneur thermique, l'actionneur piézoélectrique et la pression de la solution injectée créée par les pertes de charge dans la conduite et la pression artérielle ;
- $q(t)$ le débit de solution ;
- $u_e(t)$ la tension image du volume de solution injecté.

Annexe 2 - Code informatique pour le pivot de Gauss (Q9).

Version Python

```
import numpy as np
def Augmente(A,B):
    n=len(A)
    m=len(B.T)
    M=np.concatenate((A,B.T),axis=1)
    return M

def Echligne(M,i,j):
    N=np.copy(M[i])
    M[i]=M[j]
    M[j]=N
    return M

def Pivot(M,i):
    n=len(M)
    j=i
    for k in range(i+1,n):
        if abs(M[k,i])>abs(M[j,i]):
            j=k
    return j

def Elimine(M,i,j):
    a=-M[j,i]/M[i,i]
    M[j]+=a*M[i]
    return M

def Normalise_diagonale(M):
    n=len(M)
    for i in range(n):
        M[i]/=M[i,i]
    return M

def Gauss(A,B):
    M=Augmente(A,B)
    n=len(M)
    for i in range(n-1):
        p=Pivot(M,i)
        if i!=p:
            M=Echligne(M,i,p)
        for j in range(i+1,n):
            Elimine(M,i,j)
        print(M,"\n")   #Affichage1
    Normalise_diagonale(M)
    for i in range(n-1,0,-1):
        for j in range(i-1,-1,-1):
            Elimine(M,i,j)
    return (M)
```

Version Scilab

```
function M=Augmente(A,B)
    n=size(A,"r")
    m=size(B,"r")
    M=A
    for i=1:n
        M(i,n+1)=B(i)
    end
endfunction

function M=Echligne(M,i,j)
    x=M(i,:)
    M(i,:)=M(j,:)
    M(j,:)=x
endfunction

function r=Pivot(M,i)
    n=size(M,"r")
    r=i
    for k=i+1:n
        if abs(M(k,i))>abs(M(r,i)) then
            r=k
        end
    end
endfunction

function M=Elimine(M,i,j)
        a=-M(j,i)/M(i,i)
        M(j)=M(j)+a*M(i)
endfunction

function M=Normalisediagonale(M)
    n=size(M,"r")
    for i=1:n
        M(i)=M(i)/M(i,i)
    end
endfunction

function M=Gauss(A,B)
    n=size(A,"r")
    M=Augmente(A,B)
    for i=1:n-1
        i_piv=Pivot(M,i)
        if (i!=i_piv) then
            M=Echligne(M,i,i_piv)
        for j=i+1:n
            M=Elimine(M,i,j)
        disp (M,"\n")   //Affichage1
    M=Normalisediagonale(M)
    for i=n-1:-1:1
        for j=i-1:-1,0
            M=Elimine(M,i,j)
endfunction
```

Fin de l'énoncé

CONCOURS COMMUNS POLYTECHNIQUES

EPREUVE SPECIFIQUE - FILIERE TPC

MODELISATION

Durée : 4 heures

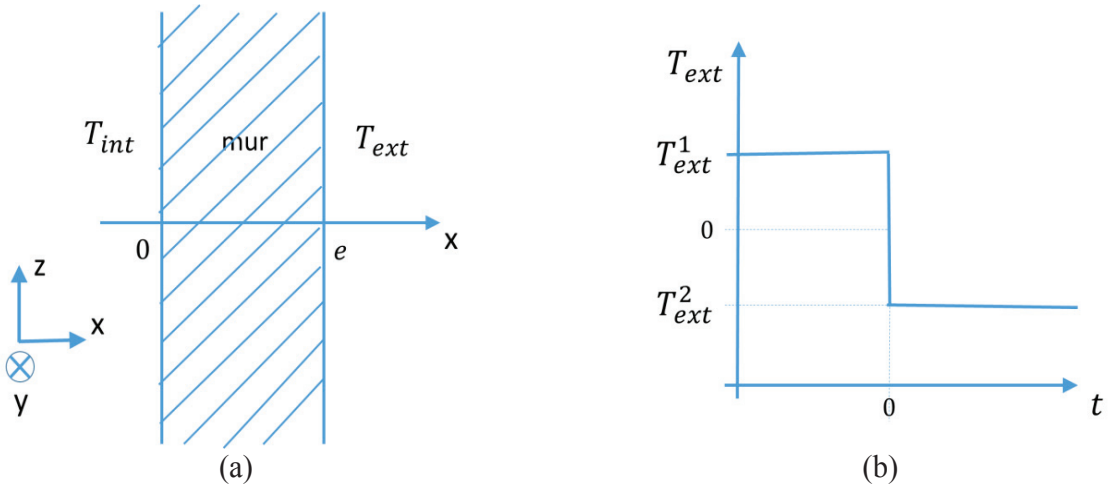
N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Les calculatrices sont interdites

Le sujet comporte deux parties indépendantes. Le candidat précisera au début de sa copie le langage de programmation (Python ou Scilab) qu'il a choisi et toutes les questions seront traitées dans le même langage. Un bonus sera accordé aux copies soignées avec des programmes bien commentés. Plusieurs fonctions du langage Scilab sont rappelées en annexe A. Les candidats choisissant le langage Python pourront utiliser les bibliothèques numpy et matplotlib.pyplot. Une documentation simplifiée de plusieurs fonctions de ces bibliothèques est présente en annexes B et C.

SIMULATION NUMERIQUE DU TRANSFERT THERMIQUE DANS UN MUR EN REGIME TRANSITOIRE

On étudie les transferts thermiques dans le mur d'une maison, figure 1(a). La température à l'intérieur de la maison est constante dans le temps et égale à $T_{int} = 20$ °C. Aux temps négatifs ($t < 0$), la température extérieure est égale à $T_{ext1} = 10$ °C. A $t = 0$, elle chute brusquement à $T_{ext2} = -10$ °C et elle reste égale à cette valeur aux temps positifs ($t > 0$), figure 1(b). On souhaite étudier l'évolution du profil de température dans le mur au cours du temps.



Figures 1 (a) - Schéma du mur étudié. (b) - Evolution de la température extérieure au cours du temps.

Le mur a une épaisseur $e = 40$ cm. Les propriétés physiques du mur sont constantes : conductivité thermique $\lambda = 1{,}65\ \mathrm{W.m^{-1}.K^{-1}}$, capacité thermique massique $c_p = 1\,000\ \mathrm{J.kg^{-1}.K^{-1}}$, masse volumique $\rho = 2\,150\ \mathrm{kg.m^{-3}}$.

PARTIE I : ETUDE PRELIMINAIRE

Dans cette partie, on établit l'équation gouvernant les variations de la température et on la résout en régime permanent.

I.A. Equation gouvernant la température

On suppose que la température dans le mur T ne dépend que du temps t et de la coordonnée x.

I.A.1. A quelle condition peut-on supposer que la température ne dépend pas des coordonnées y et z ?

I.A.2. Donner l'équation générale qui décrit le transport de chaleur dans un solide en l'absence de source d'énergie. Comment cette équation se simplifie-t-elle sous les hypothèses de la question I.A.1 ?

I.B. Conditions aux limites

On envisage plusieurs types de conditions aux limites.

(i) La température est imposée aux limites du système.

(ii) La paroi extérieure est isolée par un matériau de très faible conductivité.

I.B.1. Traduire chacune de ces conditions aux limites sur la fonction $T(x,t)$ et/ou sa dérivée.

Dans toute la suite, on adoptera des conditions aux limites de type température imposée.

I.C. Solutions en régime permanent

I.C.1. Résoudre l'équation obtenue à la question I.A.2. en régime permanent, avec les conditions aux limites de type températures imposées (question I.B.(i)) :

- pour un instant particulier négatif $t_1 < 0$,
- pour un instant particulier positif $t_2 > 0$, très longtemps après la variation de température extérieure, quand le régime permanent est de nouveau établi dans le mur.

I.C.2. Quelle est la nature des profils $T(x)$ obtenus (en régime permanent) à ces deux instants ? Tracer à la main les deux profils sur un même graphique sur la copie.

I.C.3. Sur le même graphique, tracer à la main qualitativement les profils intermédiaires à différents instants entre la variation brutale de la température extérieure ($t = 0$) et l'instant t_2 où le régime est de nouveau permanent.

PARTIE II : RESOLUTION NUMERIQUE

II.A. Equation à résoudre

On cherche à résoudre numériquement l'équation aux dérivées partielles :

$$\alpha \frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} \qquad (1)$$

où α est une constante. A l'équation (1) sont associées les conditions :

$$\begin{array}{ll} T(0,t) = T_{int} & \text{pour tout } \ t > 0 \\ T(e,t) = T_{ext2} & \text{pour tout } \ t > 0 \\ T(x,0) = ax + b & \text{pour tout } \ x \in [0,e] \end{array}$$

II.A.1. Quelle est l'expression de α en fonction des paramètres physiques du mur ?

II.A.2. Exprimer a et b en fonction de T_{int}, T_{ext1} et e.

Pour effectuer la résolution de l'équation (1), nous utiliserons la méthode des différences finies présentée dans la partie II.B.

II.B. Méthode des différences finies

II.B.1. Discrétisation dans l'espace et dans le temps

On divise l'intervalle $[0, e]$, représentant l'épaisseur du mur, en $N + 2$ points, numérotés de 0 à $N + 1$, régulièrement espacés de Δx (figure 2, page suivante). Cette division est appelée « discrétisation ». La distance Δx est appelée le « pas d'espace ». A l'intérieur du mur (frontières intérieure et extérieure exclues) se trouvent donc N points. On cherche à obtenir la température en ces points particuliers à chaque instant.

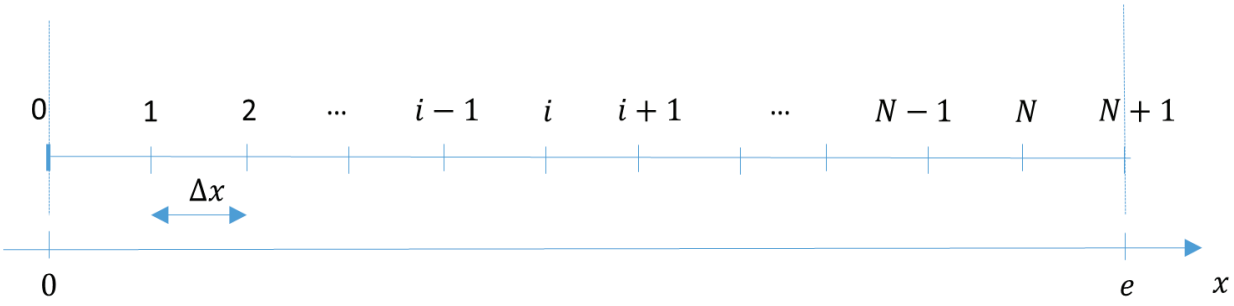


Figure 2 - Discrétisation spatiale dans la direction x.

II.B.1.a. Donner l'expression de Δx en fonction de N et de l'épaisseur du mur e.

II.B.1.b. Donner l'abscisse x_i du i^e point en fonction de i et Δx, sachant que $x_0 = 0$ et $x_{N+1} = e$.

Le temps est discrétisé en $ItMax$ intervalles de durée Δt et on ne s'intéresse au profil de température qu'aux instants particuliers $t_k = k.\Delta t$. L'intervalle élémentaire de temps Δt est appelé le « pas de temps ».

Pour résoudre l'équation (1), deux méthodes sont proposées :

- méthode utilisant un schéma explicite,
- méthode utilisant un schéma implicite.

II.B.2. Méthode utilisant un schéma explicite

II.B.2.a. A l'aide d'un développement limité de la fonction $x \longmapsto T(x,t)$, donner une expression de $T(x+\Delta x, t)$ à l'ordre 3 ($o(\Delta x^3)$) en fonction de T et de ses dérivées partielles par rapport à x évaluées en (x,t). De même, donner une expression de $T(x-\Delta x, t)$ à l'ordre 3.

II.B.2.b. En déduire une expression approchée à l'ordre 1 ($o(\Delta x)$) de $\left.\frac{\partial^2 T}{\partial x^2}\right|_{x,t}$ (dérivée partielle spatiale seconde de T évaluée au point x à l'instant t) en fonction de $T(x+\Delta x, t)$, $T(x-\Delta x, t)$ et $T(x,t)$ et Δx.

On note T_i^k la température $T(x_i, t_k)$, évaluée au point d'abscisse x_i à l'instant t_k. De même, on note $T_{i+1}^k = T(x_i + \Delta x, t_k)$ et $T_{i-1}^k = T(x_i - \Delta x, t_k)$.

II.B.2.c. Déduire de la question précédente une expression approchée de $\left.\frac{\partial^2 T}{\partial x^2}\right|_{x_i,\, t_k}$ (dérivée partielle spatiale seconde de T évaluée en x_i à l'instant t_k) en fonction de T_i^k, T_{i+1}^k et T_{i-1}^k et Δx.

La dérivée partielle temporelle de l'équation (1) est maintenant approchée grâce à un développement limité.

II.B.2.d. A l'aide d'un développement limité de la fonction $t \longmapsto T(x,t)$, donner une expression de $T(x, t+\Delta t)$ à l'ordre 1 ($o(\Delta t)$) en fonction de T et de sa dérivée partielle par rapport à t évaluées en (x,t).

II.B.2.e. En déduire une valeur approchée de $\left.\frac{\partial T}{\partial t}\right|_{x,t}$ (dérivée partielle par rapport au temps de T évaluée au point x à l'instant t) à l'ordre 0 ($o(1)$) en fonction de $T(x, t+\Delta t)$, $T(x,t)$ et Δt.

II.B.2.f. Donner une expression de $\left.\frac{\partial T}{\partial t}\right|_{x_i,\, t_k}$ (dérivée partielle par rapport au temps de T évaluée en x_i à l'instant t_k) en fonction de Δt, T_i^k et T_i^{k+1}, avec $T_i^{k+1} = T(x_i, t_k + \Delta t)$.

L'équation (1) est valable en chaque point d'abscisse x_i et à chaque instant t_k.

II.B.2.g. Ecrire la forme approchée de cette équation au point i et à l'instant k en approchant $\left.\frac{\partial^2 T}{\partial x^2}\right|_{x,t}$ avec la formule obtenue à la question II.B.2.c. et en approchant $\left.\frac{\partial T}{\partial t}\right|_{x,t}$ avec la formule obtenue à la question II.B.2.f.

II.B.2.h. Montrer que l'équation obtenue à la question II.B.2.g peut s'écrire sous la forme :

$$T_i^{k+1} = rT_{i-1}^k + (1-2r)T_i^k + rT_{i+1}^k \qquad (2)$$

en précisant la valeur du paramètre r en fonction de $\Delta x, \Delta t$ et α.

L'équation (2) est appelée schéma numérique explicite. Si on connaît la température en tous les points $x_1, x_2, \dots, x_{N-1}, x_N$ à l'instant t_k, on peut calculer grâce à elle la température en tous les points à l'instant ultérieur t_{k+1}.

II.B.2.i. L'équation (2) est-elle valable dans tout le domaine, c'est-à-dire pour toute valeur de i, $0 \le i \le N+1$? Que valent T_0^k et T_{N+1}^k ?

II.B.2.j. Dans cette question, on élabore une fonction `schema_explicite` permettant de calculer la température en chaque point au cours du temps selon la formule (2). Parmi les variables d'entrée se trouvera un vecteur `T0` de dimension N, défini en dehors de la fonction, contenant les valeurs de la température aux points de discrétisation à l'instant initial. Au sein de la fonction, un algorithme calculera itérativement la température avec un nombre maximal d'itérations `ItMax`. En sortie de la fonction, on récupérera le nombre d'itérations réellement effectuées, `nbIter` et une matrice `T_tous_k`, de dimensions $N \times ItMax$. Chaque colonne de cette matrice contient le vecteur $\boldsymbol{T^k}$ dont les éléments sont les valeurs de la température aux N points $x_1, \dots, x_N$ (points à l'intérieur du mur) à l'instant k :

$$\boldsymbol{T}^k = \begin{pmatrix} T_1^k \\ T_2^k \\ \dots \\ T_{N-1}^k \\ T_N^k \end{pmatrix} \quad \text{et} \quad \texttt{T\_tous\_k} = \begin{pmatrix} T_1^1 & T_1^2 & \dots & T_1^k & \dots & T_1^{k-1} & T_1^k \\ T_2^1 & T_2^2 & & T_2^k & & T_2^{k-1} & T_2^k \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ T_{N-1}^1 & T_{N-1}^2 & \dots & T_{N-1}^k & \dots & T_{N-1}^{k-1} & T_{N-1}^k \\ T_N^1 & T_N^2 & & T_N^k & & T_N^{k-1} & T_N^k \end{pmatrix}.$$

On souhaite arrêter le calcul lorsque la température ne varie presque plus dans le temps. Dans ce but, on évaluera la norme 2 de $\boldsymbol{T}^k - \boldsymbol{T}^{k-1}$ à chaque itération. La définition de la norme 2 est rappelée à la question II.B.2.j.*(vi)*.

II.B.2.j.*(i)* Ecrire l'en-tête de la fonction en précisant bien les paramètres d'entrée et de sortie.

II.B.2.j.*(ii)* Le schéma numérique (2) permet d'approcher avec succès la solution à la condition $r < 1/2$. Programmer un test qui avertit l'utilisateur si cette condition n'est pas respectée.

II.B.2.j.*(iii)* Affecter la valeur 2 000 à `ItMax`. Créer la matrice `T_tous_k` de dimensions $N \times ItMax$ en la remplissant de zéros.

II.B.2.j.*(iv)* Remplacer la première colonne de `T_tous_k` par le vecteur des valeurs initiales `T0`.

II.B.2.j.*(v)* Calculer le profil de température à l'instant $k = 1$ $(t = \Delta t)$, en distinguant le cas $i = 1$, le cas $2 \leq i \leq N - 1$ et le cas $i = N$. Affecter ces valeurs à la deuxième colonne de `T_tous_k`.

II.B.2.j.*(vi)* Ecrire une fonction `calc_norme` qui calcule la norme 2 d'un vecteur. On rappelle que la norme 2 d'un vecteur $\boldsymbol{V}$ s'écrit :

$$\|\boldsymbol{V}\|_2 = \sqrt{\sum_{i=1}^{N} V_i^2} \qquad \text{avec} \qquad \boldsymbol{V} = \begin{pmatrix} V_1 \\ \vdots \\ V_i \\ \vdots \\ V_N \end{pmatrix}.$$

II.B.2.j.*(vii)* Elaborer une boucle permettant de calculer itérativement le profil de température aux instants $t_k = k.\Delta t$ avec $k \geq 2$. Cette boucle sera interrompue lorsque la norme 2 du vecteur $\boldsymbol{T^k} - \boldsymbol{T^{k-1}}$ deviendra inférieure à 10^{-2} ou lorsque le nombre d'itérations atteindra la valeur $ItMax$ (prévoir les deux cas). Utiliser, pour cela, la fonction `calc_norme` définie à la question II.B.2.j.*(vi)*.

II.B.2.j.*(viii)* Ecrire la fin de la fonction afin de renvoyer tous les arguments de sortie définis au début de la question II.B.2.j.

II.B.3. Méthode utilisant un schéma implicite

Le schéma explicite (2) ne converge que si le pas de temps Δt est suffisamment faible par rapport au pas d'espace Δx. Si l'on souhaite effectuer un calcul pour un temps physique long, beaucoup d'itérations seront nécessaires et le temps de calcul sera très long. C'est pourquoi on préfère d'autres types de schémas appelés schémas implicites.

Dans cette partie, la dérivée partielle seconde par rapport à x de la température apparaissant dans l'équation (1) est évaluée au point d'abscisse x_i et à l'instant $k + 1$:

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{x,t} \approx \left.\frac{\partial^2 T}{\partial x^2}\right|_{x_i, t_{k+1}}$$

et la dérivée partielle par rapport à t est évaluée au point d'abscisse x_i et à l'instant k :

$$\left.\frac{\partial T}{\partial t}\right|_{x,t} \approx \left.\frac{\partial T}{\partial t}\right|_{x_i, t_k}.$$

II.B.3.a. Donner la nouvelle expression approchée de l'équation (1) définie en page 3.

II.B.3.b. Montrer que l'équation obtenue à la question II.B.3.a. peut être mise sous la forme

$$T_i^k = -rT_{i-1}^{k+1} + (1 + 2r)T_i^{k+1} - rT_{i+1}^{k+1} \ . \qquad (3)$$

L'équation (3) est appelée schéma implicite car la température à l'instant t_k est exprimée en fonction de la température à l'instant ultérieur t_{k+1}.

Le système d'équations ainsi obtenu peut être écrit sous la forme :

$$M\boldsymbol{T}^{k+1} = \boldsymbol{T}^k + r\,\boldsymbol{v} \qquad (4)$$

où M est une matrice carrée $N \times N$ et $\boldsymbol{T}^k$ et $\boldsymbol{T}^{k+1}$ sont les vecteurs de dimension N définis par :

$$\boldsymbol{T}^k = \begin{pmatrix} T_1^k \\ T_2^k \\ \dots \\ T_{N-1}^k \\ T_N^k \end{pmatrix} \quad \text{et} \quad \boldsymbol{T}^{k+1} = \begin{pmatrix} T_1^{k+1} \\ T_2^{k+1} \\ \dots \\ T_{N-1}^{k+1} \\ T_N^{k+1} \end{pmatrix}$$

et $\boldsymbol{v}$ est un vecteur de taille N faisant intervenir les conditions aux limites.

II.B.3.c. Préciser l'expression de la matrice M et l'expression du vecteur $\boldsymbol{v}$.

A chaque pas de temps, il faut inverser le système matriciel :

$$M\boldsymbol{T}^{k+1} = \boldsymbol{T}^k + r\,\boldsymbol{v}$$

pour obtenir $\boldsymbol{T}^{k+1}$à partir de $\boldsymbol{T}^k$.

II.B.3.d. Le but de cette question est d'écrire une fonction `CalcTkp1` qui permet de résoudre un système matriciel tridiagonal en utilisant l'algorithme de Thomas présenté ci-dessous.

Algorithme de Thomas :
On cherche à résoudre un système matriciel tridiagonal de la forme :

$$M\,\boldsymbol{u} = \boldsymbol{d} \qquad (5)$$

où M est une matrice de dimensions $N \times N$ tridiagonale, c'est-à-dire une matrice dont tous les éléments sont nuls, sauf sur la diagonale principale, la diagonale supérieure et la diagonale inférieure

$$M = \begin{pmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & 0 & \\ & a_3 & b_3 & c_3 & & & \\ & & \ddots & \ddots & \ddots & & \\ & 0 & & & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & & & a_N & b_N \end{pmatrix}$$

et où les vecteurs $\boldsymbol{u}$ et $\boldsymbol{d}$, de dimension N, s'écrivent **:**

$$\boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} \quad \text{et} \quad \boldsymbol{d} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{N-1} \\ d_N \end{pmatrix}$$

Dans cet algorithme, on calcule d'abord les coefficients suivants :

$$c'_1 = \frac{c_1}{b_1}$$
$$c'_i = \frac{c_i}{b_i - a_i c'_{i-1}} \quad \text{pour} \quad i = 2, 3, \dots, N-1$$

et

$$d'_1 = \frac{d_1}{b_1}$$
$$d'_i = \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} \quad \text{pour} \quad i = 2, 3, \dots, N\,.$$

Les inconnues $u_1, u_2, \dots, u_N$ sont alors obtenues par les formules :

$$u_N = d'_N$$
$$u_i = d'_i - c'_i u_{i+1} \quad \text{pour} \quad i = N-1, N-2, \dots, 2, 1\ .$$

II.B.3.d.*(i)* En utilisant l'algorithme de Thomas, écrire une fonction `CalcTkp1` qui permet de calculer le vecteur $\boldsymbol{u}$, solution du système matriciel (5), à partir de la matrice M et du vecteur $\boldsymbol{d}$.

II.B.3.e. Dans cette question, une fonction `schema_implicite` est élaborée avec les mêmes arguments d'entrée et de sortie que la fonction `schema_explicite` (définis à la question II.B.2.j.) et qui utilise les mêmes critères d'arrêt (définis à la question II.B.2.j.*(vii)*).

II.B.3.e.*(i)* Ecrire l'en-tête de la fonction en précisant les paramètres d'entrée et de sortie.

II.B.3.e.*(ii)* Affecter la valeur 2 000 à `ItMax`. Créer la matrice `T_tous_k` dont les dimensions sont $N \times ItMax$ en la remplissant de zéros.

II.B.3.e.*(iii)* Remplacer la 1re colonne de `T_tous_k` par le vecteur des valeurs initiales `T0`.

II.B.3.e.*(iv)* Définir la matrice `M` et le vecteur `v` qui interviennent dans l'équation (4).

II.B.3.e.*(v)* Calculer le profil de température à l'instant $k = 1$ $(t = \Delta t)$. Affecter ces valeurs à la deuxième colonne de `T_tous_k`.

II.B.3.e.*(vi)* Ecrire une boucle permettant de calculer itérativement le profil de température aux instants ultérieurs $t_k = k \times \Delta t$ avec $k \geq 2$, en prévoyant un arrêt lorsque la norme 2 du vecteur $\boldsymbol{T^k - T^{k-1}}$ devient inférieure à 10^{-2} ou lorsque le nombre d'itérations atteint la valeur `ItMax` (prévoir les deux cas). Utiliser pour cela la fonction `calc_norme` définie à la question II.B.2.j.*(vi)*.

II.B.3.e.*(vii)* Ecrire la fin de la fonction afin de renvoyer tous les arguments de sortie définis au début de la question II.B.2.j.

II.C. Programme principal

II.C.1. Début du programme

II.C.1.a. Définir les variables `epais` (épaisseur du mur), `conduc` (conductivité thermique), `rho` (masse volumique), `Cp` (capacité thermique massique), `Tint` (température intérieure), `Text1`

(température extérieure pour les instants $t < 0$), `Text2` (température extérieure pour les instants $t > 0$), `N` (nombre de points de calcul **à l'intérieur du mur**) et `Dt` (intervalle de temps élémentaire) et leur affecter les valeurs correspondant au problème physique défini au début de l'énoncé. On prendra un nombre de points de discrétisation $N = 60$ et un pas de temps Δt de 25 secondes.

II.C.1.b. Calculer les coefficients `a` et `b` avec la formule trouvée à la question II.A.2.

II.C.1.c. Créer un vecteur `x` dont les éléments $x_1, x_2, \ldots, x_N$ sont définis à la question II.B.1.b.

II.C.1.d. Calculer le vecteur des températures initiales `T0`.

II.C.1.e. Calculer `alpha` selon la formule trouvée à la question II.A.1. Calculer `r` en utilisant la formule calculée à la question II.B.2.h.

II.C.2. Calcul des températures

II.C.2.a. Ecrire un morceau de programme qui demande à l'utilisateur quel schéma (explicite ou implicite) il souhaite utiliser et qui appelle la fonction correspondante.

II.C.3. Analyse du résultat

II.C.3.a. Ecrire un morceau de programme permettant de tracer sur un même graphique le profil de température en fonction de `x` tous les 100 pas de temps.

II.C.3.b. Faire afficher le temps en heures au bout duquel le régime permanent est établi.

Fin de l'énoncé

ANNEXE A : COMMANDES ET FONCTIONS USUELLES DE SCILAB

A=[a b c d;e f g h;i j k l]

Description : commande permettant de créer une matrice dont la première ligne contient les éléments a, b, c, d, la seconde ligne contient les éléments e, f, g, h et la troisième, les éléments i, j, k, l.

Exemple : A=[1 2 3 4 5;3 10 11 12 20;0 1 0 0 2]

```
⇨  1.  2.   3.   4.   5.
   3.  10.  11.  12.  20.
   0.  1.   0.   0.   2.
```

A(i,j)

Arguments d'entrée : les coordonnées de l'élément dans le tableau A.

Argument de sortie : l'élément (i, j) de la matrice A.

Description : fonction qui retourne l'élément (i, j) de la matrice A. Pour obtenir toute la colonne j de la matrice A, on utilise la syntaxe $A(:, j)$. De même, pour accéder à l'intégralité de la ligne i de la matrice A, on écrit $A(i, :)$.

<u>Exemple</u> : A=[1 2 3 4 5;3 10 11 12 20;0 1 0 0 2]

```
A(2,4)
⇨  12
A(:,3)
⇨  3.
   11.
   0.
A(2,:)
⇨  3.  10.  11.  12.  20.
```

x=[x1:Dx:x2]

Description : commande permettant de créer un vecteur dont les éléments sont espacés de Dx et dont le premier élément est x_1 et le dernier élément est le plus grand multiple de Dx inférieur ou égal à x_2.

ATTENTION : le vecteur ainsi créé est un vecteur ligne. Pour convertir un vecteur ligne en un vecteur colonne, on le transpose en utilisant l'apostrophe « ' » : x_trans=x'.

<u>Exemple</u> *:* x=[2:0.5:6.3]

```
⇨  2.  2.5  3.  3.5  4.  4.5  5.  5.5  6.
x_trans=x'
⇨  2.
   2.5
   3.
   3.5
   4.
   4.5
   5.
   5.5
   6.
```

zeros(n,m)

Arguments d'entrée : deux entiers n et m correspondant aux dimensions de la matrice à créer.

Argument de sortie : un tableau (matrice) d'éléments nuls.

Description : fonction créant une matrice (tableau) de dimensions $n \times m$ dont tous les éléments sont nuls.

Exemple : zeros(3,4)

⇨ 0. 0. 0. 0.
0. 0. 0. 0.
0. 0. 0. 0.

plot(x,y)

Arguments d'entrée : un vecteur d'abscisses x (tableau de dimension n) et un vecteur d'ordonnées y (tableau de dimension n).

Description : fonction permettant de tracer sur un graphique n points dont les abscisses sont contenues dans le vecteur x et les ordonnées dans le vecteur y.

Exemple :
```
x= [3:0.1:5]
y=sin(x)
plot(x,y)
```

ANNEXE B : BIBLIOTHEQUE NUMPY DE PYTHON

Dans les exemples ci-dessous, la bibliothèque numpy a préalablement été importée à l'aide de la commande : **import numpy as np**

On peut alors utiliser les fonctions de la bibliothèque, dont voici quelques exemples :

np.array(liste)

Argument d'entrée : une liste définissant un tableau à 1 dimension (vecteur) ou 2 dimensions (matrice).

Argument de sortie : un tableau (matrice).

Description : fonction permettant de créer une matrice (de type tableau) à partir d'une liste.

Exemple :
```
np.array([4,3,2])
⇨ [4 3 2]
np.array([[5],[7],[1]])
⇨ [[5]
   [7]
   [1]]
np.array([[3,4,10],[1,8,7]])
⇨ [[3 4 10]
   [1 8 7]]
```

$\boldsymbol{A[i,j]}$.

Arguments d'entrée : un tuple contenant les coordonnées de l'élément dans le tableau A.

Argument de sortie : l'élément $(\mathrm{i}+1, \mathrm{j}+1)$ de la matrice A.

Description : fonction qui retourne l'élément $(i+1, j+1)$ de la matrice A. Pour obtenir toute la colonne j+1 de la matrice A, on utilise la syntaxe $A[:,j]$. De même, pour accéder à l'intégralité de la ligne i+1 de la matrice A, on écrit $A[i,:]$.

ATTENTION : en langage Python, les lignes d'un $\boldsymbol{A}$ de dimension $\boldsymbol{n \times m}$ sont numérotées de 0 à $\boldsymbol{n-1}$ et les colonnes sont numérotées de 0 à $\boldsymbol{m-1}$

Exemple :
```
A=np.array([[3,4,10],[1,8,7]])
A[0,2]
⇨ 10
A[:,2]
⇨ [10 7]
A[1,:]
⇨ [1 8 7]
```

np.zeros((n,m))

Arguments d'entrée : un tuple de deux entiers correspondant aux dimensions de la matrice à créer.
Argument de sortie : un tableau (matrice) d'éléments nuls.
Description : fonction créant une matrice (tableau) de dimensions $n \times m$ dont tous les éléments sont nuls.
Exemple : np.zeros((3,4))

⇨ [[0 0 0 0]
[0 0 0 0]
[0 0 0 0]]

np.linspace(Min,Max,nbElements)

Arguments d'entrée : un tuple de 3 entiers.
Argument de sortie : un tableau (vecteur).
Description : fonction créant un vecteur (tableau) de $nbElements$ nombres espacés régulièrement entre Min et Max. Le 1er élément est égal à Min, le dernier est égal à Max et les éléments sont espacés de $(Max - Min)/(nbElements - 1)$:
Exemple : np.linspace(3,25,5)

⇨ [3 8.5 14 19.5 25]

ANNEXE C : BIBLIOTHEQUE MATPLOTLIB.PYPLOT DE PYTHON

Cette bibliothèque permet de tracer des graphiques. Dans les exemples ci-dessous, la bibliothèque matplotlib.pyplot a préalablement été importée à l'aide de la commande :
import matplotlib.pyplot as plt

On peut alors utiliser les fonctions de la bibliothèque, dont voici quelques exemples :

plt.plot(x,y)

Arguments d'entrée : un vecteur d'abscisses x (tableau de dimension n) et un vecteur d'ordonnées y (tableau de dimension n).
Description : fonction permettant de tracer sur un graphique de n points dont les abscisses sont contenues dans le vecteur x et les ordonnées dans le vecteur y. Cette fonction doit être suivie de la fonction **plt.show()** pour que le graphique soit affiché.
Exemple :

```
x= np.linspace(3,25,5)
y=sin(x)
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

plt.xlabel(nom)

Argument d'entrée : une chaîne de caractères.
Description : fonction permettant d'afficher le contenu de nom en abscisse d'un graphique.

plt.ylabel(nom)

Argument d'entrée : une chaîne de caractères.
Description : fonction permettant d'afficher le contenu de nom en ordonnée d'un graphique.

plt.show()

Description : fonction réalisant l'affichage d'un graphe préalablement créé par la commande **plt.plot(x,y)**. Elle doit être appelée après la fonction plt.plot et après les fonctions plt.xlabel et plt.ylabel.

Exercice n°1

On appelle A la matrice carrée d'ordre $n > 0$ ayant des 2 sur la diagonale et des 1 ailleurs, c'est-à-dire :

$$A = \begin{pmatrix} 2 & 1 & \cdots & \cdots & 1 \\ 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 1 & \cdots & \cdots & 1 & 2 \end{pmatrix}$$

1. Soit J la matrice carrée d'ordre $n > 0$ dont le terme général est égal à 1, c'est à dire :

$$J = \begin{pmatrix} 1 & \cdots & \cdots & \cdots & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & \cdots & \cdots & \cdots & 1 \end{pmatrix}$$

   (a) Exprimer la matrice J^2 en fonction de la matrice J.

(b) En déduire un polynôme annulateur du second degré de la matrice A.

(c) Montrer que la matrice A est inversible et calculer son inverse A^{-1}.

2. (a) Montrer qu'il existe une matrice Q appartenant au groupe orthogonal $O_n(\mathbb{R})$ et une matrice diagonale D telles que :
$$A = QDQ^{-1}.$$

(b) Déterminer les valeurs propres et sous-espaces propres de la matrice A, en déduire la matrice D.

3. Soit $\mathcal{E}$ l'ensemble des matrices U carrées d'ordre $n > 0$ à coefficients réels vérifiant la relation :
$${}^tUA = AU$$

(a) Montrer que $\mathcal{E}$ est un sous-espace vectoriel de l'ensemble des matrices carrées d'ordre $n > 0$ à coefficients réels.

(b) Pour toute matrice U de l'ensemble $\mathcal{E}$, on pose $V = {}^tQUQ$ où la matrice Q est la matrice de la question 2)a). Montrer que :
$${}^tVD = DV$$
où D désigne la matrice diagonale de la question 2).

(c) Soit $\mathcal{F}$ l'ensemble des matrices V carrées d'ordre $n > 0$ à coefficients réels vérifiant la relation :
$${}^tVD = DV$$
On admet que $\mathcal{F}$ est un sous-espace vectoriel de l'ensemble des matrices carrées d'ordre $n > 0$ à coefficients réels, montrer que : $\dim \mathcal{E} = \dim \mathcal{F}$.

(d) En déduire la dimension de l'espace $\mathcal{E}$.

4. On appelle φ l'application de $(\mathbb{R}^n)^2$ dans $\mathbb{R}$ définie par : $\varphi(x, y) = {}^tXAY$ où X et Y désignent les matrices colonnes formées par les coordonnées des vecteurs x et y dans la base canonique de $\mathbb{R}^n$.

(a) Montrer que φ est un produit scalaire sur $\mathbb{R}^n$.

(b) A chaque matrice U de l'espace $\mathcal{E}$, on associe l'application linéaire u de $\mathbb{R}^n$ dans lui-même telle que la matrice de u dans la base canonique soit la matrice U.
Montrer que l'application linéaire u est symétrique pour le produit scalaire φ.

(c) En déduire que pour toute matrice U de l'espace $\mathcal{E}$, il existe une matrice B appartenant à $GL_n(\mathbb{R})$ et une matrice Δ diagonale appartenant à $\mathcal{M}_n(\mathbb{R})$ vérifiant
$$\begin{cases} U = B\Delta B^{-1} \\ {}^tBAB = I_n \end{cases}$$

Exercice n°2

A toute suite $(a_n)_{n\geqslant 1}$ de réels et à toute suite de réels non nuls $(b_n)_{n\geqslant 1}$ on associe la matrice A_n

de $\mathcal{M}_n(\mathbb{R})$ où n est un entier strictement positif.

$$A_n = \begin{pmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ b_1 & a_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & a_{n-1} & b_{n-1} \\ 0 & \cdots & 0 & b_{n-1} & a_n \end{pmatrix}$$

On note $P_n(X)$ son polynôme caractéristique qui est égal à $\det(XI_n - A_n)$.

1. Déterminer une relation de récurrence entre les polynômes $P_{n+1}(X)$, $P_n(X)$ et $P_{n-1}(X)$.

2. (a) Justifier que la matrice A_n est diagonalisable.
   (b) Soit λ une valeur propre réelle de la matrice A_n, calculer le déterminant de la matrice

$$\begin{pmatrix} -b_1 & 0 & 0 & \cdots & 0 \\ \lambda - a_2 & & \ddots & \ddots & \vdots \\ -b_2 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & -b_{n-2} & 0 \\ & \cdots & -b_{n-2} & \lambda - a_{n-1} & -b_{n-1} \end{pmatrix}$$

   extraite de la matrice $\lambda I_n - A_n$ en supprimant sa première colonne et sa dernière ligne.
   (c) En déduire le rang de la matrice $\lambda I_n - A_n$ pour λ valeur propre de la matrice A_n.
   (d) En déduire que le polynôme caractéristique $P_n(X)$ de la matrice A_n admet n racines distinctes.

3. On appelle $\Delta_n(x)$ le déterminant de la matrice $\begin{pmatrix} P'_{n+1}(x) & P'_n(x) \\ P_{n+1}(x) & P_n(x) \end{pmatrix}$ où $P'_{n+1}(x)$ et $P'_n(x)$ désignent respectivement les fonctions polynômes dérivées des fonctions polynômes $P_{n+1}(x)$ et $P_n(x)$.

   (a) Soit $n \geqslant 2$. Montrer que : $\forall x \in \mathbb{R}, \quad \Delta_n(x) = P_n^2(x) + b_n^2 \Delta_{n-1}(x)$.
   (b) Montrer que $\Delta_1(x)$ est strictement positif pour tout x réel. En déduire le signe de $\Delta_n(x)$ pour tout $n \geqslant 2$.

4. Montrer que l'application $x \mapsto P_{n+1}(x)$ s'annule entre deux zéros consécutifs de P_n.

   (On pourra considérer l'application $x \mapsto \dfrac{P_{n+1}(x)}{P_n(x)}$)

Exercice n°3

1. Montrer que l'intégrale $\displaystyle\int_0^{+\infty} \frac{\sin x}{x} dx$ existe. On admet alors que :

$$\int_0^{+\infty} \frac{\sin x}{x} dx = \frac{\pi}{2}.$$

2. (a) Montrer que pour tout α strictement positif et pour tout x réel, l'application $t \mapsto \dfrac{1-\cos \alpha t}{t^2} e^{-itx}$ est prolongeable par continuité en 0.

   (b) Montrer que : $\forall \alpha > 0$, $\forall x \in \mathbb{R}$, l'application $t \mapsto \dfrac{1-\cos \alpha t}{t^2} e^{-itx}$ ainsi prolongée est intégrable sur $\mathbb{R}$.

3. On note, $\forall \alpha > 0$, $\forall x \in \mathbb{R}$:

$$I = \int_{-\infty}^{+\infty} \frac{1-\cos \alpha t}{t^2} e^{-itx} dt.$$

   (a) Montrer que I est réelle.

   (b) Soit $A > 0$ et $B > 0$. On admet l'existence de l'intégrale $\displaystyle\int_A^{+\infty} \frac{\cos Bx}{x^2} dx$.
   Montrer que :

$$\int_A^{+\infty} \frac{\cos Bx}{x^2} dx = \frac{\cos AB}{A} - B \int_{AB}^{+\infty} \frac{\sin t}{t} dt.$$

   (c) En déduire le calcul de l'intégrale $\displaystyle\int_0^{+\infty} \frac{1-\cos Bx}{x^2} dx$ pour $B > 0$ puis pour B quelconque.

   (d) En déduire le calcul de l'intégrale I .

Exercice 4

Des bits d'information, c'est-à-dire des 1 ou 0, sont transmis par l'intermédiaire d'un canal. Ce canal n'est pas complètement fiable. On observe qu'un bit envoyé, un 1 ou un 0, peut être altéré en sortie, c'est-à-dire qu'un 1 (respectivement un 0) en entrée du canal peut devenir un 0 (respectivement un 1) en sortie.

On note b le bit envoyé et b' le bit en sortie ($b \in \{0,1\}$ et $b' \in \{0,1\}$).

$$b \rightarrow \boxed{\text{canal bruité}} \rightarrow b'$$

Après observation, on modélise la transmission d'un bit de façon probabiliste :

- Le bit envoyé définit une variable aléatoire b : On note α la probabilité qu'un 1 soit envoyé, (c'est-à-dire $\alpha = P(b = 1)$) et donc $1 - \alpha$ la probabilité qu'un 0 soit envoyé.
- La perturbation dans le canal est aussi modélisée de façon probabiliste :
  - On désigne par p la probabilité qu'un 1 en entrée ne soit pas altéré pendant la transmission (c'est-à-dire $p = P(b' = 1|b = 1)$) et donc $1 - p$ désigne la probabilité qu'un 1 en entrée devienne un 0 en sortie.
  - On désigne par q la probabilité qu'un 0 en entrée ne soit pas altéré pendant la transmission, et donc $1 - q$ désigne la probabilité qu'un 0 en entrée devienne un 1 en sortie.

1. On a écrit ci-dessus $p = P(b' = 1|b = 1)$. Exprimer de la même manière $1 - p, q$ et $1 - q$ en terme de probabilités conditionnelles.
2. Un bit est envoyé. Quelle est la probabilité de recevoir un 1 en sortie ?
3. On reçoit le bit 1. Quelle est alors la probabilité qu'un 1 ait été envoyé en entrée ?

Soit n un entier supérieur ou égal à 2. On décide d'envoyer n fois le même bit b. On note $b'_1, ..., b'_n$ les n bits obtenus en sortie et on note X la variable aléatoire qui compte le nombre de 1 en sortie. On remarque que les valeurs possiblement prises par X sont $0, 1, ..., n$.

$$b \rightarrow \left\{ \begin{array}{ccc} \boxed{\text{canal bruité}} & \rightarrow & b'_1 \\ \boxed{\text{canal bruité}} & \rightarrow & b'_2 \\ \vdots & \vdots & \vdots \\ \boxed{\text{canal bruité}} & \rightarrow & b'_n \end{array} \right.$$

4. Soit k un entier entre 0 et n. Exprimer $P(X = k)$ en fonction des paramètres p, q, α.
5. En déduire l'espérance de X en fonction des paramètres p, q, α.
6. Soit k un entier entre 0 et n. Exprimer la probabilité que le bit 1 ait été envoyé sachant que le nombre de 1 en sortie vaut k.

Le canal est désormais supposé symétrique, c'est-à-dire que chaque bit, que ce soit un 0 ou un 1, peut-être altéré avec la même probabilité $(1 - p)$. On suppose $\frac{1}{2} < p < 1$.

7. (a) Déterminer en fonction des paramètres p et α, l'ensemble des valeurs k prises par X pour lesquelles il est plus probable (au sens strict) qu'un 1 ait été envoyé plutôt qu'un 0 ?

   (b) Que devient ce résultat lorsque $\alpha = \frac{1}{2}$?

8. On suppose $\alpha = \frac{1}{2}$. On note $f(n)$ la probabilité que l'interprétation de l'observation en sortie soit fausse, c'est-à-dire que le bit en entrée n'est pas celui le plus probable en sortie.

   (a) Exprimer $f(n)$ en fonction des $P(X = k)$, pour des entiers k entre 0 et n.

   (b) Donner une expression de $f(n)$ en fonction de n et p.

   (c) Ecrire une fonction `binome` en langage Python qui prend en entrée un entier naturel N et un entier naturel k compris entre 0 et N et retourne la valeur du coefficient binomial $\binom{N}{k}$.

   (d) On suppose $p = 0.95$. Ecrire un programme en langage Python qui prend en entrée l'entier naturel n et donne une estimation de $f(n)$.

Véhicule intelligent RobuCar

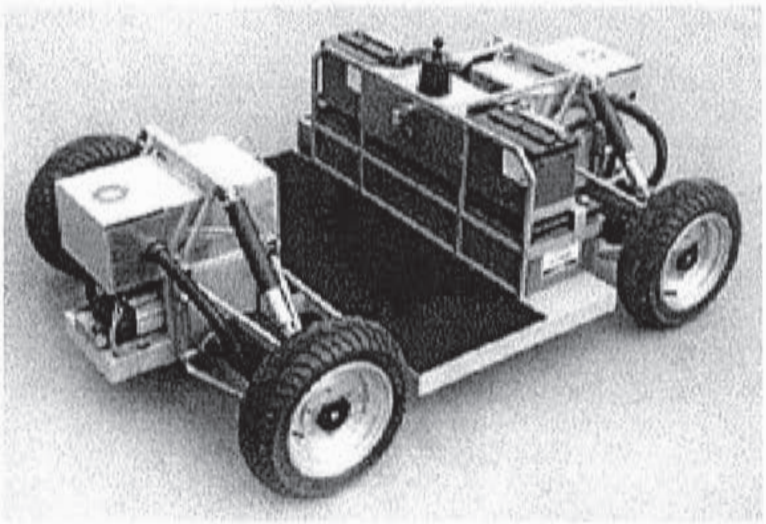
Le candidat est invité à formuler toute hypothèse cohérente qui lui semblerait nécessaire pour pouvoir répondre aux questions posées.

Contexte

L'optimisation des transports de demain passera par l'emploi de véhicules dits intelligents. Des expériences ont déjà été menées pour la réalisation de trains de poids lourds capables de se suivre en toute sécurité et ceci à distance fixe, le lien étant télémétrique au lieu d'être physique.

Cette capacité "d'accrochage télémétrique" entre véhicule maître (leader) et véhicule suiveur (esclave) peut-être transposée à de nombreuses applications : transbordeurs de containers, exploitation minière / terrassement, transports en commun du futur…).

Afin d'étudier les comportements possibles de ces trains de véhicules intelligents dans différentes situations, normales et dégradées, le laboratoire d'Automatique Génie Informatique et Signal (LAGIS UMR8219) situé à Polytech-Lille utilise un démonstrateur composé d'un véhicule maître et deux véhicules suiveurs dont on peut voir l'architecture sur la figure 1.



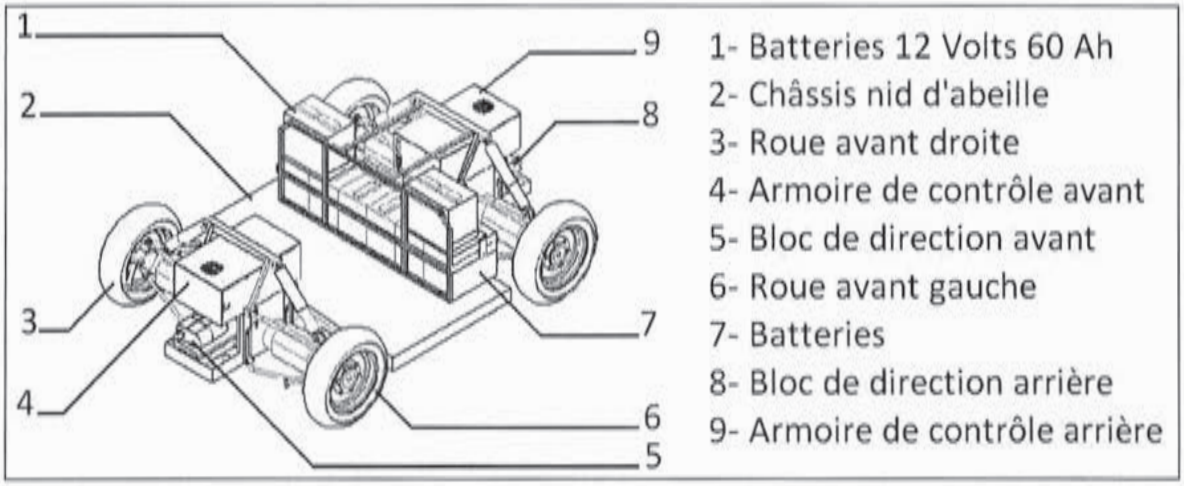


Figure 1 – Ensemble Châssis – Train roulant du véhicule RobuCar

Description de la morphologie du système

Le véhicule suiveur intelligent est un châssis à quatre roues motrices et directrices pilotables séparément (figure 1). Il y a donc deux paramètres de commande pour chaque roue (orientation (direction), vitesse de rotation (motricité))

Le schéma suivant présente un diagramme de définition des blocs pour une roue. Il permet de visualiser les deux chaînes d'énergie d'entraînement de la roue et d'orientation de la roue, ainsi que les éléments de la chaîne d'information tels que l'armoire de commande, les capteurs d'orientation et de vitesse de la roue.

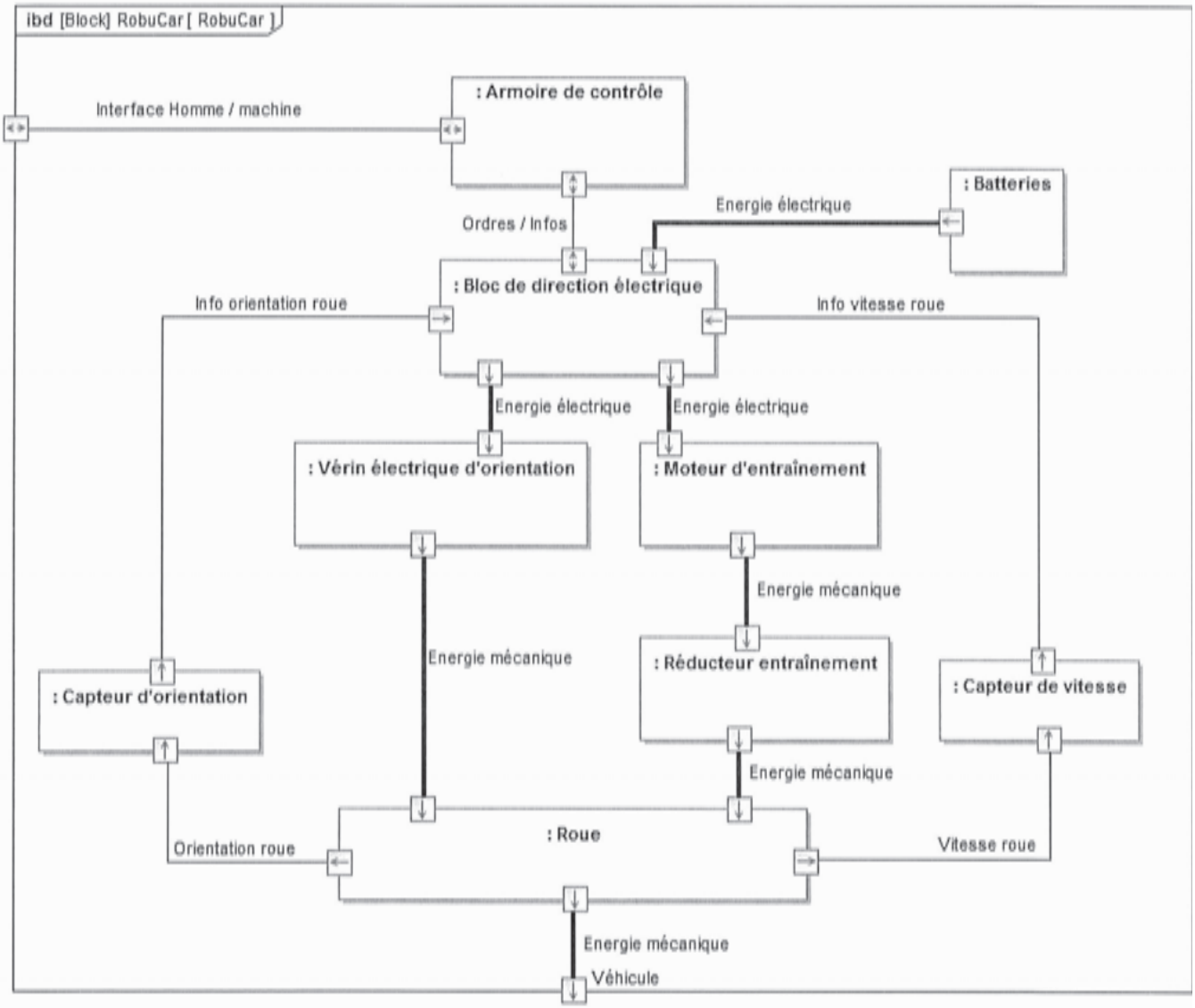


Diagramme de définition des blocs pour le système de commande d'une roue

Objectif

L'objectif de l'étude proposée est d'obtenir un modèle analytique liant le comportement dynamique intrinsèque du véhicule suiveur aux paramètres de la chaîne de commande des différents actionneurs.
Le modèle obtenu doit ainsi permettre d'étudier les stratégies de commande adaptées aux différentes situations.

A cet effet, le sujet est articulé autour de cinq parties I.A, I.B, I.C, II.A et II.B, chacune étant indépendante de la précédente. Le candidat est tout de même invité à les traiter dans l'ordre.

I. Etude des différentes trajectoires possibles

Hypothèses

- L'hypothèse maîtresse de l'étude consiste à considérer que les quatre roues du véhicule sont en contact ponctuel sur un sol parfaitement horizontal, et qu'il y a roulement sans glissement.
- On admet que le contact roue/sol vérifie la loi d'adhérence de Coulomb. Le non-glissement est donc assuré dès lors que :

$$\left\|\vec{T}\right\| < f\left\|\vec{N}\right\|$$

Où $\left\|\vec{T}\right\|$ est le module de l'effort tangentiel sol/roue, $\left\|\vec{N}\right\|$ est le module de l'effort normal sol/roue, et f le coefficient d'adhérence de Coulomb du contact roue/sol considéré. Sur sol sec et avec des pneumatiques en bon état, on admet que $f \approx 1$.

Dans ces conditions, le déplacement du châssis est assimilable à un mouvement plan sur plan. Le respect d'une trajectoire revient à piloter les valeurs des paramètres de mouvement de chaque roue (orientation et vitesse de rotation), les valeurs de ces paramètres devant être cohérentes entre elles pour respecter le roulement sans glissement et sans dérapage de chacune des roues.

L'objectif de l'étude cinématique est d'écrire les relations liant les paramètres de mouvement dans les deux cas de figure les plus simples : la ligne droite et le virage.

Dans le cas de la translation, tous les points du châssis ont la même vitesse par rapport au repère fixe.

Les caractéristiques du groupe moto-propulseur sont résumées dans le tableau ci-dessous :

Moteur	Vitesse de rotation maximale	N_m = 3200 tr/mn
	Couple nominal	2,7 Nm
	Puissance Maximale	900 W
Réducteur	Rapport de réduction	N = 13
Roue	Rayon	R = 0,20 m

Tableau 1. Caractéristiques du groupe moto-réducteur-roue

Un extrait du cahier des charges est fourni ci-dessous :

Entraîner et freiner le véhicule sur sol sec.	Vitesse à atteindre	15 km/h
	Décélération sans glissement des roues	- 6 m.s^{-2}
Orienter le véhicule via un rayon de virage standard de 6,4 m	Temps pour un virage de 45°	Inférieur à 3 s

Q1. A l'aide des données fournies dans le tableau 1, et en tenant compte des hypothèses de l'étude, calculer la vitesse maximale '*Vmax*' en ligne droite du véhicule. Vérifier la performance attendue par le cahier des charges.

A. Etude des phases transitoires accélération / freinage en ligne droite.

Dans ces phases transitoires, l'altitude du centre de gravité du véhicule par rapport au sol influe sur l'équilibre dynamique de celui-ci, et notamment sur la répartition avant/arrière de la charge normale (= verticale) supportée par les roues. La motricité des roues dépend directement de celle-ci. On doit donc calculer la charge normale (= verticale) appliquée sur chaque roue afin d'en déduire la capacité de transmission d'effort tangentiel correspondant. On pourra ainsi optimiser la répartition de l'effort tangentiel de freinage ou d'accélération entre train avant et train arrière.
Le but de cette partie est d'établir un modèle permettant de valider le cahier des charges.
Pour cette étude, le problème étant symétrique, on adopte le modèle simplifié ramené au plan de symétrie conformément au schéma suivant :

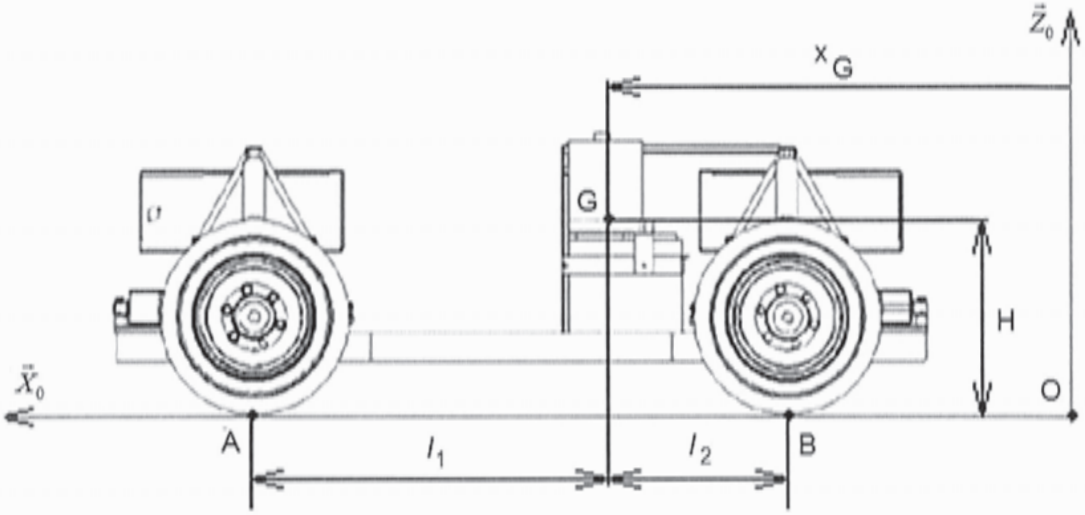


Figure 2 - Modèle plan proposé -

Hypothèses :
On néglige l'influence de la rotation des roues sur le comportement dynamique du véhicule, de sorte que celles-ci peuvent être considérées comme appartenant au solide 'Véhicule'.
Les roues restent en contact avec le sol et les suspensions sont suffisamment rigides pour que le phénomène de plongée soit négligé (Le châssis ne tangue pas vers l'avant).
Le contact sol/roue est assimilable à un contact ponctuel sans glissement, c'est-à-dire capable de transmettre une composante tangentielle d'effort.

On isole l'ensemble du véhicule considéré comme un unique solide.

On applique le principe fondamental de la dynamique au problème plan constitué du Véhicule (solide (S)), en mouvement par rapport au sol (solide (0)) assimilable à un repère galiléen, et on note :

A : Point de contact sol/roue pour le train avant du modèle plan
B : Point de contact sol/roue pour le train arrière du modèle plan
G : Centre de gravité du véhicule

Constantes :
M : Masse du véhicule
g : Valeur du champ de pesanteur

H : Position verticale du centre de gravité par rapport à l'origine du repère

Variable :

$x_G(t)$: Abscisse du centre de gravité par rapport à l'origine du repère

On se place en phase de freinage. Les composantes de la résultante des actions de contact du sol sur les roues sont notées T_A, N_A en A pour les roues avants, T_B et N_B en B pour les roues arrières, où T définit la composante tangentielle au sol et N la composante normale. Le moment de chaque action de contact, au point de contact est négligé (absence de résistance au roulement).

Le torseur des actions de contact sol roues-avant, exprimé au point de contact A, et pour le modèle plan adopté est donc :

$$\left\{ \mathbf{T}(sol \to roue\_avant) \right\}_A = \left\{ \vec{F}_A \middle| \vec{0} \right\}_A = \begin{Bmatrix} T_A & - \\ - & 0 \\ N_A & - \end{Bmatrix}_{A/B_0} \quad \text{avec } B_0 = \left(\vec{X}_0, \vec{Y}_0, \vec{Z}_0\right).$$

Les résultats aux questions 2 à 9 peuvent être exprimés en fonction de M, f, g, $\ddot{x}_G$, H, l_1 et l_2.

Q2. Représenter le véhicule isolé avec l'ensemble des actions mécaniques extérieures agissant sur celui-ci.

Q3. Ecrire l'équation de résultante dynamique en projection sur $\vec{X}_0$.

Q4. Ecrire l'équation de résultante dynamique en projection sur $\vec{Z}_0$.

Q5. Le mouvement étant une translation, donner l'expression du moment dynamique en G du véhicule par rapport au sol.

Q6. Ecrire l'équation du moment dynamique en A en projection sur $\vec{Y}_0$.

Q7. En déduire l'expression des modules des actions verticales sol/roue s'exerçant en B (roues arrières) et s'exerçant en A (roues avants) en fonction de la valeur de la décélération $\ddot{x}_G$ et des autres paramètres.

Q8. Déduire des lois de Coulomb l'effort maximum tangentiel transmissible sol/roue en B et A en fonction de la valeur f du coefficient d'adhérence et des autres paramètres.

Q9. En déduire la valeur de la décélération $\ddot{x}_G$ maximale envisageable en fonction de la valeur f du coefficient de frottement et des autres paramètres. Vérifier la performance attendue par le cahier des charges.

Application numérique :

M = 310 kg

$l_1 = 0{,}75\,m$

$l_2 = 0{,}55\,m$

$H = 0{,}4\,m$

$f = 1$

Dans la pratique, le coefficient de frottement n'est pas constant mais varie de façon complexe en fonction de plusieurs paramètres. On adopte un modèle simplifié basé sur la valeur de la vitesse du véhicule régit par la loi :

$$f = 0,8 + 0,2.e^{-\frac{V}{VREF}} \text{ avec } VREF = 5 \text{ m.s}^{-1}$$

Q10. Les annexes 1 et 2 proposent une ébauche des programmes en langage Python (annexe 1) et Scilab (annexe 2). L'objectif du programme est de calculer la durée et la distance de la phase de freinage. Compléter sur votre copie les zones manquantes d'un des deux programmes fournis. Ceci doit conduire au bon fonctionnement sans l'introduction de nouvelle variable.

Q10.a Compléter la condition de la boucle while : zone A
Q10.b Compléter la zone d'intégration en temps: zone B
Q10.c Compléter la zone d'affectation des résultats : zone C

B. Etude du suivi de trajectoire / Modélisation du virage

On considère que les virages sont effectués en pilotant de façon conjuguée les orientations des trains avant et arrière conformément au modèle de la figure 3. De cette façon, à la valeur près des paramètres a et b, le pilotage du train arrière (roues 3 et 4) est une image du pilotage du train avant (roues 1 et 2).

L'objectif de cette partie est de définir les lois de pilotage d'entraînement des roues afin de suivre une trajectoire définie. On se limite à un mouvement de rotation.

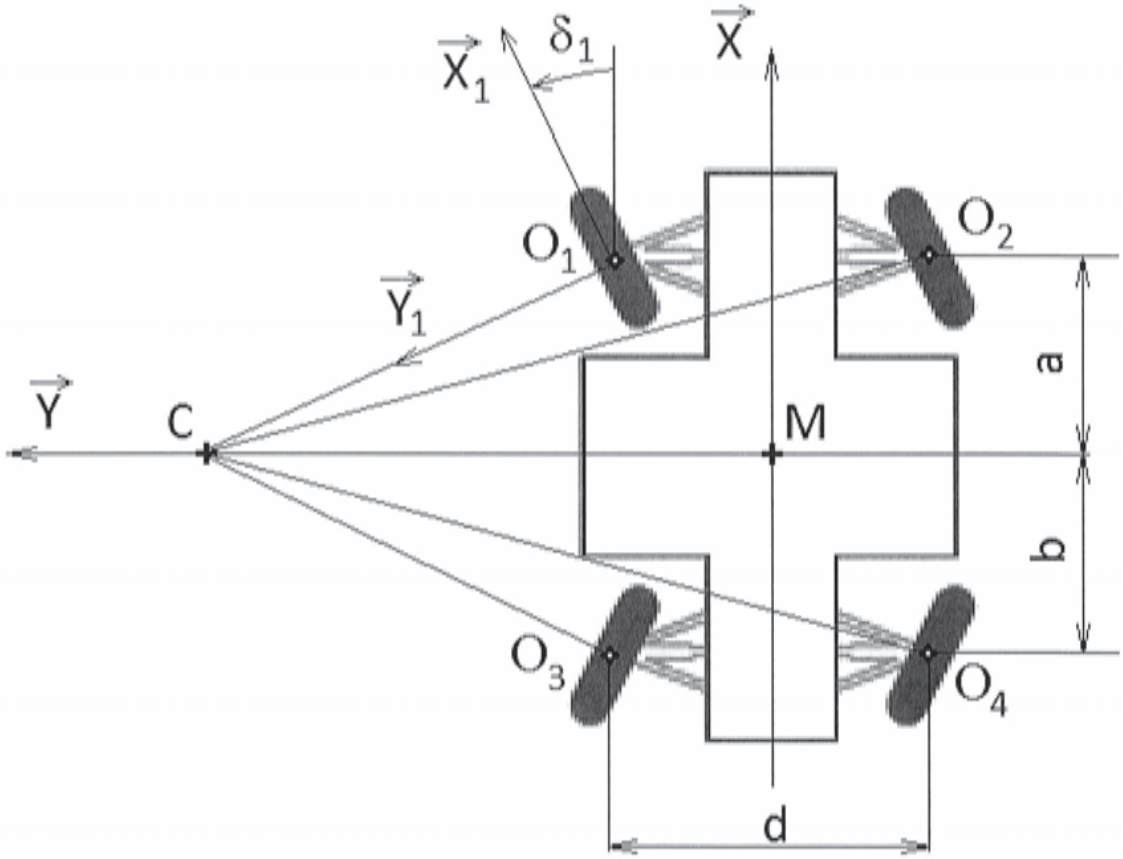


Figure 3. - Véhicule en virage vue du dessus -

Le pilotage conjugué des directions avant et arrière conduit à l'identification d'un point M de l'axe longitudinal du châssis défini par rapport à C, centre de la trajectoire circulaire à l'instant t par : $\overrightarrow{CM} \perp \vec{X}$ avec $\overrightarrow{CM} = -\rho\,\vec{Y}$.

On définit les bases suivantes :

$\left(\vec{X}_0, \vec{Y}_0, \vec{Z}_0\right)$ liée au sol (solide 0),

$\left(\vec{X}, \vec{Y}, \vec{Z}_0\right)$ liée au châssis du véhicule (solide S),

$\left(\vec{X}_i, \vec{Y}_i, \vec{Z}_0\right)$ liée à l'axe de la roue *i*.

Et le point O_i : point du plan médian de la roue i appartenant à l'axe de rotation (cf figures 3 et 4).

Le mouvement du véhicule est un mouvement de rotation autour du point C supposé fixe.

On note la vitesse angulaire d'orientation du châssis par rapport au repère fixe lié au sol, due à la trajectoire circulaire autour de C par : $\vec{\Omega}(S/0) = \dot{\psi}\ \vec{Z}_0$.

On note V la vitesse du point M : $\vec{V}(M \in S/0) = V.\vec{X} = \rho\dot{\psi}.\vec{X}$ avec V constant

Pour chaque roue, on peut considérer localement la figure suivante :

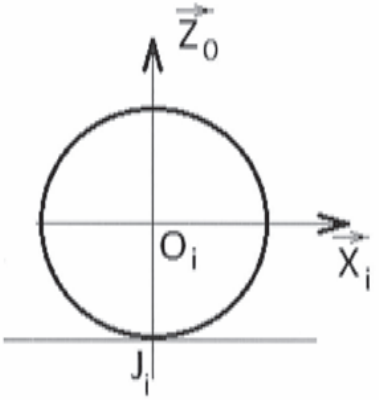


Figure 4. - Modélisation du contact roue/sol dans le plan médian de la roue -

Soit :

$\vec{\Omega}(Roue_i / axe\ roue\ i) = \dot{\theta}_i \vec{Y}_i$: vitesse de rotation de la roue i par rapport à l'axe de la roue i.

R : la valeur du rayon des roues.

Q11. Donner l'expression vectorielle dans la base $\left(\vec{X}, \vec{Y}, \vec{Z}_0\right)$ de $\vec{V}(O_1 \in S/0)$ et $\vec{V}(O_2 \in S/0)$ en fonction de ρ, a, d et $\dot{\psi}$.

Q12. Donner l'expression vectorielle dans la base $\left(\vec{X}_i, \vec{Y}_i, \vec{Z}_0\right)$ de $\vec{V}(O_1 \in Roue_1/0)$ et $\vec{V}(O_2 \in Roue_2/0)$ en fonction de R et $\dot{\theta}_i$ en faisant l'hypothèse de roulement sans glissement en J_i.

Q13. En remarquant que $\vec{V}(O_1 \in Roue_1/S) = \vec{0}$, établir deux relations scalaires après projection dans la base $\left(\vec{X}, \vec{Y}, \vec{Z}_0\right)$.

Q14. Faire de même pour la roue 2.

Q15. En déduire les expressions de $\tan(\delta_1)$ et de $\tan(\delta_2)$ en fonction de ρ, a et d. Effectuer l'application numérique. a = 0,65 m ; d = 1,2 m ; ρ = 5 m

Q16. Etant donné les valeurs de δ_1 et δ_2, on adopte la simplification suivante : $\cos\delta_i \approx 1$ et $\sin\delta_i \approx 0$. En déduire les expressions simplifiées de $\dot{\theta}_1$ et de $\dot{\theta}_2$.

C. Etude des vitesses de rotation des roues 1 et 2 au cours d'un virage

L'objectif de cette partie est d'établir les lois de commande et de valider le cahier des charges du point de vue du temps de virage pour obtenir un changement de direction de 45°. On souhaite conserver V (module de la vitesse du point M centre du châssis) constant au cours de la phase de virage.
Le scénario retenu pour piloter un virage imposé par le suivi de trajectoire est décomposé en cinq phases :

- $t < t_0$: translation rectiligne
- $t_0 < t < t_1$: mouvement transitoire
- $t_1 < t < t_2$: mouvement de rotation autour de C
- $t_2 < t < t_3$: mouvement transitoire
- $t_3 < t$: translation rectiligne

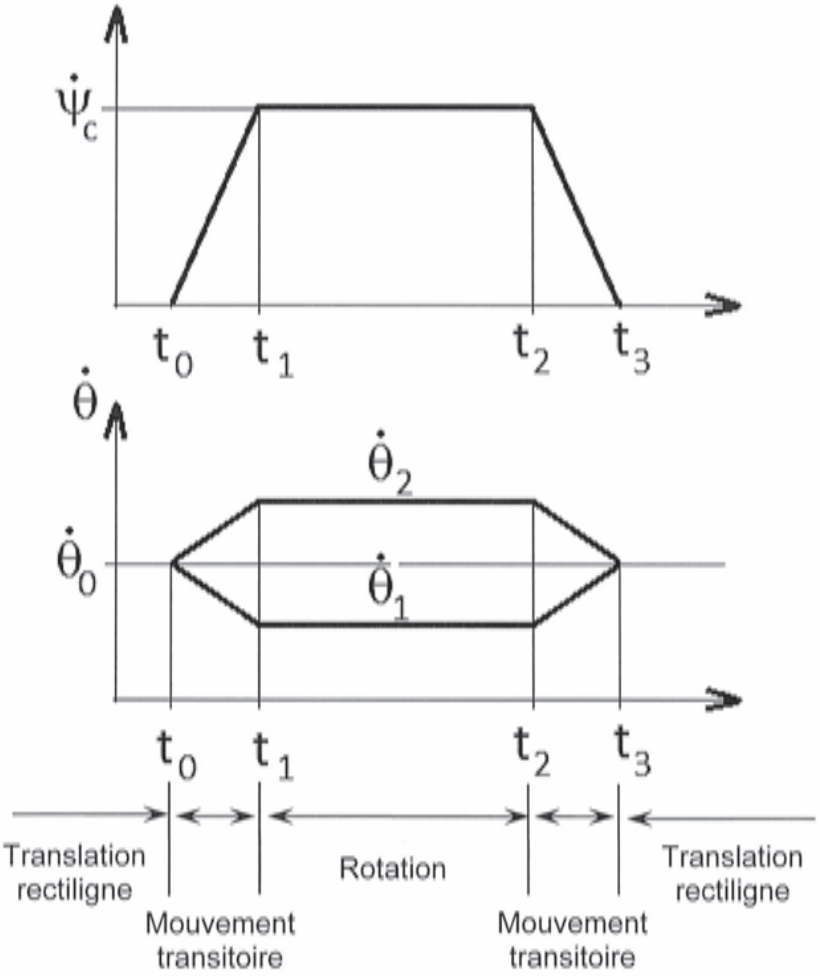


Figure 5. - Scénario retenu pour effectuer un virage -

On définit $\vec{V}(M \in S/0) = V.\vec{X}$ avec le vecteur $\vec{X}$ lié au véhicule donc mobile par rapport au sol.

$\dot{\psi}_C$ représente la valeur de consigne et est directement calculée par $\dot{\psi}_C = \dfrac{V}{\rho}$, avec ρ le rayon de virage imposé par le parcours à l'instant t et V la vitesse du point M au même instant t.

Ce scénario est nécessaire car on ne peut pas passer brutalement de la ligne droite à un virage de rayon donné. En effet, cela imposerait un changement instantané des vitesses de rotation des roues, ce qui est physiquement impossible.

On peut aisément montrer que : $\dot{\psi} = \dfrac{(\dot{\theta}_2 - \dot{\theta}_1)}{d} R$ et $\rho\dot{\psi} = V = \dfrac{(\dot{\theta}_2 + \dot{\theta}_1)}{2} R$

On donne : $|\ddot{\psi}_{Max}| = \ddot{\psi}_0 = 1 \quad rd.s^{-2}$, valeur constante pendant les deux phases transitoires.

On admet donc que $(t_3 - t_2) = (t_1 - t_0) = \dfrac{\dot{\psi}_C}{\ddot{\psi}_0}$

Q17. Pour chaque phase $[t_i ; t_{i+1}]$, on se propose de donner l'expression permettant de calculer la valeur instantanée de $\rho(t)$, $\dot{\psi}(t)$, $\psi(t)$ sous forme d'un tableau. Déterminer les expressions de A, B, C et D qui apparaissent dans le tableau suivant :

	$t < t_0$	$[t_0 ; t_1]$	$[t_1 ; t_2]$	$[t_2 ; t_3]$	$t > t_3$
$\rho(t)$	∞	$\dfrac{V}{\dot{\psi}(t)} = \dfrac{V}{\ddot{\psi}_0 (t - t_0)}$	$\rho = cste = \dfrac{V}{\dot{\psi}_C}$	$\dfrac{V}{\dot{\psi}(t)} = \dfrac{V}{\dot{\psi}_C - \ddot{\psi}_0 (t - t_2)}$	∞
$\ddot{\psi}(t)$	0	$\ddot{\psi}_0 = 1 \; rd/s^2$	0	$-\ddot{\psi}_0 = -1 \; rd/s^2$	0
$\dot{\psi}(t)$	0	$\dot{\psi}(t) = \ddot{\psi}_0 (t - t_0)$	A	C	0
$\psi(t)$	0	$\psi(t) = \ddot{\psi}_0 \dfrac{(t - t_0)^2}{2}$	B	D	0

Q18. En déduire l'expression littérale du changement d'orientation total effectué ψ_{TOT} en fonction de $\ddot{\psi}_0$, $\dot{\psi}_C$, $(t_2 - t_1)$.

Q19. Calculer $(t_2 - t_1)$ pour un virage à gauche de rayon $\rho = 6{,}4 \; m$ effectué à $V = 10 \, km/h$ provoquant un changement d'orientation de 45°.

Q20. En déduire le temps total $(t_3 - t_0)$ nécessaire pour effectuer ce changement de direction.

Q21. Dans ces conditions et pour $t_0 = 0$, calculer t_1, t_2, t_3, $\psi(t_1)$, $\psi(t_2)$ et vérifier que la valeur de la durée du virage de 45° valide bien le cahier des charges.

II. Modélisation et Commande du véhicule

Modélisation du comportement de l'ensemble moto-réducteur-roue

La trajectoire et la vitesse du véhicule autonome sont assurées par la commande de 4 roues indépendantes pilotées chacune par un moteur électrique. Le schéma de principe de chaque système de commande et le schéma-bloc représentés par les figures 6 (a) et (b) sont constitués d'un moteur électrique à courant continu, d'un réducteur de vitesse et de la roue. La partie électrique est pilotée par une source de tension $u_a(t)$ alimentant le stator représenté par la mise en série d'une résistance électrique (R_e) et d'une inductance (L). Le courant induit $i(t)$ fournit un couple moteur $c_m(t)$ ($c_m(t) = K_{em} i(t)$) à l'arbre du moteur soumis à des frottements visqueux f_m et ayant une inertie J_m. Le couple moteur est transmis avec un rapport 1/N à l'arbre de la roue de raideur K et soumis à des frottements visqueux f_r et ayant une inertie J_r. La roue est soumise à un couple résistant $c_p(t)$ du contact pneu-chaussée dû à la force longitudinale F_p. Le tableau 2 définit la nomenclature des variables utilisées.

L'objectif est d'obtenir les équations du comportement dynamique dans la direction longitudinale pour la synthèse du système asservi de la vitesse angulaire. La figure 7 représente le schéma de principe de la commande du véhicule.

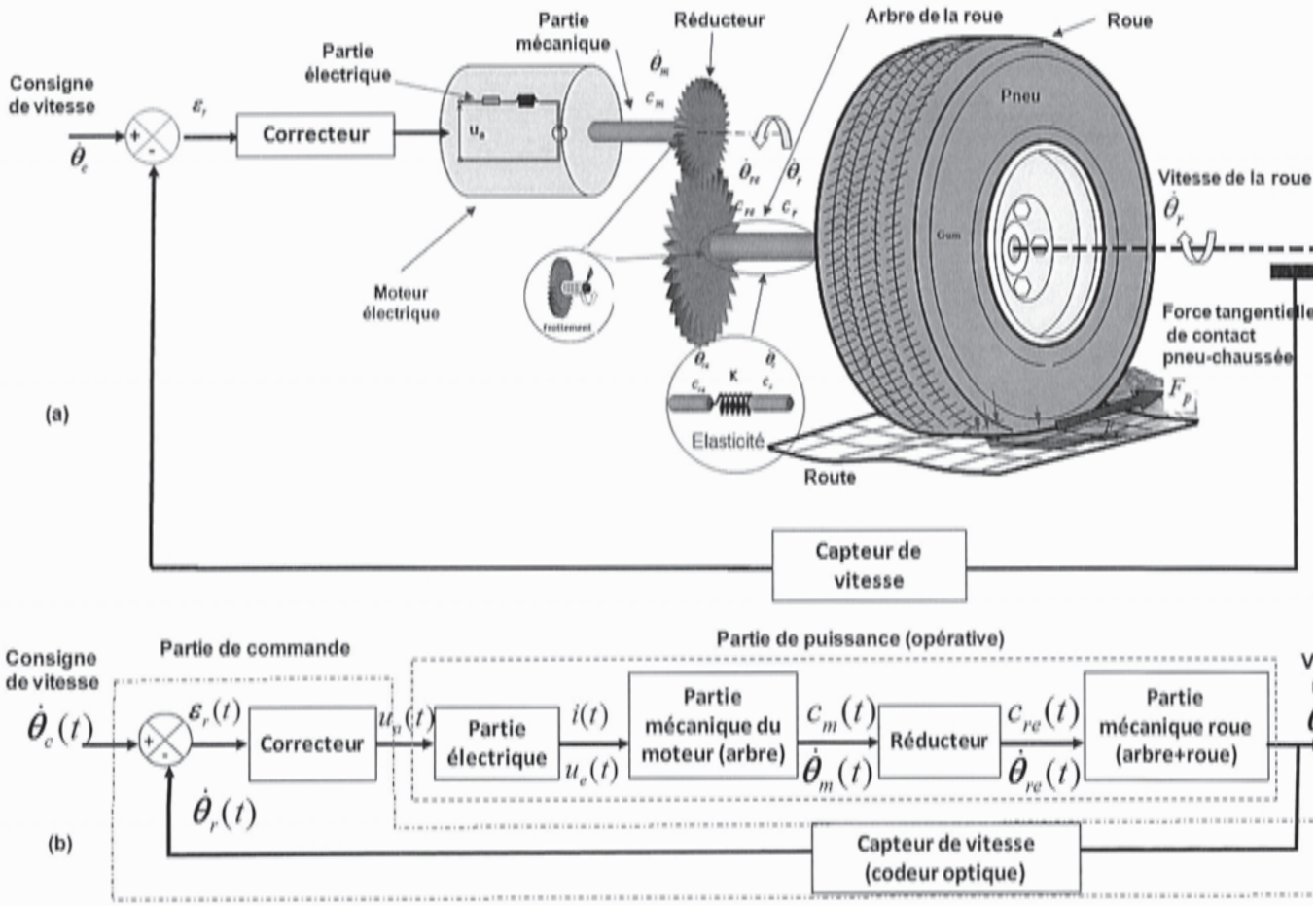


Figure 6 : - Schéma de principe (a) et schéma-bloc (b) du système moto-réducteur-roue

Hypothèses de modélisation

La propulsion est assurée par les couples fournis par les 4 moteurs de roues. La direction est contrôlée entre-autre, à partir d'une différenciation de vitesse sur les trains arrière $\dot{\theta}_3$ et $\dot{\theta}_4$, et avant $\dot{\theta}_1$ et $\dot{\theta}_2$. Les vitesses de rotation des roues $\dot{\theta}_i$ $(i = 1-4)$ sont mesurées par un codeur incrémental fixé sur l'arbre du moteur. Les courants des moteurs $i_i(t)$ sont également mesurés. La variable de commande de chaque moteur est la tension d'alimentation notée $u_a(t)$ de valeur maximale 48 V.

Paramètre	Désignation	Valeur
	Partie électrique du moteur et contrôleur	
$\dot{\theta}_c(t), \dot{\theta}_r(t)$	Vitesse angulaire de consigne introduite par l'opérateur et réelle de la roue (fournie par le capteur)	Variable (rad.s^{-1})
$\varepsilon_r(t) = \dot{\theta}_c(t) - \dot{\theta}_r(t)$	Ecart de réglage (différence entre vitesse réelle et de consigne)	Variable (rad.s^{-1})
$u_a(t)$	Tension d'alimentation contrôlée	48 (V) valeur maximale
L	Inductance	0.075 (H)
R_e	Résistance électrique	1.6 (Ω)
K_{em}	Constante f.e.m.	0.122 (V/rad/s)
i(t)	Courant électrique	Variable (A)
$u_e(t)$	Force électromotrice du bobinage	Variable (V)
	Partie mécanique du moteur	
$c_m(t)$	Couple en sortie du moteur électrique	Variable (N.m)
J_m	Moment d'inertie de l'arbre du moteur	0.0095 (kg.m^2)
f_m	Coefficient de frottement visqueux arbre moteur / carter moteur	0.0043 (N.m.s/rad)
$\dot{\theta}_m(t)$	Vitesse angulaire de l'arbre moteur	Variable (rad.s^{-1})
$\theta_m(t)$	Position angulaire de l'arbre moteur	Variable (rad)
	Partie roue et réducteur	
N	Rapport de réduction	13
K	Raideur de l'arbre sortie réducteur	10000 N.m/rad
J_r	Moment d'inertie roue+axe	8 (kg.m^2)
f_r	Coefficient de frottement visqueux axe de roue / palier	17 (N.m.s/rad)
$c_{re}(t)$	Couple en sortie du réducteur	Variable (N.m)
$\dot{\theta}_{re}(t)$	Vitesse angulaire en sortie du réducteur	Variable (rad.s^{-1})
$c_p(t)$	Couple du contact pneu-chaussée du à la force longitudinale	0.5 (N.m) considéré constant par hypothèse simplificatrice
$\dot{\theta}_r(t)$	Vitesse angulaire de la roue	Variable (rad.s^{-1})
$\theta_r(t)$	Position angulaire de la roue	Variable (rad)
G	Centre de gravité	
ψ (t)	Angle de lacet	Rad
d	Distance entre roues (fig.3)	1.2 m
a=b	Distance entre le point M et les trains avant et arrière (fig.3)	0.65 m

Tableau 2 : Nomenclature des variables utilisées

A. Modélisation des différents éléments

• Modélisation de la partie électrique du moteur

On suppose que dans les différents régimes de fonctionnement considérés, le comportement reste linéaire. Le bobinage du moteur est équivalent à un circuit électrique de résistance R_e, d'inductance L en série et d'une force contre-électromotrice $u_e(t)$, le tout alimenté par la tension $u_a(t)$. La partie électrique de l'induit est traversée par un courant $i(t)$. Les équations de couplage couple moteur-intensité et force contre-électromotrice/vitesse angulaire sont décrites par les relations: $c_m(t) = K_{em}.i(t)$ et $u_e(t) = K_{em}.\dot{\theta}_m(t)$ respectivement. Dans ce qui suit, les variables temporelles seront notées en minuscules, et les variables dans le domaine de Laplace seront en majuscules, par exemple :

$$L\left\{\dot{\theta}(t)\right\} = \Omega(p),\ \ L\left\{u_e(t)\right\} = U_e(p),\ \ L\left\{u_a(t)\right\} = U_a(p),\ L\left\{c_m(t)\right\} = C_m(p),$$

p étant l'opérateur de Laplace

L'équation électromécanique du moteur en régime dynamique dans le domaine temporel liant $u_a(t)$, $u_e(t)$ et $i(t)$ s'écrit :

$$u_a(t) = R_e i(t) + L\frac{di}{dt} + u_e(t)$$

• Modélisation de la partie mécanique

On isole l'arbre moteur avec le réducteur (réducteur dont l'inertie est négligée). Les actions mécaniques extérieures qui agissent sur cet ensemble sont : le couple moteur $c_m(t)$, le couple de l'axe de la roue sur l'arbre de sortie du réducteur $-c_{re}(t)$ et les frottements visqueux du palier sur l'arbre moteur.

Q22. Ecrire l'équation différentielle issue du théorème de l'énergie cinétique (ou théorème de l'énergie-puissance) qui régit l'évolution de la variable $\dot{\theta}_m(t)$ en fonction de $c_m(t)$, $c_{re}(t)$, $\dot{\theta}_m(t)$, $\ddot{\theta}_m(t)$ et de J_m, f_m et N.

On isole la roue et son axe. Les actions mécaniques extérieures qui agissent sur cet ensemble sont : le couple de la chaussée sur la roue $c_p(t)$, le couple de l'axe de sortie du réducteur sur l'axe de la roue $c_{re}(t)$ et les frottements visqueux du palier sur la roue.

Q23. Ecrire l'équation différentielle issue du théorème de l'énergie cinétique qui régit l'évolution de la variable $\dot{\theta}_r(t)$ en fonction de $c_p(t)$, $c_{re}(t)$, $\dot{\theta}_r(t)$, $\ddot{\theta}_r(t)$ et de J_r et f_r.

On isole l'arbre de sortie du réducteur effectuant le lien avec l'axe de la roue. Cet élément déformable est supposé de comportement linéaire et de loi de déformation $C = K.\Delta\theta$, $\Delta\theta$ étant l'écart angulaire entre les sections externes. Cet arbre transmet le couple $c_{re}(t)$.

Q24. En déduire une expression de $c_{re}(t)$ en fonction de K, $\theta_{re}(t)$ et $\theta_r(t)$.

Q25. En combinant les 3 équations précédentes, identifier les paramètres α, β et γ dans les deux équations suivantes :

$$J_m\ddot{\theta}_m(t) = C_m(t) - \alpha\dot{\theta}_m(t) - \beta\left(\frac{\theta_m(t)}{N} - \theta_r(t)\right)\frac{1}{N}$$

$$J_r\ddot{\theta}_r(t) = C_p(t) - \gamma\dot{\theta}_r(t) + \beta\left(\frac{\theta_m(t)}{N} - \theta_r(t)\right)$$

Simulation du système de commande de la vitesse angulaire

Q26. Le schéma de simulation du système de commande de la vitesse est fourni en figure 7. En se basant sur les équations électriques et dynamiques de la question précédente, donner l'expression des 4 fonctions de transfert : $H_1(p)$, $H_2(p)$, $H_3(p)$ et $H_4(p)$ en fonction entre autres, de α et γ.

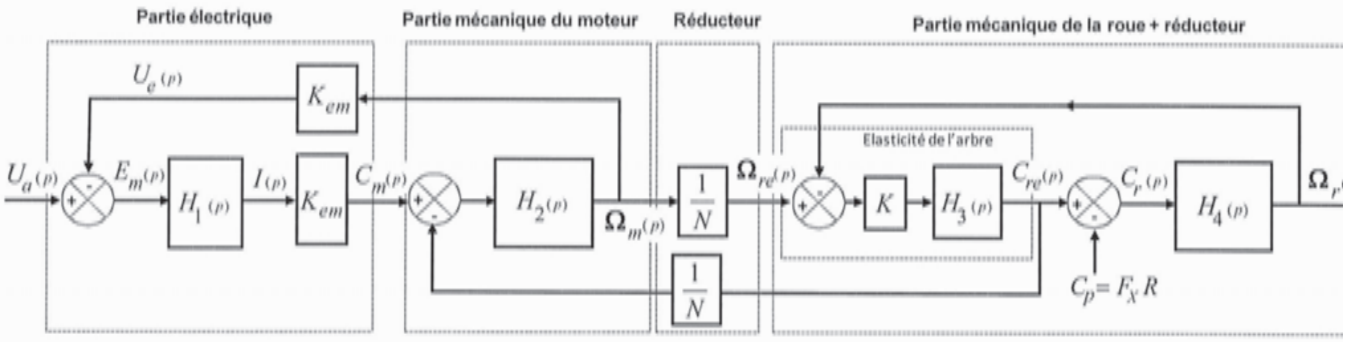


Figure 7. - Schéma bloc de simulation du système moto-réducteur-roue pour la commande de la vitesse -

La partie électrique peut être facilement modélisée car les paramètres fournis par le constructeur sont bien identifiés. Afin de déterminer la fonction de transfert de l'ensemble du système mécanique (moto réducteur et arbre de la roue), nous avons isolé la partie mécanique du système et réalisé un essai indiciel en introduisant un couple moteur $c_m(t)$ d'un échelon de 4N.m à l'entrée de l'arbre du moteur, et nous avons enregistré en sortie de la roue la variation de la vitesse angulaire comme indiqué par le schéma de la figure 8. La réponse à cet échelon de couple est donnée figure 9.

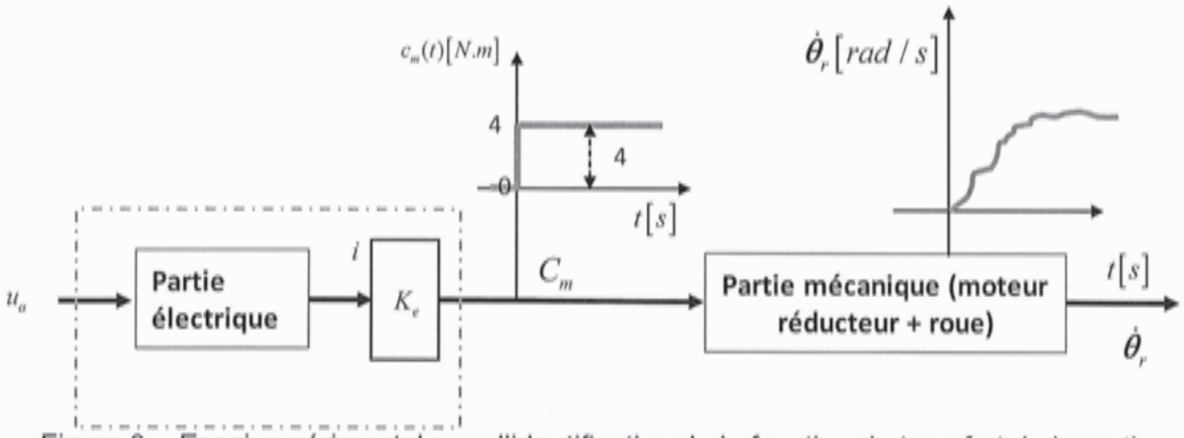


Figure 8. - Essai expérimental pour l'identification de la fonction de transfert de la partie mécanique -

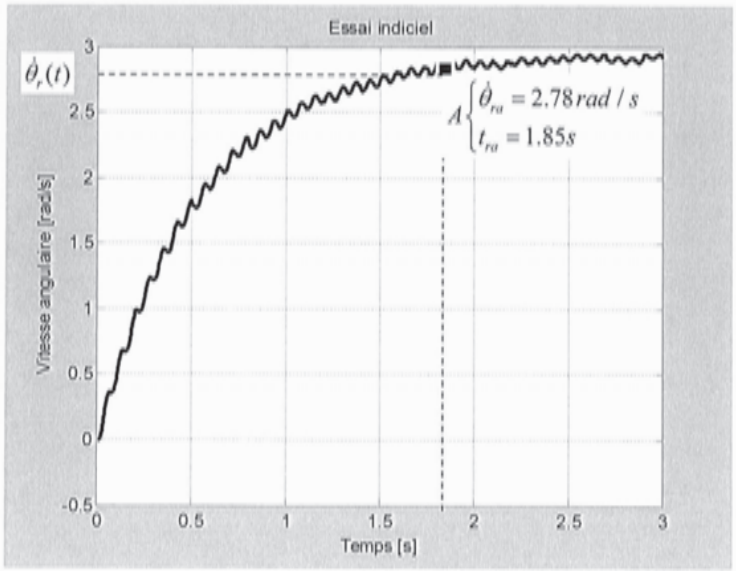


Figure 9. - Réponse indicielle de la vitesse angulaire de la roue suite à un couple moteur d'un échelon de couple 4 N.m

On assimile la fonction de transfert à un élément du premier ordre et on relève la vitesse en régime permanent : $\Delta\dot{\theta}(\infty) = 2.93\ rad.s^{-1}$, et ceci pour un échelon d'entrée d'amplitude $c_m = 4N.m$.

Q27.1 Identifier les valeurs numériques (en précisant les unités) des paramètres caractéristiques de la fonction de transfert du 1er ordre.

On souhaite écrire une fonction qui permet de valider l'identification précédente. Les données mesurées sont stockées dans deux listes $t = [t_1 \quad t_2 \quad .. \quad .. \quad t_n]$ et $y = [y_1 \quad y_1 \quad .. \quad .. \quad y_n]$. La première contient les différents instants de mesure et la seconde les valeurs mesurées correspondant à la vitesse angulaire.

Q27.2 Ecrire une fonction utilisant les langages Python ou Scilab qui prend pour argument les deux listes *t* et *y*, ainsi que les grandeurs du 1er ordre identifiées manuellement. Cette fonction doit renvoyer la valeur absolue maximale de l'écart entre les valeurs mesurées et celle du modèle. On rappelle que l'expression de la réponse indicielle d'un système du premier ordre s'écrit en fonction de $1 - e^{-\frac{t}{T}}$ avec T la constante de temps du premier ordre.

B. Validation des performances du modèle

Les fonctions de transfert étant identifiées, le schéma fonctionnel du système moto-réducteur-roue peut alors être représenté par le schéma bloc de la figure 10a. Le schéma bloc de la figure 10b montre l'influence du couple perturbateur $C_p(p)$ et de la tension d'alimentation $U_a(p)$ sur la vitesse angulaire de la roue. Le couple de contact pneu-chaussée $C_p(p)$ est alors considéré comme une perturbation. En vertu de la propriété de superposition, $\mathbf{\Omega}_r(p)$ subit l'influence de la tension $U_a(p)$ et de la perturbation $C_p(p)$:

$$\mathbf{\Omega}_r(p) = W_{Ua}(p).U_a(p) - W_{Cp}(p).C_p(p)$$

On notera dans la suite $(K_{em})^2 K_m K_e = K_s$

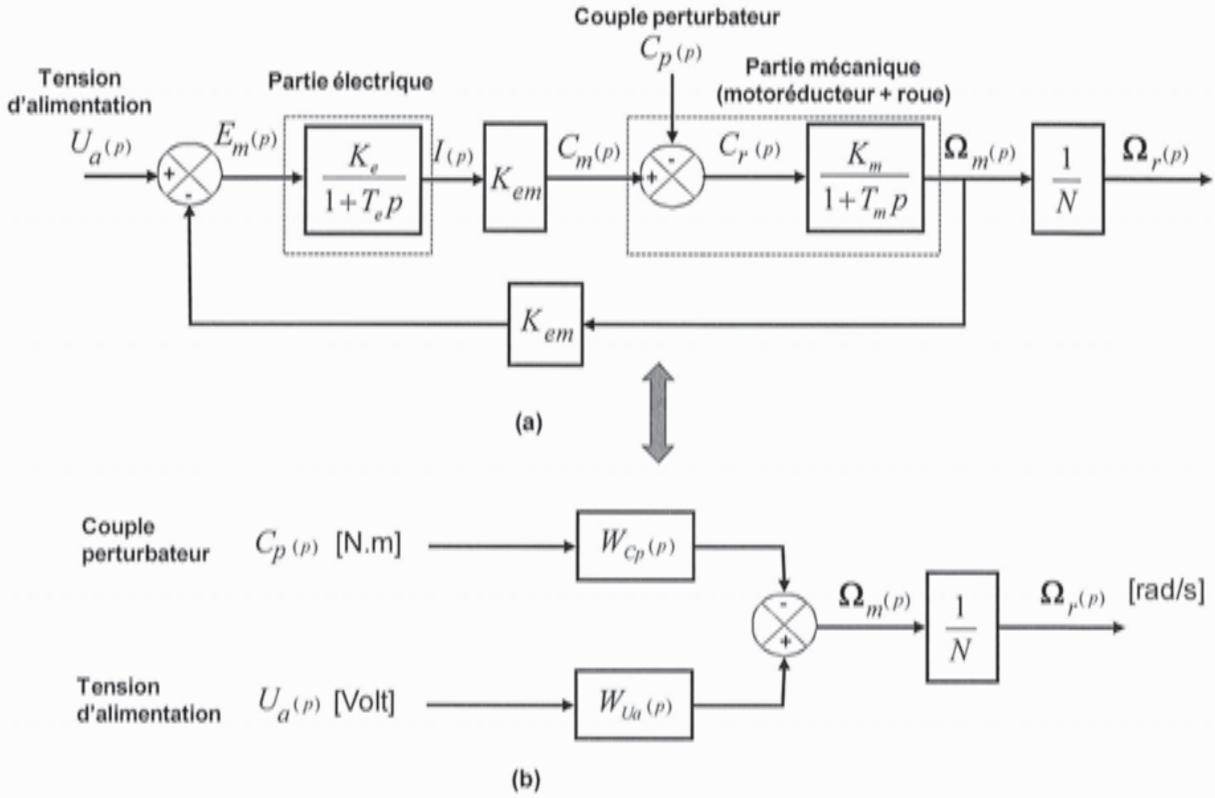


Figure 10 : - Schéma fonctionnel détaillé (a) et compact (a) du système moto-réducteur-roue -

On prendra K_m = 9,56 rad/s.N.m et T_m = 0,6 s.

Q28.1 Expliciter les paramètres caractéristiques T_e et K_e en utilisant les résultats de question 26 (on pourra poursuivre cette partie sans ce résultat).

Q28.2 Donner les expressions des fonctions de transfert $W_{Ua}(p)$ et $W_{Cp}(p)$ (à mettre sous forme canonique). Notez $(K_{em})^2 K_m K_e = K_s$.

Synthèse de correcteurs pour le suivi de trajectoire d'un véhicule intelligent

En raison de la présence de perturbation, la vitesse angulaire de la roue doit être contrôlée afin de maintenir une inter-distance entre les véhicules, ce qui impose un écart statique nul. Afin de choisir le bon correcteur, nous allons analyser les performances d'un ensemble de correcteurs de fonction de transfert $W_R(p)$ d'action P (proportionnelle) puis PI (proportionnelle intégrale) pour l'asservissement de la vitesse angulaire de chaque roue. Le schéma bloc du système asservi peut alors être représenté par la figure 11 suivante :

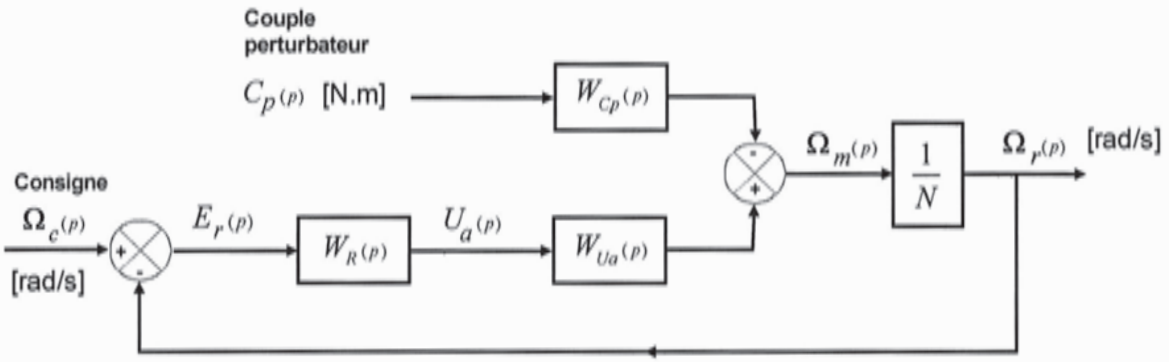


Figure 11 : - Schéma fonctionnel du système asservi de la vitesse de chaque roue -

Lors d'un virage, chaque roue possède une fréquence de rotation différente. Pour suivre une trajectoire (rectiligne ou circulaire), on doit connaître la vitesse de rotation à imposer à chaque roue. On fait l'hypothèse que les virages sont effectués à vitesse constante V=10 km/h et ρ=6,4m (voir partie I).

Q29. Trouver les valeurs numériques des vitesses désirées $\dot{\theta}_1$ et $\dot{\theta}_2$ à partir des expressions utilisées dans la partie I.C sachant que ce seront alors les consignes d'asservissement pour chaque moteur 1 et 2 respectivement (on pourra poursuivre cette partie sans ce résultat).

- **Asservissement de la vitesse de rotation du moteur avec un régulateur proportionnel.**

On se propose d'abord de réguler la vitesse de rotation de la roue par une boucle de commande. On utilise d'abord un correcteur à action proportionnelle $W_R(p) = K_p$. Pour cette partie et les suivantes, on suppose que $C_p(p) = 0$.

Q30.1 Lors d'un changement de consigne ou d'une perturbation et afin de maintenir une inter-distance entre le véhicule suiveur et leader, déterminer la valeur du gain K_p pour que l'écart statique $E_r(\infty)$ en position suite à une variation de consigne soit inférieure à 10% de Ω_0 (l'amplitude de l'échelon de consigne : $\Omega_c(p) = \frac{\Omega_0}{p}$).

Q30.2 Analyser la stabilité du système en fonction du gain K_p. Quelle est la marge de gain ?

- **Synthèse par correcteur proportionnel intégral.**

Pour assurer une meilleure précision, on place un PI régulateur. $W_R(p) = K_R + \frac{K_I}{p}$ où K_R caractérise l'action proportionnelle et K_I l'action intégrale.

Q31.1 Quelle est l'erreur statique pour une entrée échelon ?

Q31.2 Dans un plan avec le paramètre K_I en abscisse et K_R en ordonnée, représenter le domaine de stabilité de la fonction de transfert en boucle fermée.

Remarque : On indique que le critère algébrique de Routh Hurwitz appliqué à un système de fonction de transfert en boucle fermée $W_f(p) = \frac{N_f}{D_f}$ *ayant un dénominateur* $D_f = a_3p^3 + a_2p^2 + a_1p + a_0$

est stable si et seulement si : a_3, a_2, a_1 et $a_0 \geq 0$ et $a_1 a_2 > a_0 a_3$ *L'égalité* $a_1 a_2 = a_0 a_3$ fournit la limite de stabilité, à la condition que tous les coefficients soient de même signe et non nuls.

- **Synthèse des 2 correcteurs**.

Q32 Dans le cas de l'asservissement en vitesse, quelles performances en termes de précision statique, temps de réponse, de stabilité et de filtrage des bruits peut-on améliorer en rajoutant un correcteur P ou un correcteur PI ? Les réponses seront portées sur un tableau tel que défini ci-après. On utilisera les termes « bonne » et « mauvaise ».

Type régulateur	P	PI
Précision		
Stabilité		
Rapidité		
Filtrage des bruits		

Tableau 3 : Performances d'un P et PI régulateur

Annexe 1 : programme de la question 10 en Python.

```
from math import *
## Calcul de la distance de freinage d un vehicule
## Methode : Tangente amelioree
print("Calcul de la distance de freinage d"un vehicule de masse m")
print(" Vitesse initiale v0")
print(" coefficient de frottement sol roue sur sol sec f=0.8+0.2*e(-V/Vref)")
print(" position initiale x0=0")
##
V0=float(input("vitesse initiale v0=? en m/s" ))
h=float(input("pas de temps d"integration h=? en s" ))
##==============================================
## Donnees
g=9.81 #m/s2
## paramètres du coefficient de frottement f=a+b*e(-V/Vref)
a=0.8
b=0.2
Vref=5 # m/s
##==============================================
## t - temps (discretise avec le pas h)
## y1 - abscisse x calculee au pas i-1
## y2 - vitesse horizontale calculee au pas i-1
## y1n - abscisse x calculee au pas i
## y2n - vitesse horizontale calculee au pas i
##==============================================
## conditions initiales
t=0
x0=0
y1=x0
y2=V0
y2n=V0
##==============================================
while [                    ]        # ZONE A
    t=t+h
    ##======
    ## Euler
                                    # DEBUT ZONE B
    [                    ]
                                    # FIN ZONE B
    ##=====================
    y1=y1n
    y2=y2n
##==============================================
                                    # DEBUT ZONE C
DA=  [                    ]
Vf=
tf=
                                    # FIN ZONE C
##==============================================
print("Temps de freinage " + "tf=" + str(tf))
print("Distance d"arret " + "DA="+ str(DA))
print("Verification vitesse nulle " + "Vf=" + str(Vf))
```

Annexe 2 : programme de la question 10 en Scilab.

```
// Calcul de la distance de freinage d un vehicule
// Methode : Tangente amelioree
disp("Calcul de la distance de freinage d"un vehicule de masse m");
disp(" Vitesse initiale v0");
disp(" coefficient de frottement sol roue sur sol sec f=0.8+0.2*e(-V/VREF)");
disp(" position initiale x0=0");
disp("=====================================================");
V0=input("vitesse initiale v0=? " );
h=input("pas de temps d"integration h=? " );
//==============================================
// Donnees
g=9.81;
// paramètres du coefficient de frottementf=a+b*e(-V/VREF)
a=0.8;
b=0.2;
Vref=5;
//==============================================
// t - temps (discretise avec le pas h)
// y1 - abscisse x calculee au pas i-1
// y2 - vitesse horizontale calculee au pas i-1
// y1n - abscisse x calculee au pas i
// y2n - vitesse horizontale calculee au pas i
//==============================================
// conditions initiales
t=0;
x0=0;
y1=x0;
y2=V0;
y2n=V0;
//===============================================
while [                    ]        // ZONE A
t=t+h;
//======
// Euler
                                   // DEBUT ZONE B
      [                    ]
                                   // FIN ZONE B
//===================
y1=y1n;
y2=y2n;
end
//=================================================
                                   // DEBUT ZONE C
tf=
DA=    [                    ]
Vf=
                                   // FIN ZONE C
disp("Temps de freinage  " + "tf=" + string(tf));
disp("Distance d"arret  " + "DA="+ string(DA));
disp("Verification vitesse nulle  " + "Vf=" + string(Vf));
```

Exercice 1

Pour tout $n \in \mathbb{N}$ tel que $n \geqslant 2$, on désigne par $A_n = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ n-1 & 0 & 2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 2 & 0 & n-1 \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}$

la matrice de $\mathcal{M}_n(\mathbb{R})$ de coefficients $a_{i,j}$ tous nuls exceptés ceux tels que :

$$\forall k \in [\![1, n-1]\!], \qquad a_{k+1,k} = n-k, \quad a_{k,k+1} = k.$$

1 **1.a)** On prend $n = 2$. A_2 est- elle diagonalisable ? A_2 est-elle inversible ?

1.b) On prend $n = 3$. A_3 est -elle diagonalisable? A_3 est- elle inversible?

2 Dans cette question seulement on prend $n = 4$. Soit $A_4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 3 & 0 & 2 & 0 \\ 0 & 2 & 0 & 3 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ notée simplement A et soit le polynôme $Q(X) = (X^2-1)(X^2-9)$.

2.a) Calculer les matrices A^2 et $Q(A)$.

2.b) Justifier que A est diagonalisable.

2.c) Déterminer une matrice $\Delta \in \mathcal{M}_4(\mathbb{R})$, à coefficients entiers, inversible et telle que $\Delta^{-1}A\Delta = D$ soit une matrice diagonale notée $diag(a, b, c, d)$ avec $a > b > c > d$.

2.d) Montrer que $\Phi : M \mapsto \Delta M \Delta^{-1}$ est un endomorphisme de l'espace vectoriel réel $\mathcal{M}_4(\mathbb{R})$ et justifier que Φ est bijectif.

2.e) Montrer qu'une matrice $N \in \mathcal{M}_4(\mathbb{R})$ commute avec D si et seulement si N est une matrice diagonale.

2.f) Montrer que l'ensemble $\mathcal{C}_A = \{M \in \mathcal{M}_4(\mathbb{R}) / \quad AM = MA\}$ est un sous-espace vectoriel de $\mathcal{M}_4(\mathbb{R})$ et, en utilisant les questions précédentes, déterminer sa dimension.

2.g) Justifier que la famille (I_4, A, A^2, A^3) est une base de $\mathcal{C}_A$ (où $I_4 = diag(1,1,1,1)$).

3 On note ici $E = \mathbb{R}_{n-1}[X]$ l'espace vectoriel réel des polynômes réels de degré $\leqslant n-1$, et $\mathcal{B} = (1, X, \dots, X^{n-1})$ la base canonique de E.

Si $P(X) \in E, P'(X)$ désigne le polynôme dérivé de $P(X)$.

3.a) Justifier que l'application $\varphi : P(X) \mapsto (n-1)XP(X) + (1-X^2)P'(X)$ est un endomorphisme de l'espace vectoriel E et calculer $\varphi(X^h)$ pour tout $h \in [\![0, n-1]\!]$.

3.b) Comparer la matrice de φ dans la base $\mathcal{B}$ et la matrice A_n définie en préliminaire.

3.c) Pour tout $h \in [\![0, n-1]\!]$, on définit le polynôme $P_h = (X-1)^h(X+1)^{n-1-h}$. Déterminer le réel λ_h tel que $\varphi(P_h) = \lambda_h P_h$. Que peut-on déduire de ce calcul ?

3.d) Justifier que la matrice A_n est diagonalisable dans $\mathcal{M}_n(\mathbb{R})$.

3.e) Préciser selon n dans quel cas A_n est inversible.

3.f) Montrer l'existence d'une matrice $U_n \in \mathcal{M}_n(\mathbb{R})$ telle que $(U_n)^3 = A_n$.

Exercice 2

1 **1.a)** Rappeler la définition de la fonction Arctan, ainsi que son tableau de variation et sa dérivée. Justifier que pour tout réel x, on a : $|\operatorname{Arctan}(x)| \leqslant |x|$.

1.b) Étudier et représenter la fonction $g\ :]-\frac{\pi}{2}, \frac{\pi}{2}[\, \cup\,]\frac{\pi}{2}, \frac{3\pi}{2}[\to \mathbb{R}$ définie par :

$$\forall x \in]-\tfrac{\pi}{2}, \tfrac{\pi}{2}[\, \cup\,]\tfrac{\pi}{2}, \tfrac{3\pi}{2}[, \qquad g(x) = \operatorname{Arctan}(\tan(x)).$$

1.c) Pour $x > 0$, soit $\psi(x) = \operatorname{Arctan}(x) + \operatorname{Arctan}\left(\frac{1}{x}\right)$. Calculer la dérivée $\psi'(x)$.

En déduire une relation entre $\operatorname{Arctan}(x)$ et $\operatorname{Arctan}\left(\frac{1}{x}\right)$ pour $x > 0$.

1.d) Déterminer le développement en série entière de la fonction Arctan sur $]-1, 1[$.

2 On considère la fonction h définie par $h(0) = 1$ et pour tout réel $x \neq 0$,

$$h(x) = \frac{\operatorname{Arctan}(x)}{x}.$$

2.a) Justifier que pour tout $x \in [-1, 1]$, on a : $h(x) = \displaystyle\sum_{k=0}^{+\infty} \frac{(-1)^k x^{2k}}{(2k+1)}$.

On traitera le cas $|x| < 1$ et on étendra le résultat au cas $|x| = 1$, en utilisant un argument à préciser sur la continuité sur $[-1, 1]$ de la somme de la série de fonctions.

2.b) Justifier que h est de classe $\mathcal{C}^\infty$ sur $\mathbb{R}$ (*on raisonnera sur $]-1, 1[$ puis sur $\mathbb{R}$.*)

3 Soit $H(x) = \displaystyle\int_0^x \frac{\operatorname{Arctan}(t)}{t}\, dt$ pour $x \in \mathbb{R}$.

3.a) Montrer que H est bien définie et est de classe $\mathcal{C}^1$ sur $\mathbb{R}$; préciser H'.

3.b) Trouver une relation entre $H(x)$ et $G(x) = H(\frac{1}{x})$ pour $x > 0$.
Pour cela, on pourra utiliser G'.

3.c) Soit $f(x) = \dfrac{H(x)}{x}$ pour $x \neq 0$ et $f(0) = 1$.
Montrer que f est développable en série entière sur $[-1, 1]$, et f de classe C^∞ sur $\mathbb{R}$.

3.d) Donner une relation entre $f(x)$ et $f(\frac{1}{x})$ pour $x > 0$.

4 Pour $x \in \mathbb{R}$, on pose : $\varphi(x) = \displaystyle\int_0^{+\infty} \frac{\operatorname{Arctan}(tx)}{t\,(t^2+1)}\, dt$.

4.a) Montrer que φ est définie sur $\mathbb{R}$ et impaire.

4.b) Montrer que φ est de classe $\mathcal{C}^1$ sur $\mathbb{R}$. *On précisera bien le théorème utilisé.*

4.c) Déterminer $\varphi'(x)$ pour $x \in \mathbb{R}_+$ (avec une expression sans intégrale), si $x \neq 1$ puis $x = 1$. *Pour cela on pourra utiliser la relation :*

$$\forall t \in \mathbb{R}_+, \quad \frac{1}{(1+t^2)(1+x^2t^2)} = \frac{1}{1-x^2}\left(\frac{1}{1+t^2} - \frac{x^2}{1+x^2t^2}\right).$$

4.d) En déduire la valeur de $\varphi(x)$ pour tout réel x.

4.e) Si $x \in \mathbb{R}$, justifier l'existence et calculer $K(x) = \displaystyle\int_0^{\frac{\pi}{2}} \frac{\operatorname{Arctan}(x\tan(\theta))}{\tan(\theta)}\, d\theta$.

4.f) En déduire la valeur de $J = \displaystyle\int_0^{\frac{\pi}{2}} \ln(\sin(\theta))\, d\theta$.

Exercice 3

On considère l'espace vectoriel euclidien orienté $E = \mathcal{M}_{3,1}(\mathbb{R})$ muni de son produit scalaire canonique, la base canonique $\mathcal{B}$ étant orthonormale et directe. On notera $\langle .,. \rangle$ le produit scalaire tel que si $X, Y \in E$, $\langle X, Y \rangle = {}^tXY$.

$\mathcal{L}(E)$ désigne l'espace vectoriel des endomorphismes de E.

1 On considére les deux sous-espaces vectoriels de $\mathcal{L}(E)$ définis par :

$$\mathcal{S}(E) = \{u \in \mathcal{L}(E) / \ \forall (X,Y) \in E^2, \quad \langle u(X), Y \rangle = \langle X, u(Y) \rangle\},$$
$$\mathcal{A}(E) = \{u \in \mathcal{L}(E) / \ \forall (X,Y) \in E^2, \quad \langle u(X), Y \rangle = -\langle X, u(Y) \rangle\}.$$

1.a) Justifier que $\mathcal{S}(E)$ est bien un sous-espace vectoriel de $\mathcal{L}(E)$ (on l'admettra pour $\mathcal{A}(E)$) et montrer que $\mathcal{S}(E) \cap \mathcal{A}(E)$ est réduit à l'endomorphisme nul.

1.b) Soit $u \in \mathcal{L}(E)$ et $M = Mat_{\mathcal{B}}(u)$ la matrice de u dans la base $\mathcal{B}$, avec $u : X \mapsto MX$.

Montrer que :

(i) $u \in \mathcal{S}(E) \Longleftrightarrow {}^tM = M$, $\qquad$ (ii) $u \in \mathcal{A}(E) \Longleftrightarrow {}^tM = -M$.

1.c) Soit $u \in \mathcal{L}(E)$ et $M = Mat_{\mathcal{B}}(u)$. On lui associe $\hat{u} \in \mathcal{L}(E)$ tel que $Mat_{\mathcal{B}}(\hat{u}) = {}^tM$.

Montrer que $(u + \hat{u}) \in \mathcal{S}(E)$ et que $(u - \hat{u}) \in \mathcal{A}(E)$

1.d) Conclure que $\mathcal{S}(E)$ et $\mathcal{A}(E)$ sont deux sous-espaces supplémentaires de $\mathcal{L}(E)$.

2 **2.a)** Soit $\sigma \in \mathcal{S}(E)$. Justifier l'existence de (e_1, e_2, e_3) base orthonormale de E et de trois réels $\lambda_1, \lambda_2, \lambda_3$ tels que :

$$\forall\, V \in E, \quad \sigma(V) = \lambda_1 \langle e_1, V \rangle e_1 + \lambda_2 \langle e_2, V \rangle e_2 + \lambda_3 \langle e_3, V \rangle e_3.$$

2.b) Préciser dans quel cas σ est une projection orthogonale, ou alors une symétrie orthogonale.

2.c) Exemple 1. Soit σ tel que $Mat_{\mathcal{B}}(\sigma) = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & -2 \\ 2 & -2 & 1 \end{pmatrix}$.

Déterminer (e_1, e_2, e_3) base orthonormale de E et $\lambda_1, \lambda_2, \lambda_3$ pour σ comme en **2.a)**.

3 Soit $\alpha \in \mathcal{A}(E)$ non nul.

3.a) Montrer que $Ker(\alpha)$ et $Im(\alpha)$ sont orthogonaux, puis que $E = Ker(\alpha) \oplus Im(\alpha)$.

3.b) Montrer que $Im(\alpha)$ est stable par α et que l'endomorphisme $\tilde{\alpha}$ induit par α sur $Im(\alpha)$ ne peut pas avoir de valeur propre (réelle). En déduire que α est de rang 2.

3.c) Justifier l'existence de (e_1, e_2, e_3) base orthonormale directe de E et d'un réel $k \neq 0$ tels que : $\qquad \forall\, V \in E, \quad \alpha(V) = k\left(\langle e_1, V \rangle e_2 - \langle e_2, V \rangle e_1\right).$

3.d) Exemple 2. Soit α tel que $Mat_{\mathcal{B}}(\alpha) = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$

Déterminer (e_1, e_2, e_3) base orthonormale directe de E et $k \neq 0$ associés à α (cf. **3.c)**).

4 Soit $\theta \in \mathbb{R}$ et la rotation $r \in SO(E)$ telle que $Mat_{\mathcal{B}}(r) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

On considère aussi $\hat{r}$ tel que $Mat_{\mathcal{B}}(\hat{r}) = {}^t\left(Mat_{\mathcal{B}}(r)\right)$.

Donner la décomposition de $\sigma = \dfrac{r + \hat{r}}{2}$ comme en **2.a)**, ainsi que de $\alpha = \dfrac{r - \hat{r}}{2}$ comme en **3.c)** lorsque $\alpha \neq 0$.

Exercice 4

On s'intéresse aux nombres de **Fibonacci** définis par récurrence par $F_0 = 0$, $F_1 = 1$ et:

$$\forall n \in \mathbb{N}, \qquad F_{n+2} = F_{n+1} + F_n.$$

1 **1.a)** Écrire en langage Python une fonction "$fib(n)$" permettant le calcul de F_n.

1.b) On peut démontrer et on l'admettra, que ces nombres vérifient les relations suivantes pour tout $n \in \mathbb{N}^*$:

$$F_n = 2F_{\frac{n}{2}-1}F_{\frac{n}{2}} + (F_{\frac{n}{2}})^2 \text{ si } n \text{ est pair}, \qquad F_n = (F_{\frac{n-1}{2}+1})^2 + (F_{\frac{n-1}{2}})^2 \text{ si } n \text{ est impair.}$$

Vérifier cette propriété en calculant ainsi F_6.

Compléter l'écriture de la fonction "$fibb(n)$" ci-dessous permettant de calculer F_n, on commencera au dernier "else" sur la copie en rajoutant les instructions manquantes.

```
def fibb(n):
    if n <= 1:
            return n
    else :
            if n%2 == 0:
                a = fibb(n//2)
                b = fibb(n//2 - 1)
                return a * (2 * b + a)
            else : ...
```

2 **2.a)** Soit $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$. Pour tout $n \in \mathbb{N}^*$, exprimer les coefficients de A^n au moyen des nombres de Fibonacci F_{n+1}, F_n et F_{n-1}.

2.b) Écrire en langage Python une fonction "$fibbb(n)$" permettant le calcul de F_n en utilisant le calcul de A^n (dont le calcul doit intervenir dans le programme).

2.c) Montrer que pour tout $n \in \mathbb{N}^*$, on a : $\quad F_{n+1}F_{n-1} - (F_n)^2 = (-1)^n$.

3 **3.a)** Montrer la convergence de la série de terme général $\dfrac{(-1)^n}{F_{n+1}F_n}$ $(n \in \mathbb{N}^*)$ et exprimer $S = \displaystyle\sum_{n=1}^{\infty} \frac{(-1)^n}{F_{n+1}F_n}$ au moyen de $L = \displaystyle\lim_{p \to +\infty} \frac{F_p}{F_{p+1}}$ (dont l'existence résulte de la convergence de la série).

3.b) Grâce à une propriété concernant les séries alternées et que l'on rappellera, étant donné $\varepsilon > 0$, écrire un algorithme (en français ou alors en langage Python) permettant de calculer une valeur approchée de L au moyen d'une somme partielle de cette série, avec une précision inférieure à ε.

4 On note $\Phi = \dfrac{1+\sqrt{5}}{2}$ et $\varphi = \dfrac{1-\sqrt{5}}{2}$. On peut montrer et on l'admettra, que l'on a:

$$\forall n \in \mathbb{N}, \quad F_n = \frac{1}{\sqrt{5}}\left(\Phi^n - \varphi^n\right)$$

4.a) Calculer L de **3.a)** au moyen de Φ.

4.b) Pour tout $x \in]\varphi, -\varphi[$, montrer la convergence de la série ainsi que l'égalité :

$$\sum_{n=1}^{+\infty} F_n x^n = \frac{x}{1 - x - x^2}.$$

CONCOURS ARTS ET MÉTIERS ParisTech - ESTP - POLYTECH

Épreuve de Physique-Modélisation PC

Durée 4 h

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, d'une part il le signale au chef de salle, d'autre part il le signale sur sa copie et poursuit sa composition en indiquant les raisons des initiatives qu'il est amené à prendre.

L'usage de calculatrices est autorisé.

AVERTISSEMENT

Les parties A, B, C, D et E d'une part, et les parties F, G et H d'autre part, sont à rédiger sur copies séparées.

- Les durées indicatives pour chacune des parties sont les suivantes :
  - Première partie (A+B+C) : environ 1h50.
  - Deuxième partie (D+E) : environ 1h30.
  - Troisième partie (F+G+H) : environ 40 minutes.
- Les explications des phénomènes étudiés interviennent dans la notation au même titre que les développements analytiques et les applications numériques ; les résultats exprimés sans unité ne seront pas comptabilisés.
- Tout au long de l'énoncé, les paragraphes en italiques ont pour objet d'aider à la compréhension du problème mais ne donnent pas lieu à des questions.
- Tout résultat fourni dans l'énoncé peut être admis et utilisé par la suite, même s'il n'a pas été démontré par le(la) candidat(e).
- Les questions comportant le mot calculer demandent une application numérique.

Ce problème, qui comporte trois parties indépendantes, s'intéresse au ***phénomène de marées****.*
La première partie traite du phénomène en lui-même, la seconde décrit la mesure des hauteurs d'eau par un marégraphe à ultrasons et enfin la troisième exploite une base de données.
Les trois parties sont largement indépendantes.

Données :

- distance Terre Lune : $d_L = 3,8.10^8\ m$
- distance Terre Soleil : $d_S = 1.5.10^{11}\ m$
- rayon de la Terre : $R_T = 6,4.10^6\ m$
- masse du Soleil : $m_S = 2.10^{30}\ kg$
- masse de la Terre : $m_T = 6.10^{24}\ kg$
- masse de la Lune : $m_L = 7,3.10^{22}\ kg$
- constante de gravitation universelle : $G = 6,7.10^{-11}\ kg^{-1}.m^3.s^{-2}$

Lexique :

- pleine mer : hauteur maximale de la marée
- basse mer : hauteur minimale de la marée
- marnage : différence de hauteurs entre une pleine mer et une basse mer consécutives
- vive-eau : marée pendant laquelle le marnage est maximal
- phase de la pleine mer : heure à laquelle la pleine mer est atteinte

PREMIERE PARTIE
LE PHENOMENE DE MAREES

A / NOTIONS QUALITATIVES SUR LES MAREES

La carte reproduite dans la figure 1 représente l'évolution de la marée réelle dans la Manche. Les nombres indiqués sous certains ports sont la phase de la pleine mer et le marnage par vive-eau. On trouve deux types de courbes :

- *Les lignes cotidales (avec une indication en heures) représentent les points dans le même «état de marée» (pleine mer) à un instant donné (les valeurs données correspondent à la date de la pleine mer par rapport à une référence arbitraire).*
- *Les lignes iso-marnage (avec une indication en mètres) représentent les points avec un même marnage (le marnage est la différence de hauteur entre la pleine mer et la basse mer).*

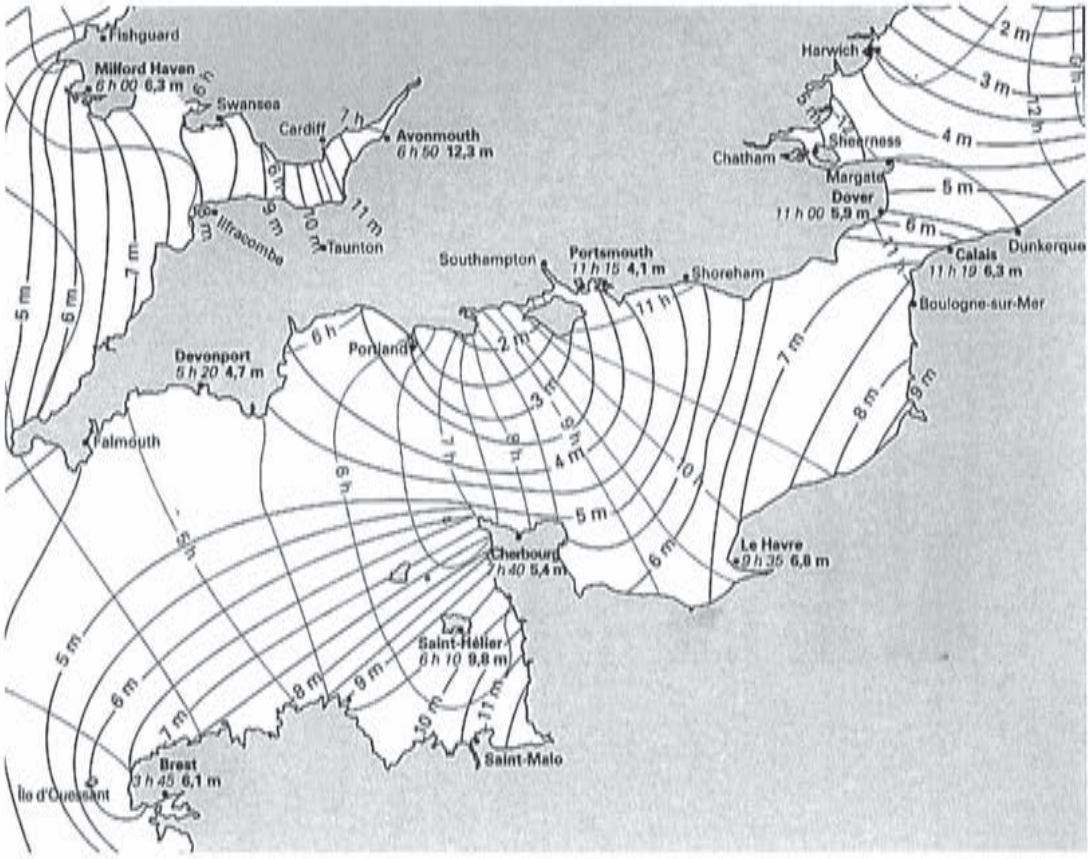


FIGURE 1

A1. A quels endroits de la carte les marées sont-elles les plus importantes ? Est-ce dû à une particularité géographique ? On attend une réponse brève.

A2. On peut envisager l'évolution spatiale et temporelle de la hauteur d'eau due aux marées comme résultant de la propagation d'une *onde de marée*. Dans quel sens se déplace cette onde de marée dans la Manche ? Une explication basée sur la rotation propre de la Terre est-elle satisfaisante ? Justifier.

A3. Donner (sans explication) un ordre de grandeur de la périodicité des marées océaniques. Donner un ordre de grandeur de la vitesse de déplacement de l'onde de marée dans la Manche (à titre de point de repère, la distance entre Saint-Malo et Brest est de l'ordre de 200km). En déduire un ordre de grandeur de la longueur d'onde associée.

A4. Dans la Manche la marée est déviée vers les côtes françaises, ce qui a pour conséquence des marnages plus importants que sur les côtes anglaises. Interpréter cette déviation (un schéma clair est attendu).

A5. A l'ouest de la ville de Saint-Malo, on distingue sur la carte l'estuaire de la Rance où est implantée une usine marémotrice. Justifier ce choix d'implantation ; pourquoi ne pas avoir fait de même sur l'estuaire de la Seine, au niveau du Havre (Normandie, Seine-Maritime) ?

B / CHAMP DE MAREE

Les marées sont dues aux champs de gravitation au niveau de la Terre des différents astres du système solaire, principalement la Lune et le Soleil. On considérera que les astres ont une distribution de masse à symétrie sphérique.

B1. Donner sans justification l'expression du champ de gravitation $\overrightarrow{g_A}(M)$ créé par l'astre A, de masse m_A et de centre O, en un point M en dehors de l'astre. On pourra noter $r = OM$ la distance entre O et M et $\overrightarrow{e_r}$ un vecteur unitaire dirigé de O vers M.

On cherche maintenant à établir cette expression.

B2. Montrer par des considérations de symétrie que $\overrightarrow{g_A}(M) = g_A(r)\overrightarrow{e_r}$.

B3. Enoncer le théorème de Gauss dans le cadre de l'électromagnétisme, puis le transposer au cas de la gravitation. Utiliser ce résultat pour retrouver l'expression donnée au B1. Préciser quelle est la simplification dans l'expression du champ de gravitation en dehors de l'astre apportée par la symétrie sphérique de la distribution de masses.

L'influence d'un astre sur les marées découle d'une petite différence entre la force de gravitation qu'il exerce et la force d'inertie dont il est responsable dans le référentiel géocentrique . On établit ici l'expression du champ de marée en prenant le Soleil comme exemple (dans les trois questions qui suivent, on ne considère que les forces de gravitation dues au soleil), mais le résultat est valable pour n'importe quel astre. Dans toute la suite, on considérera le référentiel héliocentrique (R_h) comme galiléen.

B4. Décrire le mouvement du référentiel géocentrique par rapport au référentiel héliocentrique et en déduire l'expression de la force d'inertie d'entrainement sur un point matériel M de masse m dans le référentiel géocentrique. On notera $\overrightarrow{a}_{T/R_h}$ l'accélération de T (centre de la Terre) dans le référentiel héliocentrique.

B5. Etablir que $\overrightarrow{a}_{T/R_h} = \overrightarrow{g_S}(T)$, où $\overrightarrow{g_S}(T)$ est le champ de gravitation créé par le Soleil au centre de la terre T. La Terre sera supposée avoir une distribution de masse à symétrie sphérique, ce qui fait que la force de gravitation exercée par le Soleil sur la Terre est assimilable au produit de la masse de la terre par le champ de gravitation du Soleil en son centre.

B6. En déduire que la résultante de la force de gravitation et de la force d'inertie d'entrainement dans le référentiel géocentrique dues au Soleil sur un point matériel M de masse m s'écrit :

$$m\overrightarrow{C_S}(M) \quad \text{avec} \quad \overrightarrow{C_S}(M) = -Gm_S\left(\frac{\overrightarrow{SM}}{SM^3} - \frac{\overrightarrow{ST}}{ST^3}\right)$$

où S désigne le centre du Soleil, m_S sa masse, et G la constante de gravitation universelle. $\overrightarrow{C_S}(M)$ est appelé *champ de marée* du Soleil au point M.

Les marées sont essentiellement dues à l'influence de la Lune, celle du Soleil se traduisant par une plus ou moins grande amplitude (marées de vives eaux et de mortes eaux). Dans la suite on ne considère que l'influence de la Lune. Le résultat de la question B6 est transposable à n'importe quel astre, l'expression du champ de marée dû à la Lune est donc (L désignant le centre de la Lune) :

$$\overrightarrow{C_L}(M) = -Gm_L\left(\frac{\overrightarrow{LM}}{LM^3} - \frac{\overrightarrow{LT}}{LT^3}\right)$$

Sur le schéma de la figure 2 on indique quelques points particuliers à la surface de la Terre, relativement à la position de la Lune.

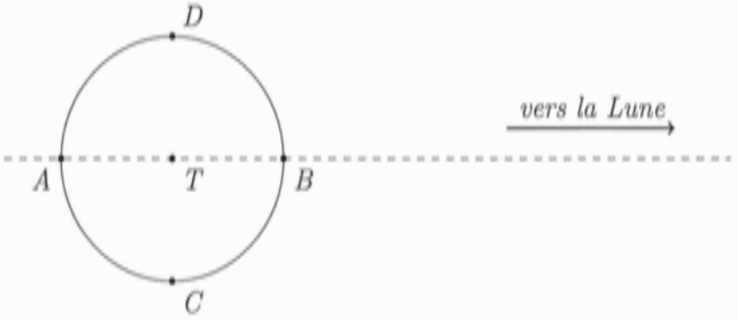


FIGURE 2

B7. Reprendre le dessin précédent et représenter en A, B, C et D la force gravitationnelle et la force d'inertie dues à la lune, ainsi que leur résultante (proportionnelle au champ de marée).

B8. Indiquer les points (parmis A, B, C et D) de marée haute et de marée basse. Dans quel plan sont situés tous les points de marée basse ?

B9. En utilisant la troisième loi de Kepler, donner un ordre de grandeur de la période de révolution de la Lune dans le référentiel géocentrique.

B10. Donner un ordre de grandeur de la période de rotation propre de la Terre. Conclure sur la périodicité (approximative) des marées.

On cherche à simplifier l'expression du champ de marée, en tenant compte du fait que, pour un point M à la surface de la terre, $TM << TL$ et en effectuant un développement limité au premier ordre en $\frac{TM}{TL}$. On posera $TM = r$ et $TL = d_L$, et on repérera la position de M, dans le plan contenant L, T et M, en coordonnées polaires (voir la figure 3).

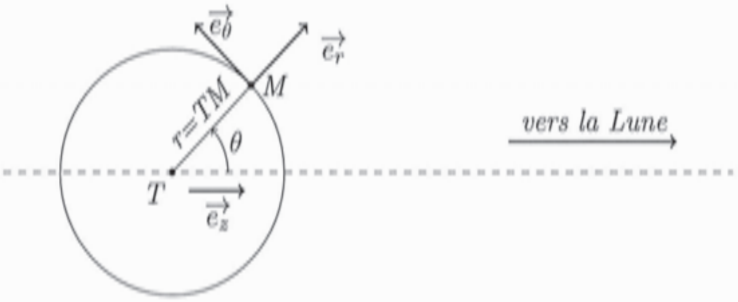


FIGURE 3

B11. Montrer que $\overrightarrow{LM} = -d_L\overrightarrow{e_z}+r\,\overrightarrow{e_r}$. En déduire que, au premier ordre en $\frac{r}{d_L}$, on a $\frac{1}{LM^3} = \frac{1}{{d_L}^3}\left(1+\frac{3r\cos(\theta)}{d_L}\right)$.

B12. En déduire que, toujours au premier ordre en $\frac{r}{d_L}$:

$$\overrightarrow{C_L}(M) = \frac{Gm_L r}{{d_L}^3}\left(3\cos(\theta)\overrightarrow{e_z} - \overrightarrow{e_r}\right)$$

En projetant $\overrightarrow{e_z}$ sur la base $(\overrightarrow{e_r}, \overrightarrow{e_\theta})$, on obtient finalement :

$$\overrightarrow{C_L}(M) = \frac{G m_L r}{{d_L}^3}\left(\left(3\cos^2(\theta) - 1\right)\overrightarrow{e_r} - 3\sin(\theta)\cos(\theta)\overrightarrow{e_\theta}\right)$$

B13. Montrer que le l'influence de la Lune sur les marées est de l'ordre de 2 fois plus importante que celle du soleil.

B14. Préciser les positions relatives de la Terre, de la Lune et du Soleil pour les marées de vives eaux (amplitude maximale, les effets de la Lune et du Soleil s'ajoutent) et pour les marées de mortes eaux (amplitude minimale, les effets de la Lune et du Soleil se compensent partiellement). Attention à bien indiquer deux configurations distinctes pour chaque cas. Indiquer le lien avec les phases de la Lune et donner un ordre de grandeur de la périodicité de l'alternance vives-eaux / mortes-eaux.

C / AMPLITUDE DES MAREES OCEANIQUES

On considère dans cette partie un modèle simple : la Terre est entièrement recouverte d'eau. On obtient ainsi des résultats pertinents pour l'amplitude des marées en haute mer, mais qui n'expliquent pas les phénomènes observés près des côtes. On se place dans le cadre d'un modèle quasi-statique où la forme des océans à un instant donné obéit à la loi de l'hydrostatique (p désigne la pression et $\overrightarrow{f_v}$ la résultante des forces volumiques) :

$$\overrightarrow{grad}(p) = \overrightarrow{f_v}$$

Dans toute cette partie on ne considère que l'influence de la Lune (et pas celle du Soleil). Le marnage, *que l'on notera Δh, est la différence de hauteur d'eau entre la marée haute et la marée basse en un endroit donné.*

C1. On commence par une approche dimensionnelle : on considère que, outre le facteur $\frac{m_L}{{d_L}^3}$ mis en évidence précédemment, la masse de la Terre m_T et son rayon R_T interviennent sur le marnage et on pose :

$$\Delta h = \frac{m_L}{{d_L}^3} {m_T}^\alpha {R_T}^\beta$$

Déterminer les coefficients α et β pour que Δh ait bien les dimensions d'une longueur et calculer numériquement la valeur de Δh qui en résulte.

C2. Que traduit la loi de l'hydrostatique ? Quelle est la dimension de ses termes ? Le terme $\overrightarrow{f_v}$ traduit ici l'attraction gravitationnelle due à la Terre, celle due à la Lune ainsi que la force d'inertie d'entrainement dans le référentiel géocentrique due à la Lune. Donner l'expression de $\overrightarrow{f_v}$ en utilisant, pour ce qui est des effets dus à la Lune, le résultat donné dans la partie B. On notera μ la masse volumique de l'eau.

On pose $\overrightarrow{f_v} = -\overrightarrow{grad}(V_T + V_L)$, où V_T est l'énergie potentielle volumique associée à l'attraction gravitationnelle de la Terre et V_L l'énergie potentielle volumique associée aux effets dus à la Lune. On donne l'expression du gradient d'une fonction scalaire f en coordonnées sphériques :

$$\overrightarrow{grad}(f) = \frac{\partial f}{\partial r}\overrightarrow{e_r} + \frac{1}{r}\frac{\partial f}{\partial \theta}\overrightarrow{e_\theta} + \frac{1}{r\sin(\theta)}\frac{\partial f}{\partial \varphi}\overrightarrow{e_\varphi}$$

C3. Etablir l'expression de V_T, en expliquant bien le choix de la constante.

On donne $V_L = -\mu G \frac{m_L}{{d_L}^3} \frac{r^2}{2} \left(3\cos^2(\theta) - 1\right)$. *La pression atmosphérique est considérée uniforme à la surface de l'eau, et on néglige les phénomènes de tension superficielle.*

C4. Montrer que dans ces conditions la surface de l'eau vérifie $V_T + V_L = cste_1$, et en déduire que, toujours à la surface de l'eau, $\frac{m_T}{r} + \frac{m_L}{{d_L}^3} \frac{r^2}{2} \left(3\cos^2(\theta) - 1\right) = cste_2$

On détermine (en écrivant que, l'eau étant considérée comme incompressible, le volume des océans est le même avec et sans déformation) que la constante introduite précédemment vaut $\frac{m_T}{R_T}$.

C5. La déformation h étant petite par rapport à R_T, on pose $r = R_T + h$ avec $\frac{h}{R_T} << 1$. En effectuant les développements limités au premier ordre en $\frac{h}{R_T}$ adéquats, montrer que :

$$h\left(\frac{m_T}{{R_T}^2} - \frac{m_L R_T}{{d_L}^3}\left(3\cos^2(\theta) - 1\right)\right) \simeq \frac{m_L {R_T}^2}{2{d_L}^3}\left(3\cos^2(\theta) - 1\right)$$

Vérifier numériquement que $\frac{m_L R_T}{{d_L}^3} << \frac{m_T}{{R_T}^2}$ et simplifier l'expression en conséquence. On utilisera cette expression dans les deux questions suivantes.

Pour simplifier, on considère que la Lune reste dans le plan équatorial.

C6. Où le marnage Δh est-il le plus important ? Que peut-on dire du marnage aux pôles ?

C7. Etablir l'expression du marnage à l'Equateur et faire l'application numérique.

DEUXIEME PARTIE
LE MAREGRAPHE A ULTRASONS

Dans le contexte mondial actuel, mesurer le niveau des mers présente un intérêt certain. Outre les prévisions marégraphiques, l'étude de la hausse du niveau moyen des mers est devenu un sujet sensible. Le marégraphe côtier numérique (MCN) étudié ici fait partie d'un réseau de marégraphes installés sur les côtes françaises. Il est situé à Brest dans l'embouchure de la Penfeld.

D / PUITS DE TRANQUILLITE

La surface de l'eau en mer ou sur les côtes n'étant pas plane la plupart de temps, il importe de mesurer les variations du niveau de la mer en s'affranchissant des fluctuations de hauteur. C'est le rôle du puits de tranquillité. A l'intérieur du bâtiment (voir schéma de la figure 4), le puits de tranquillité est constitué d'un tube cylindrique vertical où l'eau rentre par le bas et peut monter librement. Les mesures de hauteur d'eau se font dans ce tube de diamètre 1,5 mètre pour le marégraphe de Brest. Même si le bâtiment est fermé, isolé du soleil et des intempéries, la hauteur du puits (plus de 8 mètres) fait que sa température interne n'est pas uniforme. Il faut donc envisager un gradient de température à l'intérieur du puits. On donne dans la figure 4 une vue en coupe du puits de tranquillité ainsi qu'un enregistrement de la température en fonction de l'altitude (les carrés correspondent aux points de mesure).

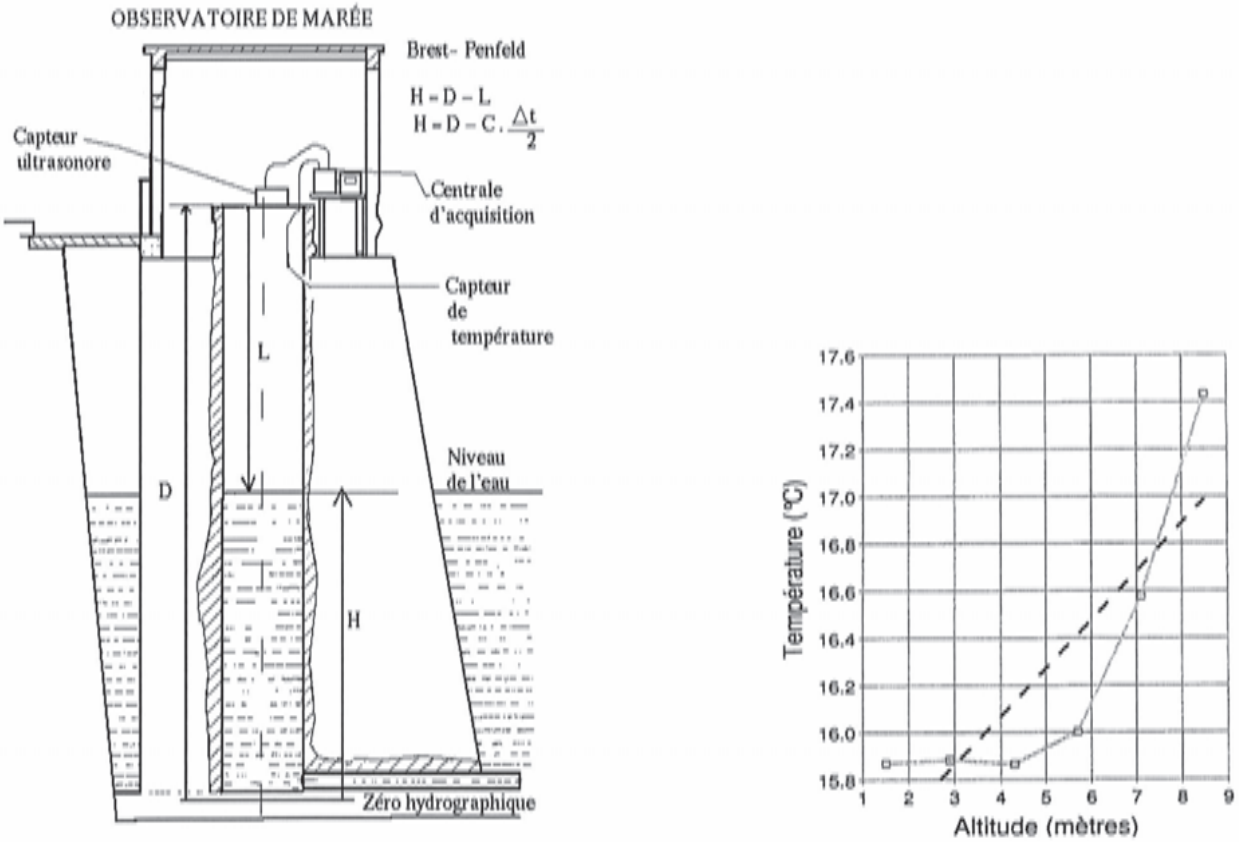


FIGURE 4

D1. Comment peut-on qualifier l'effet du puits de tranquillité sur les variations de hauteur d'eau en termes de filtrage ?

D2. Evaluer numériquement la norme du gradient de température en haut du puits (environ $8,5\ m$ d'altitude).

Pour rendre compte du gradient de température (dans l'air) qui s'établit dans le puits, on adopte le modèle suivant : Le puits est cylindrique, de rayon R_1 (on note $s = \pi {R_1}^2$ sa section) et de hauteur L (entre le niveau de l'eau et le haut du puits). L'air contenu dans le puits est assimilé à un matériau de masse volumique invariable μ, de capacité thermique massique c et de conductivité thermique λ que l'on considère globalement au repos (on néglige ainsi la convection et la dilatation, ce qui revient à dire que, pour simplifier la modélisation, on raisonne comme si l'air était un solide indilatable). La température de l'air dans le puits ne dépend que de la profondeur z (voir figure 5).

L'eau impose en $z = L$ (point P) une température T_1 tandis que la partie supérieure impose en $z = 0$ (point O) une température T_0. Les parois du puits, d'épaisseur e et de conductivité thermique λ', sont comprises entre les rayons R_1 et R_2 (donc $e = R_2 - R_1$) et caractérisées par un coefficient r_{th} homogène à une résistance thermique multipliée par une longueur et défini par $R_{th} = r_{th}/\ell$ où R_{th} est la résistance thermique associée à une longueur ℓ de parois. L'extérieur des parois est à la température de l'eau, T_1.

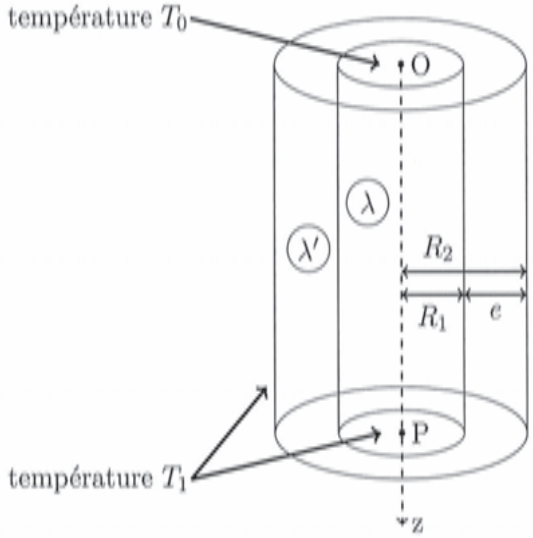


FIGURE 5

D3. On cherche à écrire l'équation de diffusion thermique à une dimension vérifiée par la température à l'intérieur du puits. On considère des évolutions à pression constante. On introduira un terme p qui représente une puissance par unité de longueur (selon (Oz)) et qui traduit les échanges thermiques au travers des parois du puits (p devant être positive si de l'énergie est effectivement reçue par l'air à l'intérieur du puits). Montrer que l'équation de diffusion thermique se met sous la forme :

$$\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial z^2} + \alpha\, p$$

Donner les expressions de D et α en fonction de μ, c, λ et s.

On se place en régime stationnaire. Dans un premier temps, on considère qu'il n'y a pas d'échanges thermiques au travers des parois.

D4. Montrer que cette hypothèse implique un gradient de température uniforme.

On prend maintenant en compte les échanges thermiques au travers des parois (et on est toujours en régime stationnaire).

D5. On propose pour r_{th} les expressions suivantes :

1. $\frac{2\pi}{\lambda'} \ln\left(\frac{R_1}{R_2}\right)$
2. $\frac{2\pi}{\lambda'} (R_2 - R_1)$
3. $\frac{1}{2\pi\lambda'} \ln\left(\frac{R_2}{R_1}\right)$
4. $\frac{\lambda'}{2\pi} \frac{R_2}{R_1}$
5. $\frac{1}{2\pi\lambda'} \frac{R_1}{R_2 - R_1}$

Déterminer quelle est la bonne expression en expliquant pourquoi les quatre autres ne peuvent être correctes.

D6. En partant de l'équation obtenue en D3, compte tenu de l'hypothèse de régime stationnaire, montrer que la température T à l'intérieur du puits vérifie l'équation :

$$\frac{d^2T}{dz^2} - k^2T = -k^2T_1$$

Donner l'expression de k en fonction de λ, s et r_{th}.

D7. Donner la forme des solutions de l'équation différentielle de la question D6 sans chercher à expliciter les constantes d'intégration. On admet ensuite que l'on doit se restreindre à une expression de la forme $T(z) = Ae^{-kz} + B$. Donner les expressions de A et B en fonction de T_1 et T_0.

D8. Représenter graphiquement T en fonction de z. Est-ce qualitativement en accord avec les données expérimentales ?

D9. En déduire l'expression du gradient de température dans le puits, commenter son sens et donner en particulier $\left\|\overrightarrow{grad}(T)\right\| (z = 0)$.

D10. On donne les valeurs numériques suivantes :

- $d = 1,5m$ (diamètre du puits)
- $\lambda = 0,023 W.m^{-1}.K^{-1}$ (conductivité thermique de l'air)
- $\lambda' = 1,5 W.m^{-1}.K^{-1}$ (conductivité thermique des parois)
- $e = 2m$ (épaisseur des parois)

Calculer numériquement r_{th}, k et $\left\|\overrightarrow{grad}(T)\right\| (z = 0)$. Critiquer cette dernière valeur.

E / TEMPS DE PROPAGATION

Le marégraphe de Brest est un marégraphe à ultrasons. Au sommet du puits de tranquillité se trouvent deux capteurs gérés par la centrale d'acquisition (voir le schéma en partie D ci-dessus). Un capteur assure l'émission-réception d'ondes ultrasonores de fréquence 41,5 kHz et un autre assure la collecte des températures mesurées par six sondes régulièrement espacées dans le puits.

A partir de l'intervalle de temps entre l'émission et la réception du signal ultrasonore, on peut déduire le tirant d'air L puis la hauteur H d'eau dans le puits. Cela suppose connue la célérité du son dans l'air du puits. Celle-ci est donnée par la formule suivante admise :

$$c = 331,2(1 + 0,97\frac{U}{P} + 1,9 \times 10^{-3}T)$$

- *c est la célérité du son (en $m.s^{-1}$),*
- *P est la pression atmosphérique en hPa,*
- *U est l'humidité relative de l'air,*
- *T est la température de l'air en ^oC.*

On pourra utiliser les développements limités (au voisinage de 0) suivants :

$$\ln(1+x) = x + ... \qquad et \qquad e^x - 1 = x + \frac{x^2}{2} + ...$$

E1. Pourquoi la fréquence des ondes ultrasonores n'intervient-elle pas dans la formule de la célérité donnée ci-dessus ? Comment évolue la célérité des ondes ultrasonores, pour une température donnée, en fonction de l'humidité de l'air ? Proposer une explication. On rappelle que la célérité v des ondes sonores dans un gaz parfait est donnée, avec les notations et approximations habituelles, par la relation $v = \sqrt{\frac{\gamma RT}{M}}$ où M est la masse molaire du gaz parfait.

E2. Pourquoi la variation de l'humidité relative est-elle faible dans le puits de tranquillité ?

Une étude quantitative plus approfondie montre que les variations de pression et les variations relatives de l'humidité de l'air ont peu d'effet sur la célérité des ondes ultrasonores. En revanche, la température reste le paramètre important. Lorsqu'on néglige l'influence de la température, la mesure de la hauteur L est entachée d'une erreur de près de 2 cm pour un transducteur placé à 10 m au-dessus de l'eau. Il faut donc étudier l'effet d'un gradient de température. On fait les hypothèses suivantes :

- *la célérité de l'onde est indépendante de sa fréquence*
- *l'expression de la célérité est localement valable sur le chemin de l'onde*
- *il n'y a pas de réflexion de l'onde ailleurs que sur la surface de l'eau*
- *on considère que seule la température influence la valeur de la célérité*

La relation entre le temps de parcours Δt du train d'onde et le tirant d'air est alors donnée par :

$$\Delta t = 2\int_0^L \frac{dz}{c(z)} \qquad \text{avec} \qquad c = c_0(1 + aT) \qquad \text{où } c_0 \text{ et } a \text{ sont des constantes}$$

E3. Expliquer la présence du "2" dans l'expression ci-dessus. Donner l'expression de L en fonction de c_0, Δt, a et T_0 en considérant une température uniforme égale à T_0 dans le puits.

Approximation par un gradient constant : On suppose dans un premier temps le développement au premier ordre suivant de $T(z)$:

$$T(z) = T_0 + Gz$$

G est la valeur du gradient en $z = 0$ et T_0 la valeur de la température au même niveau (on choisit le niveau $z = 0$ pour la plus haute sonde de température dans le puits).

E4. Exprimer l'intégrale permettant de calculer Δt puis en déduire l'expression de Δt en fonction de c_0, L, G, a et T_0.

E5. Montrer que L est alors donnée par : $L = \frac{1+aT_0}{aG}(e^{\frac{c_0 aG\Delta t}{2}} - 1)$

E6. Pour simplifier encore, on peut effectuer un développement limité à l'ordre deux de l'exponentielle dans l'expression précédente en supposant le gradient faible. On suppose pour cela l'inégalité : $L \ll \frac{1}{aG}$. Trouver la nouvelle expression de L et montrer que l'on retrouve le résultat de la question E3 plus un terme correctif à expliciter, noté δL_1.

On cherche à tester un autre modèle. On suppose maintenant la forme suivante pour la fonction température :

$$T(z) = T_0 - \Delta T(1 - e^{-\frac{z}{L_0}})$$

- *ΔT est la différence de température entre le haut du puits et la surface de la mer*
- *T_0 est la température du haut du puits*
- *L_0 est la hauteur caractéristique du gradient*

E7. Pourquoi cette nouvelle expression de $T(z)$ semble-t-elle mieux convenir ?

E8. Montrer que la nouvelle expression de Δt est de la forme :

$$\Delta t = B\int_0^L \frac{dz}{A + e^{-\frac{z}{L_0}}}$$

Exprimer les constantes A et B et préciser leur dimension.

Le calcul de l'intégrale de la question E8 permet d'exprimer Δt en fonction de L, puis d'en déduire L en fonction de Δt. On met ainsi en évidence un nouveau terme correctif δL_2 :

$$L = \frac{c_0\Delta t}{2}(1 + aT_0) + \delta L_2 \qquad \text{avec} \qquad \delta L_2 = a\Delta T\left(-L_0\left(1 - e^{-\frac{c_0(1+aT_0-a\Delta T)\Delta t}{2L_0}}\right) + \frac{c_0\Delta t}{2}\right)$$

On donne les valeurs numériques suivantes :

- $G = 0,21°C.m^{-1}$
- $a = 1,9.10^{-3}\ °C^{-1}$
- $c_0 = 331,2\ m.s^{-1}$
- $T_0 = 17,42\ °C$
- $\Delta T = 1,55\ °C$
- $\Delta t = 41,95\ ms$
- $L_0 = 0,82\ m$

E9. A l'aide des données des parties D et E, retrouver les 5 premières valeurs numériques proposées ci-dessus.

E10. Calculer les corrections δL_1 et δL_2 des deux modèles (gradient linéaire et gradient exponentiel) envisagés. Quel modèle vous paraît le mieux convenir à la situation ?

TROISIEME PARTIE
HAUTEURS DE MAREE A BREST

Dans cette partie, on respectera les consignes suivantes :

— lorsque du code est demandé, il doit être écrit en langage Python

— on se limitera aux types suivants : entiers, flottants, chaines de caractères, listes et tuples

— on se limitera aux mots clés suivants : `if`, `elif`, `else`, `is`, `while`, `for`, `in`, `def`, `return`, `lambda`, `and`, `or`, `not`, `True`, `False` et `None`

— on se limitera aux fonctions et méthodes préprogrammées suivantes : `print`, `input`, `plot`, `range`, `enumerate`, `len` et `append`

F / BASE DE DONNEES

Le service hydrographique et océanographique de la marine (shom) dispose de données très complètes sur les hauteurs de marée, certaines remontant à plusieurs siècles. Actuellement les hauteurs sont relevées toutes les minutes sur 63 sites d'observation en France métropolitaine.

On envisage ici une base de données simplifiée qui ne contient les données que pour l'année 2013 et pour des hauteurs d'eau relevées toutes les 10 minutes, sur 63 sites dont celui de Brest.

Table `station` (extrait)

id	nom	latitude	longitude
1	DUNKERQUE	51.048	2.366
2	CALAIS	50.969	1.868
3	BOULOGNE-SUR-MER	50.727	1.578
...			
7	SAINT-MALO	48.641	-2.028
8	ROSCOFF	48.718	-3.966
9	LE CONQUET	48.359	-4.781
10	BREST	48.383	-4.495
...			

Table hauteurs (extrait)			
idStation	date	heure	hauteur
10	01/01/2013	00:00:00	2.0
10	01/01/2013	00:10:00	1.995
10	01/01/2013	00:20:00	1.999
10	01/01/2013	00:30:00	2.029
10	01/01/2013	00:40:00	2.105
10	01/01/2013	00:50:00	2.145
...			
10	31/12/2013	23:40:00	3.505
10	31/12/2013	23:50:00	3.687
7	01/01/2013	00:00:00	5.665
7	01/01/2013	00:10:00	5.381
7	01/01/2013	00:20:00	5.072
...			

La table `station` *contient :*

— *un identifiant propre à chaque site d'observation, de type entier (la numérotation suit l'ordre géographique le long des côtes, en commençant par Dunkerque et en terminant par Monaco)*

— *le nom du site d'observation, de type chaine de caractères*

— *la latitude et la longitude du site, de type flottant*

La table `hauteurs` *contient :*

— *l'identifiant du site d'observation (entier)*

— *la date et l'heure de la mesure (chaines de caractères)*

— *la hauteur d'eau relevée (flottant)*

F1. Donner un choix de clé primaire possible pour la table `station` ? Peut-on en définir une pour la table `hauteurs` ?

F2. Combien de lignes contient chacune des tables ?

F3. Ecrire en langage SQL les requêtes pour obtenir :

1. la latitude et la longitude de Saint-Malo
2. les sites d'observation situés à l'ouest du méridien de Greenwich
3. la hauteur d'eau à Brest le 6 avril 2013 à 14h00

G / REPRESENTATIONS GRAPHIQUES

A partir de la base de données précédente, on peut récupérer dans une liste l'ensemble des hauteurs de marée à Brest pour l'année 2013, accompagnées de la date et de l'heure. On utilise pour cela une variable `data` *qui a la structure d'une liste de listes, chaque sous-liste correspondant à un triplet* `[date,heure,hauteur]`. *Pour illustrer les choses, voici le début de* `data` *:*

```
[['01/01/2013', '00:00:00', 2.0],
['01/01/2013', '00:10:00', 1.995],
['01/01/2013', '00:20:00', 1.999],
['01/01/2013', '00:30:00', 2.029],
['01/01/2013', '00:40:00', 2.105], ...]
```

Dans cette partie on va exploiter `data` *dans le but d'obtenir les représentations graphiques suivantes (figure 6) :*

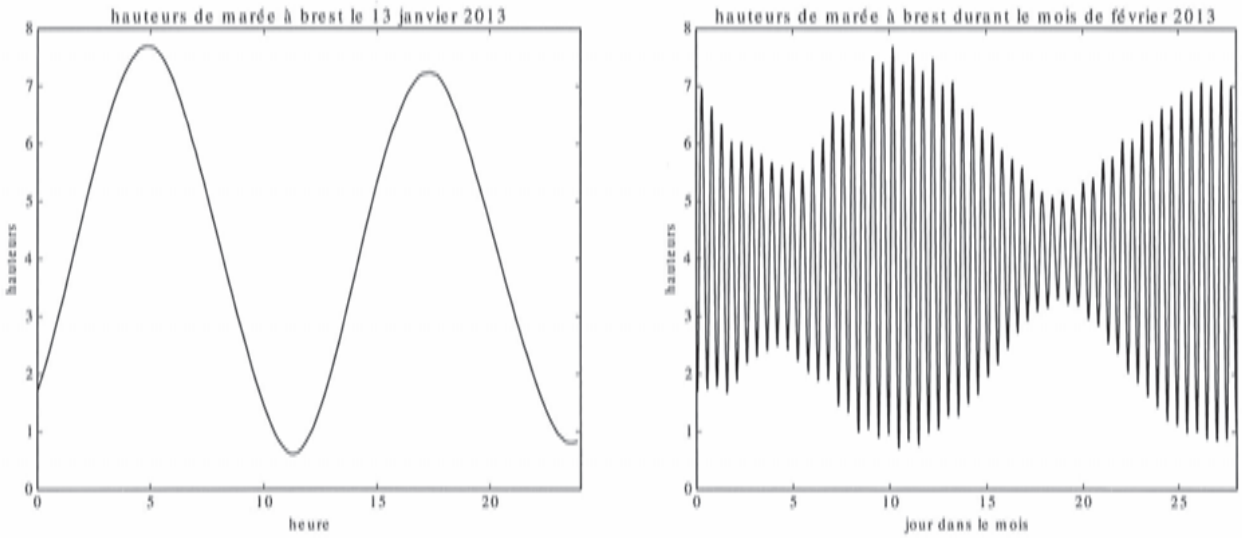


FIGURE 6

G1. Que renvoie `data[2][1]` ?

G2. Ecrire le code Python permettant de récupérer la liste des hauteurs (et uniquement les hauteurs) dans une variable (de type liste) `height`.

G3. Écrire le code Python destiné à récupérer les extraits de `height` nécessaires pour construire les deux représentations graphiques présentées ci-dessus (on les nommera respectivement `h_jan13` et `h_fev`).

G4. Écrire le code Python permettant de créer une liste de flottants `dates1` contenant le même nombre d'éléments que `h_jan13` et qui contient les instants correspondant aux différentes hauteurs, exprimés en heures. Faire de même en créant une liste de flottants `dates2` associée à `h_fev` en exprimant les instants en jours. Comment obtenir, à partir des différentes listes créées, les représentations graphiques de la figure 6 en utilisant la fonction `plot` du module `matplotlib` (on ne se préoccupera pas de l'importation de ce module, ni des légendes) ?

H / PERIODE ET AMPLITUDE DES MAREES

On se propose dans cette partie d'extraire de la liste `height` *diverses informations, le but final étant d'obtenir la période moyenne des variations de hauteur ainsi qu'une liste de leurs amplitudes (l'amplitude de la marée, égale à la différence de hauteur entre une pleine mer et une basse mer successives, est aussi appelée* marnage*).*

H1. Ecrire une fonction `minimum` qui prend en argument une liste de nombres et qui renvoie la plus petite valeur contenue dans la liste. Décrire son fonctionnement (préciser les valeurs successives prises par la (les) variable(s) interne(s) à la fonction) sur l'entrée [5,6,3,4,3,8].

H2. Modifier cette fonction (en une fonction que l'on appellera `min2`) pour qu'elle renvoie la position dans la liste de la plus petite valeur. Comment utiliser cette nouvelle fonction, avec les listes `data` et `height` pour obtenir le jour et l'heure de la plus petite hauteur d'eau à Brest lors de l'année 2013 ?

H3. Écrire une fonction `moyenne` qui prend en argument une liste de nombres et renvoie la moyenne de ses éléments.

H4. On propose la fonction «inconnue» suivante :

```
def fonction(liste) :

    m = moyenne(liste)
    e = (m-minimum(liste))/10

    val = []
    i = 0
    while i < len(liste) :
        if liste[i] > m-e and liste[i] < m+e :
            a = i
            while liste[i] > m-e and liste[i] < m+e :
                i += 1
            b = i-1
            val.append((a+b)/2)
        i += 1

    s = 0
    n = len(val)
    for i in range(n-1) :
        s += val[i+1]-val[i]

    return 2*s/(n-1)
```

Expliquer son fonctionnement. Que renvoie-t-elle appliquée à `height` ?

H5. Écrire une fonction qui permette d'obtenir la liste des amplitudes (marnages) à partir de la liste `height`. On pourra utiliser les fonctions précédentes, ainsi qu'une fonction `maximum` qui prend en arguments une liste de nombres et renvoie la plus grande valeur.

On obtient à partir de cette liste la seconde courbe de la figure 7 (pour les trois premiers mois de l'année), à comparer avec la courbe des hauteurs de marée de la même figure.

La figure 7 est destinée à illustrer ce qui précède, et ne donne lieu à aucune question.

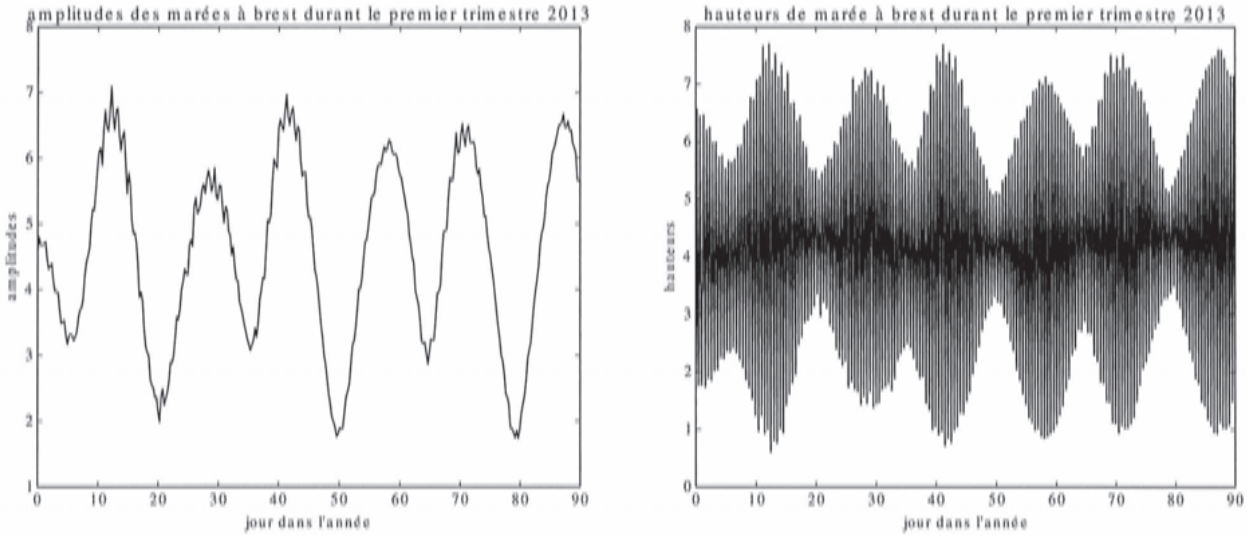


FIGURE 7

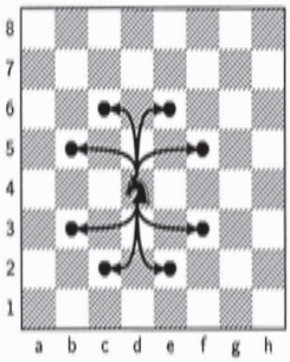
Fin de l'épreuve...

Le sujet est constitué de quatre exercices qui testent plusieurs compétences complémentaires des programmes de mathématiques et d'« informatique pour tous » de la filière PSI.
Il est donc demandé au candidat de répartir équitablement son travail sur les quatre exercices proposés. Il en sera tenu compte dans l'évaluation de l'épreuve.

Exercice 1.

But de l'exercice

Le jeu d'échec se joue sur un échiquier, c'est à dire sur un plateau de 8×8 cases. Ces cases sont référencées de a1 à h8 (cf. figure).
Une pièce, appelée le cavalier, se déplace suivant un "L" imaginaire d'une longueur de deux cases et d'une largeur d'une case.
Exemple : un cavalier situé sur la case d4 atteint, en un seul déplacement, une des huit cases b5, c6, e6, f5, f3, e2, c2 et b3 (voir figure ci-contre).



Dans toute la suite de l'exercice, on appellera **case permise** toute case que le cavalier peut atteindre en un déplacement à partir de sa position.

Le but de cet exercice est d'écrire un programme faisant parcourir l'ensemble de l'échiquier à un cavalier **en ne passant sur chaque case qu'une et une seule fois**.

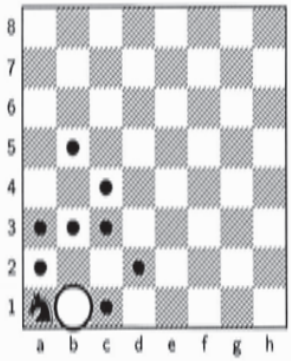
Motivation et méthode retenue

Une première idée est de faire parcourir toutes les cases possibles à un cavalier en listant à chaque déplacement les cases parcourues. Lorsque celui-ci ne peut plus avancer on consulte le nombre de cases parcourues.

- Si ce nombre est égal à $64 = 8 \times 8$, alors le problème est résolu.
- Sinon, il faut revenir en arrière et tester d'autres chemins.

0. Exemple : on considère le parcours suivant d'un cavalier démarrant en a1 (figure ci contre) :

a1, b3, c1, a2, c3, b5, a3, c4, d2.



Avec ce début de parcours, au déplacement suivant :

a. le cavalier va en b1. Peut-il accomplir sa mission ?

b. le cavalier ne va pas en b1. Peut-il accomplir sa mission ?

Il convient donc dans la résolution du problème proposé d'éviter de se retrouver dans la situation repérée à la question **0.**

Dans tout ce qui suit, nous nommerons **coordonnées** d'une case la liste d'entiers $[i, j]$ où i représente le numéro de ligne et j le numéro de colonne (tous deux compris entre 0 et 7). Par exemple, la case b3 a pour **coordonnées** $[2, 1]$.

D'autre part, les cases sont numérotées de 0 à 63 en partant du coin gauche comme indiqué par la figure ci-contre.

Nous appellerons **indice** d'une case le numéro n compris entre 0 et 63 ainsi déterminé. Ainsi la case b3 a pour **indice** 17.

8	56	57	58	59	60	61	62	63
7	48	49	50	51	52	53	54	55
6	40	41	42	43	44	45	46	47
5	32	33	34	35	36	37	38	39
4	24	25	26	27	28	29	30	31
3	16	17	18	19	20	21	22	23
2	8	9	10	11	12	13	14	15
1	0	1	2	3	4	5	6	7
	a	b	c	d	e	f	g	h

Les questions

1. Écrire une fonction `Indice` qui aux **coordonnées** $[i, j]$ d'une case renvoie son **indice**.
Ainsi, `Indice` appliquée à $[2, 1]$ doit renvoyer 17.

2. Écrire une fonction `Coord` qui, à l'**indice** n d'une case renvoie la liste $[i, j]$ de ses **coordonnées**.
Ainsi `Coord` appliquée à 17 doit renvoyer $[2, 1]$.

3. On considère la fonction Python `CasA` suivante :

```
def CasA(n):
    Deplacements = [[1, -2], [2, -1], [2, 1], [1, 2], [-1, 2], [-2, 1], [-2, -1], [-1, -2]]
    L = []
    i, j = Coord(n)
    for d in Deplacements:
        u = i +d[0]
        v = j +d[1]
        if u >=0 and u <8 and v >=0 and v < 8:
            L.append(Indice([u, v]))
    return L
```

3.1 Que renvoient `CasA(0)` et `CasA(39)` ?

3.2 Expliquer **en une phrase** ce que fait cette fonction.

4. Écrire une fonction `Init` ne prenant aucun argument et qui modifie deux variables globales `ListeCA` et `ListeCoups`.
`ListeCoups` recevra la liste vide [].
`ListeCA` recevra une liste de 64 éléments. Chaque élément `ListeCA[n]` (pour $0 \leqslant n \leqslant 63$) devra contenir la liste des **indices** des cases qu'un cavalier peut atteindre en un coup à partir de la case d'indice n.

5. Après exécution de la fonction `Init()`, la commande `>>> ListeCA[0]` renvoie :

a - $[5]$
b - $[10, 17]$
c - $[10, 17, 0]$
d - $[17, 0, 10]$
e - []
f - Elle renvoie une autre valeur.

6. Au cours de la recherche, lorsqu'on déplace le cavalier vers la case d'indice n, cet indice n doit être retiré de la liste des **cases permises** à partir de la position n.

Exemple :
Après exécution de la fonction `Init()`, la liste des cases permises depuis b1 est $[a3, c3, d2]$, et `ListeCA[1]` vaut $[16, 18, 11]$.
La liste des cases permises depuis a3 est $[b5, c4, c2, b1]$ et `ListeCA[16]` vaut $[33, 26, 10, 1]$.
Puis on choisit de commencer le parcours en posant le cavalier en b1. Cette case doit donc être retirée de la liste des cases permises de a3, c3 et d2.
En particulier pour a3, la liste `ListeCA[16]` devient : $[33, 26, 10]$.

Cette méthode nous permet de détecter les blocages :
Le cavalier arrive sur la case d'indice n : n est alors retiré de toutes les listes `ListeCA[k]` pour toute case d'indice k permise pour n.
Si dès lors l'une de ces listes devient vide, nous dirons alors que nous sommes dans une **situation critique**, cela signifiera que la case d'indice k ne peut plus être atteinte que depuis la case d'indice n. Par conséquent :

- si le cavalier se déplace sur une autre case que celle d'indice k, alors cette dernière ne pourra plus jamais être atteinte ;
- si le cavalier se déplace sur la case d'indice k, le cavalier est bloqué pour le coup suivant. Et :
  - soit la mission est accomplie,
  - soit le cavalier n'a pas parcouru toutes les cases..

Le programme va réaliser la recherche en maintenant à jour la variable globale `ListeCoups` afin qu'elle contienne en permanence la liste des positions successives occupées par le cavalier au cours de ses tentatives de déplacement.
Nous avons alors besoin d'écrire trois fonctions :

6.1 Écrire une fonction `OccupePosition` qui :

- prend comme argument un entier n (indice d'une case), l'ajoute à la fin de la variable globale `ListeCoups`,
- puis enlève n de toutes les listes `ListeCA[k]` pour toutes les cases d'indice k permises depuis la case d'indice n,
- renvoie enfin la valeur `True`, si nous sommes dans une situation critique et `False` sinon.

On pourra utiliser la méthode `remove` *qui permet de retirer d'une liste le premier élément égal à l'argument fourni. Si l'argument ne fait pas partie de la liste, une erreur sera retournée.*

```
L = [1, 2, 3, 4, 2, 5]
L.remove(2)
L
```

Les commandes précédentes renvoient la liste $[1,3,4,2,5]$.

```
L = [1, 2, 3, 4, 2, 5]
L.remove(6)
```

Les commandes précédentes provoquent une erreur.

6.2 Écrire une fonction `LiberePosition` qui ne prend pas d'argument et qui

- récupère le dernier élément n de la variable globale `ListeCoups` (i.e. n est l'indice de la dernière case jouée à l'aide de la fonction `OccupePosition(n)`),
- puis l'enlève de `ListeCoups`,
- et enfin, qui ajoute n à toutes les listes `ListeCA[k]` pour toutes les cases d'indice k permises depuis la case d'indice n.

À la fin de cette fonction les listes `ListeCoups` et `ListeCA` seront donc dans le même état qu'avant l'exécution de la fonction `OccupePosition(n)`.

On pourra utiliser la méthode `pop` *qui renvoie le dernier élément d'une liste et le supprime de cette même liste.*

```
L = [1, 2, 3, 4, 2, 5, 2]
n = L.pop()
```

Les commandes précédentes affectent la valeur 2 à la variable n, la liste L étant ensuite : $[1,2,3,4,2,5]$.

6.3 Écrire une fonction `TestePosition` d'argument un entier n (indice d'une case) qui :

- occupe la position d'indice n.
- vérifie si la situation est critique.

  Si tel est le cas,
  - la fonction vérifiera si 63 cases sont occupées et dans ce cas renverra `True` pour indiquer que la recherche est terminée.
  - Si les 63 cases ne sont pas occupées, la fonction libérera la case d'indice n et renverra `False`.

  Dans le cas contraire,
  - la fonction vérifiera, à l'aide de la fonction `TestePosition(k)`, toutes les cases d'indice k jouables après la case d'indice n. On prendra garde à affecter une variable locale avec la liste `ListeCA[n]` puisque celle-ci risque d'être modifiée lors des appels suivants.
  - La fonction retournera `True` dès qu'un appel à `TestePosition(k)` retourne `True` ou libérera la case d'indice n et retournera `False` si tous les appels à `TestePosition(k)` retournent `False`.

7. Afin de réduire notablement la complexité temporelle du programme on part du principe qu'il faut tester en priorité les cases ayant le moins de cases permises possible. On appellera **valuation** d'une case d'indice n le nombre de cases permises pour cette case.

7.1 Écrire une fonction `valuation` qui prend comme argument un indice n de case en entrée et renvoie la valuation de cette case.

7.2 Écrire une fonction `Fusion` qui prend comme argument deux listes `A` et `B` constituées chacune d'entiers naturels entre 0 et 63 (`A` et `B` sont donc des listes d'indice de cases) ; on suppose que ces listes, `A` et `B` sont triées par ordre croissant de valuation de leurs éléments ; la fonction `Fusion(A,B)` retourne alors comme valeur la liste fusionnée de tous les éléments de `A` et `B` triée par ordre croissant de valuation de ses éléments.

7.3 Écrire une fonction `TriFusion` qui prend comme argument une liste `L` d'entiers compris entre 0 et 63, a priori non supposée triée par valuation croissante de ses éléments, et qui retourne comme valeur la liste de tous les éléments de `L` triée par valuation croissante de ses éléments.

7.4 Modifier la fonction `TestePosition` pour qu'elle agisse ainsi que l'on a décidé en début de question.

Exercice 2.

1. Soit $A = \begin{pmatrix} 0 & 1 \\ y-4 & 2x \end{pmatrix} \in \mathscr{M}_2(\mathbb{R})$.

Déterminer une condition nécessaire et suffisante pour que la matrice A soit diagonalisable dans $\mathscr{M}_2(\mathbb{R})$.

2. On note $E_1 = \{u \in \mathbb{R}_+,\ u^2 \notin \mathbb{N}\}$ et E_2 son complémentaire dans $\mathbb{R}_+$.

Prouver que E_2 est un ensemble dénombrable.

3. Soient $(\Omega, \mathscr{A})$ un espace probabilisable et f l'application de $\mathbb{R}_+$ dans $\mathbb{R}$ définie par :

$$\forall u \in \mathbb{R}_+,\quad f(u) = \begin{cases} 0 \text{ si } u^2 \notin \mathbb{N} \\ \dfrac{\lambda}{2^{u^2}} \text{ sinon} \end{cases}$$

Déterminer $\lambda \in \mathbb{R}$ pour qu'il existe une probabilité $\mathbb{P}$ tel que f soit la loi de probabilité d'une variable aléatoire X définie sur Ω et à valeurs dans $\mathbb{R}_+$.

Préciser $X(\Omega)$.

4. Déterminer $X^2(\Omega)$ et la loi de probabilité de X^2.

5. Déterminer l'espérance $\mathrm{E}(X^2)$ de la variable aléatoire X^2.

6. Déterminer la fonction génératrice de la variable aléatoire X^2.

Retrouver alors la valeur de $\mathrm{E}(X^2)$ obtenue à la question précédente.

7. Soit Y une variable aléatoire définie sur Ω, indépendante de la variable aléatoire X et suivant la loi :

$$\forall u \in \mathbb{R}_+,\ \mathbb{P}(Y = u) = \begin{cases} 0 \text{ si } u \notin \mathbb{N} \\ \dfrac{1}{2^{u+1}} \text{ sinon} \end{cases}$$

Soit alors Z la variable aléatoire définie sur Ω par : $Z = X^2 + Y$.

Déterminer la fonction génératrice de Z. En déduire sa loi de probabilité.

8. Déterminer enfin la probabilité pour que la matrice $A = \begin{pmatrix} 0 & 1 \\ Y-4 & 2X \end{pmatrix} \in \mathscr{M}_2(\mathbb{R})$ soit diagonalisable.

Exercice 3. On pose, lorsque cela est possible, $f(x) = \displaystyle\int_1^{+\infty} \frac{\mathrm{d}t}{t^x\sqrt{t^2-1}}$.

1. Déterminer l'ensemble de définition I de f.

2. En justifiant son existence, calculer $\displaystyle\int_0^{+\infty} \frac{\mathrm{d}x}{\mathrm{e}^x+\mathrm{e}^{-x}}$.

3. Calculer $f(1)$. (On pourra utiliser l'application $\varphi : u \in \mathbb{R}_+^* \longmapsto \varphi(u) = \operatorname{ch}(u)$).

4. Calculer $f(2)$. (On pourra remarquer que la dérivée de $x \longmapsto \dfrac{\operatorname{sh}(x)}{\operatorname{ch}(x)}$ est égale à $x \longmapsto \dfrac{1}{\operatorname{ch}^2(x)}$)

5. Vérifier que f est positive sur I.

6. Montrer que f est décroissante sur I.

7. Prouver que f est de classe C^1 sur I et préciser l'expression de $f'(x)$.

Retrouver alors le résultat de la question précédente.

8. Soit $x \in I$. Démontrer la relation :

$$f(x+2) = \frac{x}{x+1} f(x)$$

On pourra effectuer, en la justifiant, une intégration par parties.

9. Soit $p \in \mathbb{N}^*$. Donner l'expression de $f(2p)$ à l'aide de factorielles.

10. Pour tout réel x strictement positif, on pose :

$$\phi(x) = x\, f(x)\, f(x+1)$$

Prouver que $\phi(x+1) = \phi(x)$.

Calculer $\phi(n)$ pour tout entier naturel n non nul.

11. En utilisant la question précédente, déterminer un équivalent simple de $f(x)$ lorsque x tend vers 0 par valeurs supérieures.

12. Vérifier que : $\forall n \in \mathbb{N}^*,\ f(n)\, f(n+1) = \dfrac{\pi}{2n}$.

En déduire que $\forall n \in \mathbb{N}^*,\ f(n) \underset{n\to+\infty}{\sim} \sqrt{\dfrac{\pi}{2n}}$.

13. En utilisant des parties entières, prouver que : $f(x) \underset{x\to+\infty}{\sim} \sqrt{\dfrac{\pi}{2x}}$

14. Déduire des questions précédentes le tableau des variations de f sur I et tracer sa courbe représentative dans un repère orthonormé.

15. Prouver que la fonction ϕ est constante sur $\mathbb{R}_+^*$.

Exercice 4.

Dans tout l'exercice, pour tout entier naturel k, on identifie polynôme de $\mathbb{R}_k[X]$ et fonction polynomiale associée pour la structure d'espace vectoriel normé.

1. Soit P un élément de $\mathbb{R}[X]$ unitaire (le terme de plus haut degré de P est égal à 1).

1.1 Soit $\alpha \in \mathbb{R}$.

Montrer que : $\forall z \in \mathbb{C}, |z - \alpha| \geqslant |\mathrm{Im}(z)|$.

1.2 On suppose dans cette question que P est scindé sur $\mathbb{R}$.

En utilisant une factorisation de P, montrer que :

$$\forall z \in \mathbb{C}, \quad |P(z)| \geqslant |\mathrm{Im}(z)|^{\deg(P)}$$

où $\deg(P)$ désigne le degré du polynôme P.

1.3 On prend dans cette question $P(X) = X^3 + 1$.

(a) Donner une factorisation de P dans $\mathbb{C}[X]$.

(b) Trouver $z_0 \in \mathbb{C}$ tel que : $|P(z_0)| < |\mathrm{Im}(z_0)|^{\deg(P)}$

1.4 On suppose dans cette question que : $\forall z \in \mathbb{C}, \quad |P(z)| \geqslant |\mathrm{Im}(z)|^{\deg(P)}$

Montrer que toutes les racines de P sont réelles. En déduire que P est scindé sur $\mathbb{R}$.

1.5 Enoncer clairement le résultat obtenu.

2. Soient q un entier naturel non nul et $(A_n)_{n\in\mathbb{N}}$ une suite de matrices trigonalisables de $\mathscr{M}_q(\mathbb{R})$ qui converge vers une matrice A.

On appelle pour tout entier naturel n, P_n le polynôme caractéristique de A_n et P celui de la matrice A.

2.1 Donner le degré et le coefficient dominant de P_n.

2.2 Prouver que : $\forall x \in \mathbb{R}, \lim\limits_{n\to+\infty} P_n(x) = P(x)$.

2.3 En déduire que A est trigonalisable.

2.4 Qu'en conclut-on pour l'ensemble des matrices trigonalisables de $\mathscr{M}_q(\mathbb{R})$?

3. On prend dans cette question $q = 2$ et $A_n = \begin{pmatrix} 1 - \dfrac{1}{n} & 1 - \dfrac{\sin n}{n} \\ 0 & 1 + \dfrac{1}{n} \end{pmatrix}$. Où n est un entier non nul.

3.1 Déterminer $A = \lim\limits_{n\to+\infty} A_n$.

3.2 Etudier la diagonalisabilité des matrices A_n et A dans $\mathscr{M}_2(\mathbb{R})$.

3.3 Conclure.

024

CONCOURS ARTS ET MÉTIERS ParisTech - ESTP - POLYTECH

Épreuve de Physique - Modélisation PSI

Durée 3 h

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, d'une part il le signale au chef de salle, d'autre part il le signale sur sa copie et poursuit sa composition en indiquant les raisons des initiatives qu'il est amené à prendre.

L'usage de calculatrices est autorisé.

AVERTISSEMENT

Les parties A, B, C, D, et les parties E, F, G sont à rédiger sur copies séparées.

Remarques importantes : il est rappelé aux candidat(e)s que

- Il faudra exclusivement utiliser les notations de l'énoncé.
- Les explications des phénomènes étudiés interviennent dans la notation au même titre que les développements analytiques et les applications numériques (données avec un nombre de chiffres significatifs adapté) ; les résultats exprimés sans unité ne seront pas comptabilisés (S.I. n'est pas une unité).
- Tout au long de l'énoncé, les paragraphes en italiques ont pour objet d'aider à la compréhension du problème.
- Tout résultat fourni dans l'énoncé peut être admis et utilisé par la suite, même s'il n'a pas été démontré par le(la) candidat(e).
- **Un document réponse est à rendre non plié avec la copie de la deuxième partie.**

Ce problème traite de systèmes d'identification par radio fréquence (RFID). Aucune connaissance particulière sur les antennes n'est demandée.

*La **radio-identification**, le plus souvent désignée par le sigle **RFID** (de l'anglais radio frequency identification), est une méthode pour mémoriser et récupérer des données à distance en utilisant des marqueurs appelés « radio-étiquettes » (« RFID tag » ou « RFID transponder » en anglais).*
Les radio-étiquettes peuvent être des étiquettes autoadhésives, pouvant être collées sur des objets. Les radio-étiquettes comprennent une antenne associée à une puce électronique qui leur permet de recevoir et de répondre aux requêtes radio émises (signal radiofréquence) depuis un émetteur-récepteur.
Cette puce électronique contient un identifiant et éventuellement des données complémentaires.
Les puces RFID tentent aujourd'hui de supplanter les codes à barres en jouant de leurs avantages, à savoir qu'il est possible d'écrire, d'effacer et de réécrire les données stockées dans la puce un grand nombre de fois, que leur portée peut être supérieure aux lecteurs optiques utilisés pour les codes à barres, et que la communication peut se faire à travers certains obstacles contrairement aux systèmes à lecture optique.

Un système RFID passif est composé de deux entités qui communiquent entre elles (Fig. 0) :

- *Un TAG passif (dénommé TAG par la suite) ou radio-étiquette, associé à l'élément à identifier. Il est capable de répondre à une demande venant du système émetteur - récepteur. Le TAG n'a pas d'alimentation de type batterie ou pile mais est autoalimenté par l'onde électromagnétique reçue (il existe cependant des TAG actifs alimentés par pile).*
- *Une station de base ou lecteur RFID qui a pour mission d'identifier le TAG. Le lecteur envoie une onde électromagnétique en direction de l'élément à identifier, cette onde alimente le TAG qui peut alors communiquer avec le générateur - lecteur grâce à sa puce électronique interne. En retour, le générateur - lecteur reçoit l'information renvoyée par le TAG.*

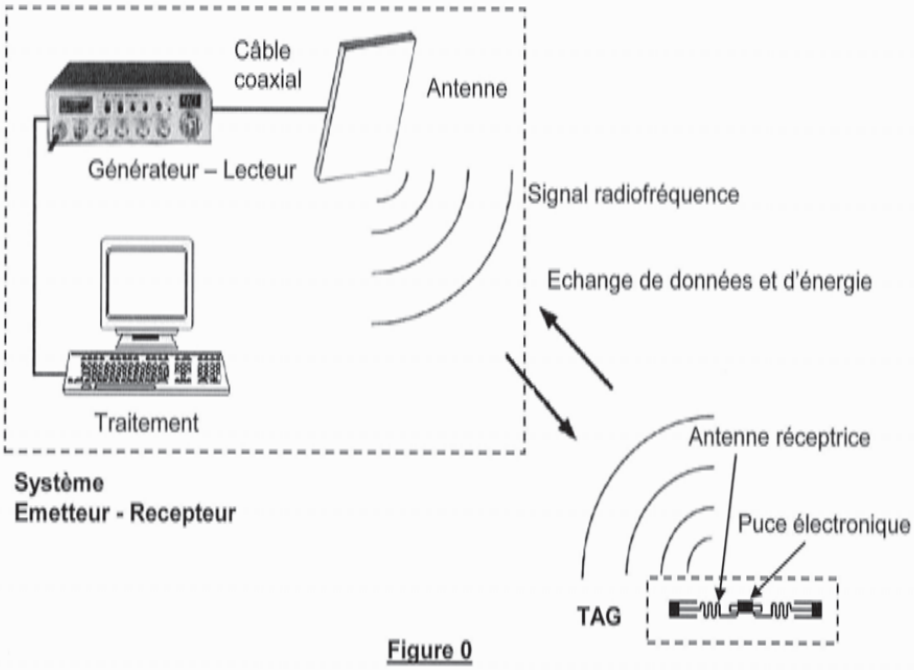


Figure 0

La figure 0 présente le fonctionnement général d'un système RFID. Le générateur - lecteur relié à une antenne émettrice agit en maître par rapport au TAG : si le TAG est dans la zone de lecture du générateur - lecteur, ce dernier l'active en lui envoyant une onde électromagnétique et entame la communication. Le TAG est quant à lui, constitué d'une antenne et d'une puce électronique qui

module l'onde réémise vers le générateur - lecteur. En démodulant le signal reçu, le générateur - lecteur relié à une application interne récupère l'information pour la traiter, il est chargé de l'interface et de la gestion de l'identification des TAGs qui se présentent à lui.
Il existe plusieurs familles de systèmes RFID dont le principal critère de différentiation est la fréquence de fonctionnement. Les système RFID utilisent des bandes de fréquence diverses allant de 125 kHz à 2,45 GHz. Le sujet étudie quelques aspects d'un système RFID à 860 MHz. Les parties **A**, **B**, **C**, **D**, **E**, **F** *et* **G** *de ce sujet sont largement indépendantes les unes des autres.*

Question préliminaire. Quel peut être l'avantage majeur d'un système RFID actif par rapport à un système RFID passif ?

Partie I : Système RFID à 860MHz

A Adaptation d'impédance

La notation complexe est utilisée et à $x(t) = X_{eff}\sqrt{2}\cos(\omega t + \varphi)$ *est associé la grandeur complexe* $\underline{X} = X_{eff}\sqrt{2}e^{j(\omega t+\varphi)}$.
Un générateur de tension, de fem interne $\underline{E_G}$ *et d'impédance interne* $\underline{Z_G} = R_G + jX_G$ *est connecté à une impédance de charge variable* $\underline{Z_U} = R_U + jX_U$ *(Fig. 1). La fréquence de fonctionnement est f = 860MHz. On note* P_{E_G} *la puissance moyenne délivrée par la fem* $\underline{E_G}$, P_{Z_G} *la puissance moyenne dissipée dans* $\underline{Z_G}$ *et* P_{Z_U} *la puissance moyenne reçue par* $\underline{Z_U}$.

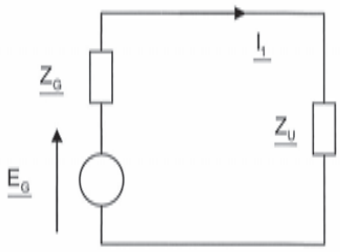


Figure 1

A1. Déterminer le courant $\underline{I_1}$ débité par le générateur, puis déterminer l'expression de P_{Z_U} en fonction de $|\underline{E_G}|$, R_G, X_G, R_U *et* X_U.

A2. Pour $\underline{Z_G}$ et $\underline{E_G}$ fixés, montrer que P_{Z_U} est maximale quand l'impédance de la charge vérifie : $R_U + jX_U = R_G - jX_G$. Dans ce cas on dit que l'impédance de charge est adaptée à celle de la source (adaptation en puissance). Dans ces conditions, que valent les puissances P_{Z_U}, P_{E_G} et P_{Z_G} ?

On insère entre le générateur et la charge les deux composants de réactance X_1 et X_2 *(Fig. 2). La condition d'adaptation n'est plus réalisée et maintenant,* $\underline{Z_G} = R_G$ *et* $X_G = 0$ *ce qui correspond à un générateur standard; on a encore* $\underline{Z_U} = R_U + jX_U$.
Pour les applications numériques prendre $R_G = 50{,}0\,\Omega$, $R_U = 73{,}2\,\Omega$ et $X_U = 42{,}5\,\Omega$. *On appelle* $\underline{Z_{IN}} = R_{IN} + jX_{IN}$ *l'impédance équivalente comprenant* jX_1, jX_2 *et* $\underline{Z_U}$.

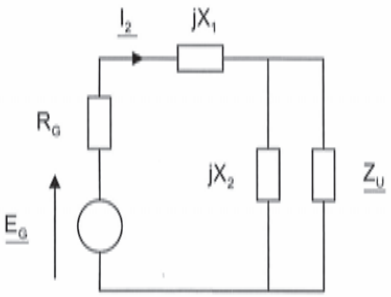


Figure 2

A3. Montrer qualitativement que les deux réactances X_1 et X_2 modifient l'impédance de la charge vue du générateur et permettent éventuellement une adaptation entre le générateur et le reste du montage.
Application numérique : on a tracé sur la figure 3 les lieux des points de coordonnées (X_1, X_2) donnant d'une part $X_{IN} = 0\,\Omega$ et $R_{IN} = 50{,}0\,\Omega$ d'autre part. En analysant cette figure, déterminer chaque couple de (X_1, X_2) assurant l'adaptation d'impédance ; donner le type de composant correspondant (bobine ou condensateur) et sa caractéristique (inductance ou capacité). Quel est le meilleur choix ? Expliquer.

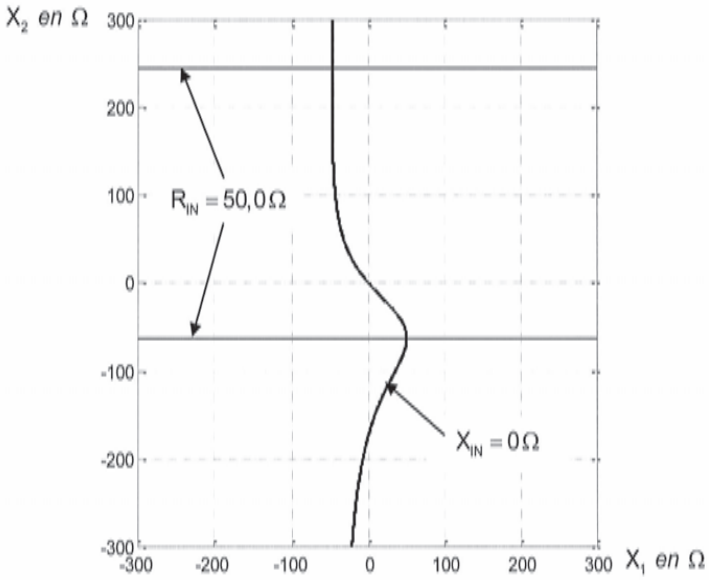


Figure 3

Le montage de la figure 2 ne convient pour réaliser l'adaptation que dans la situation où $R_G < R_U$.

A4. Sans calcul et en invoquant des propriétés de symétrie, proposer une modification simple de la figure 2 pour permettre une adaptation d'impédance dans le cas où $R_G > R_U$.

B influence du câble coaxial

On étudie un câble coaxial d'axe (Ox), *de longueur* ℓ_L *intercalé entre le générateur et la charge équivalente adaptée en puissance de la figure 2 (cf.* **A2.**). *Le schéma équivalent est donné par la figure 4 où* e_G *est la fem du générateur.*

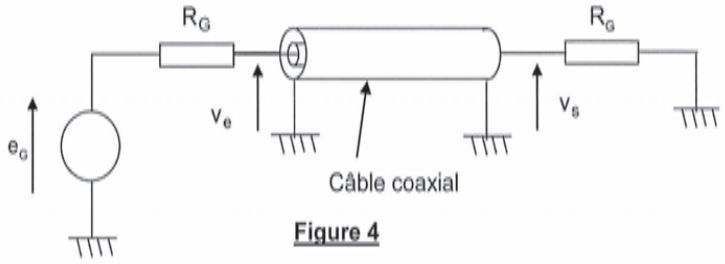


Figure 4

On admet que la portion de câble comprise entre les plans de section droite repérés par x *et par* $x + dx$ *peut être modélisée par le circuit de la figure 5. Elle possède une capacité* Γdx *entre les conducteurs et une inductance propre* Λdx *entre les sections d'entrée et de sortie. La tension*

entre l'âme et la gaine du câble s'écrit $v(x,t)$ *et le courant traversant une section de l'âme est* $i(x,t)$.

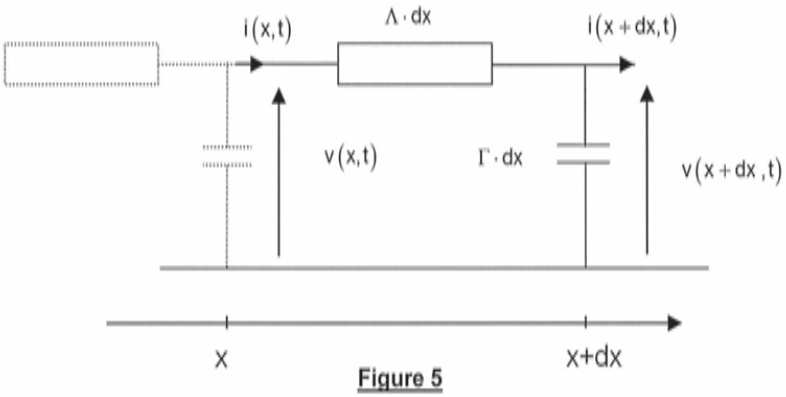


Figure 5

B1. Démontrer les relations suivantes : $\dfrac{\partial v}{\partial x} = -\Lambda \dfrac{\partial i}{\partial t}$ et $\dfrac{\partial i}{\partial x} = -\Gamma \dfrac{\partial v}{\partial t}$.

B2. En déduire l'équation de propagation vérifiée par $v(x,t)$ et $i(x,t)$. Exprimer c_L la vitesse de propagation.

On admet que les solutions générales sont de la forme : $v(x,t) = v_1(t - x/c_L) + v_2(t + x/c_L)$ et $i(x,t) = i_1(t - x/c_L) + i_2(t + x/c_L)$.

B3. Interpréter les significations physiques des grandeurs d'indice 1 et 2. Illustrer avec un graphique et expliquer.

Le générateur de tension (Fig. 4) de fem interne e_G *et de résistance interne* R_G *impose un régime sinusoïdal forcé de pulsation* ω *dans le câble coaxial. Les grandeurs* v_1, v_2, i_1 *et* i_2 *précédentes deviennent des fonctions sinusoïdales dépendant de* $\omega(t - x/c_L)$ *pour les grandeurs d'indice 1 et de* $\omega(t + x/c_L)$ *pour les grandeurs d'indice 2. On pose :* $k_L = \dfrac{\omega}{c_L}$.

B4. En utilisant la notation complexe, montrer que : $v_1(t - x/c_L) = R_C \cdot i_1(t - x/c_L)$ et $v_2(t + x/c_L) = -R_C \cdot i_2(t + x/c_L)$ où R_C est la résistance caractéristique du câble. Exprimer R_C en fonction de Γ et de Λ. Vérifier l'homogénéité.

La résistance équivalente R_G *est branchée en sortie du câble. Les grandeurs à l'entrée du câble sont notées avec l'indice e et les grandeurs à la sortie du câble avec l'indice s de sorte que :* $v_e(t) = v(0,t)$, $v_s(t) = v(\ell_L,t)$, $i_e(t) = i(0,t)$ *et* $i_s(t) = i(\ell_L,t)$. Avec *la notation complexe:*

$\underline{v_1} = \underline{V_{10}} \cdot e^{j(\omega t - k_L x)}$, $\underline{v_2} = \underline{V_{20}} \cdot e^{j(\omega t + k_L x)}$, $\underline{i_1} = \left(\underline{V_{10}}/R_C\right) \cdot e^{j(\omega t - k_L x)}$, $\underline{i_2} = -\left(\underline{V_{20}}/R_C\right) \cdot e^{j(\omega t + k_L x)}$

où $\underline{V_{10}}$ et $\underline{V_{20}}$ *sont des nombres complexes constants. On utilisera les amplitudes complexes* $\underline{V}(x)$ *et* $\underline{I}(x)$ *associées à* $v(x,t)$ *et* $i(x,t)$ *telles que :*

$$\begin{cases} \underline{V}(x) = \underline{V_{10}} \cdot e^{-jk_L x} + \underline{V_{20}} \cdot e^{jk_L x} \\ \underline{I}(x) = \left(\underline{V_{10}}/R_C\right) \cdot e^{-jk_L x} - \left(\underline{V_{20}}/R_C\right) \cdot e^{jk_L x} \end{cases}$$

A l'entrée du câble, on note $\underline{V}(0) = \underline{V_e}$, $\underline{I}(0) = \underline{I_e}$ *et à la sortie* $\underline{V}(\ell_L) = \underline{V_s}$, $\underline{I}(\ell_L) = \underline{I_s}$.

B5. En éliminant $\underline{V_{10}}$ et $\underline{V_{20}}$ Trouver les deux fonctions f et g telles que :

$$\begin{cases} \underline{V_e} = f(k_L\ell_L)\cdot\underline{V_s} + jR_C\cdot g(k_L\ell_L)\cdot\underline{I_s} \\ \underline{I_e} = jg(k_L\ell_L)/R_C\cdot\underline{V_s} + f(k_L\ell_L)\cdot\underline{I_s} \end{cases}$$

B6. En déduire l'impédance d'entrée $\underline{Z_e} = \dfrac{\underline{V_e}}{\underline{I_e}}$ en fonction de R_G, R_C, k_L et ℓ_L.

B7. Pour quelles valeurs particulières de ℓ_L a-t-on $\underline{Z_e} = R_G$? Interpréter et discuter les différents cas.
Application numérique : calculer la longueur ℓ_L qu'il faut donner au câble coaxial de type RG58C/U entre le générateur - lecteur et l'antenne (Fig. 0) pour qu'il ne perturbe pas la condition d'adaptation d'impédance, si la distance D entre les deux vaut trois mètres ? On donne $\Lambda = 250\,\text{nH}\cdot\text{m}^{-1}$ et $\Gamma = 100\,\text{pF}\cdot\text{m}^{-1}$.

C Antenne filaire

Une antenne a_1 (Fig. 6) est modélisée par un conducteur filiforme de longueur ℓ, centré sur O, dirigé suivant l'axe Oz de vecteur unitaire $\overrightarrow{u_z}$. L'antenne est parcourue par un courant de pulsation ω et de longueur d'onde λ, dont la répartition est de la forme : $i(z,t) = I_0(z)\sqrt{2}\cos(\omega t)$, soit $\underline{I}(z,t) = I_0(z)\sqrt{2}\exp(j\omega t)$ en complexe avec $-\ell/2 < z < \ell/2$. Le générateur d'alimentation (qui est le générateur - lecteur de la figure 0) n'est pas représenté, mais fournit à l'antenne le courant $i(0,t)$ par l'intermédiaire d'un câble coaxial. La répartition des charges sur le conducteur est donnée par la densité linéique de charges $\lambda_c(z,t)$; ($\lambda_c(z,t) = \dfrac{\delta q(z,t)}{\delta z}$ où $\delta q(z,t)$ est la charge contenue dans le tronçon de longueur δz situé à z). On se place dans le cas où $I_0(z) = I_0\cos(kz) = I_0\cos\left(\dfrac{2\pi z}{\lambda}\right)$ avec la condition $\ell = \lambda/2$ (antenne demi-onde). Dans la suite du problème c désigne *la célérité des ondes électromagnétiques dans le vide. Pour les applications numériques,* $c \approx 3\cdot10^8\,\text{m}\cdot\text{s}^{-1}$. *On rappelle également que la valeur de la perméabilité magnétique du vide vaut :* $\mu_0 = 4\pi\cdot10^{-7}\,\text{H}\cdot\text{m}^{-1}$.

C1. Rappeler les équations de Maxwell dans une région vide de charge et de courant. Peut-on se placer dans l'hypothèse de l'ARQS ? Que peut-on dire de l'intensité aux extrémités de l'antenne ? Expliquer le nom donné à cette antenne : «antenne demi-onde».
Application numérique : si $f = 860\,\text{MHz}$, calculer la longueur ℓ de l'antenne demi-onde.

C2. Etablir une condition sur ω, c et ℓ pour que $i(z,t)$ soit solution d'une équation de d'Alembert (dont les ondes se propagent à la célérité c)? Que représente I_0 ?

C3. Faire un bilan de charges sur un tronçon de conducteur infinitésimal de longueur δz pendant l'intervalle de temps δt. En déduire une relation entre une dérivée de $\lambda_c(z,t)$ et une dérivée de $i(z,t)$.

C4. Exprimer $\lambda_c(z,t)$ sachant de plus que pour tout z tel que $-\ell/2 < z < \ell/2$: $\displaystyle\int_0^{\frac{2\pi}{\omega}} \lambda_c(z,t)\,dt = 0$. Déterminer la charge q+(t) portée par la moitié supérieure de l'antenne ? Que vaut la charge totale q(t) portée par la totalité de l'antenne ?

Formulaire en coordonnées sphériques:

si $\vec{B} = B_r\vec{u_r} + B_\theta\vec{u_\theta} + B_\varphi\vec{u_\varphi}$

$$\overrightarrow{\mathrm{rot}}(\vec{B}) = \begin{cases} \dfrac{1}{r^2\sin\theta}\left(\dfrac{\partial}{\partial\theta}(B_\varphi r\sin\theta) - \dfrac{\partial}{\partial\varphi}(rB_\theta)\right) & \text{sur } \vec{u_r} \\ \dfrac{1}{r\sin\theta}\left(\dfrac{\partial}{\partial\varphi}B_r - \dfrac{\partial}{\partial r}(B_\varphi r\sin\theta)\right) & \text{sur } \vec{u_\theta} \\ \dfrac{1}{r}\left(\dfrac{\partial}{\partial r}(rB_\theta) - \dfrac{\partial}{\partial\theta}B_r\right) & \text{sur } \vec{u_\varphi} \end{cases}$$

Et : $\overrightarrow{dS} = r^2\sin\theta\cdot d\varphi\cdot d\theta\cdot\vec{u_r}$

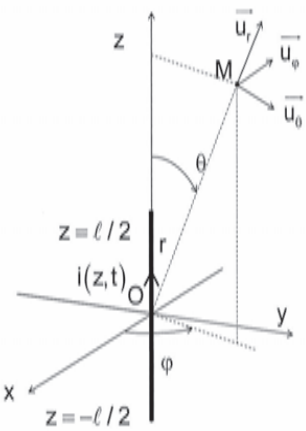


Figure 6

On admet que le champ magnétique complexe $\underline{\vec{B}}(M,t)$ *produit par l'antenne en un point M éloigné (zone de champ lointain) repéré par* $(OM = r \gg \lambda)$ *et* $\theta = (\overrightarrow{Oz}, \overrightarrow{OM})$ *est à l'instant t :*

$\underline{\vec{B}}(M,t) = \underline{B_\varphi}\vec{u_\varphi} = \dfrac{j\mu_0 I_0\sqrt{2}}{2\pi r}\dfrac{\cos\left(\dfrac{\pi}{2}\cos\theta\right)}{\sin\theta}\exp(j(\omega t - kr))\vec{u_\varphi}$ *avec* $k = \dfrac{\omega}{c}$. *Dans la suite on ne gardera dans les calculs que les termes en* $\dfrac{k}{r}$, *en négligeant les termes en* $\dfrac{1}{r^2}$.

C5. Justifier que $\underline{\vec{B}}(M,t)$ soit polarisé selon $\vec{u_\varphi}$.

C6. A partir d'une équation de Maxwell à préciser (utiliser le formulaire), trouver les composantes complexes ($\underline{E_r}, \underline{E_\theta}, \underline{E_\varphi}$) du champ $\underline{\vec{E}}(M,t)$ et montrer en particulier qu'il est orthoradial. Donner les composantes réelles correspondantes de $\vec{E}(M,t)$ associé au champ complexe $\underline{\vec{E}}(M,t)$.

C7. En un point A_1 de l'axe (Ox) tel que $OA_1 = d$, déterminer l'amplitude et la polarisation du champ $\vec{E}(A_1,t)$ dans la base $(\vec{u_x}, \vec{u_y}, \vec{u_z})$.

C8. Donner les composantes réelles correspondantes du champ $\vec{B}(M,t)$ associé au champ complexe $\underline{\vec{B}}(M,t)$.

C9. Vérifier que le champ électromagnétique en M a la structure d'une onde plane progressive harmonique.

C10. Calculer le vecteur de Poynting $\vec{\Pi}(M,t)$ et sa valeur moyenne temporelle $\langle\vec{\Pi}(M,t)\rangle$ au point M.

On donne le résultat de l'intégrale suivante : $\displaystyle\int_0^\pi \frac{\cos^2\left(\frac{\pi}{2}\cos\theta\right)}{\sin\theta}d\theta \approx 1{,}22$ *évaluée en partie* **F**.

C11. Quelle est la puissance moyenne P_a rayonnée à travers la sphère de centre O et de rayon r $(r \gg \lambda)$? Identifier une résistance R_a équivalente (dite de rayonnement) en $z = 0$ telle que : $P_a = R_a \cdot I_0^2$. Application numérique : calculer R_a et I_0 dans le cas où $P_a = 3{,}30\,W$.

D Système RFID passif

Une antenne peut être émettrice comme dans la partie précédente mais peut être également réceptrice comme l'est une antenne de téléphone portable. En conséquence, on admettra par la suite qu'une antenne est équivalente à un circuit d'impédance $\underline{Z_a} = R_a + jX_a$ *parcouru par le courant émetteur dans le cas d'une émission. De même, dans le cas d'une réception, cette antenne est équivalente à un générateur comprenant en série une fem induite* $\underline{E_R}$ *et l'impédance interne de l'antenne* $\underline{Z_a} = R_a + jX_a$. *Dans les deux cas (émission et réception), la résistance* R_a *modélise la puissance moyenne* $P_a = R_a \cdot I^2$ *rayonnée par celle-ci lorsqu'elle est parcourue par un courant efficace I. Pour les applications numériques, on prendra* $R_a = 73{,}2\,\Omega$ et $X_a = 42{,}5\,\Omega$. *On pourra remarquer que* $R_a \approx \sqrt{3}\,X_a$.

En mode émission le schéma équivalent de l'antenne alimentée par un générateur de tension $\underline{E_G}$ *et d'impédance de sortie* $\underline{Z_G} = R_G + jX_G$ *est donné à la figure 7.*

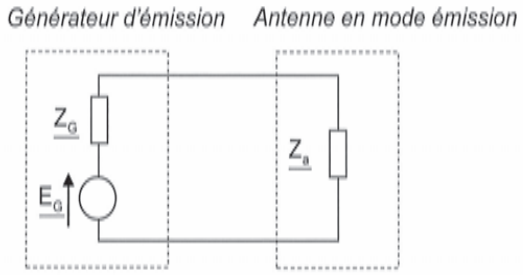


Figure 7

En mode réception l'antenne est connectée à une charge d'impédance $\underline{Z_C} = R_C + jX_C$. *Le schéma équivalent de l'antenne et de sa charge est donné à la figure 8.* $\underline{E_R}$ *est la fem induite dans l'antenne due au champ électromagnétique incident.*

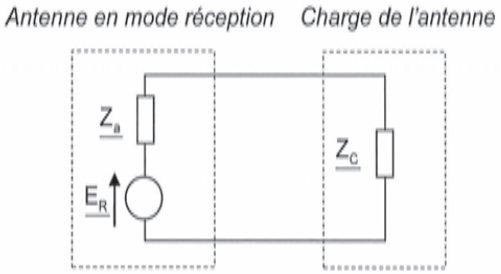


Figure 8

D1. En mode réception (Fig. 8), soit I le courant efficace circulant dans le circuit. Expliquer pourquoi l'antenne « re-rayonne » un champ électromagnétique. Déterminer la puissance P_R « re-rayonnée » par l'antenne en fonction de R_a et de I.

Soit une deuxième antenne a_2 *demi onde, du même type que* a_1 *et centrée sur le point* A_1 *de coordonnées* $(x = d, y = 0, z = 0)$ *et parallèle à* (Oz). *On rappelle que la permittivité du vide vaut* $\varepsilon_0 \approx 8{,}85 \cdot 10^{-12}\ F \cdot m^{-1}$. *On admet que la tension efficace de la fem induite dans l'antenne* a_2 *et due*

au champ électromagnétique incident produit par a_1 s'écrit $V_{beff} = \dfrac{\lambda \cdot I_0}{2\pi^2 \varepsilon_0 c d}$ *(où I_0 est le courant dans a_1 (cf.* **C.11.**)*) . Pour les applications numériques prendre* $d = 5\,m$ *et* $I_0 = 212\,mA$ *. On fera l'hypothèse que l'antenne a_2 ne perturbe pas l'émission de a_1.*

D2. Faire une figure faisant apparaître a_1 et a_2. Application numérique : calculer V_{beff} .

L'antenne a_2 est chargée par une charge d'impédance $\underline{Z_c} = R_C + jX_C = R_a - jX_a$ *pour qu'il y ait adaptation en puissance (cf.* **A.2.**).

D3. Exprimer la puissance moyenne P_C reçue par la charge en fonction de V_{beff} et R_a. Rappeler la signification physique de $\left\|\left\langle \vec{\Pi}(A_1,t)\right\rangle\right\|$. Montrer que a_2 reçoit une puissance $P_{Ra2} = k_b \lambda^2 \left\|\left\langle \vec{\Pi}(A_1,t)\right\rangle\right\|$ où k_b sera explicitée.

On définit la grandeur $A_e = k_b \lambda^2$ *appelée « aire équivalente de l'antenne ».*

D4. Pourquoi un tel nom pour A_e ? Application numérique : calculer k_b et A_e .

Le TAG est modélisé par a_2 et pour simplifier le fonctionnement, les parties émission et réception du système émetteur - récepteur de la figure 0 sont disjointes et modélisées par a_1 et a_3, qui est une troisième antenne demi-onde, du même type que a_1 et centrée sur le point B_1 de coordonnées $\left(x=0, y=0, z=\dfrac{3\ell}{2}\right)$ *et parallèle à (Oz). L'antenne a_3 est toujours connectée à une charge adaptée en puissance. On admet que la puissance reçue par l'antenne a_3 ne vient que de a_2. Lorsque l'antenne a_2 est connectée à une charge adaptée en puissance, la puissance reçue par la charge de a_3 vaut P_{30} ; enfin on pourra remarquer que a_3 vis-à-vis de a_2 joue le même rôle que a_2 vis-à-vis de a_1.*

D5. Faire une figure faisant apparaître a_1, a_2 et a_3. Déterminer P_{30} en fonction de $k_b, \lambda, V_{beff}, \varepsilon_0, c, d$ et R_a . Application numérique : calculer P_{30} .

D6. Lorsque la charge de a_2 a une impédance $\underline{Z_c} = R_C + jX_C$ quelconque (non adaptée en puissance), montrer alors que la charge de a_3 reçoit la puissance P_3 telle que $P_3 = \dfrac{k'_b R_a^2}{\left|\underline{Z_a} + \underline{Z_c}\right|^2} \cdot P_{30}$ où k'_b est une constante à déterminer.

Application numérique : calculer $\dfrac{P_3}{P_{30}} = \dfrac{k'_b R_a^2}{\left|\underline{Z_a} + \underline{Z_c}\right|^2}$ pour $\underline{Z_c} = 0$, puis pour $\underline{Z_c} = \underline{Z_a}^*$ (conjugué de $\underline{Z_a}$) et enfin pour $\underline{Z_c} = \infty$.

D7. Montrer à partir d'un schéma reprenant les éléments de la figure 8 comment réaliser en pratique ces trois conditions ($\underline{Z_c} = 0$, $\underline{Z_c} = \underline{Z_a}^*$ et $\underline{Z_c} = \infty$) avec seulement deux interrupteurs commandables.

D8. Montrer que la modulation (appellée ici « rétromodulation ») de la charge de a_2 (réalisée par la puce électronique) permet de communiquer avec le générateur - lecteur RFID.

Partie II : quelques aspects numériques du problème

L'objectif des parties **E**, **F** *et* **G** *est de décrire quelques méthodes algorithmiques et numériques permettant d'obtenir des résultats admis dans la première partie de ce sujet.*

E Résolution d'une équation de deux variables

On souhaite obtenir la courbe de la figure 3 utilisée à la question **A3.**. *Supposons qu'on dispose d'une fonction* `XIN(X1,X2)` *qui pour deux valeurs des réactances* X_1 *et* X_2 *et pour une impédance de charge* $\underline{Z}_U = 73{,}2 + 42{,}5j$ *calcule la valeur de la réactance équivalente* X_{IN} *correspondante. On souhaite résoudre numériquement l'équation* $X_{IN}(X_1, X_2) = 0$.
On propose la fonction `Resolution(N,eps)` *donnée en annexe.*

E1. La fonction proposée ne fonctionne pas. Lorsqu'on l'utilise, on reçoit le message d'erreur suivant :

En langage Python :	En langage Scilab :
`local variable 'sol1' referenced before assignment`	`variable non définie: sol1`

Ajouter dans le code d'une des deux fonctions en annexe les instructions manquantes pour son bon fonctionnement.

E2. Expliquer le fonctionnement de l'algorithme. Préciser en particulier ce qu'il retourne et ce que représentent les variables d'entrées `N` et `eps`.

E3. Quelles contraintes informatiques empêchent d'utiliser ce programme pour des valeurs arbitrairement grandes de `N` ?

F Calcul d'une intégrale

L'objectif de cette partie est d'étudier une méthode générale d'intégration. On l'utilisera pour obtenir une valeur numérique approchée de : $I = \int_0^{\pi} \frac{\cos^2\left(\frac{\pi}{2}\cos\theta\right)}{\sin\theta} d\theta$

L'application $f : x \mapsto \frac{\cos^2\left(\frac{\pi}{2}\cos(x)\right)}{\sin(x)}$ *est continue sur* $]0,\pi[$ *et est prolongeable par continuité en* 0 *et* π *en posant* $f(0) = f(\pi) = 0$. *L'intégrale* I *est donc bien définie.*
La méthode des rectangles consiste à approcher l'intégrale par la suite $(R_n)_{n\in\mathbb{N}^*}$ *définie pour notre fonction* f *par :*

$$R_1 = 0 \text{ et pour tout } n \geq 2,\ R_n = \lambda_n \sum_{k=1}^{n-1} f\left(\frac{k\pi}{n}\right)$$

où λ_n *désigne un nombre réel qu'il faudra préciser.*

Tous les algorithmes demandés dans cette partie devront être réalisés dans un même langage. Ce langage pourra être le langage Python ou le langage Scilab. On n'utilisera aucune librairie particulière mais on considérera que les fonctions **cos** *et* **sin** *ainsi que le nombre* **pi** *ont été importés dans Python. On rappelle que le nombre π se note* **%pi** *dans Scilab.*
Il n'est pas demandé de vérifier dans les algorithmes que les variables d'entrée sont bien du type voulu. Il est ainsi par exemple inutile dans la question **F2.** *de tester si la variable d'entrée* **x** *est bien dans l'intervalle $]0,\pi[$.*

F1. Illustrer le principe de la méthode des rectangles en représentant R_{10} sur le graphe de la fonction f fourni en annexe. Donner l'expression de λ_n en fonction de n.

F2. Écrire une fonction **f(x)** qui retourne la valeur de f pour un nombre réel x de $]0,\pi[$.

F3. Ecrire une fonction **Rectangle1(n)** qui, pour un nombre entier $n \geq 1$, retourne la valeur de R_n.

F4. Combien d'évaluations de l'application f nécessite cet algorithme pour une valeur de n donnée ?

Il faut déterminer une valeur de n *pour laquelle* R_n *fournit une bonne estimation de* I. *La méthode 1 consiste à calculer les termes successifs de la suite* R_n *jusqu'à ce que celle-ci semble se stabiliser.*

F5. Ecrire une fonction **Integrale1(eps)** qui calcule, pour $\text{eps} > 0$, les termes successifs de la suite $(R_n)_{n\geq 1}$ jusqu'à ce qu'on obtienne $|R_n - R_{n-1}| < \text{eps}$, puis qui retourne la dernière valeur R_n calculée.

F6 Pour $\text{eps} = 10^{-7}$, **Integrale1(eps)** s'arrête pour $n = 34$. Vérifier que cela a nécessité plus de 500 évaluations de la fonction f.

F7. Quelle contrainte informatique empêche d'obtenir un résultat pertinent pour des valeurs arbitrairement petites de **eps** ?

Il est possible d'améliorer la méthode 1. La méthode 2 repose sur la relation de récurrence suivante :

$$\forall n \in \mathbb{N}^*, R_{2n} = \frac{R_n}{2} + \frac{\pi}{2n}\sum_{k=1}^{n} f\left(\frac{\left(k-\frac{1}{2}\right)\pi}{n}\right)$$

F8. Écrire la relation de récurrence pour un entier n de la forme $n = 2^{m-1}$. Écrire une fonction récursive **Rectangle2(m)** qui pour un nombre entier m retourne la valeur de R_{2^m}.

F9. Combien d'évaluations de la fonction f nécessite cet algorithme pour une valeur de m donnée ?

F10. Écrire une fonction **Integrale2(eps)** qui calcule, pour $\text{eps} > 0$, les termes successifs de la suite $(R_{2^m})_{m\geq 0}$ jusqu'à ce qu'on obtienne $|R_{2^m} - R_{2^{m-1}}| < \text{eps}$, puis qui retourne la dernière valeur R_{2^m} calculée. Cet algorithme doit impérativement être élaboré dans le but de minimiser le nombre total d'évaluations de f.

F11. Pour $\text{eps} = 10^{-7}$, **Integrale2(eps)** s'arrête pour $m = 7$. Combien d'évaluations de la fonction f ont été nécessaires ?

G Puissance rayonnée de l'antenne réelle du TAG

Dans les questions **C5.** *et* **C6.** *de la première partie, on cherche à déterminer la résistance de rayonnement R_a de l'antenne. On souhaite proposer une façon de déterminer R_a dans un cadre plus général.*

Une antenne est en réalité rarement filaire et l'expression analytique du champ électromagnétique produit est très difficile (voire impossible) à obtenir en tout point. Une antenne de TAG RFID est representée figure 9.

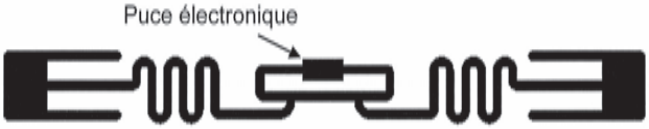


Figure 9

On considère une antenne centrée en l'origine dont les dimensions sont inférieures à $16\,\text{cm}$. *Supposons qu'on ait estimé numériquement le champ électrique et le champ magnétique en tout point d'un maillage de pas* 0,1 m *contenant l'origine et pour des temps espacés d'un centième de période du courant d'alimentation. On a ainsi accès aux coordonnées cartésiennes* (E_x, E_y, E_z) *et* (B_x, B_y, B_z) *des champs* $\vec{E}(M,t)$ *et* $\vec{B}(M,t)$ *en chaque point* $M = (x, y, z)$ *de la forme* $\left(\frac{a}{10}, \frac{b}{10}, \frac{c}{10}\right)$ *où* a,b et c *désignent des nombres entiers et en tout instant* t *de la forme* $\frac{p}{100f}$ *où* p *est un nombre entier et* $f = 860\,\text{MHz}$. *Enfin, on supposera* $I_0 = 1\,\text{A}$. *Ainsi la valeur numérique de* R_a *est simplement donnée par la valeur de* P_a, *la puissance rayonnée par l'antenne.*

L'objectif de cette partie est de proposer une démarche numérique permettant d'estimer Ra. On ne demande ni programme, ni justification mathématique des formules proposées.

G1. Proposer une façon d'estimer, à partir des données disponibles, la valeur moyenne temporelle $\langle \vec{\Pi}(M,t) \rangle$ du vecteur de Poynting dû à l'antenne en un point M du maillage.

G2. On souhaite maintenant estimer la puissance moyenne rayonnée par l'antenne à travers une surface fermée contenant l'antenne. Compte tenu des données disponibles, quelle est parmi les surfaces suivantes la plus pertinente pour obtenir une estimation précise de P_a (et donc de R_a) : la surface du cube $\left[\frac{-1}{10}, \frac{1}{10}\right]^3$, celle du cube $[-5,5]^3$, la sphère de centre O et de rayon $\frac{1}{10}$ ou la sphère de centre O et de rayon 5 ?

G3. Notons $(\langle \Pi_x(M,t) \rangle, \langle \Pi_y(M,t) \rangle, \langle \Pi_z(M,t) \rangle)$ les coordonnées de la valeur moyenne du vecteur de Poynting en un point M du maillage. Proposer une formule permettant d'estimer numériquement la résistance de rayonnement R_a.

G4. Proposer une démarche, qui en faisant varier la surface choisie, permettrait de s'assurer une bonne précision de la valeur estimée de R_a.

FIN DE L'EPREUVE

Document réponse

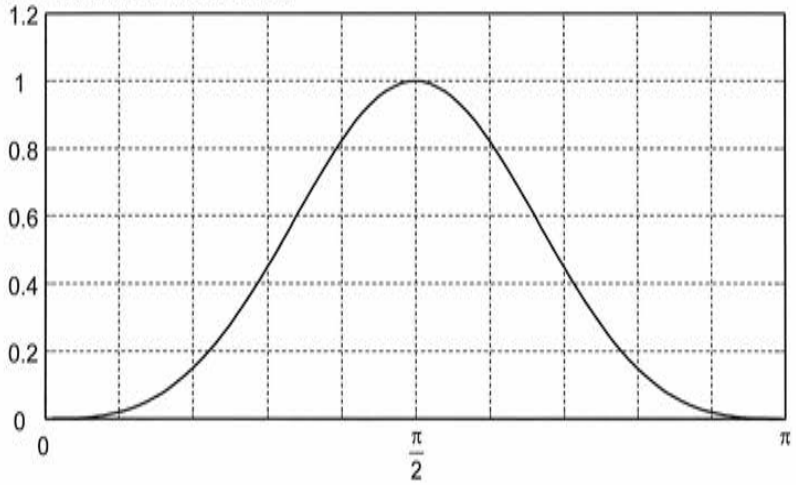
Annexe : Document réponse

Question E1 : fonction `Resolution` en langages Python et Scilab.

En langage Python :	En langage Scilab :

`def Resolution(N,eps) :`
`  valeurs=[i*300/N for i in range(-N,N+1)]`
`  for X1 in valeurs :`
`  for X2 in valeurs :`
`  if abs(Xin(X1,X2))<eps :`
`  sol1=sol1.append(X1)`
`  sol2=sol2.append(X2)`
`  return [sol1,sol2]`
	`function [sol1,sol2]=Resolution(N,eps)`
	`  valeurs=-300 : 300/N : 300 ;`
	`  for X1=valeurs`
	`  for X2=valeurs`
	`  if abs(Xin(X1,X2))<eps then`
	`  sol1=[sol1, X1] ;`
	`  sol2=[sol2, X2] ;`
	`  end ;`
	`  end ;`
	`  end ;`
	`endfunction`

Question F1 : graphe de la fonction f.



ETUDE D'UNE CELLULE D'ASSEMBLAGE POUR AVION FALCON

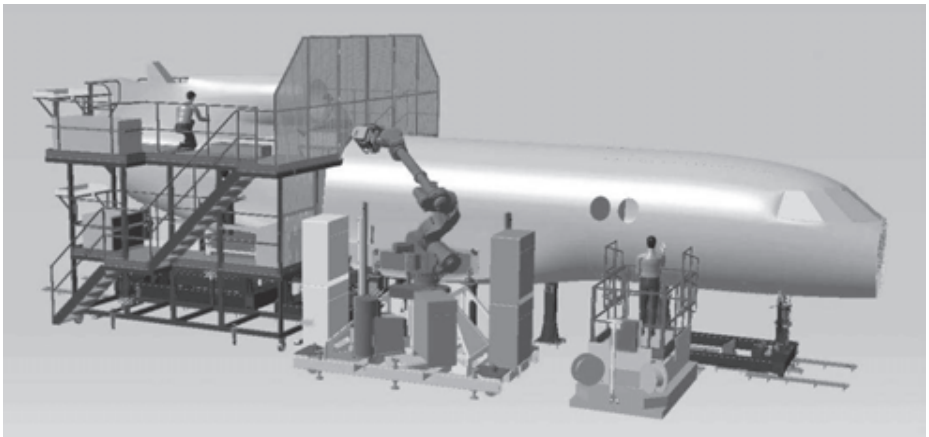


Figure 1 : vue générale CAO de la cellule d'assemblage en cours de travail

Le sujet comprend :

- un questionnaire ;
- un dossier annexe ;
- un document réponse.

Le questionnaire est composé de 4 parties :

- Partie 1 : choix du robot ;
- Partie 2 : étude de l'assemblage ;
- Partie 3 : étude des déplacements ;
- Partie 4 : étude de la sélection des fixations.

Présentation du support d'étude

1 Introduction

Dans un contexte mondial de plus en plus concurrentiel la société DASSAULT doit en permanence améliorer les procédés de production de ses avions.
Une des étapes importantes de la réalisation d'un avion est l'assemblage de sa structure. Comme le montre la Figure 2, la structure d'un avion est composée de plusieurs éléments devant être assemblés entre eux pour donner la structure finale de l'appareil.

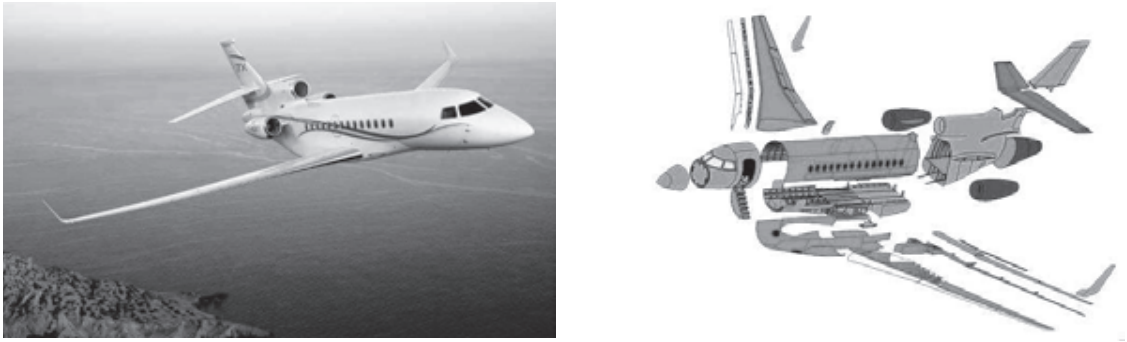


Figure 2 : FALCON 7X et vue éclatée des différents sous-ensembles d'un FALCON 7X

Afin de répondre à des exigences de qualité croissantes et permettre une amélioration de sa productivité, la société DASSAULT développe en permanence de nouveaux moyens d'assemblage des éléments de structure. La cellule d'assemblage de cette étude répond à cette problématique. Elle permet d'assister les opérateurs dans la réalisation des tâches d'assemblage.
Les éléments de structure sont assemblés entre eux par des éléments de fixation appelés rivets : c'est l'opération de rivetage. L'assemblage complet correspond à une succession d'opérations à répéter pour chacun des points de fixations :

- mise en place des éléments à assembler ;
- perçage des éléments ;
- dépose d'un rivet ;
- pose d'une bague déformable ;
- serrage du rivet par déformation de la bague.

Ces opérations devant être répétées un très grand nombre de fois (environ 300 heures d'opérations d'assemblages sur un avion) le gain de productivité apporté par la cellule est important.
De plus, l'utilisation d'un robot permet de diminuer le nombre d'opérations de montage / démontage des éléments à assembler (comparativement à un travail manuel) ce qui permet un gain de travail supplémentaire.
Le support de cette étude, la cellule d'assemblage, permet la réalisation de l'assemblage du tronçon central du fuselage du Falcon 7X. La Figure 3 présente l'extrémité du robot en cours de travail sur ce tronçon central (composé des tronçons 1 et 2).

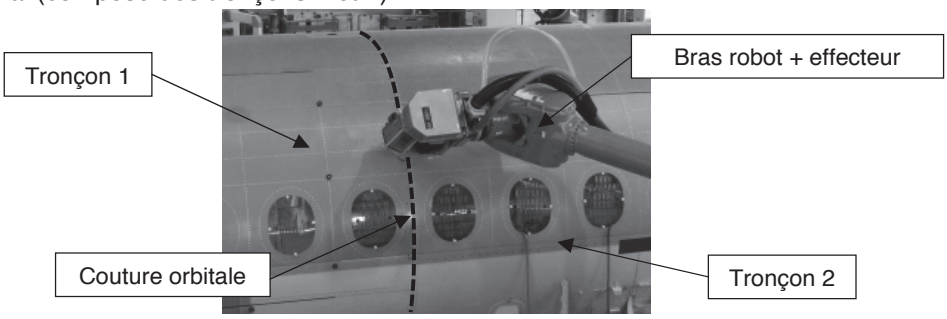


Figure 3 : structure de Falcon 7X en cours d'assemblage par la cellule

2 Présentation du procédé d'assemblage par rivetage

2.1 Positionnement des éléments à assembler

Les différents éléments de l'appareil sont assemblés par rivetage. Pour permettre cet assemblage, chacun des éléments possède à son extrémité un épaulement (partie moins épaisse) permettant l'assemblage. Lorsque les deux éléments à assembler sont en vis-à-vis, les deux tôles des extrémités se superposent permettant ensuite l'obtention d'une structure unique d'épaisseur uniforme.

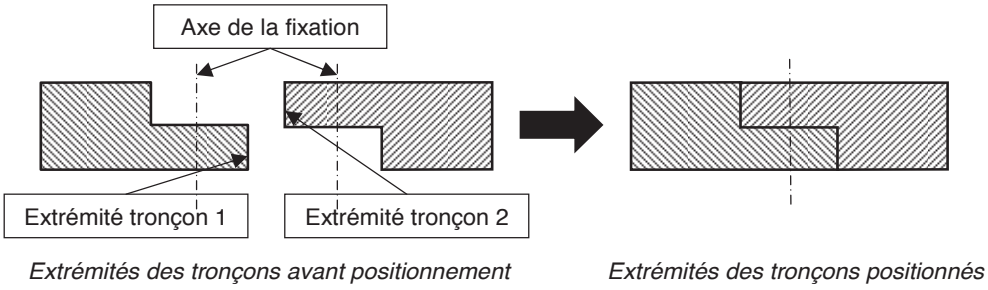


Figure 4 : principe de l'assemblage par superposition de tôles

Les deux extrémités ainsi positionnées sont prêtes à être percées pour recevoir l'élément de fixation. Dans notre étude, nous supposerons que tous les éléments sont déjà positionnés et que des éléments de maintien en position permettent le respect de ce positionnement durant la totalité des opérations de fixation.

2.2 Assemblage des éléments

Le rivetage consiste à assembler deux pièces de façon permanente, il permet donc la réalisation d'une liaison encastrement non démontable. Les opérations de rivetage connaissent également des évolutions technologiques liées à l'emploi de matériaux plus performants comme le titane. Le rivet en aluminium, massivement employé dans la construction aéronautique, est partiellement remplacé par le rivet en titane. Ces rivets en titane permettent de meilleures performances mécaniques ; les fixations obtenues ont une tension entre tôles assemblées plus importante que pour les rivets en aluminium. Les rivets en titane permettent ainsi de réduire le nombre de rivets comparativement aux rivets en aluminium.

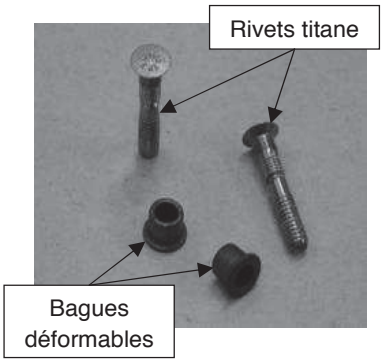


Figure 5 : bagues et rivets

Pour ces raisons, notre étude ne portera que sur l'assemblage par rivetage avec rivets de type titane. La phase d'assemblage étudiée est réalisée conjointement par le bras robot de la cellule et un opérateur. Le bras robot est situé à l'extérieur de l'avion (voir Figure 3) tandis que l'opérateur est situé à l'intérieur de l'avion. L'opérateur contrôle le bras robot à l'aide d'une télécommande et il dispose les bagues déformables (voir Figure 5) nécessaires à la réalisation de l'opération de rivetage.

La phase d'assemblage se décompose en cinq opérations :

- opération 1 : mise en position des tronçons d'avion à assembler sur un châssis de montage ;
- opération 2 : perçage des tronçons par le bras robot ;
- opération 3 : introduction dans le trou (percé) d'un rivet titane par le bras robot ;
- opération 4 : pose d'une bague déformable par l'opérateur sur l'extrémité du rivet située à l'intérieur de l'avion ;
- opération 5 : déformation de la bague et rupture du rivet.

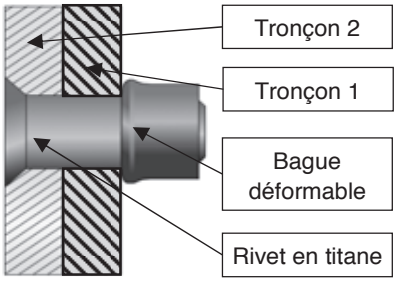


Figure 6 : rivet installé

La fixation obtenue à la fin de ces cinq opérations est celle de la Figure 6. Les cinq opérations sont répétées sur chacune des fixations de l'avion. Chaque rivet installé permet l'établissement d'une action mécanique (appelée tension installée) entre les deux tôles assemblées. La somme des tensions installées, correctement réparties sur les éléments, permet d'obtenir un assemblage rigide non démontable permettant de résister aux contraintes mécaniques subies pendant l'utilisation de l'avion. Le détail des opérations 4 et 5 est donné en annexe 1.

3 Structure de la cellule d'assemblage

3.1 Présentation fonctionnelle

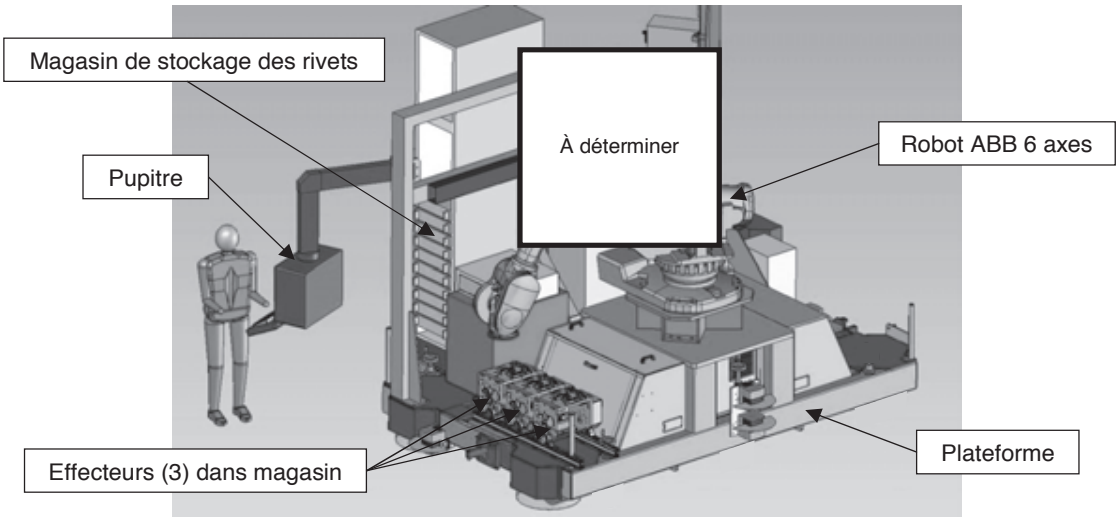


Figure 7 : vue partielle de la cellule d'assemblage

La cellule d'assemblage (Figure 7) est un système permettant de réaliser, en collaboration avec l'opérateur, l'ensemble des opérations d'assemblage décrites précédemment.
Les exigences fonctionnelles de la cellule sont données dans l'annexe 2.

3.2 Description structurelle de la cellule

La cellule est composée de plusieurs sous-ensembles fonctionnels (voir Figure 7 et diagramme en annexe 2) :

- un pupitre de commande permettant de piloter et paramétrer la cellule, ce pupitre peut être déporté grâce à une télécommande ce qui permet le pilotage à distance ;
- une plateforme permettant d'assurer la liaison au bâti et le positionnement des autres sous-ensembles ;
- un robot six axes de marque ABB équipé d'un effecteur, cet effecteur intègre les outils nécessaires à l'assemblage ;
- un magasin de stockage des rivets ;
- une unité d'aspiration des copeaux (non représentée sur la Figure 7).

Partie 1 : choix du robot

1 Objectif

L'objectif de cette partie sera de choisir un robot permettant de satisfaire aux exigences fonctionnelles 1 et 1.3. Pour cela l'étude va porter sur :

- la position de la cellule dans le hall de production (exigence 1.3) ;
- l'assemblage des tronçons à l'aide des rivets en choisissant le robot permettant d'atteindre tous les points de la couture (exigence 1) ;
- la stabilité de la plateforme (exigence 1.3).

2 Étude de la liaison plateforme (1) / sol (0)

La plateforme réalise l'interface entre le robot et le hall de production. Chaque plateforme est composée :

- d'une partie supérieure qui permet de fixer le robot et les accessoires associés à la plateforme ;
- d'une partie inférieure qui permet de lier la plateforme (1) au sol du hall (0).

Le positionnement de la plateforme au sol du hall (annexe 3) est réalisé par trois liaisons. Afin de pouvoir déplacer la plateforme sur les différentes zones d'assemblage, elles sont dupliquées dans le hall de production.

Question 1
En vous aidant de l'annexe 3, tracer le graphe de structure entre le sol (0) et la plateforme (1).

Question 2
Nommer et caractériser les liaisons entre le sol (0) et la plateforme (1).

Question 3
Le positionnement proposé par ces 3 liaisons est-il isostatique ? Justifier la réponse.

Question 4
Donner le nom de la liaison équivalente entre le sol (0) et la plateforme (1). Cette liaison doit-elle être démontable ? Justifier la réponse.

3 Choix du robot – Exigence 1

L'implantation est considérée comme optimale lorsque la totalité des points visés est accessible : l'extrémité du robot doit atteindre le point de fixation de la demi-couture des tronçons. Dans le cas de l'étude, le robot doit réaliser une couture orbitale entre deux tronçons et éviter les collisions éventuelles (annexe 5). La masse de l'effecteur positionnée à l'extrémité du robot est de 100 kg.

Ce choix sera une estimation : cette pré-étude sera validée si plus de 90 % de la zone à couturer est atteinte. Pour cela, les enveloppes de travail de l'annexe 5 seront à utiliser.

Une analyse plus fine sera réalisée par une étude d'implantation dans un environnement numérique 3D de l'atelier de production, des tronçons et des robots permettant ainsi la validation du choix.

Question 5
À l'aide de la documentation des annexes 4 et 5, choisir le robot qui permettra de réaliser la couture orbitale de la position extrême 1 à la position extrême 2 en complétant le schéma du document réponse. Votre schéma devra faire apparaitre :

- la position de l'embase de rotation du robot (point O_1) sur l'axe $\vec{y}_p$;
- les dimensions utiles (permettant d'atteindre la zone à couturer) des enveloppes de travail des robots.

4 Validation du non-basculement du robot - Exigence 1.4

Afin de garantir le non-basculement de la plateforme lorsque le robot se situe dans les positions extrêmes, un dispositif complémentaire d'équilibrage statique doit être adapté sur la plateforme.
Ce dispositif devra :

- être manœuvré manuellement par un opérateur ;
- utiliser l'énergie musculaire de l'opérateur ;
- s'adapter au sol du hall d'assemblage ;
- permettre un réglage fin.

Question 6
Proposer une solution technique permettant de garantir le non-basculement de la plateforme. Vous utiliserez une représentation cinématique en numérotant les solides et en nommant les différentes liaisons qui composent votre solution.
Le sol du hall de production sera numéroté (0) et la plateforme sera numérotée (1).

Question 7
Donner une description de son fonctionnement.

Question 8
Sur un schéma de la plateforme, proposez une implantation de ce dispositif afin de garantir son efficacité (position et nombre) ?

Partie 2 : étude de l'assemblage

1 Objectif

L'objectif de cette partie est de vérifier que le robot choisi permet d'assurer le perçage des tronçons de l'avion.

2 Détermination des actions mécaniques – Exigence 1.2

2.1 Objectif

L'objectif est de déterminer l'effort lié au perçage des tôles dans le cas le plus défavorable (exigence 1.2 en annexe 2).

2.2 Données

L'effort lié au perçage peut être déterminé par le modèle suivant : $F = K'.K_c.R.f$
Avec :

- F : effort lié au perçage (en N) ;
- K' : coefficient lié à la forme de l'outil ;
- K_c : pression spécifique de coupe fonction du matériau (en $N.mm^{-2}$) ;
- R : rayon de l'outil de perçage (en mm) ;
- f : avance de l'outil en ($mm.tour^{-1}$).

Les essais expérimentaux de perçage sont donnés dans les tableaux de l'annexe 6. Ces essais ont été réalisés sur les deux matériaux les plus utilisés pour les tronçons de l'avion (aluminium et composite aluminium/carbone).

Question 9
Déterminer l'effort F dans chacun des cas.

Question 10
Quel est le cas le plus défavorable pour l'opération de perçage ?

Cette valeur d'effort sera ensuite choisie pour les validations des caractéristiques robots.

3 Validation des caractéristiques du robot – Exigence 1.2

3.1 Objectif

L'objectif est de déterminer le couple articulaire C_{12} à appliquer sur le bras 2 afin de garantir l'effort de perçage et l'effort presseur (exigence 1.2).

3.2 Notations

Les éléments de réduction d'un torseur d'action mécanique du solide i (noté S_i) sur le solide j (noté S_j) au point O dans le repère R_0 seront notés :

$$\{S_i \rightarrow S_j\} = \begin{Bmatrix} \vec{R}(S_i \rightarrow S_j) \\ \vec{M}(O, S_i \rightarrow S_j) \end{Bmatrix} = \begin{Bmatrix} X_{ij} & L_{ij} \\ Y_{ij} & M_{ij} \\ Z_{ij} & N_{ij} \end{Bmatrix}_{O \text{ dans } R_0}$$

3.3 Hypothèses

- l'étude est réalisée pour une demi couture orbitale (couture supérieure) ;
- le repère $R_0(O_0 ; \vec{x}_0 ; \vec{y}_0 ; \vec{z}_0)$ sera supposé galiléen ;
- $\vec{y}_0$ est l'axe vertical ascendant et $\vec{g} = -g.\vec{y}_0$ avec g = 9.81 m.s^{-2} ;
- toutes les liaisons sont supposées parfaites.

3.4 Repérage et paramétrage (Figure 8)

Le repère associé à **l'embase fixe (0)** est le repère $R_0(O_0 ; \vec{x}_0 ; \vec{y}_0 ; \vec{z}_0)$, $\vec{y}_0$ étant l'axe vertical ascendant.

L'embase de rotation (1), en liaison pivot d'axe $(O_1 ; \vec{y}_1)$, par rapport au bâti (0), a pour repère associé le repère $R_1(O_1 ; \vec{x}_1 ; \vec{y}_1 ; \vec{z}_1)$ tel que $O_0 = O_1$, $\vec{x}_0 = \vec{x}_1$, $\vec{y}_0 = \vec{y}_1$, $\vec{z}_0 = \vec{z}_1$.

Le bras (2), en liaison pivot d'axe $(O_2 ; \vec{z}_2)$ par rapport à l'embase de rotation (1), a pour repère associé le repère $R_2(O_2 ; \vec{x}_2 ; \vec{y}_2 ; \vec{z}_2)$ tel que $\overrightarrow{O_1O_2} = L_1.\vec{x}_1 + L_2.\vec{y}_1$, $\vec{z}_1 = \vec{z}_2$ et $(\vec{x}_1, \vec{x}_2) = (\vec{y}_1, \vec{y}_2) = \theta_{12}$.

Le bras (3), en liaison pivot d'axe $(O_3 ; \vec{z}_3)$ par rapport au bras (2), a pour repère associé le repère $R_3(O_3 ; \vec{x}_3 ; \vec{y}_3 ; \vec{z}_3)$ tel que $\overrightarrow{O_2O_3} = L_3.\vec{x}_2$, $\vec{z}_1 = \vec{z}_3$ et $(\vec{x}_1, \vec{x}_3) = (\vec{y}_1, \vec{y}_3) = \theta_{13}$.

Le bras (4), en liaison pivot d'axe $(O_4 ; \vec{x}_4)$ par rapport au bras (3), a pour repère associé le repère $R_4(O_4 ; \vec{x}_4 ; \vec{y}_4 ; \vec{z}_4)$ tel que $\overrightarrow{O_3O_4} = L_4.\vec{x}_3 + L_5.\vec{y}_3$, $\vec{x}_3 = \vec{x}_4$ et $(\vec{y}_3, \vec{y}_4) = (\vec{z}_3, \vec{z}_4) = \theta_{34}$.

L'ensemble (E1) composé du bras (5), du poignet et de l'outil, en liaison pivot d'axe $(O_5 ; \vec{z}_5)$ par rapport au bras (4), a pour repère associé le repère $R_5(O_5 ; \vec{x}_5 ; \vec{y}_5 ; \vec{z}_5)$ tel que $\overrightarrow{O_4O_5} = L_6.\vec{x}_3$, $\vec{z}_1 = \vec{z}_5$ et $(\vec{x}_1, \vec{x}_5) = (\vec{y}_1, \vec{y}_5) = \theta_{15}$.

La masse du bras (2) est notée M_2 et la position du centre de gravité est définie par $\overrightarrow{O_2G_2} = \frac{1}{2}.L_3.\vec{x}_2$

La masse du bras (3) et du bras (4) est notée M_{34} et la position du centre de gravité est définie par $\overrightarrow{O_3G_3} = \frac{1}{3}.L_4.\vec{x}_3 + L_5.\vec{y}_3$.

La masse de l'ensemble (E1) est notée M_{E1} et la position du centre de gravité est définie par $\overrightarrow{O_5G_5} = L_7.\vec{x}_5$.
L'extrémité de l'outil est définie par le point P définie par $\overrightarrow{O_5P} = L_8.\vec{x}_5$

Le torseur d'action mécanique lié au perçage sera noté :

$$\{\text{Tronçon (perçage)} \rightarrow \text{E1}\} = \begin{Bmatrix} -F & 0 \\ 0 & 0 \\ 0 & 0 \end{Bmatrix}_{P \text{ dans } R_5}$$

Un effort presseur est de plus nécessaire pour le perçage optimal des deux tronçons. Le torseur d'action mécanique associé sera noté :

$$\{\text{Tronçon (presseur)} \rightarrow \text{E1}\} = \begin{Bmatrix} -P & 0 \\ 0 & 0 \\ 0 & 0 \end{Bmatrix}_{P \text{ dans } R_5}$$

La rotation entre les solides (0) et (1) est supposée bloquée dans la suite du sujet.

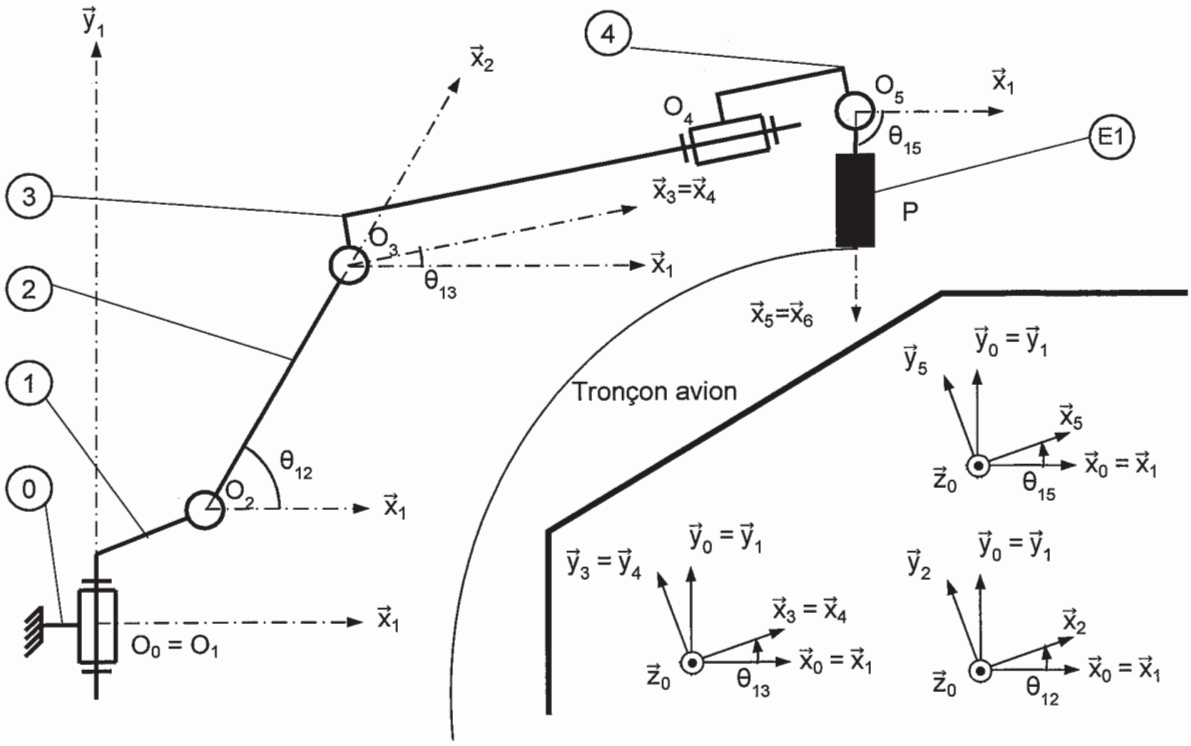


Figure 8 : paramétrage

Question 11
Réaliser le graphe de structure de l'ensemble en précisant les liaisons.

Question 12
Quel est l'ensemble Σ à isoler afin de déterminer le couple C_{12} ?

Question 13
Réaliser un bilan des actions mécaniques extérieures appliquées à Σ et écrire les éléments de réduction de chaque torseur d'actions mécaniques.

Question 14
Quel théorème doit-être appliqué et sur quel axe de projection, pour déterminer le couple C_{12} ?

La configuration correspondant à la position extrême supérieure de la couture orbitale correspond aux angles suivant : θ_{12} = 60 °; θ_{13} = -4 °;θ_{15} = -90 °.

Dans la suite de l'étude, l'angle θ_{13} sera considéré nul.

Question 15
Déterminer l'équation littérale du couple C_{12} en fonction de g, F, P, M_2, M_{34}, M_{E1}, L_3, L_4, L_5, L_6, L_7, θ_{12}, θ_{15}.

Les valeurs du robot considéré sont :

- M_2 = 264 kg ; M_{34} = 430 kg ; M_{E1} = 150 kg ; P = 150 N ;
- L_1 = 0.405 m ; L_2 = 0.433 m ; L_3 = 1.075 m ; L_4 = 1.762 m ; L_5 = 0.165 m ; L_6 = 0.250 m ; L_7 = 0.550 m ; L_8 = 0.750 m ;

Question 16
Déterminer alors la valeur du couple C_{12}.

La valeur limite supérieure du couple C_{12} est fixée par le constructeur à 9000 N.m.

Question 17
Le choix du robot permettra-t-il de garantir les conditions d'assemblage dans cette position ? Justifier la réponse.

Partie 3 : étude des déplacements

1 Contexte

En fonction des diamètres de trous à percer ou pour effectuer une maintenance sur la tête de perçage (affutage d'outils...), le robot change automatiquement de tête à l'aide d'un magasin positionné sur la plateforme. Afin de gagner du temps lors de ces changements de tête, les accélérations des moteurs sont utilisées au maximum.

2 Étude d'une phase dynamique

2.1 Objectif

L'objectif de cette partie est de déterminer le couple articulaire C_{23} sur le bras (3) afin de garantir l'accélération maximale.

2.2 Hypothèses

- le repère $R_0(O_0 ; \vec{x}_0 ; \vec{y}_0 ; \vec{z}_0)$ sera supposé galiléen ;
- $\vec{y}_0$ est l'axe vertical ascendant et $\vec{g} = -g.\vec{y}_0$ avec $g = 9.81\ m.s^{-2}$;
- toutes les liaisons sont supposées parfaites ;
- **θ_{12} est supposé constant**.

2.3 Repérage et paramétrage (Figure 9)

La modélisation proposée, différente de la partie 2 pour des raisons de simplification, est la suivante :
Le repère associé à l'embase fixe (0) est le repère $R_0(O_0 ; \vec{x}_0 ; \vec{y}_0 ; \vec{z}_0)$, $\vec{y}_0$ étant l'axe vertical ascendant.

L'embase de rotation (1), en liaison pivot d'axe $(O_1 ; \vec{y}_1)$, par rapport au bâti (0), a pour repère associé le repère $R_1(O_1 ; \vec{x}_1 ; \vec{y}_1 ; \vec{z}_1)$ tel que $O_0 = O_1$, $\vec{x}_0 = \vec{x}_1$, $\vec{y}_0 = \vec{y}_1$, $\vec{z}_0 = \vec{z}_1$.

Le bras (2), en liaison pivot d'axe $(O_2 ; \vec{z}_2)$ par rapport au bâti de rotation (1), a pour repère associé le repère $R_2(O_2 ; \vec{x}_2 ; \vec{y}_2 ; \vec{z}_2)$ tel que $\overrightarrow{O_1O_2} = L_1.\vec{x}_1 + L_2.\vec{y}_1$, $\vec{z}_1 = \vec{z}_2$ et $(\vec{x}_1, \vec{x}_2) = (\vec{y}_1, \vec{y}_2) = \theta_{12}$.

L'ensemble (E2) composé des bras (3) et (4), et du poignet, en liaison pivot d'axe $(O_3 ; \vec{z}_3)$ par rapport au bras (2), a pour repère associé le repère $R_3(O_3 ; \vec{x}_3 ; \vec{y}_3 ; \vec{z}_3)$ tel que $\overrightarrow{O_2O_3} = L_3.\vec{x}_2$, $\vec{z}_1 = \vec{z}_3$ et $(\vec{x}_1, \vec{x}_3) = (\vec{y}_1, \vec{y}_3) = \theta_{13}$.
L'effecteur, noté (EF), est fixé à l'extrémité du poignet du robot.

La masse du bras (2) est notée M_2 et la position du centre de gravité est définie par $\overrightarrow{O_2G_2} = \frac{1}{2} \cdot L_3 \cdot \vec{x}_2$.
La masse de l'ensemble (E2) est notée M_{E2} et la position du centre de gravité est définie par $\overrightarrow{O_3G_3} = \frac{1}{3}.L_4.\vec{x}_3 + L_5.\vec{y}_3$. Son moment d'inertie par rapport à l'axe $(O_3 ; \vec{z}_3)$ est noté J_{E2}.
La masse de l'effecteur est notée M_{EF} et la position du centre de gravité sera considéré comme le point G_5 avec $\overrightarrow{O_3G_5} = L_4 \cdot \vec{x}_3 + L_5 \cdot \vec{y}_3 + L_6 \cdot \vec{x}_3 + L_7 \cdot \vec{x}_3$. Son moment d'inertie par rapport à l'axe $(G_5 ; \vec{z}_3)$ est noté J_{EF}.

C_{23} est la notation du couple articulaire exercé par le moteur sur le bras (3). L'inertie de son rotor est négligée.

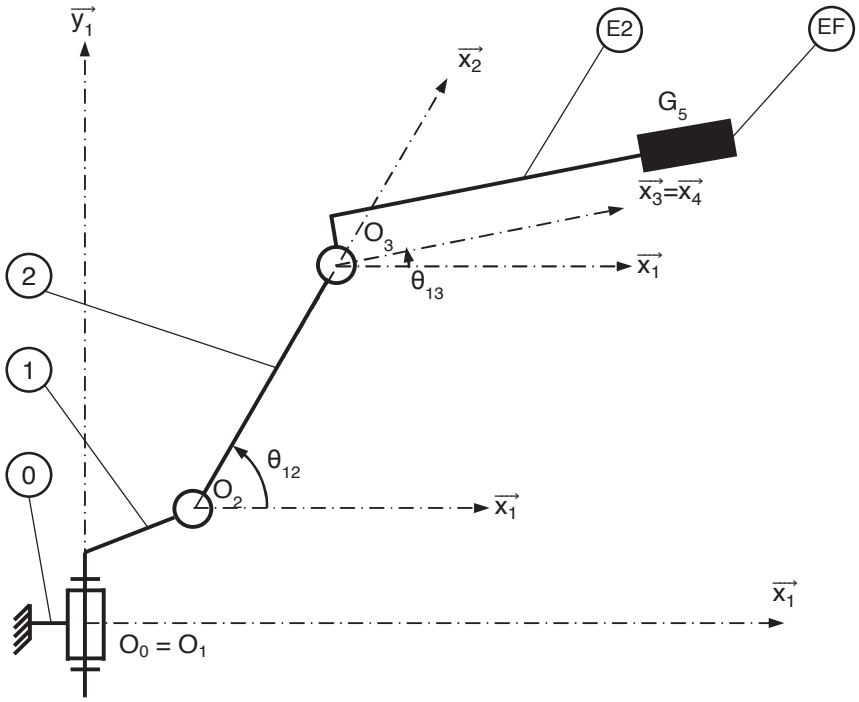


Figure 9 : paramétrage

Question 18
Déterminer la vitesse du solide (E2) au point G_3 dans son mouvement par rapport à R_1 notée $\vec{V}(G_3 \in E2/R1)$ en fonction de L_4, L_5 et $\dot{\theta}_{13}$.

Question 19
Déterminer la vitesse du solide (EF) au point G_5 dans son mouvement par rapport à R_1 notée $\vec{V}(G_5 \in EF/R1)$en fonction de L_4, L_6, L_7 et $\dot{\theta}_{13}$.

Afin de déterminer le couple C_{23}, le théorème de l'énergie sera appliqué à l'ensemble Σ = {EF U E2}.

Question 20
Donner l'expression des puissances développées par les actions extérieures à l'ensemble Σ = {EF U E2}, dans son mouvement par rapport au repère R_1.

Question 21
Donner l'expression des puissances développées par les actions intérieures à l'ensemble Σ = {EF U E2}, dans son mouvement par rapport au repère R_1. Justifier la réponse.

Question 22
Déterminer l'énergie cinétique de Σ dans son mouvement par rapport à R_1.

Question 23
En déduire l'expression du couple C_{23} en fonction de g, J_{E2}, J_{EF}, M_{E2}, M_{EF}, L_4, L_5, L_6, L_7, θ_{13}, $\ddot{\theta}_{13}$

Question 24
L'accélération $\ddot{\theta}_{13}$ est supposée constante. Pour quelle valeur de θ_{13} le couple C_{23} est-il maximal ?

L'accélération maximale est fixée à $\ddot{\theta}_{13} = 1.5$ rad.s^{-2} et le couple articulaire maxi C_{23} à $C_{23max} = 8000$ Nm.
De plus, $M_{E2} = 480$ kg.m^2 ; $M_{EF} = 100$ kg ; $J_{E2} = 272$ kg.m^2 ; $J_{EF} = 2.2$ kg.m^2 ; $L_4 = 1.762$ m ; $L_5 = 0.165$ m ; $L_6 = 0.250$ m ; $L_7 = 0.550$ m.

Question 25
Déterminer dans ces conditions le couple articulaire C_{23} et conclure sur le choix du robot.

L'étude a permis de valider le choix du robot permettant de réaliser une couture orbitale.

Partie 4 : étude de la sélection des fixations

L'objectif de cette partie est de valider les choix effectués par la société pour le sous ensemble de sélection des fixations de la cellule (exigence 1.1).

1 Fonctionnement du magasin de rivets

1.1 Présentation

Le magasin de rivets doit assurer le stockage des rivets prévus pour l'assemblage ainsi que leur distribution vers le robot. Les rivets sélectionnés dans le magasin sont acheminés vers l'effecteur au moyen d'un système d'aspiration (non étudié ici).
Avant d'être acheminés vers l'effecteur, les rivets sont stockés dans des cassettes rangées verticalement dans l'armoire de stockage (Figure 11). Un chariot de sélection se déplace verticalement pour déplacer la buse d'aspiration qui permettra d'acheminer les rivets contenus dans la cassette vers l'effecteur (Figure 10).

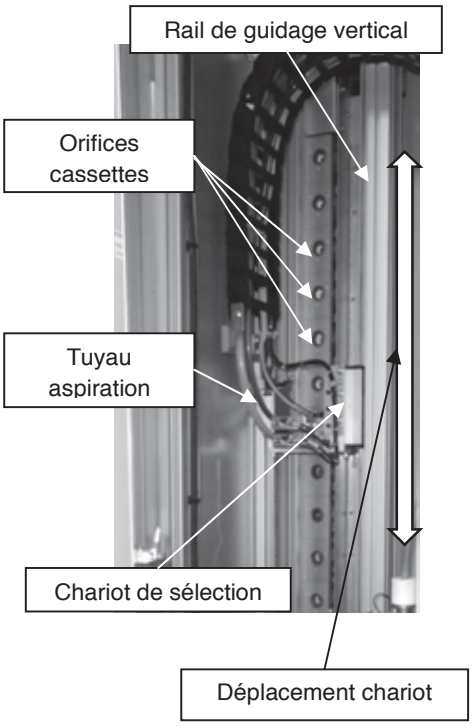


Figure 10 : vue arrière du chariot

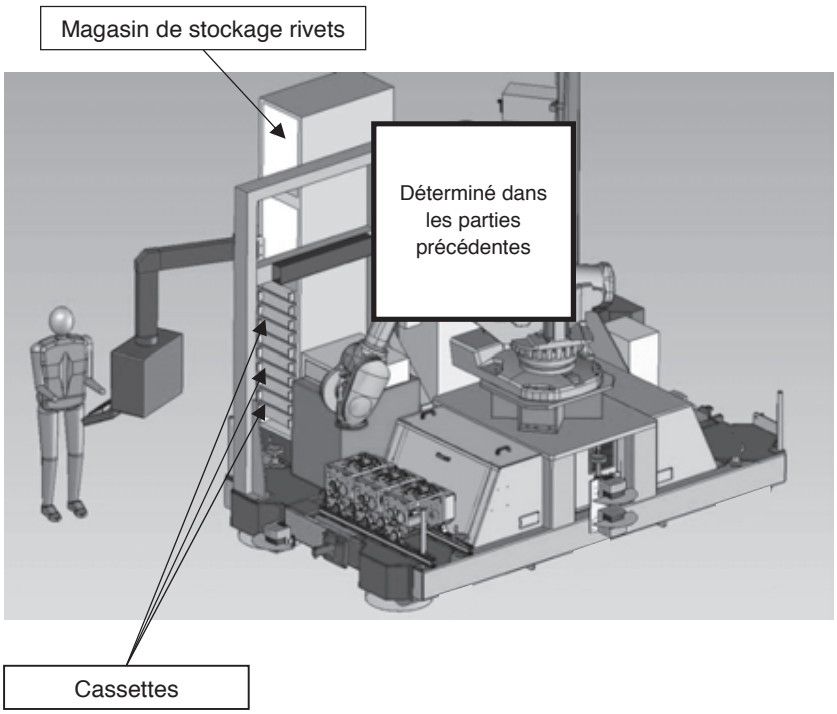


Figure 11 : localisation du magasin de stockage sur la cellule

1.2 Axe chariot

Le déplacement du chariot est assuré par un axe numérique asservi en vitesse et en position. Cet axe est composé d'un moteur à courant continu, d'un système de transmission de puissance de type poulies / courroie et d'un rail (Figure 12 et annexe 7).

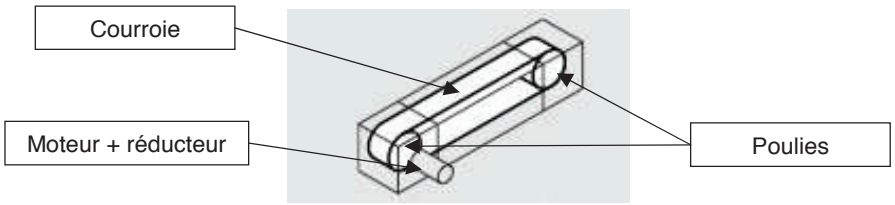


Figure 12 : schéma de principe du guidage

1.3 Modélisation du système de déplacement du chariot

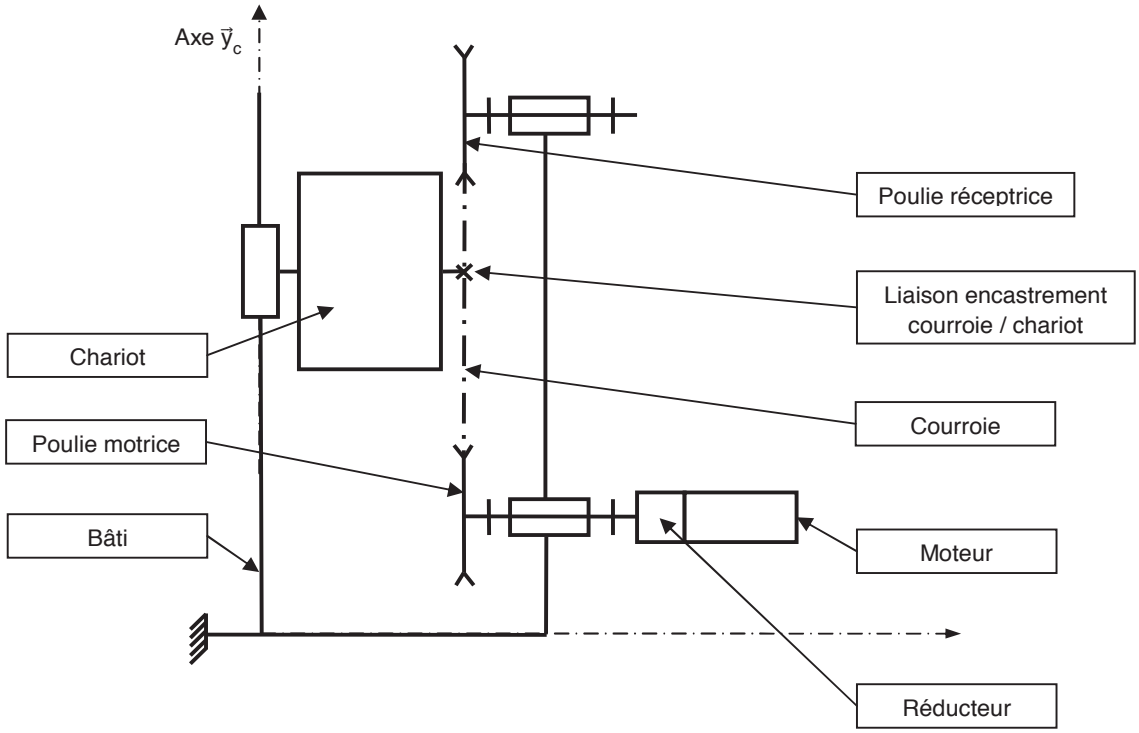


Figure 13 : modèle cinématique

2 Sélectionner les fixations - Exigence 1.1

Afin de sélectionner le type de fixation, la buse d'aspiration doit être déplacée en face de la cassette avec une erreur inférieure à 0,5 mm (voir exigences fonctionnelles). Cependant le fabricant du système poulie-courroie du rail indique déjà une erreur de +/- 0,25 mm due notamment à l'élasticité de la courroie. Par conséquent, l'erreur en position de la commande doit être nulle.
De plus, afin de ne pas perdre de temps lors de la production, le temps maximal de déplacement lors de la sélection est imposé à une seconde.

L'étude se fera dans le cas le plus défavorable c'est-à-dire un déplacement du chariot vers le haut entre les deux cassettes de rivets les plus éloignées. L'axe de déplacement est appelé $\vec{y_c}$.

2.1 Notations domaine temporel – domaine de Laplace

Les notations entre le domaine temporel et celui de Laplace sont données dans la suite. Ainsi, si la fonction f(t) possède une transformée de Laplace, elle sera notée : F(p) = L[f(t)]

Les équations caractéristiques du moteur à courant continu sont rappelées ci-dessous (les conditions de Heaviside sont respectées) :

$$u(t) = e(t) + L\cdot\frac{d\,i(t)}{dt} + R.i(t)$$

$$e(t) = K_E.\ \omega_m(t) \text{ et } C_M(t) = K_C\ .\ i(t)$$

$$J_{eq}\cdot\frac{d\,\omega_m(t)}{dt} + f.\ \omega_m(t) = C_M(t) - C_R(t)$$

Avec :

- u(t) : tension moteur ;
- i(t) : courant moteur ;
- e(t) : force contre-électromotrice ;
- $\omega_m(t)$: vitesse de rotation moteur ;
- $C_M(t)$: couple moteur ;
- $C_R(t)$: couple résistant modélisant l'action de pesanteur.

2.2 Critères à respecter pour l'exigence 1.2

Exigence	Critères	Niveaux
Déplacer le chariot	**Stabilité** • Marge de gain • Marge de phase	M_G = 6 dB mini M_φ = 45 ° mini
	Précision • Erreur statique ε_s par rapport à une consigne de vitesse constante.	nulle
	Rapidité • Temps de réponse à 5 % en réponse à une consigne échelon.	$Tr_{5\%}$ = 0,1 s maxi

2.3 Choix d'une architecture de la chaine de transmission

Question 26

À partir du document réponse, proposer sous la forme d'un schéma une autre solution permettant le déplacement du chariot. La conversion de l'énergie électrique en énergie mécanique par un moteur doit être conservée.

Question 27

Donner un avantage et un inconvénient significatif de votre solution par rapport à celle proposée Figure 13.

Compte tenu des vitesses de translation importantes, le système retenu est de type poulie-courroie.

2.4 Détermination de l'inertie équivalente

Les grandeurs caractéristiques (notations et valeurs) des éléments de l'axe du chariot sont données dans le tableau ci-dessous :

Moment d'inertie du rotor du moteur autour de son axe	J_m	140.10^{-6} kg.m^2
Moment d'inertie du réducteur ramené à l'arbre moteur	$J_{réd}$	60.10^{-4} kg.m^2
Moment d'inertie de la poulie motrice autour de son axe	J_{PM}	38.10^{-4} kg.m^2
Moment d'inertie de la poulie réceptrice autour de son axe	J_{PR}	38.10^{-4} kg.m^2
Masse totale du chariot	M	5 kg
Vitesse de rotation de l'arbre moteur	ω_m	
Vitesse de rotation de l'arbre de sortie du réducteur	ω_r	
Rayon d'une poulie motrice ou réceptrice	R_p	45 mm

Rapport de réduction réducteur (ω_r/ω_m)	λ	1/5

Question 28
À partir des grandeurs définies, déterminer l'expression littérale de l'inertie équivalente J_{eq} de l'ensemble Σ = {moteur + réducteur + poulies + chariot} ramenée sur l'arbre moteur. Cette inertie équivalente est définie par $E_c(\Sigma) = \frac{1}{2}.J_{eq}.\omega_m{}^2$

Question 29
Déterminer la valeur numérique de l'expression précédente.

2.5 Modèle de connaissance du moteur à courant continu

L'objectif de cette partie est d'établir un modèle de la motorisation de l'axe afin de simuler un déplacement.

Question 30
À partir des équations du moteur à courant continu, compléter le schéma bloc du moteur à courant continu sur le document réponse.

Question 31
En considérant $C_R(p) = 0$, déterminer la fonction de transfert $H_M(p) = \frac{\Omega_m(p)}{U(p)}$ sous sa forme canonique.

Question 32
Montrer que la fonction de transfert $H_M(p)$ peut se mettre sous la forme simplifiée $H_M(p) = \frac{K_c}{K_c.K_e+R.J_{eq}.p+L.J_{eq}.p^2}$

Justifier la réponse. Pour cette question, la valeur numérique de J_{eq} considérée sera $J_{eq} = 7.10^{-3}$ kg.m^2 indépendamment du résultat numérique calculé précédemment.

Question 33
Montrer qu'avec l'expression simplifiée $H_M(p)$ peut s'écrire sous la forme $H_M(p) = \frac{K_M}{(1+T_E.p).(1+T_M.p)}$ (avec $T_E < T_M$).

Cette expression sera maintenant utilisée pour la simulation numérique.

2.6 Étude de l'asservissement en position de l'axe

La partie précédente a permis de déterminer un modèle du moteur. La suite de l'étude va permettre, par simulation, de déterminer les réglages nécessaires de l'axe vis-à-vis du cahier des charges.

La Figure 14 ci-après présente le principe de l'asservissement de l'axe du chariot :

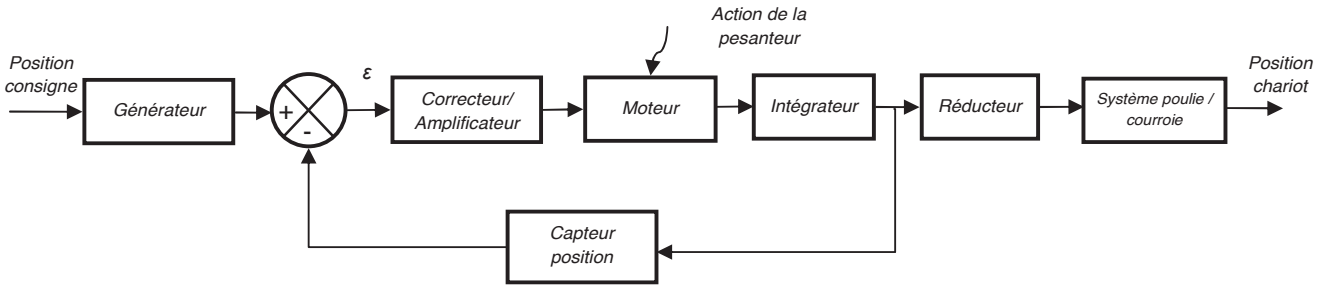


Figure 14 : schéma de principe de l'asservissement

Les grandeurs caractéristiques des blocs de l'asservissement de l'axe chariot sont données dans le tableau ci-dessous :

Générateur	K_G	À déterminer
Capteur de position	K_{capt}	5.10^{-3} V.rad^{-1}
Correcteur amplificateur	C(p)	Variable, voir énoncé

Question 34
Quelle doit être la valeur de K_G pour assurer un asservissement correct (c'est à dire l'écart ε doit être nul si la position de l'axe est identique à la consigne) ?

Question 35
Compléter le schéma bloc de l'asservissement de l'axe du document réponse.

Afin de faciliter les calculs, le schéma bloc à retour unitaire est donné Figure 15. Le couple résistant C_R dû à l'action de pesanteur est supposé constant.

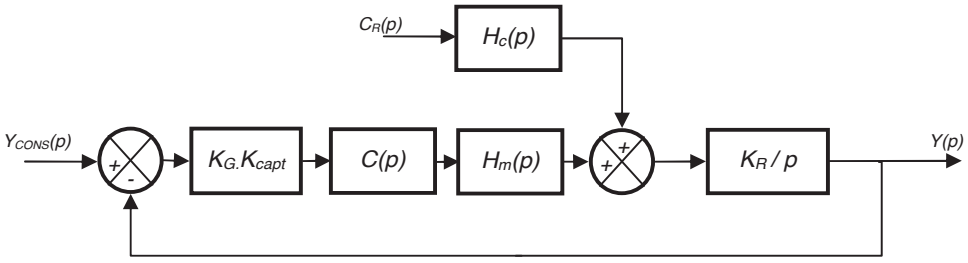


Figure 15 : schéma avec retour unitaire

Avec : $H_M(p) = \frac{K_m}{(1+T_E.p).(1+T_M.p)}$; $H_C(p) = \frac{(R+L.p).K}{(1+T_E.p).(1+T_M.p)}$; $C_R(p)=\frac{C_0}{p}$

Le diagramme de Bode de la fonction de transfert en boucle ouverte $H_{BO}(p)$ est donné sur le document réponse. Le tracé est donné pour C(p) = 1.

Question 36
Le système est-il stable ? Justifier la réponse.

Le couple résistant est un couple constant C_0 qui traduit l'action mécanique de pesanteur subie par l'ensemble mobile.

Question 37
Justifier que si $C(p) = 1$, l'exigence fonctionnelle liée à la précision (erreur nulle) ne peut être respectée.
Proposer une forme générale de fonction de transfert pour ce correcteur permettant de satisfaire à cette exigence fonctionnelle.

Afin de répondre totalement au cahier des charges, l'utilisation d'un correcteur proportionnel intégral dérivé est retenue. En effet, la commande de l'axe intègre directement ce type de correcteur.

Dans la suite du problème, le correcteur C(p) sera de la forme : $C(p) = K_I.\left(1+\frac{1}{T_i.p}\right).(1+T_D.p)$. Le réglage des coefficients a été fait par simulation numérique.

Question 38
Ce nouveau correcteur permet-il de respecter l'exigence fonctionnelle liée à la précision ? Justifier la réponse par un calcul littéral.

Le diagramme de Bode de la nouvelle fonction de transfert en boucle ouverte $H'_{BO}(p)$ est donné sur le document réponse.

Question 39
À partir du diagramme de Bode conclure sur l'exigence fonctionnelle liée à la stabilité. Les constructions graphiques permettant la justification de la réponse devront apparaître sur le document réponse.

Afin de vérifier maintenant le critère de rapidité, le document réponse donne la réponse temporelle de l'axe à un échelon de position de 1 m.

Question 40
Conclure sur la conformité au cahier des charges du système ainsi réglé.

La simulation a permis de déterminer un réglage satisfaisant vis-à-vis des exigences fonctionnelles pour ce système. Il faut maintenant vérifier ces réglages par la mesure directe d'un déplacement de l'axe.

3 Vérification des performances de l'axe du magasin de rivets

Afin de vérifier les réglages précédents, un essai sur le système réel est réalisé. L'absence de système d'acquisition dédié impose un système de mesure extérieur au système réel. C'est un dispositif d'analyse d'image qui est retenu pour ces mesures.
L'objectif de cette partie est de traiter les mesures obtenues puis de vérifier que les performances obtenues sont satisfaisantes.

3.1 Importation et tracé des mesures

Le système d'analyse d'image a permis de récupérer la position du chariot en fonction du temps. Le résultat de ces mesures est donné en annexe 8 (fichier mesure_chariot.txt). Le fichier indique les valeurs de la position en mètre du chariot associée au temps en seconde. Les mesures correspondent à un déplacement entre les deux positions théoriques extrêmes de sélection (1 m à -0,9 m dans le repère de mesure).

Le programme de traitement des mesures est partiellement fourni sur le document réponse dans le langage Python.
Dans un premier temps, les mesures du fichier doivent être récupérées pour afficher la position du chariot en fonction du temps.

Question 41
Compléter sur le document réponse les lignes de commande 8 à 13 qui permettent de créer deux tableaux (X le déplacement et T le temps) dans lesquels seront stockées les valeurs de mesures importées du fichier texte.

Question 42
Écrire les lignes de commandes (à compléter ligne 16 et 17) permettant d'obtenir le tracé de la Figure 16 (les commandes permettant d'obtenir le style du tracé et les légendes de la figure ne sont pas demandées).

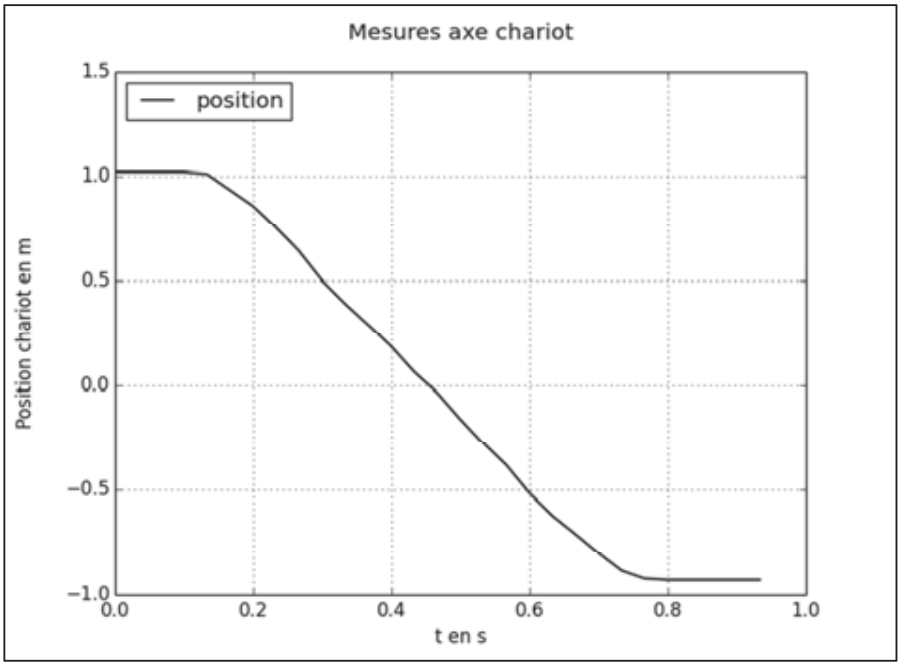


Figure 16 : graphique de la position du chariot en fonction du temps

3.2 Traitement des mesures

Le traitement des mesures récupérées précédemment va permettre de valider les performances de l'axe concernant la vitesse instantanée. Pour cela, des lignes de commandes sont ajoutées au programme précédent. L'ébauche de cet ajout est donnée dans le document réponse dans le langage Python.

Question 43
Écrire les lignes de commandes (à compléter ligne 28 à 32) permettant d'obtenir une liste de valeurs V contenant la vitesse instantanée du chariot.

Question 44
Écrire les lignes de commandes (à compléter ligne 36 à 39) permettant d'obtenir le tracé de la Figure 17 (les commandes permettant d'obtenir le style du tracé et les légendes de la figure ne sont pas demandées).

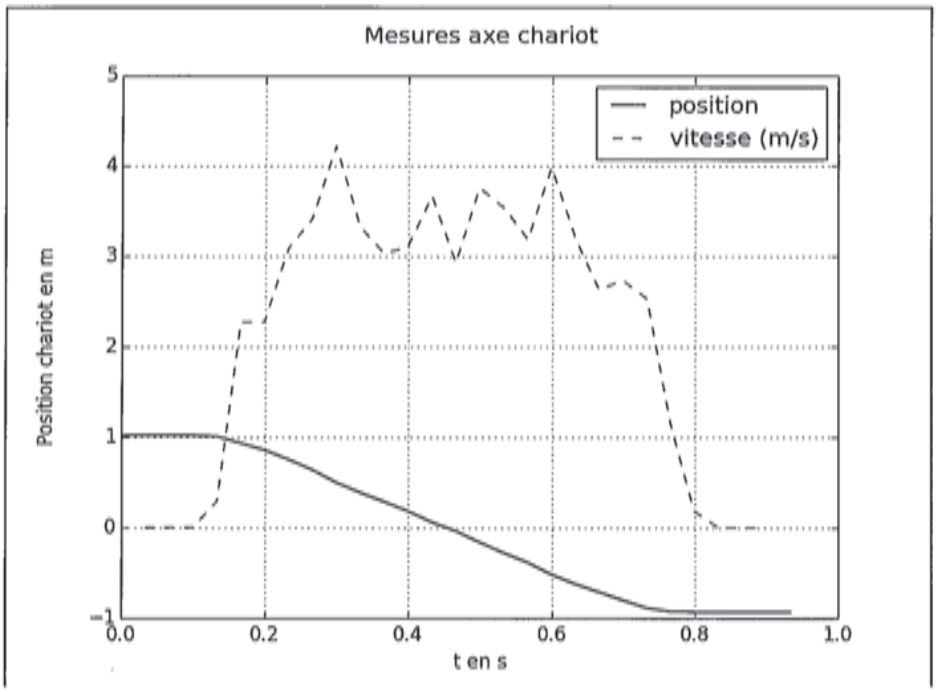


Figure 17 : graphique de la position et de la vitesse du chariot en fonction du temps

Question 45
Justifier l'allure du tracé de la vitesse entre 0,2 s et 0,6 s. Comment peut être amélioré ce tracé ?

3.3 Vérification des performances

Pour conclure l'étude, les performances de l'axe en vitesse et en position sont à vérifier.

Exigence	Critères	Niveaux
Sélectionner le type de fixation	• Temps de sélection • Erreur de position du convoyeur	1 s maxi Inférieure à 0,5 mm

Question 46
À partir des mesures précédentes (fichier de mesures et tracés), conclure sur le respect des exigences fonctionnelles de l'axe du magasin de stockage des rivets.

FIN

L'usage de calculatrices est autorisé.

Cahier réponses

Épreuve de Sciences Industrielles PSI

Document réponse

Question 1 : graphe de structure entre le sol (0) et la plateforme (1)

Question 2 : liaison entre le sol (0) et la plateforme (1)

- Liaison 1 :
- Liaison 2 :
- Liaison 3 :

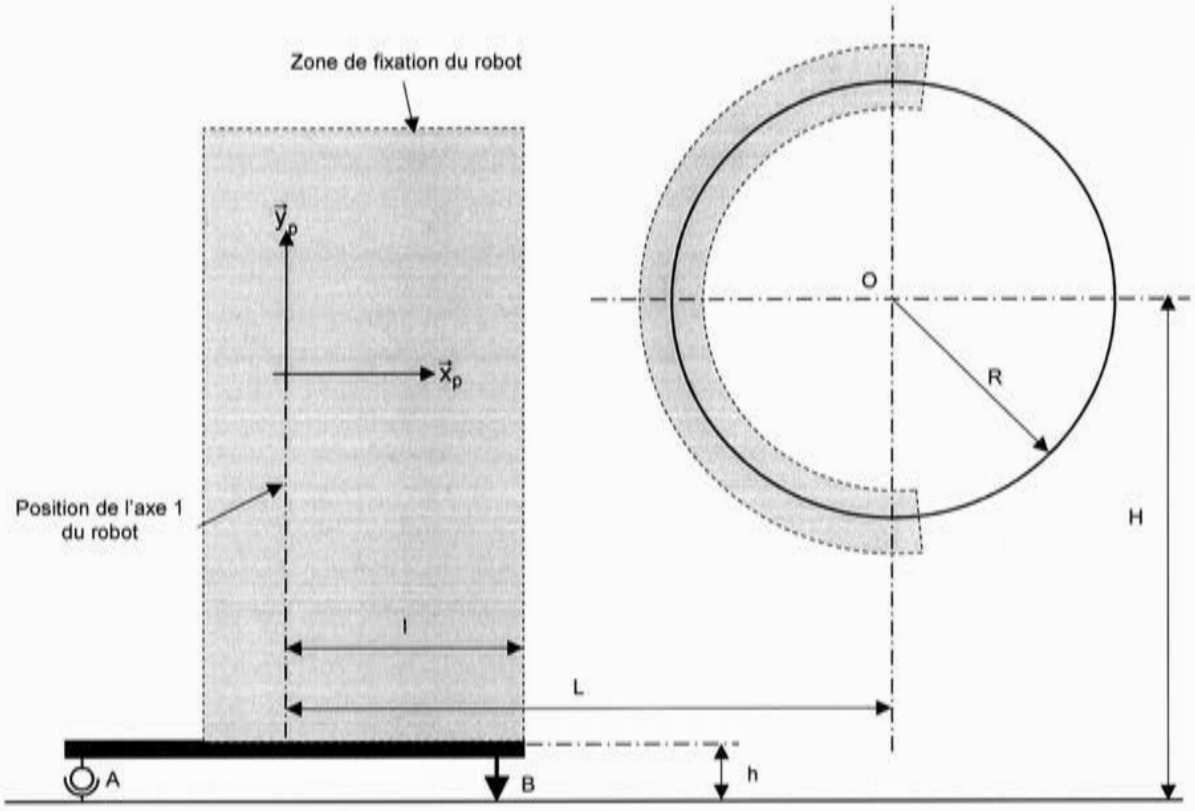
Question 3 : isostatisme du positionnement

Question 4 : liaison équivalente entre le sol (0) et la plateforme (1)

Question 5 : implantation du robot

Choix du robot :

Échelle : 1 cm = 380 mm



Justification :

Question 6 : solution technique permettant le non basculement de la plateforme

Schéma cinématique :	
Numérotation des solides :	Liste des liaisons :

Question 7 : description du fonctionnement

Question 8 : proposition d'implantation sur la plateforme

Question 9 : détermination de l'effort F

Cas 1	Cas 2	Cas 3	Cas 4

Question 10 : cas le plus défavorable

Question 11 : graphe de structure de l'ensemble

Question 12 : ensemble Σ à isoler

Question 13 : bilan des actions mécaniques extérieures à Σ

$\{ \quad \rightarrow \quad \} = \left\{ \quad \right\}$	$\{ \quad \rightarrow \quad \} = \left\{ \quad \right\}$
$\{ \quad \rightarrow \quad \} = \left\{ \quad \right\}$	$\{ \quad \rightarrow \quad \} = \left\{ \quad \right\}$

$\{ \quad \rightarrow \quad \} = \left\{ \quad \right\}$	$\{ \quad \rightarrow \quad \} = \left\{ \quad \right\}$
$\{ \quad \rightarrow \quad \} = \left\{ \quad \right\}$	

Question 14 : théorème à utiliser pour déterminer C_{12}

Question 15 : détermination de l'expression littérale du couple C_{12}

C_{12} =

Question 16 : valeur du couple C_{12}

Question 17 : validation du choix du robot et justification

Question 18 : détermination de la vitesse du solide E2 au point G_3 par rapport à R_1

Question 19 : détermination de la vitesse du solide EF au point G_5 par rapport à R_1

Question 20 : puissances développées par les actions extérieures à l'ensemble Σ

Question 21 : puissances développées par les actions intérieures à l'ensemble Σ

Question 22 : énergie cinétique de Σ dans son mouvement par rapport à R_1

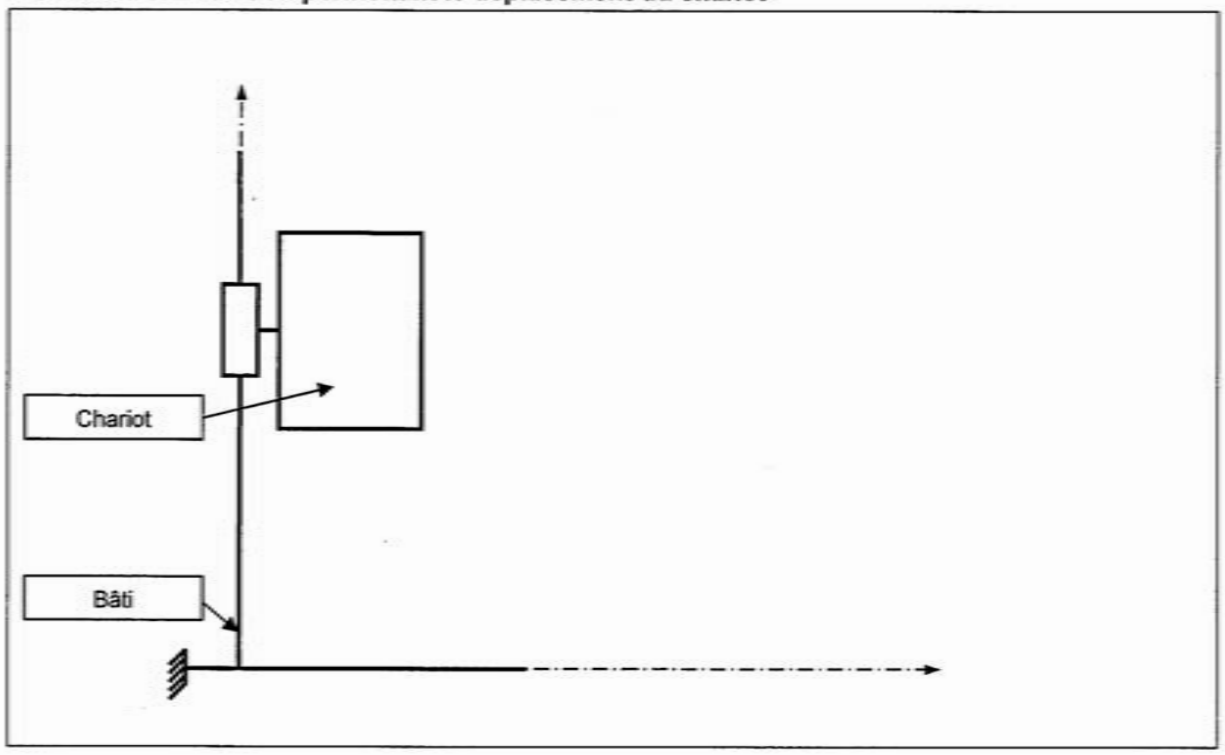
Question 23 : détermination de l'expression littérale du couple C_{23}

C_{23} =

Question 24 : valeur de de θ_{13} ou le coupe C_{23} est maximum

Question 25 : application numérique et conclusion

Question 26 : solution permettant le déplacement du chariot



Question 27 : avantage et inconvénient de votre solution

Avantage :

Inconvénient :

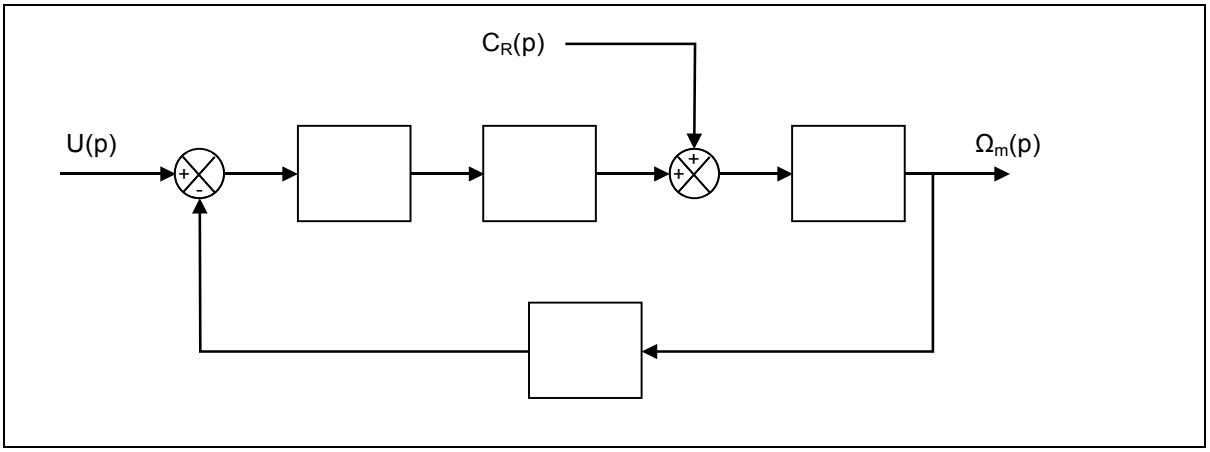
Question 28 : expression littérale de J_{eq}

Jeq =

Question 29 : valeur numérique de J_{eq}

AN : J_{eq} =

Question 30 : schéma bloc du moteur à courant continu



Question 31 : fonction de transfert $H_M(p)$

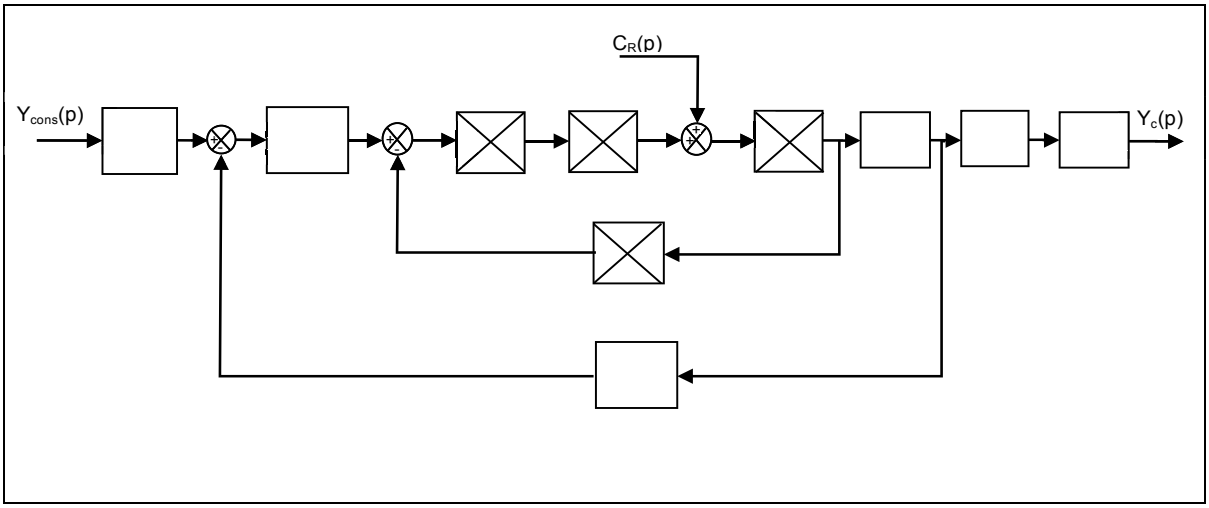
Question 32 : forme simplifiée de $H_M(p)$

Question 33 : écriture de $H_M(p)$

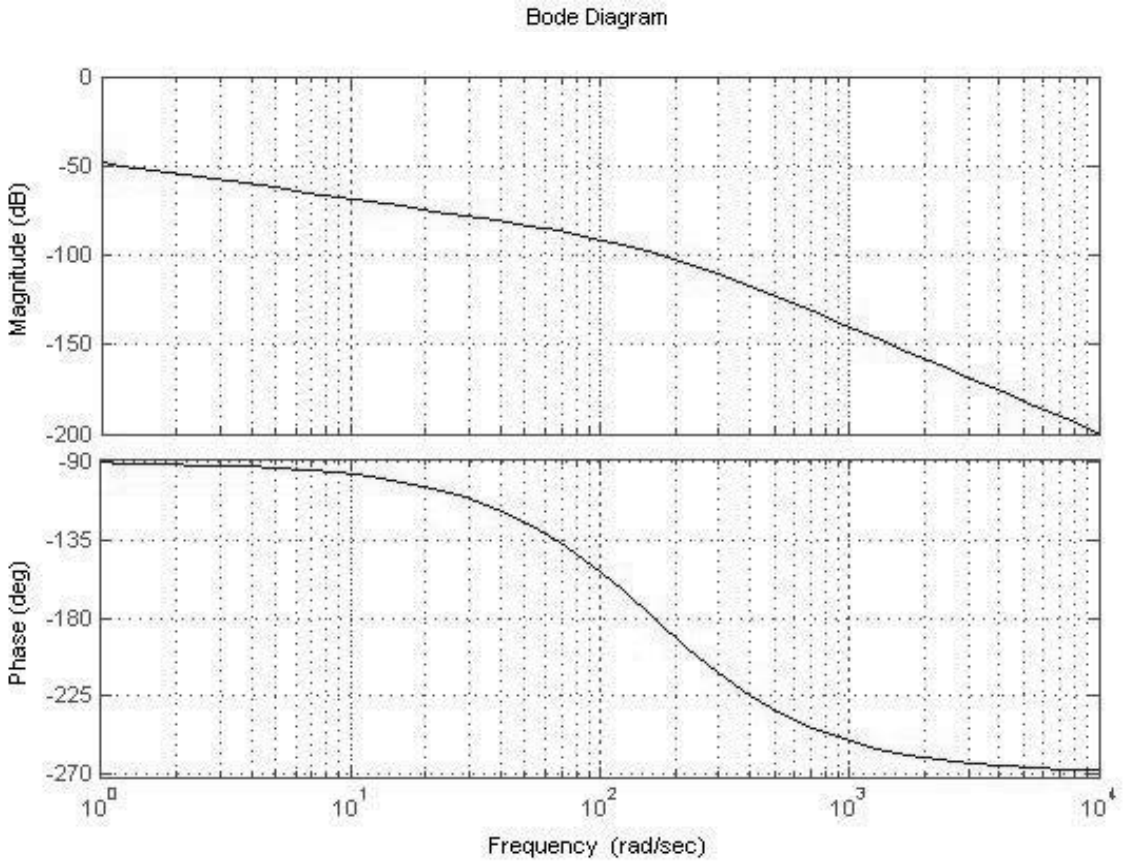
Question 34 : valeur de K_G

K_G =

Question 35 : schéma bloc de l'asservissement de l'axe



Question 36 : stabilité du système

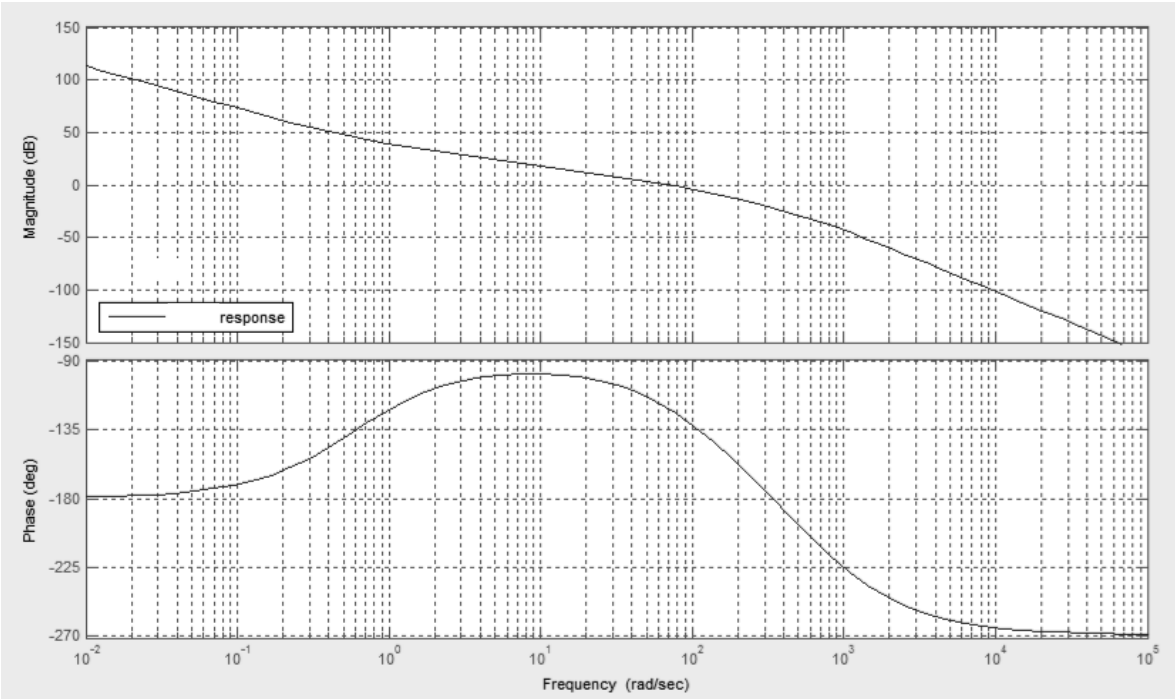


Conclusion vis-à-vis de la stabilté :

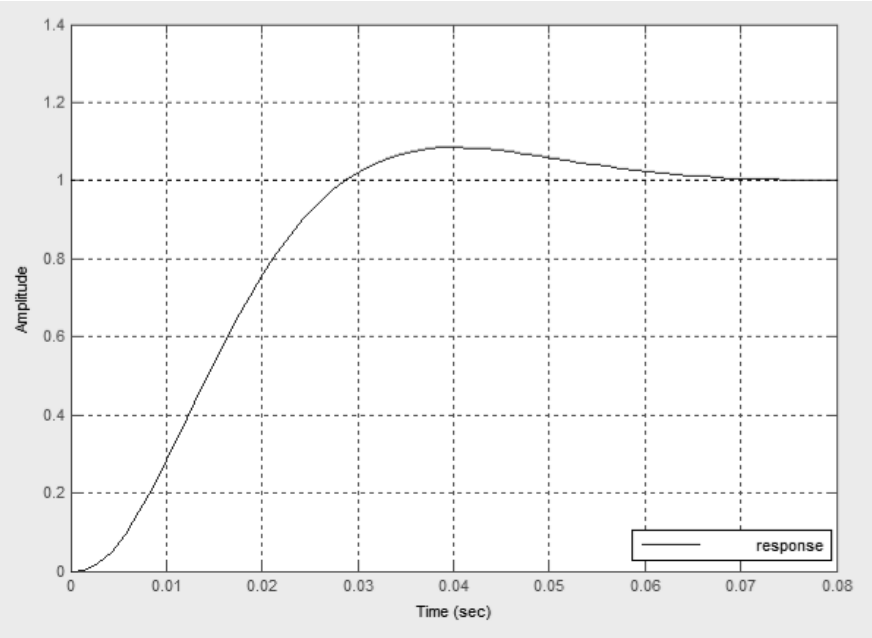
Question 37 : exigence fonctionnelle liée à la précision et proposition du correcteur C(p) = 1

Question 38 : exigence fonctionnelle liée à la précision pour le correcteur PID

Question 39 : exigence fonctionnelle liée à la stabilité



Question 40 : conformité du cahier des charges



Conformité de la commande vis-à-vis du cahier des charges :

Questions 41 et 42 : lignes 8 à 13 puis lignes 16 et 17

```
1  # Lecture du fichier de mesure chariot
2  fic=open("chariot.txt","r")          # Ouverture du fichier en lecture
3  texte=fic.readlines()                # Lecture de l'ensemble des lignes
4  fic.close()                          # Fermeture du fichier
5
6  # Lecture du tableau de données
7  texte[4:33]                          # Sélection des lignes du fichier
8                                       # Initialisation des variables T et X
9  for
10   ligne=texte[i].rstrip("\n")        # Suppression du retour à la ligne
11   ligne=ligne.split(",")             # Découpage au niveau des virgules
12                                      # Ajout au tableau T
13                                      # Ajout au tableau X
14
15   # Tracé de la courbe de déplacement
16
17
19  plt.legend(loc='upper left')
20  plt.title(texte[0])                 # Récupération ligne 1 pour le titre
21  plt.ylabel(texte[1])                # Récupération ligne 2 pour le titre de l'axe y
22  plt.xlabel(texte[2])                # Récupération ligne 3 pour le titre de l'axe x
23  plt.grid(True)                      # Affichage du quadrillage
24  plt.show()                          # Affichage du graphique
```

Question 43 et 44 : lignes 28 à 32 puis lignes 36 à 39

```
24 # plt.show()                    # Mise en commentaire de l'affichage précédent
25
27 # Calcul de la vitesse instantanée à partir des mesures
28
29
30
31
32
33 # Suppression des valeurs extrêmes
34 delT[0]
35 delT[27]
36
37 plt.legend(loc='upperleft')      # Insertion légende et placement dans le graphique
38
39
```

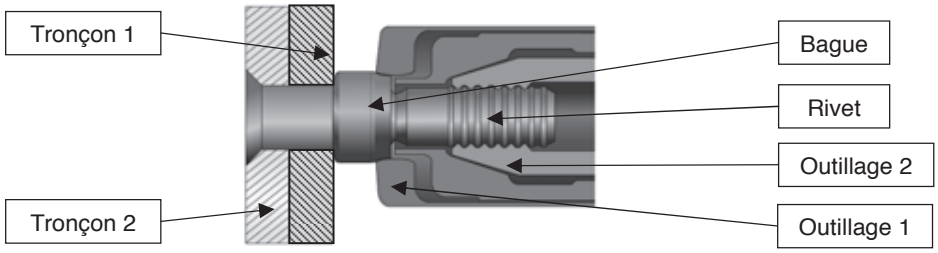
Question 45 : justification de l'allure du tracé et amélioration

Question 46 : conclusion sur exigences fonctionnelles

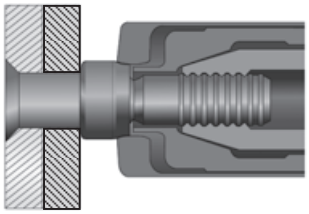
FIN

025

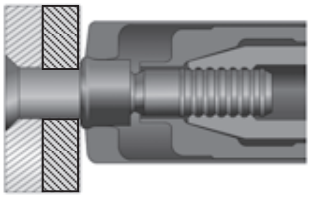
ANNEXE 1 : détail des opérations 4 et 5



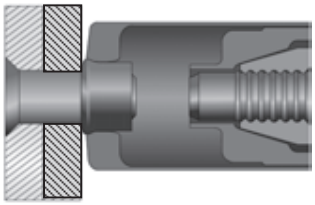
Vocabulaire



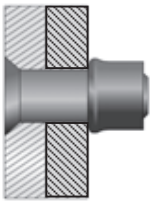
1 – Mise en place de la bague et de l'outillage sur le rivet



2 – Déformation de la bague sur le rivet par déplacement de l'outillage 1



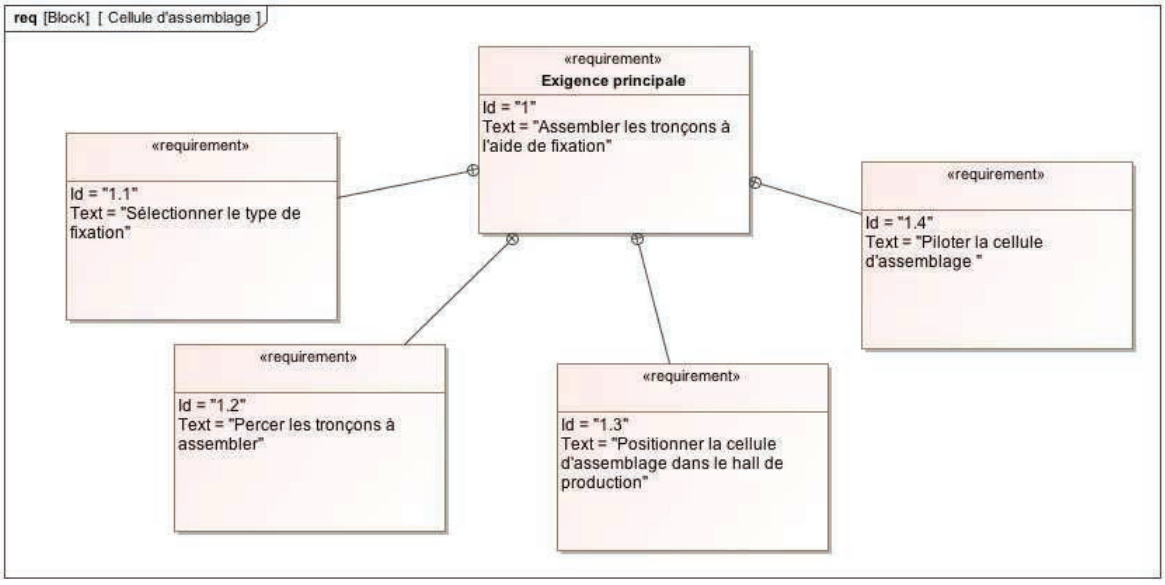
3 – Rupture du rivet par déplacement de l'outillage 2



4 – Outillage retiré et rivet installé

ANNEXE 2 : diagrammes SysML

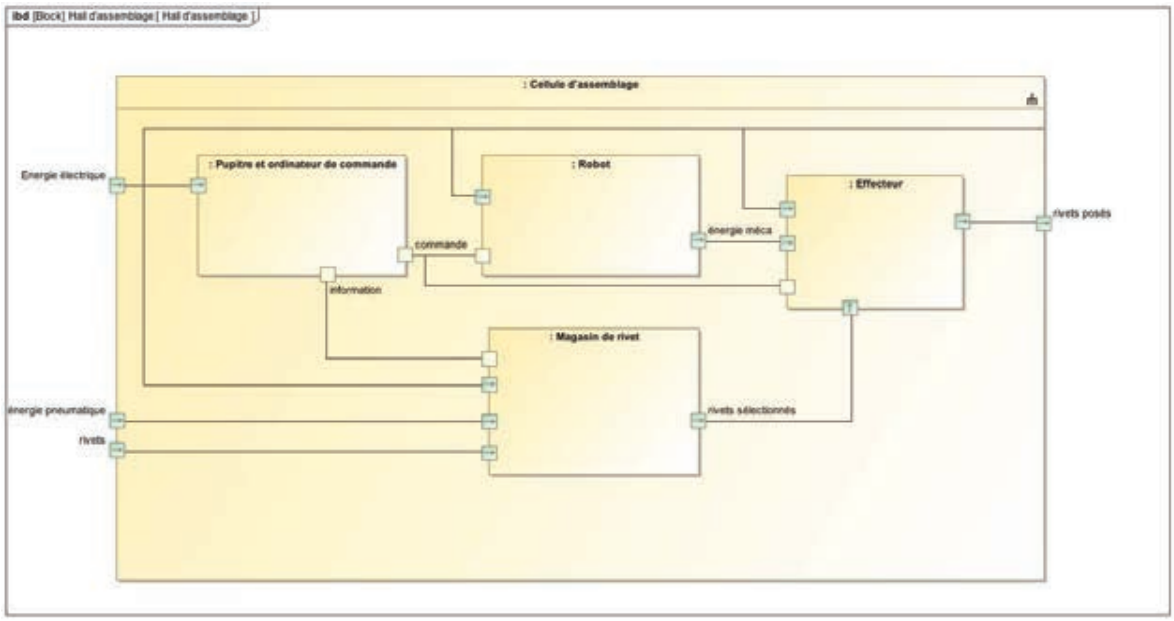
1 Diagramme des exigences



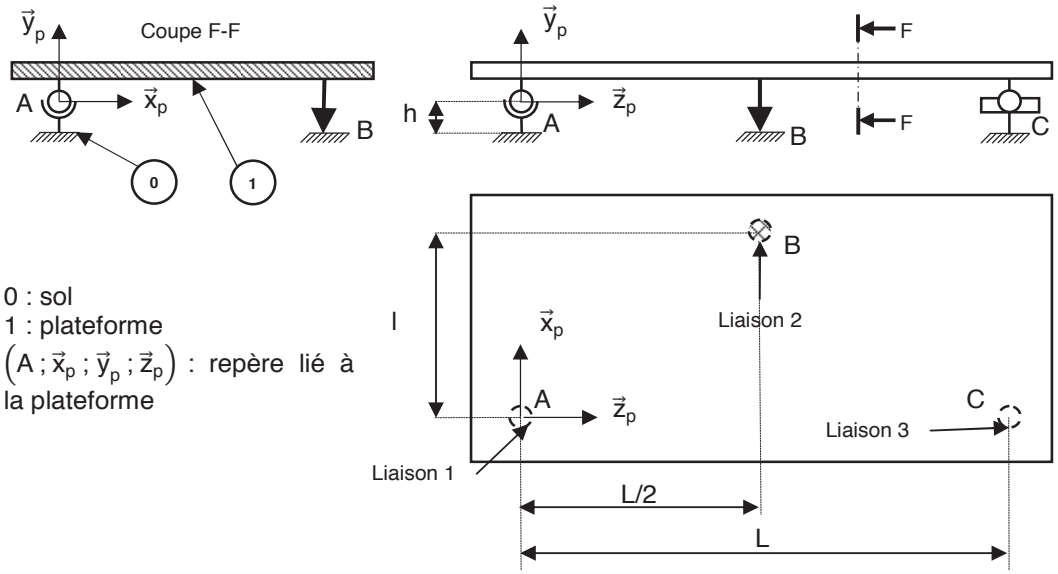
2 Critères et valeurs associées aux exigences fonctionnelles

Exigence	Id	Critères	Valeurs associées
1	Assembler les tronçons à l'aide de rivets	Tension installée entre les tôles Déplacements relatifs entre les tôles	150 daN Aucun
1.1	Sélectionner le type de fixation	Temps de sélection Erreur de position du convoyeur	1 s maxi Inférieure à 0,5 mm
1.2	Percer les tronçons	Erreur par rapport à la position nominale dans le repère avion Quantité de matière enlevée	Erreur inférieure à 0,1 mm Aucun copeau après perçage
1.3	Positionner la cellule dans le hall de production	Position de la cellule par rapport à l'avion Stabilité de la position	Position cellule +/- 1 cm Aucun basculement en fonctionnement
1.4	Piloter la cellule Aider à poser la fixation	Opérateur nécessaire pour le pilotage et la pose	Un opérateur

3 Diagramme de blocs internes de la cellule d'assemblage

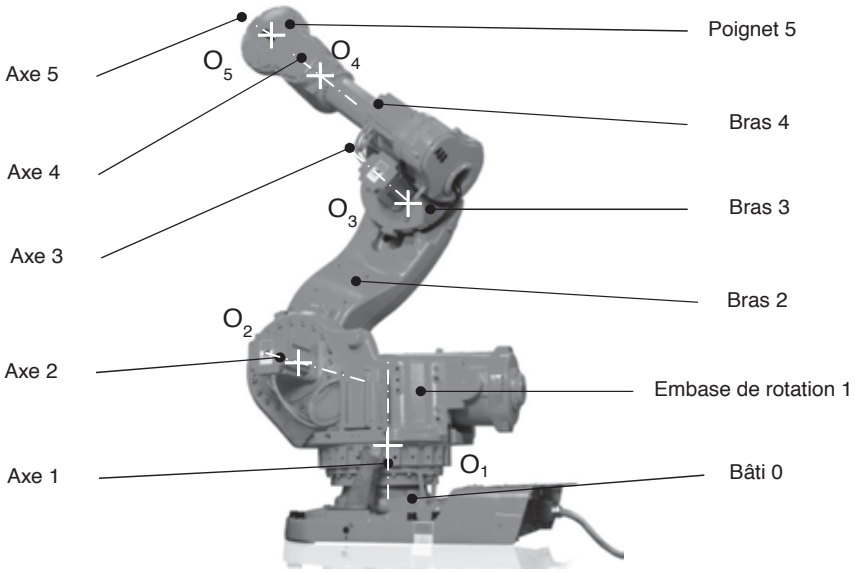


ANNEXE 3 : schéma de la plateforme



0 : sol
1 : plateforme
$\left(A\,;\vec{x}_p\,;\vec{y}_p\,;\vec{z}_p\right)$: repère lié à la plateforme

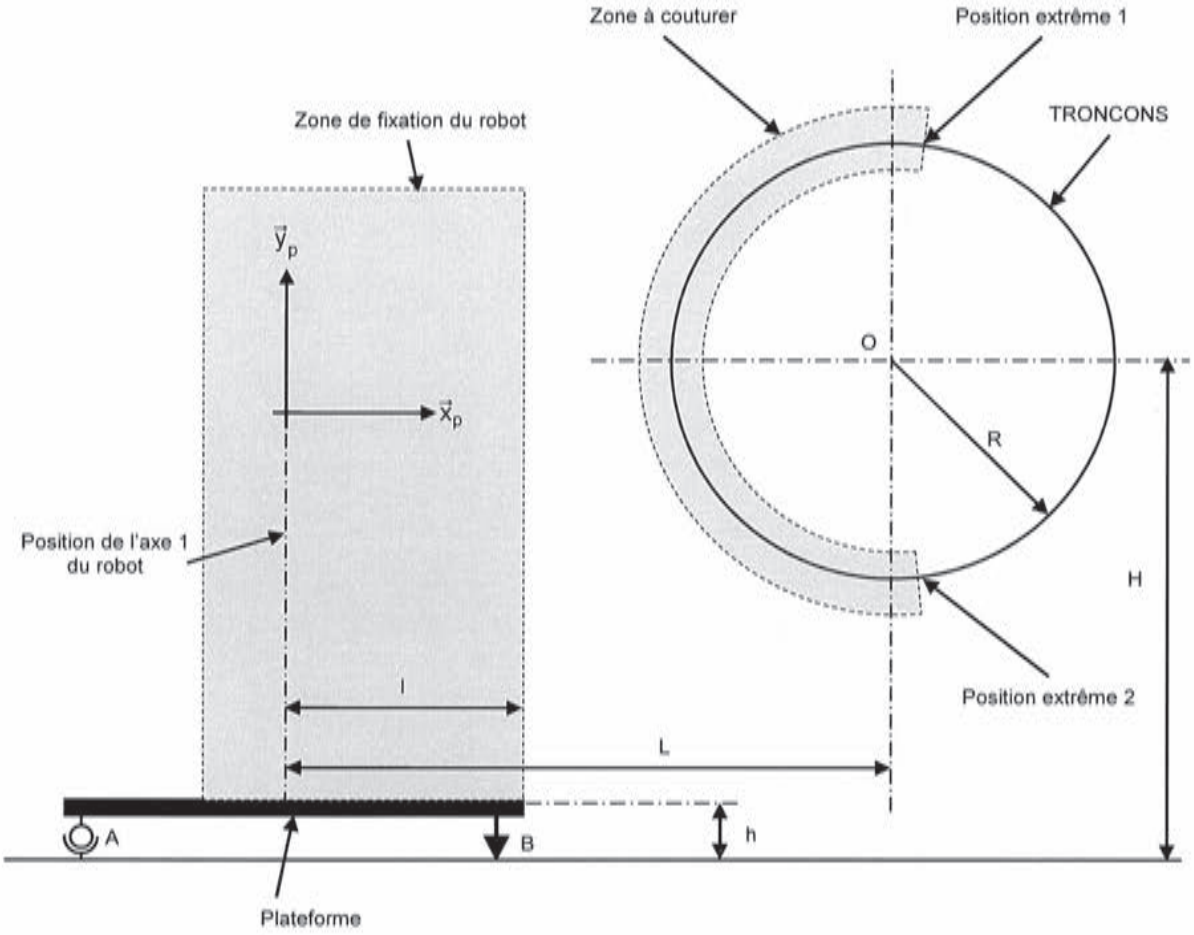
ANNEXE 4 : description du robot



Dénomination des articulations du robot

ANNEXE 5 : choix du robot

1 Schéma d'implantation



Hypothèse :

La zone de fixation du robot est dans le même plan que la zone à couturer.

Données :

- O : centre des troncons ;
- A : position de l'ancrage 1 ;
- B : position de l'ancrage 2 ;
- H : hauteur du centre du tronçon au sol du hall d'implantation H = 2700 mm ;
- h : hauteur du plan supérieur de la plateforme au sol du hall d'implatation h = 300 mm ;
- R : rayon du tronçon à couturer R = 1170 mm ;
- L : distance entre centre du tronçon et l'axe $\vec{y}_p$ L = 3180 mm ;
- l : distance entre l'axe $\vec{y}_p$ et le bord intérieur de la plateforme l = 1240 mm.

2 Documentation ABB pour les robots IRB 7600

IRB 7600

Robot industriel

CARACTÉRISTIQUES TECHNIQUES, ROBOT INDUSTRIEL IRB 7600

SPÉCIFICATIONS

Versions du robot	Rayon d'action	Capacité de charge	Centre de gravité	Couple du poignet
IRB 7600-500	2,30 m	500 kg	360 mm	3010 Nm
IRB 7600-400	2,55 m	400 kg	512 mm	3010 Nm
IRB 7600-340	2,80 m	340 kg	360 mm	2750 Nm
IRB 7600-150'	3,50 m	150 kg	360 mm	1880 Nm

Des charges supplémentaires peuvent être montées sur tous les modèles, 50 kg sur le bras supérieur et 550 kg sur le bâti de l'axe 1.

Nombre d'axes	6
Protection	Robot : IP 67
Montage	Fixé au sol

PERFORMANCES

Mouvement des axes

Axe 1	+180° à -180°
Axe 2	+80° à -60°
Axe 3	+60° à -180°
Axe 4	+300° à -300°
Axe 5	+100° à -100°
Axe 6	+300° à -300°

Vitesses maximum des axes

	500 kg	340/400 kg	150 kg
Axe 1	75°/s	75°/s	100°/s
Axe 2	60°/s	60°/s	60°/s
Axe 3	60°/s	60°/s	60°/s
Axe 4	100°/s	100°/s	100°/s
Axe 5	100°/s	100°/s	100°/s
Axe 6	160°/s	160°/s	160°/s

Une fonction de supervision évite la surchauffe lors d'applications nécessitant des mouvements fréquents.

RACCORDEMENTS ÉLECTRIQUES

Tension d'alimentation	200–600 V, 50/60 Hz

DIMENSIONS ET POIDS

Dimensions embase robot	1206,5 x 1200 mm
Poids (toutes versions)	2500 kg

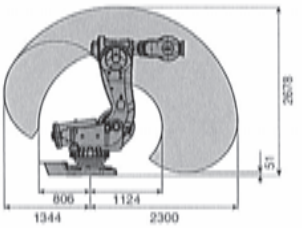
ENVIRONNEMENT

Température ambiante unité mécanique	
En fonctionnement	de +5° C à +50° C
Lors du transport et du stockage	de -25° C à +55° C
Pendant de courtes périodes	jusqu'à +70° C
Humidité relative	95 % maxi
Niveau sonore	73 dB (A) maxi
Sécurité	Circuits redondants avec supervision, arrêts d'urgence et fonctions de sécurité, palette homme-mort 3 positions
Immunité	Blindage CEM/IEM

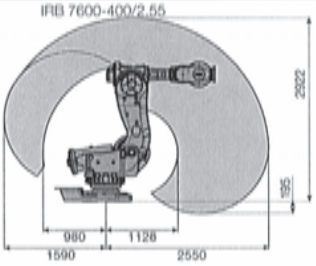
Tous droits de modification des caractéristiques techniques sans préavis.

ENVELOPPES DE TRAVAIL

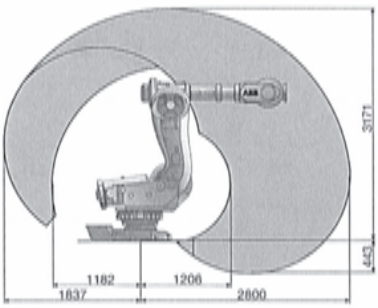
IRB 7600-500/2.30



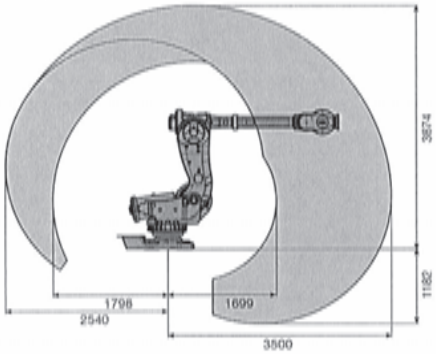
IRB 7600-400/2.55



IRB 7600-340/2.80



IRB 7600-150/3.50



JM411BF - Avril 2005

ABB France - Division Robotique
Rue de l'Equerre - ZI des Béthunes
95310 Saint-Ouen l'Aumône - France
Tél. : +33 (0) 1 34 40 25 25 - Fax : +33 (0) 1 34 40 24 24

ABB

3 Données complémentaires

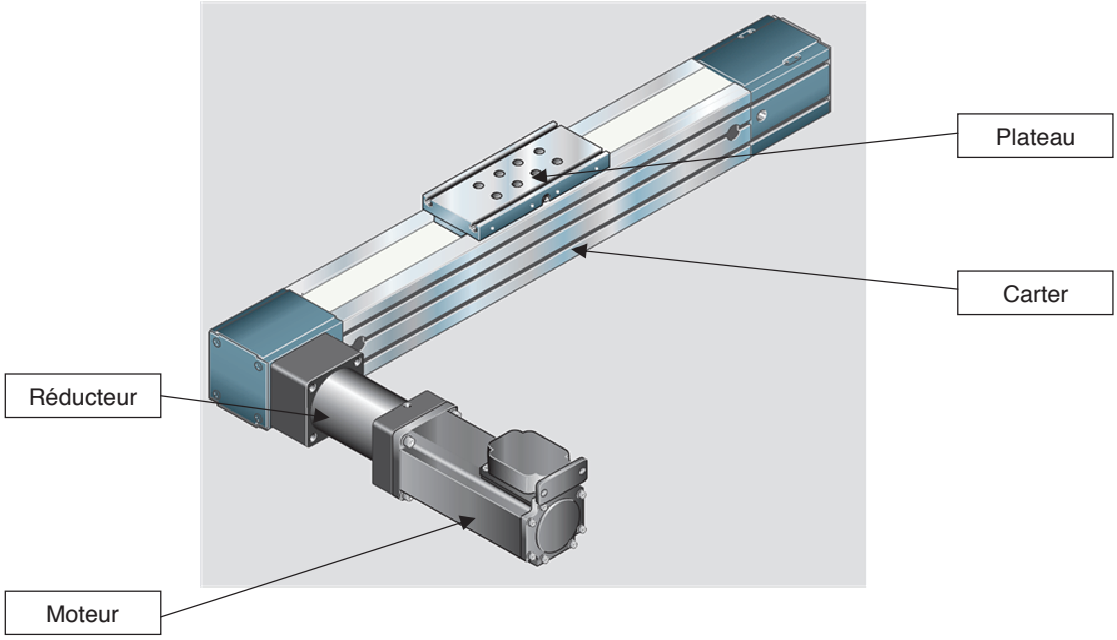
Robot	IRB 7600-500/2.30	IRB 7600-400/2.55	IRB 7600-340/2.80	IRB 7600-150/3.50
L_1	405 mm	405 mm	405 mm	405 mm
L_2	433 mm	433 mm	433 mm	433 mm
L_3	1075 mm	1075 mm	1075 mm	1075 mm
L_4	556 mm	806 mm	1056 mm	1762 mm
L_5	165 mm	165 mm	165 mm	165 mm
L_6	250 mm	250 mm	250 mm	250 mm
C_{12} limite	9000 N.m	9000 N.m	9000 N.m	9000 N.m

ANNEXE 6 : résultats des essais de perçage

Essai	1	2	3	4
Matériaux	1	1	2	2
K_c (en N/mm²)	750	750	1750	1750
Type d'outil	1	1	2	3
Ø outil (en mm)	5	6	5	5
K'	0.5	0.5	0.4	0.5
f (en mm/tour)	0.16	0.20	0.24	0.3

ANNEXE 7 : documentation axe linéaire

1 Vue 3D extérieure axe chariot de sélection



2 Caractéristiques du moteur d'axe

Vitesse nominale (N)	Couple nominal (C_{nom})	Courant nominal (I_{nom})	Courant maxi (I_{max})	Tension nominale (U_{nom})	Inductance (L)	Résistance de l'induit (R)
3000 tr.min^{-1}	3 Nm	4,8 A	30 A	400 V	9 mH	3 ohms

Coefficient de frottement visqueux (f)	Constante de couple (K_c)	Constante de fcem (K_E)
0,2.10^{-2} Nm.s.rad^{-1}	1,3 Nm.A^{-1}	1,3 V.(rad.s^{-1})$^{-1}$

ANNEXE 8 : fichier des mesures

Le fichier texte est nommé « mesure_chariot.txt », il comporte 33 lignes :

```
Mesures axe chariot
Position chariot
t en s
T,X
0,1.02
0.033,1.02
0.066,1.02
0.099,1.02
0.133,1.01
0.166,0.935
0.199,0.86
0.233,0.7549
0.266,0.6422
0.299,0.503
0.333,0.39
0.366,0.29
0.399,0.1877
0.433,0.063
0.466,-0.0337
0.499,-0.1577
0.533,-0.2779
0.566,-0.3830
0.599,-0.5145
0.633,-0.623
0.666,-0.7098
0.699,-0.80
0.733,-0.886
0.766,-0.9239
0.799,-0.93
0.833,-0.93
0.866,-0.93
0.899,-0.93
0.933,-0.93
```

Option informatique

Épreuves d'informatique

Épreuves contenant des questions d'informatique